Cryptology

MS3S05

Mark Griffiths **October 2019**

Contents

List of Figures

List of Tables

List of Definitions

List of Examples

List of Exercises

Glossary of Terminology

Cryptology – is the science of codes and ciphers and covers both their development and methods of breaking them.

Cryptography – is the science of developing codes and ciphers.

Cryptanalysis – is the science of breaking codes and ciphers.

Cleartext – is a message sent without it being put into a hidden form.

Plaintext – is the message before it is put into a hidden form.

Code – is a method of hiding the meaning of a message, it operates at the word or phrase level.

Encode – is the process of applying a code to a message.

Codetext – is the text produced by the application of a code to a message.

Decode – is the process of legitimately removing a code to regain the original message.

Cipher – is a method of hiding the meaning of a message, it operates at the symbol level.

Encipher – is the process of applying a cipher to a message.

Ciphertext – is the text produced by the application of a cipher to a message.

Decipher – is the process of legitimately removing a cipher to regain the original message.

Key – is the information that allows a specific cipher from a cryptographic system to be applied to a message.

Key space – is the set of all possible keys that can be used in a cryptographic system.

Decrypt – is the process of illegitimately removing a code or cipher to regain the original message.

Cryptanalyst – is someone who illegitimately removes a code or cipher to regain the original message.

Sender – is the originator of the ciphertext message.

Receiver – is the recipient of the ciphertext message.

Eavesdropper – is anyone who intercepts the ciphertext message.

Substitution – refers to the class of ciphers that are based on interchanging characters in a message for other characters from a cipher alphabet.

Monoalphabetic – refers to the class of substitution ciphers that are based on a single cipher alphabet.

Polyalphabetic – refers to the class of substitution ciphers that are based on multiple cipher alphabets.

Polygraphic – refers to the class of substitution ciphers that encipher two (or more) characters at a time. They can be monoalphabetic or polyalphabetic.

Transposition – refers to the class of ciphers that use an algorithm to shuffle the characters in a message.

Superencipherment – refers to the application of a cipher to a message that has already been encoded.

1-gram – is a single character from a message.

2-gram – is a pair of characters from a message; also called a bigram or digraph.

3-gram – is a triplet of characters from a message; also called a trigram or trigraph.

***n*-gram** – is a sequence of *n* characters from a message.

List of Major Cryptographic Organisations

Bletchley Park	–	the unofficial name of the UK's wartime cryptographic service. Named after the country house near Milton Keynes where it was based. Also known as Station X.
BSI	–	the Bundesamt fur Sicherheit in der Informationstechnik. The cryptographic service of the German government.
CSEC	–	the Communication Security Establishment of Canada. The cryptographic service of the Canadian government formed in 1946.
DGSE	–	the Direction Generale de la Securite Exterieure. The cryptographic service of the French government.
DSD	–	the Defence Signals Directorate. The cryptographic service of the Australian government formed in 1947.
GCHQ	–	the Government Communications Headquarters. The UK's Cryptographic service based in Cheltenham. Formed in 1919 as the Government Code and Cipher School – GC&CS, which became GCHQ in 1946.
GCSB	–	the Government Communications Security Bureau. The cryptographic service of the New Zealand government formed in 1977.
NSA	–	the National Security Agency. The USA's cryptographic service based in Maryland, formed in 1952.
UKUSA	–	the agreement signed by the UK and USA in 1946 that set up a co-ordinated approach to signal intelligence. This was later joined by Australia, Canada and New Zealand. The UKUSA Agreements was based on the wartime BRUSA Agreement signed in 1943.

Notation

The following notation will be used throughout unless otherwise stated.

A	–	is the plaintext alphabet, normally, *abc…xyz*.
m	–	is the number of different characters in the alphabet A.
Z_m	–	is the set of integers $(0,\ldots,m\text{-}1)$.
n	–	is the number of characters in the plaintext message x or in the ciphertext message y.
x	–	is a plaintext message.
x_i	–	is the i^{th} character in the plaintext message x.
$(x_1, x_2, x_3, ..., x_n)$	–	is the n-gram of characters in the plaintext message x.
X	–	is the set of all possible plaintext messages.
y	–	is a ciphertext message.
y_i	–	is the i^{th} character in the ciphertext message y.
$(y_1, y_2, y_3, ..., y_n)$	–	is the n-gram of characters in the ciphertext message y.
Y	–	is the set of all possible ciphertext messages.
e_k	–	is an encipherment key.
d_k	–	is a decipherment key.
k	–	is a cipher keyword.
k_i	–	is the i^{th} character in the keyword message k.
$(k_1, k_2, k_3, ..., k_r)$	–	is the r-gram of characters in the keyword k.
r	–	is the number of characters in the keyword.
K	–	is the set of all possible keywords, i.e., the key space.

T	–	is an encipherment transformation.
T^{-1}	–	is a decipherment transformation, i.e., the inverse of T.
$\boldsymbol{T}$	–	is the set of all encipherment transformations T.

π	–	is a permutation (typically, of characters in the alphabet A).
$P(E)$	–	is the probability of some event E occurring.
$\mathbf{P}$	–	is the one-step transition matrix of a Markov chain.
$\mathbf{\Pi}$	–	is the limiting matrix of a Markov chain, i.e., the n-step transition matrix as n goes to infinity.
$\boldsymbol{\pi}$	–	is the equilibrium row vector of a Markov chain.

R_0	–	is the seed of a pseudo random number generator (PRNG).
R_i	–	is the i^{th} random integer from a PRNG.
M	–	is the largest possible integer from a PRNG.
U_i	–	is the i^{th} random decimal in the range (0,1) from a PRNG.
$E(\ )$	–	is the expectation of a random variable.
pdf	–	is a probability distribution function.
cdf	–	is a cumulative probability distribution function.
μ	–	is the mean value of a distribution.
σ	–	is the standard deviation of a distribution.
α	–	is the significance level of a statistical test.

Chapter 1 – Models of the Plaintext Source

1.1 Alphabets

Both the plaintext and ciphertext consists of symbols from some alphabet, which is often but not necessarily the same alphabet.

Typical examples of alphabets are:

The lower case letters:

abcdefghijklmnopqrstuvwxyz

The numerals:

0123456789

The binary values of a byte of information:

00000000 00000001 … 11111111

The lower and uppercase letters:

abcde…xyzABCD…XYZ

The lower and uppercase letters plus punctuation marks:

abcde…xyzABCD…XYZ?!",.

The lower and uppercase letters plus punctuation marks and numerals:

abcde…xyzABCD…XYZ?!",.0123456789

The ASCII (American Standard Code for Information Interchange) characters:

!"#$%&'( )+,-./ 0123456789:;<=?@> ABCD…XYZ [\]^–`abcde…xyz{}~*

Note that ASCII also contains some unprintable character such as *delete*, *backspace*, *newline*, etc.

The collection of all symbols in an alphabet forms a set *A*. All the material that follows will use the plaintext alphabet of lowercase letters *abc…xyz*.

So the set that will be used is $A = (a, b, c, \ldots, x, y, z)$.

Throughout what follows the alphabet for the ciphertext will also be *A*. However, note that in worked examples, uppercase letters will be used for ciphertext in place of lowercase letters to avoid confusion between plaintext and ciphertext.

Definition 1.1 *n*-gram

An *n*-gram is a string of *n* letters from the alphabet *A*.

Example 1.1

A 1-gram from the alphabet *A* could be: *a* or *m* or *x*, etc. There are 26 possibilities.

Example 1.2

A 2-gram (also called a bigram or digraph) could be: *aa* or *ax* or *hv* or *xy*. There are $26^2 = 676$ possibilities.

Example 1.3

A 3-gram (also called a trigram or trigraph) could be: *aaa* or *bax* or *htv* or *xya*. There are $26^3 = 17576$ possibilities.

Example 1.4

A word such as "*tomorrow*" can be regarded as a sequence of eight 1-grams *t*/*o*/*m*/*o*/*r*/*r*/*o*/*w* or as four 2-grams *to*/*mo*/*rr*/*ow* or as two 4-grams *tomo*/*rrow*.

Definition 1.2 Plaintext
A plaintext *x* from the alphabet *A* is an *n*-gram.

$$\text{Plaintext: } x = (x_1, x_2, x_3, ..., x_n) \qquad x_i \in A \quad 1 \le i \le n$$

1.2 The 1-gram Model of English

If the language from which the plaintext is drawn is English there are various characteristics that are evident. For example, the letter *e* is very common, the letter *z* is uncommon, the letter *q* is always followed by a letter *u*, etc. These features could be regarded as a series of facts, however, it is more useful to consider them as statistical properties of the language (see Table 1.1, and Figures 1.1 and 1.2).

Note that the average word length in English is about 5 letters. This means that if a *space* was included as a symbol in the alphabet then it would have a frequency of approximately 20% and the frequency of the other letters in Table 1.1 would be scaled back by 20%, for example, the letter *e* would have a frequency of approximately 10%. This is one of the reasons that when using anything but modern ciphers the practice is to remove spaces as they give far too strong a clue to the cryptanalyst.

It is important to realise that the distribution in English (or any other language) of 1-grams will vary from text to text. So even for a monoalphabetic cipher it is impossible to simply match up the ciphertext and the plaintext by ranking on the probabilities. However, with the use of some judgement it does allow this type of cipher to be decrypted.

Furthermore, knowledge of the source of the text allows a better model of the plaintext to be built. For example, British government documents will use a different style of English from that of an Australian magazine article or that used in an American business letter. This means that by analysing the specific source from which the enciphered plaintext comes from a more accurate model can be built.

Alphabetically		By Probability	
1-gram	**Probability**	**1-gram**	**Probability**
a	0.08064	**e**	0.12886
b	0.01537	**t**	0.09025
c	0.02689	**a**	0.08064
d	0.04329	**o**	0.07378
e	0.12886	**n**	0.06985
f	0.02448	**i**	0.06906
g	0.01963	**s**	0.06382
h	0.06099	**r**	0.06157
i	0.06906	**h**	0.06099
j	0.00112	**d**	0.04329
k	0.00625	**l**	0.04102
l	0.04102	**u**	0.02786
m	0.02501	**c**	0.02689
n	0.06985	**m**	0.02501
o	0.07378	**f**	0.02448
p	0.01703	**w**	0.02119
q	0.00106	**g**	0.01963
r	0.06157	**y**	0.01806
s	0.06382	**p**	0.01703
t	0.09025	**b**	0.01537
u	0.02786	**v**	0.01026
v	0.01026	**k**	0.00625
w	0.02119	**x**	0.00169
x	0.00169	**j**	0.00112
y	0.01806	**q**	0.00106
z	0.00097	**z**	0.00097

Table 1.1 English 1-gram Probability Distribution

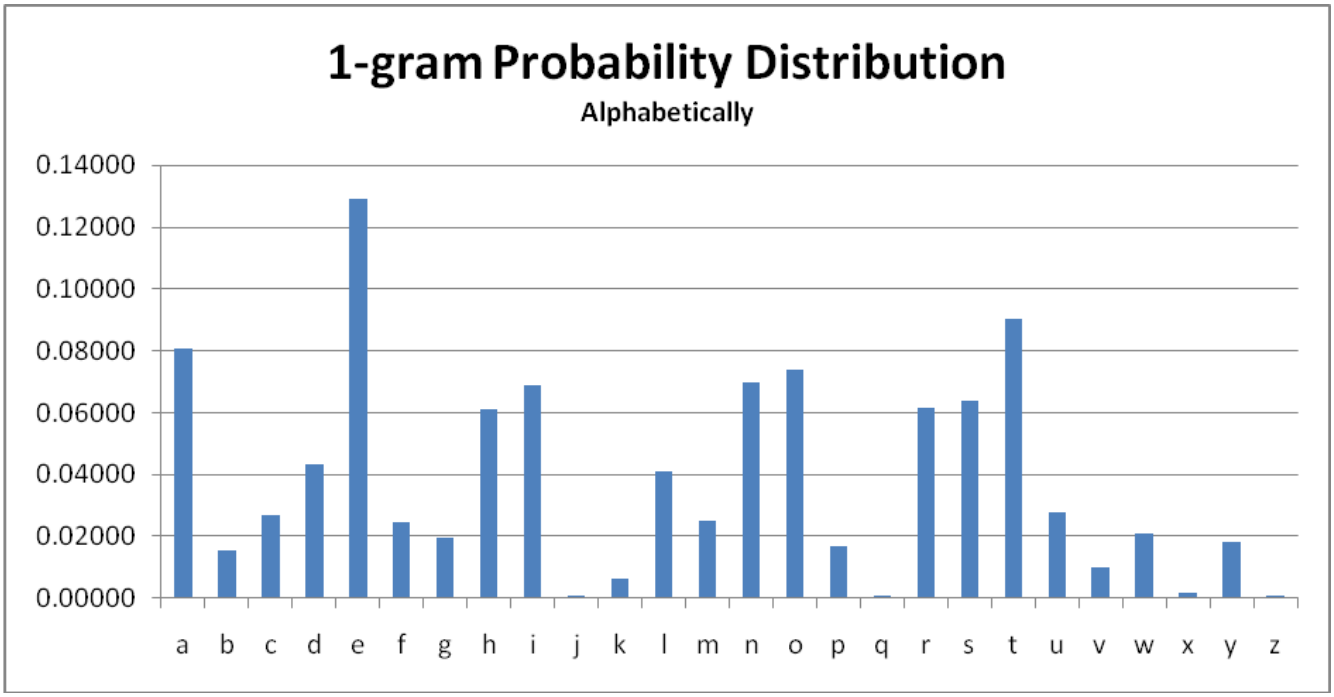

Figure 1.1 English 1-gram Probability Distribution (alphabetically)

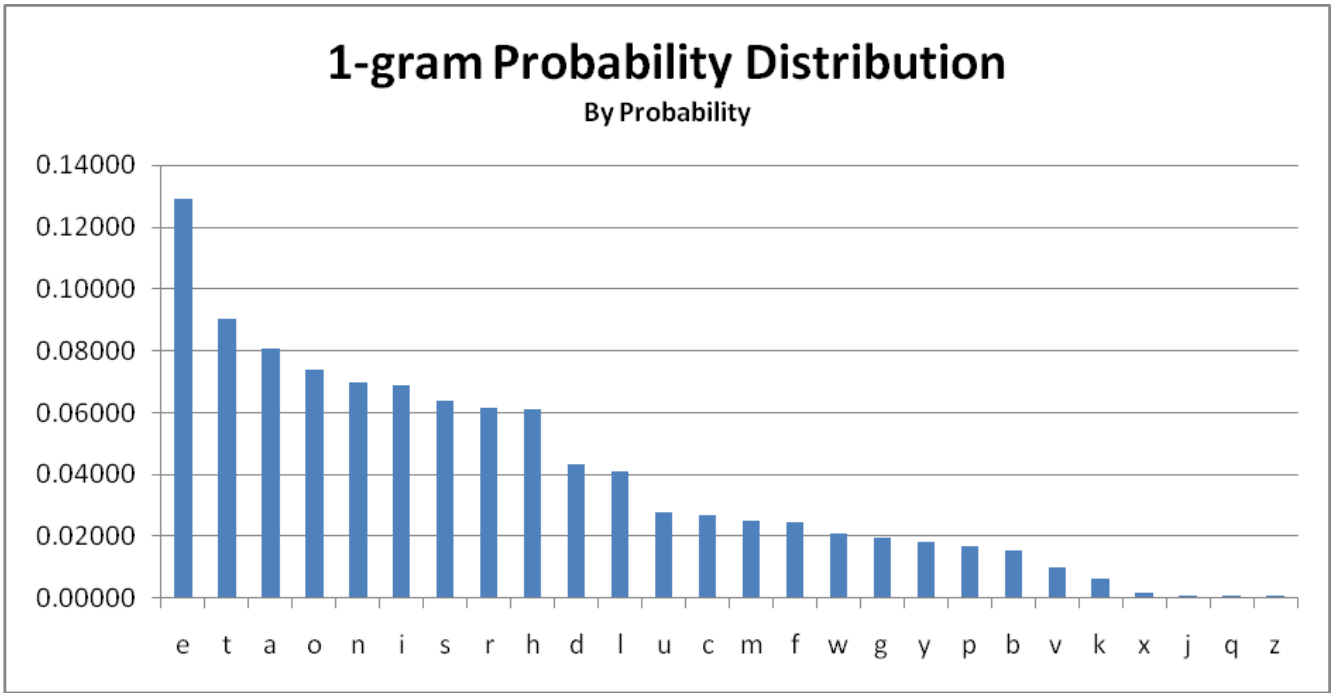

Figure 1.2 English 1-gram Probability Distribution (by probability)

It is possible to model a plaintext source of *n*-grams as a stochastic process of *n* random variables:

$$X_1, X_2, X_3, ..., X_n$$

and define the probability that the string of characters $x = (x_1, x_2, x_3, ..., x_n)$ will occur as the probability of the event

$$P\{(x_1, x_2, x_3, ..., x_n)\} = P\{X_1 = x_1, X_2 = x_2, X_3 = x_3, ..., X_n = x_n\}$$

The *n*-gram probabilities must satisfy three conditions, namely:

$$P\{(x_1, x_2, x_3, ..., x_n)\} \geq 0$$

$$1 = \sum_{(x_1, ..., x_n)} P\{(x_1, x_2, x_3, ..., x_n)\}$$

and

$$P\{(x_1, x_2, x_3, ..., x_n)\} = \sum_{(x_1, ..., x_k)} P\{(x_1, x_2, x_3, ..., x_k)\} \text{ if } k > n$$

The last condition follows from the Kolmogorov Consistency Theorem and is sometimes called the *prefix* condition, which states that, for example, $P\{sub\}$ the probability of the prefix *sub* is equal to the sum of all the probabilities of six letter words (for $k = 6$) starting with *sub*, that is all word of the form *sub****.

Definition 1.3 The 1-gram Plaintext Source
A plaintext source is generated by 1-gram independent and identical trials if

$$P\{(x_1, x_2, x_3, ..., x_n)\} = P(x_1)P(x_2)P(x_3)...P(x_n)$$

Example 1.5

Using Definition 1.3 and the 1-gram statistics for English from Table 1.1:

$$\begin{aligned} P\{stop\} &= P(s)P(t)P(o)P(p) \\ &= 0.06382 \times 0.09025 \times 0.07378 \times 0.01703 \\ &= 7.2370 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P\{tspo\} &= P(t)P(s)P(p)P(o) \\ &= 0.09025 \times 0.06382 \times 0.01703 \times 0.07378 \\ &= 7.2370 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P\{opst\} &= P(o)P(p)P(s)P(t) \\ &= 0.07378 \times 0.01703 \times 0.06382 \times 0.09025 \\ &= 7.2370 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P\{qu\} &= P(q)P(u) \\ &= 0.00106 \times 0.02786 \\ &= 2.9532 \times 10^{-5} \end{aligned}$$

$$\begin{aligned} P\{qt\} &= P(q)P(t) \\ &= 0.00106 \times 0.09025 \\ &= 9.5665 \times 10^{-5} \end{aligned}$$

In other word, in a sample of 1,000,000 4-grams it will be expected that *stop* will occur 7 times, and words of the form *qu*** will occur 30 times. If it seems strange that *qu*** occurs more frequently than *stop* this is because there is only one arrangement of 4 letters that produces *stop*, but there are many that produce *qu***. For example, some of the possible words and parts of words in English are *quit(s, e*), *quie(t, ter, test), ques(t, ts, tion, tions),* etc.

Clearly, this model does not capture all aspects of English, since it indicates that *stop* is as equally likely to occur as *stpo* and *opst*, which is unrealistic. It also indicates that *qt* could occur and even that it is more likely than *qu*, which again is unrealistic.

To try to capture the behaviour of how one letter follows another it is necessary to analyse the likelihood of pairs of letters occurring.

1.3 The 2-gram Model of English

Definition 1.4 The 2-gram Plaintext Source
A plaintext source is generated by 2-gram independent and identical trials if

$$P\{(x_1, x_2, x_3, ..., x_{2n})\} = P(x_1, x_2)P(x_3x_4)...P(x_{2n-1}, x_{2n})$$

Example 1.6

Using Definition 1.4 and the 2-gram statistics for English from Table 1.2 (parts a & b):

$$\begin{aligned} P\{stop\} &= P(st)P(op) \\ &= 0.01353 \times 0.00117 \\ &= 1.5830 \times 10^{-5} \end{aligned}$$

$$\begin{aligned} P\{tspo\} &= P(ts)P(po) \\ &= 0.00293 \times 0.00220 \\ &= 6.4460 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P\{opst\} &= P(op)P(st) \\ &= 0.00117 \times 0.01353 \\ &= 1.5830 \times 10^{-5} \end{aligned}$$

$$\begin{aligned} P\{qu\} &= P(qu) \\ &= 1.0600 \times 10^{-3} \end{aligned}$$

$$\begin{aligned} P\{qt\} &= P(qt) \\ &= 0 \end{aligned}$$

The plaintext *stop* is now predicted to occur more often than the plaintext *tspo* and plaintext *qt* is predicted to never occur. Clearly, this model captures more of the behaviour of English than the 1-gram model given in Definition 1.3 does.

However, the plaintext *stop* is still predicted to occur equally as often as the plaintext *opst*, which is unrealistic.

	2nd Letter												
1st Letter	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
a	0.00002	0.00198	0.00321	0.00370	0.00011	0.00066	0.00149	0.00013	0.00324	0.00002	0.00096	0.00676	0.00226
b	0.00090	0.00009	0.00000	0.00004	0.00529	0.00000	0.00001	0.00001	0.00074	0.00012	0.00000	0.00178	0.00002
c	0.00336	0.00000	0.00059	0.00001	0.00469	0.00000	0.00000	0.00465	0.00154	0.00000	0.00124	0.00133	0.00000
d	0.00319	0.00006	0.00003	0.00092	0.01396	0.00008	0.00062	0.00008	0.00906	0.00004	0.00002	0.00106	0.00024
e	0.01086	0.00025	0.00446	0.01679	0.00631	0.00170	0.00105	0.00035	0.00253	0.00005	0.00022	0.00677	0.00371
f	0.00249	0.00001	0.00001	0.00001	0.00362	0.00195	0.00000	0.00000	0.00320	0.00000	0.00000	0.00104	0.00000
g	0.00172	0.00001	0.00002	0.00006	0.00430	0.00001	0.00029	0.00412	0.00145	0.00000	0.00000	0.00115	0.00009
h	0.01087	0.00005	0.00014	0.00004	0.03163	0.00003	0.00000	0.00000	0.00862	0.00000	0.00000	0.00010	0.00009
i	0.00122	0.00051	0.00356	0.00266	0.00352	0.00199	0.00225	0.00001	0.00004	0.00000	0.00054	0.00386	0.00302
j	0.00024	0.00000	0.00000	0.00000	0.00021	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
k	0.00015	0.00001	0.00000	0.00002	0.00296	0.00004	0.00000	0.00001	0.00116	0.00000	0.00000	0.00017	0.00001
l	0.00487	0.00003	0.00010	0.00320	0.00794	0.00064	0.00006	0.00001	0.00531	0.00000	0.00026	0.00650	0.00027
m	0.00457	0.00085	0.00000	0.00008	0.00742	0.00005	0.00000	0.00001	0.00261	0.00000	0.00000	0.00005	0.00066
n	0.00226	0.00005	0.00398	0.01662	0.00798	0.00050	0.01118	0.00027	0.00278	0.00011	0.00067	0.00082	0.00006
o	0.00070	0.00062	0.00107	0.00171	0.00025	0.00959	0.00051	0.00016	0.00074	0.00002	0.00090	0.00227	0.00490
p	0.00233	0.00001	0.00001	0.00001	0.00402	0.00000	0.00000	0.00040	0.00098	0.00000	0.00000	0.00174	0.00002
q	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
r	0.00582	0.00025	0.00093	0.00225	0.01862	0.00045	0.00085	0.00021	0.00629	0.00000	0.00069	0.00093	0.00153
s	0.00373	0.00007	0.00168	0.00016	0.01258	0.00011	0.00005	0.00608	0.00540	0.00001	0.00057	0.00122	0.00070
t	0.00470	0.00002	0.00044	0.00001	0.01067	0.00011	0.00000	0.04008	0.00797	0.00000	0.00000	0.00175	0.00011
u	0.00077	0.00054	0.00136	0.00066	0.00091	0.00017	0.00150	0.00000	0.00088	0.00000	0.00000	0.00315	0.00091
v	0.00105	0.00000	0.00000	0.00005	0.00699	0.00000	0.00000	0.00000	0.00151	0.00000	0.00000	0.00000	0.00000
w	0.00441	0.00001	0.00001	0.00005	0.00386	0.00003	0.00000	0.00455	0.00430	0.00000	0.00001	0.00022	0.00001
x	0.00011	0.00000	0.00030	0.00000	0.00010	0.00000	0.00000	0.00002	0.00021	0.00000	0.00000	0.00000	0.00000
y	0.00048	0.00026	0.00008	0.00014	0.00375	0.00010	0.00002	0.00005	0.00094	0.00000	0.00000	0.00021	0.00026
z	0.00043	0.00000	0.00001	0.00001	0.00031	0.00000	0.00000	0.00000	0.00007	0.00000	0.00000	0.00002	0.00000

Table 1.2a English 2-gram Probability Distribution

	2nd Letter												
1st Letter	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
a	0.01863	0.00005	0.00170	0.00002	0.00960	0.00811	0.01145	0.00085	0.00266	0.00077	0.00008	0.00203	0.00016
b	0.00001	0.00150	0.00000	0.00000	0.00115	0.00035	0.00017	0.00185	0.00002	0.00000	0.00000	0.00134	0.00000
c	0.00000	0.00480	0.00000	0.00005	0.00103	0.00003	0.00248	0.00098	0.00000	0.00000	0.00000	0.00011	0.00000
d	0.00035	0.00453	0.00002	0.00000	0.00204	0.00350	0.00010	0.00204	0.00032	0.00016	0.00000	0.00087	0.00000
e	0.01623	0.00064	0.00202	0.00043	0.02602	0.01419	0.00481	0.00028	0.00313	0.00141	0.00209	0.00251	0.00004
f	0.00000	0.00603	0.00001	0.00000	0.00322	0.00015	0.00134	0.00131	0.00000	0.00001	0.00000	0.00006	0.00000
g	0.00037	0.00166	0.00002	0.00000	0.00257	0.00076	0.00018	0.00074	0.00000	0.00001	0.00000	0.00009	0.00000
h	0.00009	0.00482	0.00000	0.00002	0.00081	0.00015	0.00205	0.00090	0.00000	0.00007	0.00000	0.00050	0.00000
i	0.01916	0.00364	0.00046	0.00003	0.00320	0.00847	0.00877	0.00005	0.00173	0.00000	0.00014	0.00000	0.00024
j	0.00000	0.00030	0.00000	0.00000	0.00000	0.00000	0.00000	0.00037	0.00000	0.00000	0.00000	0.00000	0.00000
k	0.00090	0.00010	0.00001	0.00000	0.00000	0.00055	0.00002	0.00003	0.00000	0.00004	0.00000	0.00008	0.00000
l	0.00005	0.00396	0.00011	0.00000	0.00009	0.00110	0.00087	0.00069	0.00020	0.00012	0.00000	0.00464	0.00000
m	0.00009	0.00326	0.00144	0.00000	0.00050	0.00074	0.00003	0.00115	0.00000	0.00000	0.00000	0.00150	0.00000
n	0.00080	0.00614	0.00003	0.00009	0.00004	0.00401	0.00899	0.00088	0.00031	0.00008	0.00003	0.00114	0.00001
o	0.01233	0.00279	0.00117	0.00002	0.00993	0.00272	0.00416	0.01129	0.00139	0.00412	0.00005	0.00029	0.00007
p	0.00001	0.00220	0.00110	0.00000	0.00247	0.00040	0.00074	0.00049	0.00000	0.00002	0.00000	0.00008	0.00000
q	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00106	0.00000	0.00000	0.00000	0.00000	0.00000
r	0.00176	0.00730	0.00038	0.00000	0.00150	0.00364	0.00335	0.00138	0.00056	0.00016	0.00000	0.00226	0.00044
s	0.00024	0.00524	0.00307	0.00009	0.00002	0.00495	0.01353	0.00352	0.00001	0.00057	0.00000	0.00022	0.00000
t	0.00013	0.01091	0.00001	0.00000	0.00371	0.00293	0.00205	0.00222	0.00000	0.00094	0.00000	0.00145	0.00003
u	0.00342	0.00006	0.00133	0.00001	0.00481	0.00357	0.00370	0.00000	0.00003	0.00001	0.00003	0.00001	0.00003
v	0.00013	0.00044	0.00000	0.00000	0.00001	0.00000	0.00000	0.00003	0.00001	0.00000	0.00000	0.00004	0.00000
w	0.00090	0.00225	0.00000	0.00000	0.00020	0.00030	0.00004	0.00003	0.00000	0.00001	0.00000	0.00002	0.00000
x	0.00000	0.00001	0.00041	0.00001	0.00000	0.00001	0.00044	0.00004	0.00001	0.00000	0.00002	0.00000	0.00000
y	0.00015	0.00803	0.00019	0.00000	0.00033	0.00216	0.00073	0.00004	0.00000	0.00011	0.00001	0.00001	0.00001
z	0.00000	0.00007	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001	0.00003

Table 1.2b English 2-gram Probability Distribution (continued)

1.4 The Markov Model of English

It is possible to construct an even more sophisticated model based on Markov chains. A Markov chain consists of states that completely describe the system at a point in time together with the probabilities of moving from one state to another state.

In the case of using a Markov chain to model English, the states are the current letter of the plaintext. Figure 1.3 shows a small part of the Markov chain for English:

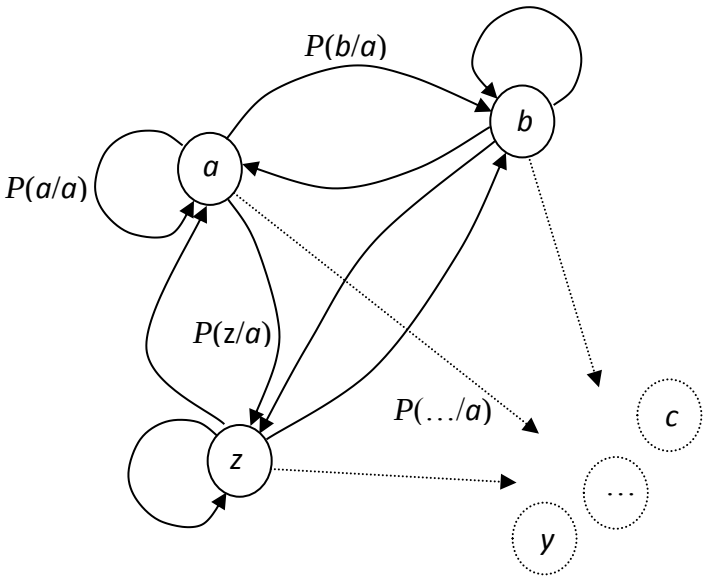

Figure 1.3 Part of the Markov Chain for English

This gives the transition matrix $\mathbf{P} = \begin{pmatrix} P(a/a) & P(b/a) & \ldots & P(z/a) \\ P(a/b) & P(b/b) & \ldots & P(z/b) \\ . & . & & . \\ . & . & & . \\ P(a/z) & P(b/z) & \ldots & P(z/z) \end{pmatrix}$

where $P(j/i)$ is the conditional probability of j given i, in other words, the probability of moving to state j from state i.

These probabilities of moving from one state to another can be calculated from the 2-gram probabilities.

If the transition matrix **P** is raised to higher and higher powers it can be observed that all the values in a column converge. This means that each row will become identical, which indicates that the chance of ending up in a particular state in the long term is

independent of the initial state. Note that this does not apply to all matrices or all Markov chains only to Markov chains that can be classified as *regular*.

Formally, if **P** is the transition matrix of a regular Markov chain, then $\mathbf{P}^n$ approaches an unique limiting matrix $\mathbf{\Pi}$, which has all of its rows equal to:

$$\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_m) \qquad m = 26 \text{ for alphabet } A$$

as *n* tends to infinity.

Now, since $\mathbf{\Pi}$ has converged multiplying it by **P** will leave it unchanged, that is, $\mathbf{\Pi P} = \mathbf{\Pi}$. Consequently, each row must also obey $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$.

The row vector can be determined by solving the equations $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$ plus the equation $\sum_{j=1}^{m} \pi_j = 1$. Note that, as this gives 27 equations in 26 unknowns one of the equations from $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$ must be left unused.

The Markov transition probabilities for English are shown in Table 1.3 (parts a & b), and the associated equilibrium values are shown in Table 1.4.

	2nd Letter												
1st Letter	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
a	0.00031	0.02459	0.03975	0.04591	0.00132	0.00816	0.01846	0.00158	0.04014	0.00025	0.01186	0.08382	0.02808
b	0.05844	0.00591	0.00008	0.00228	0.34424	0.00021	0.00043	0.00037	0.04827	0.00775	0.00000	0.11592	0.00129
c	0.12488	0.00003	0.02188	0.00053	0.17424	0.00001	0.00000	0.17279	0.05742	0.00000	0.04610	0.04964	0.00000
d	0.07380	0.00133	0.00068	0.02135	0.32246	0.00179	0.01444	0.00186	0.20927	0.00092	0.00050	0.02439	0.00565
e	0.08431	0.00198	0.03465	0.13027	0.04898	0.01323	0.00816	0.00274	0.01964	0.00040	0.00169	0.05251	0.02877
f	0.10183	0.00038	0.00051	0.00033	0.14803	0.07978	0.00003	0.00016	0.13086	0.00000	0.00003	0.04252	0.00006
g	0.08758	0.00057	0.00077	0.00297	0.21914	0.00040	0.01502	0.20994	0.07408	0.00004	0.00005	0.05841	0.00442
h	0.17827	0.00087	0.00231	0.00058	0.51861	0.00052	0.00004	0.00008	0.14132	0.00000	0.00000	0.00161	0.00147
i	0.01770	0.00733	0.05158	0.03852	0.05094	0.02876	0.03262	0.00008	0.00051	0.00000	0.00775	0.05583	0.04376
j	0.21145	0.00000	0.00000	0.00000	0.19135	0.00000	0.00000	0.00000	0.00044	0.00000	0.00000	0.00000	0.00000
k	0.02413	0.00166	0.00072	0.00244	0.47368	0.00594	0.00067	0.00111	0.18552	0.00000	0.00044	0.02652	0.00139
l	0.11882	0.00064	0.00239	0.07813	0.19368	0.01566	0.00138	0.00018	0.12946	0.00001	0.00628	0.15836	0.00647
m	0.18285	0.03397	0.00020	0.00302	0.29651	0.00211	0.00002	0.00036	0.10423	0.00002	0.00000	0.00187	0.02635
n	0.03232	0.00073	0.05696	0.23787	0.11429	0.00714	0.15999	0.00388	0.03981	0.00165	0.00960	0.01181	0.00089
o	0.00952	0.00835	0.01455	0.02324	0.00337	0.13001	0.00694	0.00212	0.01006	0.00023	0.01224	0.03075	0.06644
p	0.13682	0.00047	0.00047	0.00050	0.23577	0.00015	0.00009	0.02356	0.05739	0.00003	0.00012	0.10200	0.00116
q	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
r	0.09457	0.00414	0.01506	0.03662	0.30239	0.00730	0.01384	0.00341	0.10216	0.00006	0.01124	0.01512	0.02481
s	0.05844	0.00108	0.02636	0.00254	0.19713	0.00176	0.00078	0.09521	0.08455	0.00008	0.00899	0.01913	0.01095
t	0.05209	0.00021	0.00493	0.00009	0.11819	0.00122	0.00005	0.44407	0.08831	0.00001	0.00002	0.01937	0.00121
u	0.02776	0.01930	0.04881	0.02370	0.03280	0.00624	0.05367	0.00007	0.03166	0.00001	0.00017	0.11321	0.03277
v	0.10243	0.00000	0.00000	0.00497	0.68108	0.00000	0.00000	0.00024	0.14706	0.00000	0.00000	0.00014	0.00000
w	0.20795	0.00038	0.00034	0.00215	0.18220	0.00148	0.00004	0.21485	0.20300	0.00000	0.00057	0.01015	0.00031
x	0.06665	0.00000	0.17474	0.00112	0.06154	0.00000	0.00016	0.01467	0.12468	0.00000	0.00000	0.00159	0.00016
y	0.02662	0.01450	0.00442	0.00765	0.20755	0.00561	0.00114	0.00263	0.05215	0.00005	0.00010	0.01177	0.01440
z	0.44509	0.00079	0.00635	0.00714	0.32046	0.00000	0.00000	0.00053	0.07489	0.00026	0.00000	0.02091	0.00000

Table 1.3a English Transition Matrix

	2nd Letter												
1st Letter	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
a	0.23106	0.00059	0.02103	0.00019	0.11899	0.10063	0.14195	0.01056	0.03302	0.00956	0.00103	0.02517	0.00201
b	0.00035	0.09738	0.00002	0.00000	0.07459	0.02286	0.01082	0.12051	0.00116	0.00003	0.00000	0.08710	0.00000
c	0.00004	0.17864	0.00001	0.00174	0.03826	0.00107	0.09207	0.03639	0.00000	0.00000	0.00000	0.00426	0.00001
d	0.00802	0.10459	0.00043	0.00006	0.04710	0.08083	0.00241	0.04707	0.00731	0.00359	0.00000	0.02013	0.00000
e	0.12592	0.00499	0.01565	0.00331	0.20195	0.11009	0.03731	0.00221	0.02430	0.01095	0.01624	0.01946	0.00028
f	0.00014	0.24641	0.00027	0.00002	0.13162	0.00631	0.05453	0.05342	0.00002	0.00032	0.00000	0.00242	0.00000
g	0.01891	0.08474	0.00082	0.00000	0.13079	0.03861	0.00923	0.03789	0.00005	0.00051	0.00000	0.00484	0.00020
h	0.00146	0.07903	0.00005	0.00036	0.01322	0.00249	0.03353	0.01483	0.00000	0.00116	0.00000	0.00816	0.00000
i	0.27753	0.05275	0.00664	0.00046	0.04637	0.12271	0.12699	0.00074	0.02499	0.00002	0.00199	0.00000	0.00342
j	0.00000	0.26540	0.00000	0.00000	0.00000	0.00000	0.00000	0.33115	0.00000	0.00000	0.00000	0.00022	0.00000
k	0.14397	0.01620	0.00172	0.00000	0.00061	0.08749	0.00327	0.00427	0.00000	0.00588	0.00000	0.01237	0.00000
l	0.00114	0.09658	0.00265	0.00001	0.00216	0.02689	0.02111	0.01688	0.00486	0.00301	0.00000	0.11321	0.00003
m	0.00351	0.13049	0.05763	0.00000	0.02018	0.02969	0.00101	0.04578	0.00003	0.00014	0.00000	0.06002	0.00000
n	0.01145	0.08796	0.00045	0.00131	0.00056	0.05747	0.12867	0.01263	0.00451	0.00114	0.00045	0.01631	0.00017
o	0.16709	0.03784	0.01587	0.00026	0.13453	0.03693	0.05639	0.15305	0.01886	0.05590	0.00062	0.00388	0.00098
p	0.00062	0.12912	0.06473	0.00000	0.14512	0.02375	0.04352	0.02886	0.00000	0.00125	0.00000	0.00450	0.00000
q	0.00000	0.00000	0.00000	0.00000	0.00023	0.00023	0.00000	0.99954	0.00000	0.00000	0.00000	0.00000	0.00000
r	0.02862	0.11853	0.00619	0.00004	0.02442	0.05920	0.05435	0.02240	0.00910	0.00256	0.00001	0.03668	0.00721
s	0.00376	0.08208	0.04809	0.00139	0.00032	0.07751	0.21206	0.05516	0.00013	0.00900	0.00000	0.00350	0.00000
t	0.00146	0.12093	0.00011	0.00000	0.04109	0.03250	0.02267	0.02460	0.00000	0.01042	0.00000	0.01608	0.00037
u	0.12273	0.00214	0.04761	0.00019	0.17272	0.12818	0.13269	0.00004	0.00099	0.00021	0.00094	0.00032	0.00107
v	0.01285	0.04301	0.00000	0.00000	0.00083	0.00031	0.00000	0.00259	0.00059	0.00000	0.00000	0.00390	0.00000
w	0.04230	0.10597	0.00009	0.00001	0.00931	0.01435	0.00206	0.00133	0.00000	0.00036	0.00000	0.00078	0.00000
x	0.00000	0.00622	0.23964	0.00351	0.00000	0.00303	0.26212	0.02168	0.00574	0.00016	0.01036	0.00223	0.00000
y	0.00829	0.44460	0.01028	0.00000	0.01803	0.11984	0.04043	0.00219	0.00025	0.00591	0.00050	0.00035	0.00075
z	0.00000	0.07700	0.00000	0.00000	0.00026	0.00185	0.00053	0.00397	0.00106	0.00000	0.00000	0.01058	0.02831

Table 1.3b English Transition Matrix (continued)

1-gram	Probability
a	0.06881
b	0.00539
c	0.02356
d	0.05542
e	0.15809
f	0.01870
g	0.02182
h	0.05046
i	0.06941
j	0.00033
k	0.00618
l	0.04258
m	0.02006
n	0.07786
o	0.07726
p	0.01395
q	0.00087
r	0.07681
s	0.06696
t	0.07134
u	0.03099
v	0.01105
w	0.00892
x	0.00293
y	0.01907
z	0.00118

Table 1.4 English Markov Chain Equilibrium Values

Definition 1.5 The Markov Plaintext Source
A plaintext source is generated (1-gram at a time) by a Markov chain with a transition matrix **P** and equilibrium distribution $\boldsymbol{\pi}$ if

$$P\{(x_1, x_2, x_3, ..., x_n)\} = \pi(x_1)P(x_2 / x_1)P(x_3 / x_2)...P(x_n / x_{n-1})$$

Example 1.7

Using Definition 1.5 and the Markov transition and equilibrium statistics for English from Tables 1.3 and 1.4:

$$\begin{aligned} P\{stop\} &= \pi(s)P(t/s)P(o/t)P(p/o) \\ &= 0.06696 \times 0.21206 \times 0.12093 \times 0.01587 \\ &= 2.7251 \times 10^{-5} \end{aligned}$$

$$\begin{aligned} P\{tspo\} &= \pi(t)P(s/t)P(p/s)P(o/p) \\ &= 0.07134 \times 0.03250 \times 0.04809 \times 0.12912 \\ &= 1.4397 \times 10^{-5} \end{aligned}$$

$$\begin{aligned} P\{opst\} &= \pi(o)P(p/o)P(s/p)P(t/s) \\ &= 0.07726 \times 0.01587 \times 0.02375 \times 0.21206 \\ &= 6.1752 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P\{qu\} &= \pi(q)P(u/q) \\ &= 0.00087 \times 0.99954 \\ &= 8.6960 \times 10^{-4} \end{aligned}$$

$$\begin{aligned} P\{qt\} &= \pi(q)P(t/q) \\ &= 0.00087 \times 0 \\ &= 0 \end{aligned}$$

The plaintext *stop* is now predicted to occur more often than both the plaintexts *tspo* and *opst*, and plaintext *qt* is predicted to never occur. The Markov model captures more of the behaviour of English than either the 1-gram model or the 2-gram model.

Exercise 1.1

Consider the following English texts:

(i) x = *ship*
(ii) x = *tank*
(iii) x = *boat*

1. Given the texts $x = (x_1, x_2, x_3, ..., x_n)$ above are from an English plaintext source that is generated by 1-gram independent and identical trials, calculate $P\{x\}$ and compare the results.

2. Given the texts $x = (x_1, x_2, x_3, ..., x_{2n})$ above are from an English plaintext source that is generated by 2-gram independent and identical trials, calculate $P\{x\}$ and compare the results.

   What do you do when there is an odd number of letters?

3. Given the texts $x = (x_1, x_2, x_3, ..., x_n)$ above are from an English 1-gram plaintext source that is generated by a Markov chain, calculate $P\{x\}$ and compare the results.

1.5 Statistics for Other Languages

Data can be collected on the statistical properties of languages other than English. The 1-gram probabilities for French, German, Italian, Portuguese and Spanish shown in Table 1.5 are based on data published by Gaines (1939). Note that for this analysis all diacritical marks, such as, accents and circumflexes have been dropped. Figure 1.4 gives a comparison of these1-gram probabilities with those for English.

	French	German	Italian	Portuguese	Spanish
1-gram	Probability	Probability	Probability	Probability	Probability
a	0.0942	0.050	0.1174	0.135	0.1269
b	0.0102	0.025	0.0092	0.005	0.0141
c	0.0264	0.015	0.0450	0.035	0.0393
d	0.0338	0.050	0.0373	0.050	0.0558
e	0.1587	0.185	0.1179	0.130	0.1315
f	0.0095	0.015	0.0095	0.010	0.0046
g	0.0104	0.040	0.0164	0.010	0.0112
h	0.0077	0.040	0.0154	0.010	0.0124
i	0.0841	0.080	0.1128	0.060	0.0625
j	0.0089	0.000	0.0000	0.005	0.0056
k	0.0000	0.010	0.0000	0.000	0.0000
l	0.0534	0.030	0.0651	0.035	0.0594
m	0.0324	0.025	0.0251	0.045	0.0265
n	0.0715	0.115	0.0688	0.055	0.0695
o	0.0514	0.035	0.0983	0.115	0.0949
p	0.0286	0.005	0.0305	0.030	0.0243
q	0.0106	0.000	0.0061	0.015	0.0116
r	0.0646	0.070	0.0637	0.075	0.0625
s	0.0790	0.070	0.0498	0.075	0.0760
t	0.0726	0.050	0.0562	0.045	0.0391
u	0.0624	0.050	0.0301	0.040	0.0463
v	0.0215	0.010	0.0210	0.015	0.0107
w	0.0000	0.015	0.0000	0.000	0.0000
x	0.0030	0.000	0.0000	0.002	0.0013
y	0.0024	0.000	0.0000	0.000	0.0106
z	0.0032	0.015	0.0049	0.003	0.0035

Table 1.5 1-gram Probability Distributions for French, German, Italian, Portuguese and Spanish

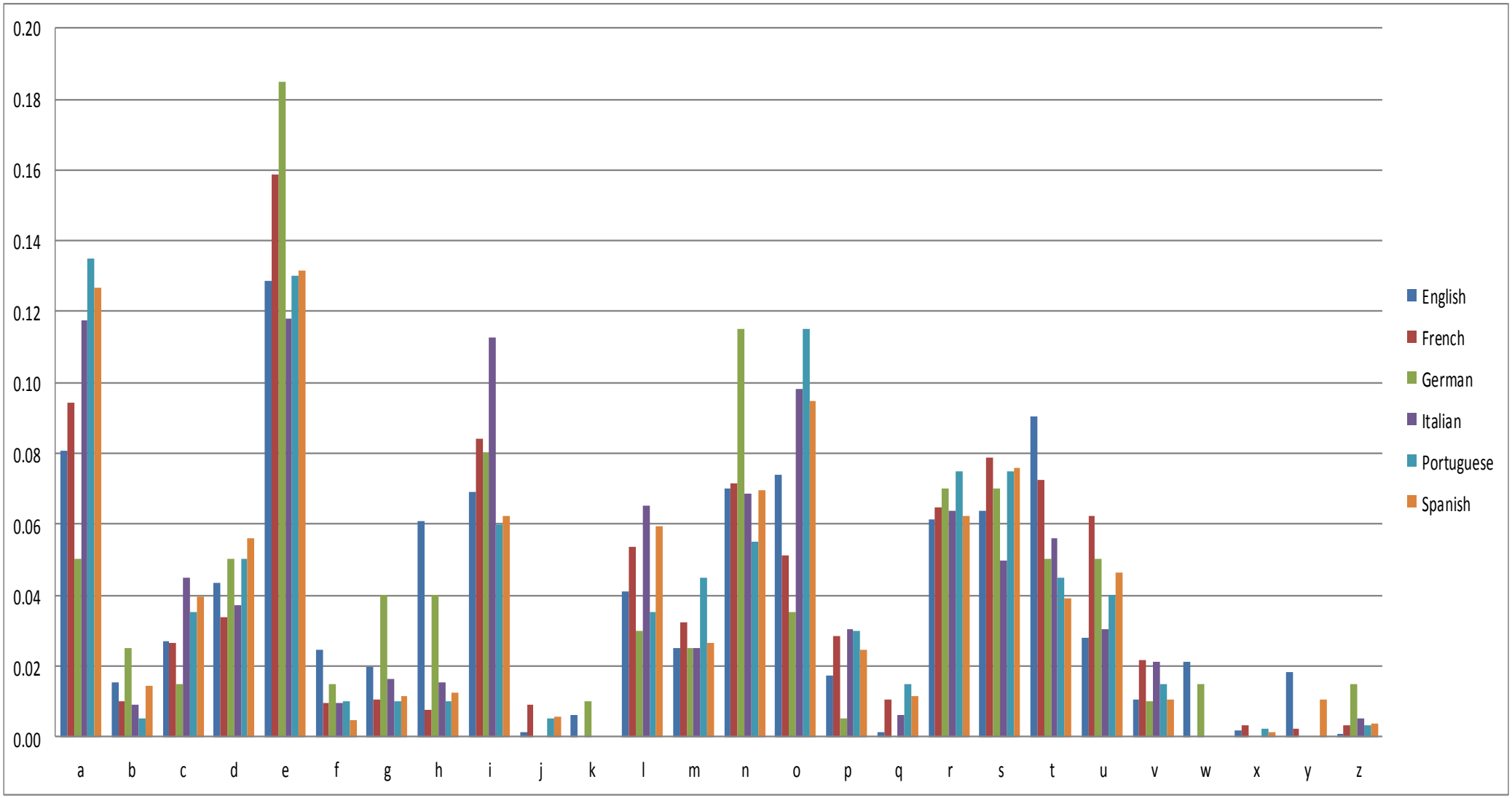

Figure 1.4 Comparison of 1-gram Probabilities of Six Languages

Exercise 1.2

1. Given the texts $x = (x_1, x_2, x_3, ..., x_n)$ below are from plaintext sources that are generated by 1-gram independent and identical trials calculate $P\{x\}$ for the texts. In each case, assume that the underlying language is English, French, German, Italian, Portuguese or Spanish. Hence identify the language that each text is most likely to come from.

   (i) x = *teller*
   (ii) x = *colle*
   (iii) x = *jabon*
   (iv) x = *martelo*
   (v) x = *kamin*
   (vi) x = *lisse*

2. The following email message x has been received, unfortunately, due to a computer glitch the letters have been completely jumbled up. Identify the language the message was written in. Hint – which languages can't it be?

   x = *que ce est haut wo wahrt sackt y pilo iy*

1.6 References

Gaines, H. F. (1939), "*Cryptanalysis*", American Photographic Publishing Co., USA.

Chapter 2 – Cipher Systems

2.1 Definition of a Cipher System

A cipher can be thought of as a transformation *T* that maps the plaintext *n*-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext *n*-gram $y = (y_1, y_2, y_3, ..., y_n)$ using an encipherment key e_k. For the cipher to be of any use, there must also exist an inverse transformation T^{-1} that uses a decipherment key d_k to transform the ciphertext $y = (y_1, y_2, y_3, ..., y_n)$ back to the plaintext *n*-gram $x = (x_1, x_2, x_3, ..., x_n)$.

$$y = T(x, e_k)$$

and

$$x = T^{-1}(y, d_k)$$

Assuming that the ciphertext alphabet is the same as the plaintext alphabet *A* (or at least the same size), then *X* can be defined as the set of all finite plaintext messages, that is, all possible *n*-grams drawn from *A*. Similarly, *Y* can be defined as the set of all finite ciphertext messages.

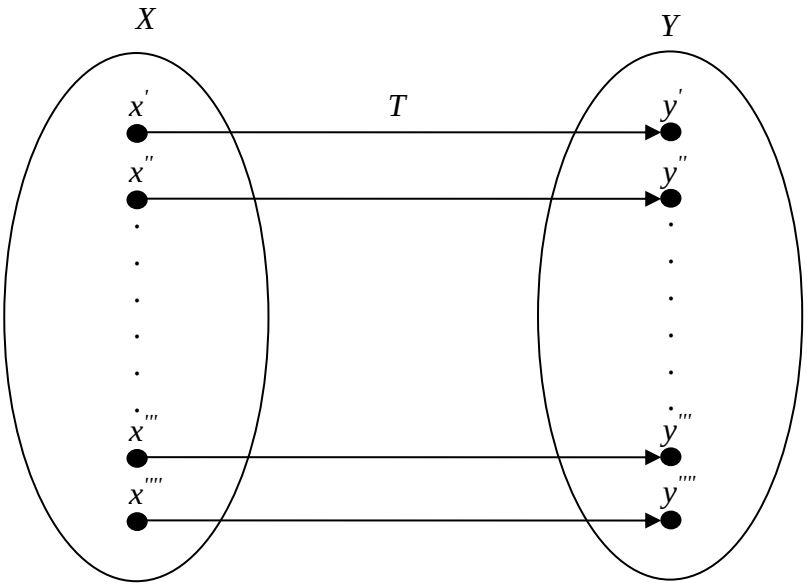

Figure 2.1 Cipher Transformation *T* from *X* to *Y*

A cipher *T* needs to have the property that every plaintext message is enciphered to a unique ciphertext.

If x' and x'' are distinct plaintext messages in X, then they must map to distinct ciphertext messages.

$$x' \neq x'' \quad \Rightarrow \quad T(x') \neq T(x'') \quad \forall x', x'' \in X$$

This means that T is an **injective** mapping (i.e., a **one-to-one** mapping).

A cipher T also needs to have the property that every ciphertext message can be deciphered, i.e., each ciphertext is an encipherment of at least one plaintext.

$$\forall y \in Y, \ \exists x \in X \ : \ y = T(x)$$

This means that T is a **surjective** mapping (i.e., an **onto** mapping).

A mapping that is both injective and surjective is **bijective**, and will have an inverse function, denoted T^{-1}.

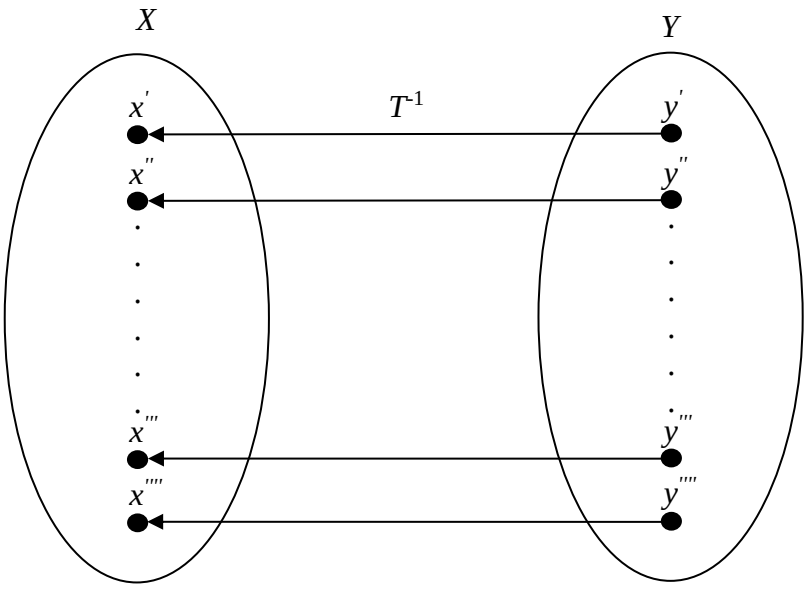

Figure 2.2 Cipher Transformation T^{-1} from Y to X

Note that there are some ciphers that do not meet the injective property above. These ciphers use the idea, originally from codes, of mapping plaintexts to one of several ciphertexts at the discretion of the encipherer, with the aim of confusing the cryptanalyst. For this to be possible the cipher alphabet needs to be larger than the plaintext alphabet. Consequently, this requirement has caused this type of cipher to drop out of use.

Figure 2.3 shows how a cipher system is applied by the *Sender* of a message to hide its meaning. The *Sender* has the plaintext and the encipherment key, which are used together with the transformation *T* to produce the ciphertext. Only the ciphertext is sent and is at risk of interception by an *Eavesdropper* or *Enemy*, all other information is kept secret by the *Sender*. On receipt of the ciphertext the *Receiver* deciphers the message using the decipherment key and the transformation T^{-1} to reproduce the plaintext. The *Eavesdropper* only has the ciphertext and has to try and break the cipher with only this information.

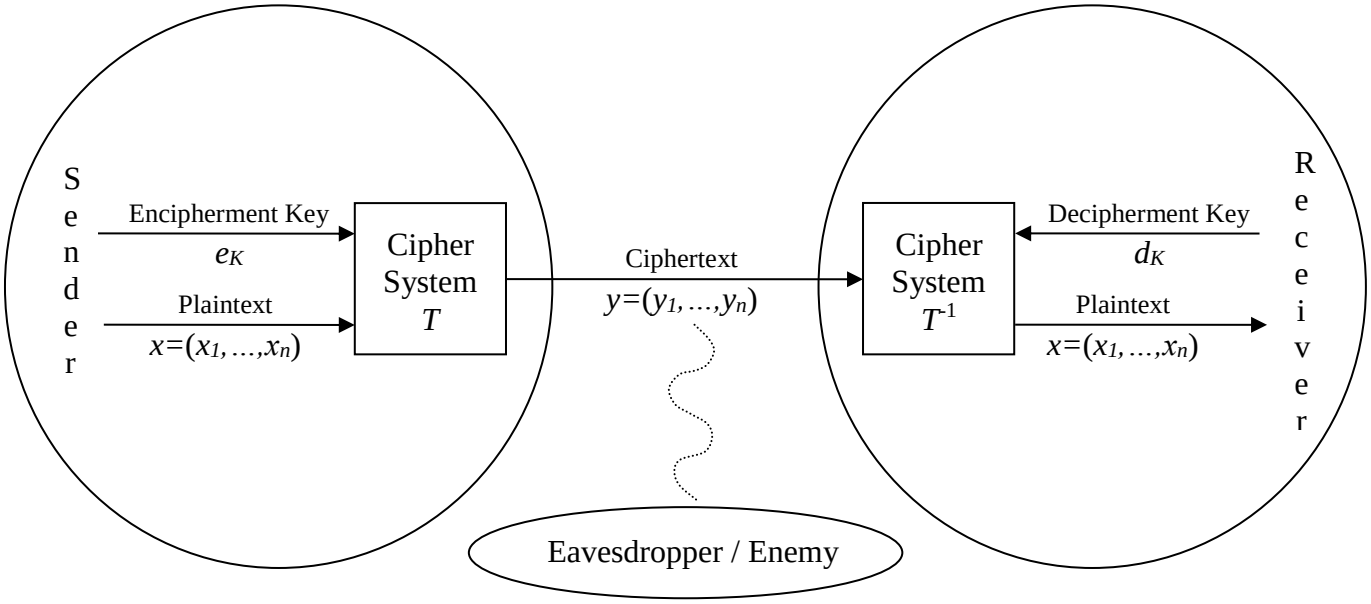

Figure 2.3 A Cipher System

Note that many modern cipher systems have had complete details published of how their transformations work, and the security of the system relies only on the secrecy of the encipherment and decipherment keys. Even when the system is a military one and the transformations are kept secret, the designers of the system assume that the transformations are known by the enemy, or at least will at some stage become known by them. As a result they have been designed to be secure as long as the keys are kept secret.

Given it is much harder to break a cipher if nothing is known about how it operates than if its transformations are known, this may seem to be a strange way of viewing a cipher system. However, it is a viewpoint that has been developed out of bitter experience. Throughout history there are numerous cases where espionage has been employed to steal a cipher or code that was too difficult to break.

John Walker a Chief Warrant Officer in the US Navy was convicted in 1985 of spying for the Soviet Union. Between 1968 and 1985 he gave the KGB details of the Navy's cryptographic systems (Earley, 1988), which allowed them to decrypt over one million messages. Heath (2001) gives an analysis of just one of the many naval

cryptographic systems that Walker compromised and the flaws in the system that allowed him to do this.

Exercise 2.1

1. Show that if T is bijective then T^{-1} exists.

2. Show that if T^{-1} exists then T is bijective.

2.2 References

Earley, P., (1988), "*Family of Spies: Inside the John Walker Spy Ring"*, Bantam Books, New York.

Heath, L. J., Major, (2001), "*An Analysis of the Systematic security Weaknesses of the U.S. Navy Fleet broadcasting System, 1967-1974, as exploited by CWO John Walker*", Masters Thesis, U.S. Army Command and General Staff College, Georgia Institute of Technology, USA.

Chapter 3 – Classical Ciphers

Although all ciphers will be applied to the alphabet *A*, it is useful to recognise that there is a one-to-one mapping between *A* and Z_{26}, (namely: $a - 0$, $b - 1$, …, $y - 24$, $z - 25$) and to define the ciphers in terms of Z_{26}, see Table 3.1. Note that in the more general case an alphabet with *m* symbols maps to Z_m. The following ciphers will be defined for the general case.

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

Table 3.1 The Mapping Between the Alphabet and Z_{26}

3.1 Monoalphabetic Ciphers

One important class of cipher is the monoalphabetic cipher. These ciphers map each distinct character in the plaintext message to a specific and unique character in the ciphertext. Each time the same character appears in the plaintext it will map to the same letter in the ciphertext. For example, the plaintext word *deed* could map to, say, *tfft*, which changes the letters but does not disguise the pattern present in the word.

3.1.1 The Shift Cipher / Caesar Cipher

One of the earliest known ciphers was the Caesar cipher, which is named after Julius Caesar. According to Suetonius the Roman historian, Caesar swapped every letter in his plaintext with a letter three places further back in the alphabet (Thomson, 1909), see Table 3.2. However, this has been misinterpreted by most authors as a shift in the opposite direction; as a result a Caesar cipher is almost universally taken to mean that shown in Table 3.3 (Gaines, 1939; Smith 1943, Kahn, 1967). Suetonius also states that Caesar's nephew, Augustus, who was the first Roman Emperor, replaced letters with letters one place further on in the alphabet.

Plaintext	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
Ciphertext	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

Table 3.2 The Cipher Caesar Actually Used

Plaintext	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
Ciphertext	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

Table 3.3 The Caesar Cipher

The weakness of the Caesar ciphers is that once it is know that it is being used an opponent can easily decipher it. However, the cipher can be generalised by using different shifts in the ciphertext alphabet and the shift can be regarded as the key. So even if an opponent knows that the cipher is being used they also need to know the key before the ciphertext can be deciphered.

Definition 3.1 The Shift Cipher
The Shift cipher is a monoalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = k$ $k \in Z_m$

and, $T: \quad y_i = x_i + k \bmod m \qquad 1 \le i \le n$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = k$ $k \in Z_m$

and, $T^{-1}: \quad x_i = y_i - k \bmod m \qquad 1 \le i \le n$

Because of the modular arithmetic used in the definition, k can only take on m distinct values, so there are m possible keys for this cipher: $0 \le k \le m-1$. Obviously, the key $k = 0$ offers no security.

Note that with a key of $k = 3$ the Shift cipher becomes the Caesar cipher, and with a key of $k = 1$ it becomes Augustus' cipher.

Example 3.1

Encipher the plaintext "*tonight*" using the Shift cipher with key $k = 8$.

The encryption equation is:

$$T: \quad y_i = x_i + 8 \bmod 26 \qquad 1 \le i \le n$$

The letter "*t*" maps to the number 19 in Z_{26}.

$$\begin{aligned} y_1 &= 19 + 8 \bmod 26 \\ &= 27 \bmod 26 \\ &= 1 \bmod 26 \end{aligned}$$

So, the first ciphertext character is "*B*". Similarly, encrypt the other characters.

Plaintext	*t*	*o*	*n*	*i*	*g*	*h*	*t*
Plaincode x_i	19	14	12	8	6	7	19
Ciphercode y_i	1	22	20	16	14	15	1
Ciphertext	*B*	*W*	*V*	*Q*	*O*	*P*	*B*

This gives ciphertext "*BWVQOPB*".

Example 3.2

Decipher the ciphertext "*EZOLJ*" using the Shift cipher with key $k = 11$.

The decryption equation is:

$$T^{-1}: \quad x_i = y_i - 11 \bmod 26 \qquad 1 \le i \le n$$

The letter "*E*" maps to the number 4 in Z_{26}.

$$\begin{aligned} x_1 &= 4 - 11 \bmod 26 \\ &= -7 \bmod 26 \\ &= 19 \bmod 26 \end{aligned}$$

So, the first plaintext character is "*t*". Similarly, decrypt the other characters.

Ciphertext	*E*	*Z*	*O*	*L*	*J*
Ciphercode y_i	4	25	14	11	9
Plaincode x_i	19	14	3	0	24
Plaintext	*t*	*o*	*d*	*a*	*y*

This gives ciphertext "*today*".

3.1.2 The Reverse-shift Cipher / Atbash Cipher

Another cipher that has ancient origins is the Atbash cipher, which has been used in the Bible. The name comes from the first two letters of the Hebrew alphabet (*aleph* and *beth*). In their cipher the Hebrews swapped the first letter of their alphabet with the last, and the second with the last but one, etc. see Table 3.4, (Kahn, 1967).

Plaintext	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
Ciphertext	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Table 3.4 The Atbash Cipher

The weakness of the Atbash ciphers is that once it is know that it is being used it can be easily deciphered by an opponent. However, the cipher can be generalised by using different shifts in the reversed ciphertext alphabet and the shift can be regarded as the key. So even if an opponent knows that the cipher is being used they also need to know the key before the ciphertext can be deciphered.

Definition 3.2 The Reverse-shift Cipher
The Reverse-shift cipher is a monoalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

$$\text{where,} \qquad e_k = k \qquad\qquad k \in Z_m$$

$$\text{and,} \quad T: \quad y_i = k - x_i \bmod m \qquad 1 \le i \le n$$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

$$\text{where,} \qquad d_k = k \qquad\qquad k \in Z_m$$

$$\text{and,} \quad T^{-1}: \quad x_i = k - y_i \bmod m \qquad 1 \le i \le n$$

Because of the modular arithmetic used in the definition, k can only take on m distinct values, so there are m possible keys for this cipher: $0 \le k \le m-1$. Obviously, the key $k = 0$ offers no security.

In the case of $k = 25$ (or in the general case $k = m$-1) the Reverse-shift cipher is reduces to the Atbash cipher.

Exercise 3.1

1. Use the Shift cipher with key $k = 7$ to encrypt the plaintext message "*meet me at the airport*".

2. The Shift cipher has been used with a key $k = 3$ to encrypt a message and has produced the ciphertext "*KLGH*". Decipher the ciphertext.

3. Use the Reverse-shift cipher with key $k = 6$ to encrypt the plaintext message "*send money*".

4. The Reverse-shift cipher has been used with a key $k = 2$ to encrypt a message and has produced the ciphertext "*JOPUWVJ*". Decipher the ciphertext.

5. For the Shift cipher, show that the encryption and decryption transformations given in Definition 3.1 are inverse functions of one another.

6. For the Reverse-shift cipher, show that the encryption and decryption transformations given in Definition 3.2 are inverse functions of one another.

3.1.3 The Multiplicative Cipher

Given it is possible to create a cipher by addition mod *m*, an obvious idea is to try to create one by using multiplication mod *m*. This is possible, however, there are more restrictions in this method.

Definition 3.3 The Multiplicative Cipher
The Multiplicative cipher is a monoalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext *n*-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext *n*-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

$$\text{where,} \qquad e_k = a \qquad a \in Z_m$$

$$\text{and,} \quad T: \quad y_i = ax_i \bmod m \qquad 1 \le i \le n$$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

$$\text{where,} \qquad d_k = a^{-1} \qquad a^{-1} \in Z_m$$

$$\text{and,} \quad T^{-1}: \quad x_i = a^{-1} y_i \bmod m \qquad 1 \le i \le n$$

where a^{-1} is the multiplicative inverse of a modulo m. It can be shown that a^{-1} exists iff a and m are coprime.

Because of the modular arithmetic, a can only take on m distinct values, so there are m possible keys for this cipher: $0 \le a \le m-1$. However, the constraint that a and m are coprime reduces this significantly. Obviously, the key $a = 1$ offers no security.

Exercise 3.2

1. Use the Multiplicative cipher with key $k = 7$ to encrypt the plaintext message "*at dawn*".

2. The Multiplicative cipher has been used with a key $k = 3$ to encrypt a message and has produced the ciphertext "*CINZYCM*". Decipher the ciphertext.

3. Given that a^{-1} exists, show that the encryption and decryption transformations for the Multiplicative cipher given in Definition 3.3 are inverse functions of one another.

4. What does it mean for the Multiplicative cipher if a^{-1} does not exist?

5. How many keys are there for the Multiplicative cipher when $m = 26$?

6. For the Multiplicative cipher, prove that a^{-1} exists iff a and m are coprime.

3.1.4 The Affine Cipher

The next level of cipher complexity is to combine the ideas of addition and multiplication mod n to create a cipher.

Definition 3.4 The Affine Cipher

The Affine cipher is a monoalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

$$\text{where,} \qquad e_k = (a, k) \qquad\qquad a, k \in Z_m$$

$$\text{and,} \quad T: \quad y_i = ax_i + k \bmod m \qquad 1 \le i \le n$$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

$$\text{where,} \qquad d_k = (a^{-1}, k) \qquad\qquad a^{-1}, k \in Z_m$$

$$\text{and,} \quad T^{-1}: \quad x_i = a^{-1}(y_i - k) \bmod m \qquad 1 \le i \le n$$

where a^{-1} is the multiplicative inverse of a modulo m. It can be shown that a^{-1} exists iff a and m are coprime.

Again, because of the modular arithmetic, a and k can only take on m distinct values, so there are at most m^2 possible keys for this cipher: $0 \le a, k \le m-1$. However, the constraint that a and m are coprime reduces this significantly.

Note that the Shift cipher and the Multiplicative ciphers can be regarded as special cases of the Affine cipher (with $a = 1$ for the Shift cipher, and $k = 0$ for the Multiplicative cipher).

3.1.5 The General Substitution Cipher

All the ciphers examined so far can be regarded as special cases of the General Substitution cipher, which maps letters from the plaintext to the ciphertext using a permutation π of the alphabet A. Figure 3.1 gives a classification of monoalphabetic ciphers that shows which ciphers are special cases of others.

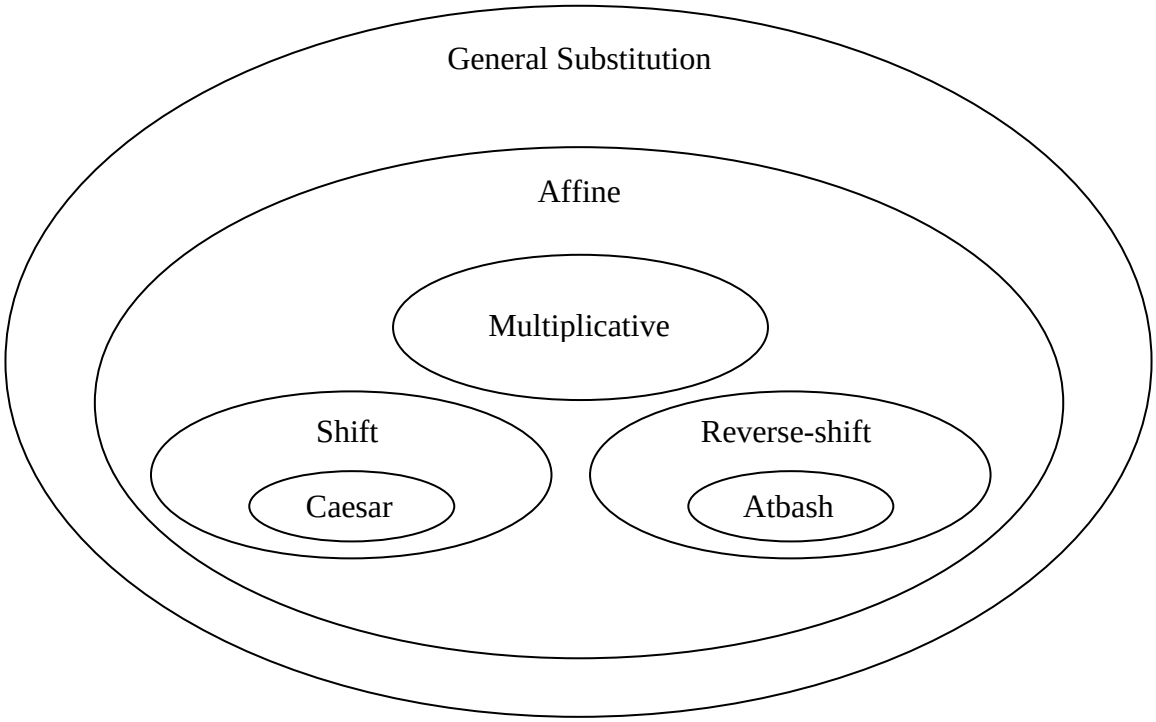

Figure 3.1 Classification of Monoalphabetic Ciphers

Definition 3.5 The General Substitution Cipher
The General Substitution cipher is a monoalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = \pi$ π a permutation of A

and, T: $y_i = \pi(x_i)$ $1 \le i \le n$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = \pi^{-1}$ π^{-1} a permutation of A_m

and, T^{-1}: $x_i = \pi^{-1}(y_i)$ $1 \le i \le n$

and where π^{-1} is the inverse permutation of π.

There are m! permutations of an alphabet of size m so there are m! keys for the General Substitution cipher.

Note that for maximum security a key should be constructed so that there is no obvious relationship between the mappings in the permutation π. This makes it harder for the cryptanalyst to break the cipher. However, this also makes it much harder to remember the key. One way around this is to use a key phrase and some rule to produce the key from it.

Example 3.3

Encipher the plaintext "*traitor*" using the General Substitution cipher with permutation π defined by the key phrase "catch".

First, set up the permutation π by writing out the alphabet from a to z. Under this write out the key phrase without repeated letters, followed by the rest of the alphabet in alphabetical order.

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

The encryption equation is:

$$T: \quad y_i = p(x_i) \qquad 1 \le i \le n$$

For the first character in the message, reading the permutation from the top row to the bottom:

$$T: \quad S = p(t) \qquad i = 1$$

So, the first plaintext letter "*t*" maps to the ciphertext letter "*S*". Similarly, encrypt the other characters.

Plaintext	*t*	*r*	*a*	*i*	*t*	*o*	*r*
Ciphertext	*S*	*Q*	*C*	*G*	*S*	*N*	*Q*

This gives the ciphertext "*SQCGSNQ*".

Example 3.4

Decipher the ciphertext "*TNMSCTS*" using the General Substitution cipher with permutation of π defined by the key phrase "catch".

First, set up the permutation π by writing out the alphabet from *a* to z. Under this write out the key phrase without repeated letters, followed by the rest of the alphabet in alphabetical order.

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

The decryption equation is:

$$T^{-1}: \quad x_i = p^{-1}(y_i) \qquad 1 \le i \le n$$

For the first character in the message, reading the permutation from the bottom row to the top:

$$T^{-1}: \quad c = p^{-1}(T) \qquad i = 1$$

So, the first ciphertext letter "*T*" maps to the plaintext letter "*c*". Similarly, decrypt the other characters.

Ciphertext	*T*	*N*	*M*	*S*	*C*	*T*	*S*
Plaintext	*c*	*o*	*n*	*t*	*a*	*c*	*t*

This gives the plaintext "*contact*".

Exercise 3.3

1. Encipher a plaintext message x by the Multiplicative cipher followed by the Shift cipher. What is the result?

2. Encipher a plaintext message x by the Shift cipher followed by the Multiplicative cipher. What is the result?

3. Given that a^{-1} exists, show that the encryption and decryption transformations for the Affine cipher given in Definition 3.4 are inverse functions of one another.

4. How many keys are there for the Affine cipher when $m = 26$?

5. How many keys are there for the Affine cipher when $m = 256$; the size of the ASCII character set?

6. Use the Affine cipher with key (a = 5, k = 3) to encrypt the plaintext message "*meet me at the airport*".

7. The Affine cipher has been used with a key (a = 11, k = 7) to encrypt a message and has produced the ciphertext "*MZXZUO*". Decipher the ciphertext.

8. Use the keyword phrase "*tomorrow is after today*" to create a General Substitution cipher, and then use it to encrypt "*send the money*".

9. The keyword phrase "*follow the path*" has been used with a General Substitution cipher to encrypt a message giving the ciphertext "*WTDFYQPAKGTIR*". What is the plaintext?

3.2 Polyalphabetic Ciphers

Polyalphabetic ciphers use more than one cipher alphabet. This means that the first and second occurrences of a letter in the plaintext are likely to map to different letters in the ciphertext. Note that, in practice, the multiple cipher alphabets are simply permutations of *A*.

Definition 3.6 The Vigenère Cipher
The Vigenère cipher is a polyalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext *n*-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext *n*-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

$$\text{where,} \qquad e_k = k = (k_1, ..., k_i, ..., k_r) \qquad k_i \in Z_m, \;\; 1 \le i \le r, \;\; 2 \le r$$

$$\text{and,} \quad T: \quad y_i = x_i + k_{i \bmod r} \mod m \qquad 1 \le i \le n$$

Note that the keyword $k = (k_1, k_2, k_3, ..., k_r)$ is an *r*-gram.

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

$$\text{where,} \qquad d_k = k = (k_1, ..., k_i, ..., k_r) \qquad k_i \in Z_m, \;\; 1 \le i \le r, \;\; 2 \le r$$

$$\text{and,} \quad T^{-1}: \quad x_i = y_i - k_{i \bmod r} \mod m \qquad 1 \le i \le n$$

For a keyword with a length of *r* characters there are m^r keys for the Vigenère cipher.

The Vigenère cipher was not invented by Blaise de Vigenère a French diplomat whose name has since become attached to it, but by the Italian cryptographer Giovan Battista Bellaso who published it in 1553 (Kahn, 1967). Bellaso's arrangement of the alphabets was not as they are today, which is instead based on the German abbot Johannes Trithemius' tabula recta (Table 3.5) first published in 1518, but written in 1508.

This cipher became known as *Le Chiffre Indéchiffrable*, despite this it was not widely used for over 200 years as it was thought to be too difficult and slow to use, and code books were thought to be adequate at the time.

When used with a very long key the Vigenère cipher is often called a Running Key cipher. A very long key makes the cipher more secure, however, it cannot be memorised, so it is commonly taken from a text such as a book. The sender and

receiver only have to agree on which book to use and where to start taking the key from (e.g., page number and line number).

			Plaintext																										
			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**
Alphabet	**0**	**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
	1	**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
	2	**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
	3	**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
	4	**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
	5	**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
	6	**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
	7	**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
	8	**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
	9	**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
	10	**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
	11	**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
	12	**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
	13	**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
	14	**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
	15	**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
	16	**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
	17	**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
	18	**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
	19	**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
	20	**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
	21	**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
	22	**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
	23	**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
	24	**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
	25	**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 3.5 Tabula Recta for the Vigenère Cipher

Definition 3.7 The General Vigenère Cipher
The General Vigenère cipher is a polyalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = (k, \pi)$ π a permutation of A

$k = (k_1, ..., k_i, ..., k_r)$ $k_i \in Z_m, \ 1 \le i \le r, \quad 2 \le r$

and, T: $y_i = \pi(x_i + k_{i \bmod r}) \bmod m \qquad 1 \le i \le n$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = (k, \pi^{-1})$ π^{-1} a permutation of A

$k = (k_1, ..., k_i, ..., k_r)$ $k_i \in Z_m, \ 1 \le i \le r, \quad 2 \le r$

and, T^{-1}: $x_i = \pi^{-1}(y_i) - k_{i \bmod r} \bmod m \qquad 1 \le i \le n$

Note that r is the key length.

For a keyword with a length of r characters there are $m^r \times m!$ keys for the General Vigenère cipher.

This was also invented by Bellaso and was published in 1555 (Kahn, 1967), although not quite in the form we use it today. Table 3.6 gives a tabula recta based on the permutation:

$$\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ \ j\ k\ l\ m\ n\ o\ p\ q\ r\ \ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ \ s\ l\ e\ \ p\ g\ y\ w\ n\ o\ m\ a\ r\ \ k\ d\ b\ \ f\ c\ j\ q\ u\ \ v\ x\ z \end{pmatrix}.$$

			Plaintext																								
			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
Alphabet	0	a	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z
	1	b	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T
	2	c	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H
	3	d	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I
	4	e	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S
	5	f	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L
	6	g	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E
	7	h	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P
	8	i	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G
	9	j	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y
	10	k	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W
	11	l	O	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N
	12	m	M	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O
	13	n	A	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M
	14	o	R	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A
	15	p	K	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R
	16	q	D	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K
	17	r	B	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D
	18	s	F	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B
	19	t	C	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F
	20	u	J	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C
	21	v	Q	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J
	22	w	U	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q
	23	x	V	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U
	24	y	X	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V
	25	z	Z	T	H	I	S	L	E	P	G	Y	W	N	O	M	A	R	K	D	B	F	C	J	Q	U	V	X

Table 3.6 Tabula Recta for the Permutation *p* = thislepgywnomarkdbfcjquvxz

Exercise 3.4

1. For the General Vigenère cipher, show that the encryption and decryption transformations given in Definition 3.7 are inverse functions of one another.

2. Encipher "*send more ammunition*" using the General Vigenère cipher with the permutation *p* = *thislepgywnomarkdbfcjquvxz* and the keyword "*daylight*".

3. Decipher "*YDXGJHCJVODXUGGZ*" using the General Vigenère cipher with the permutation *p* = *thislepgywnomarkdbfcjquvxz* and the keyword "*nighttime*".

Definition 3.8 The Autokey Cipher (plaintext version)
The Autokey cipher (plaintext version) is a polyalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = k = (k_1, ..., k_i, ..., k_r)$ $k_i \in Z_m$, $1 \le i \le r$, $1 \le r$

and, T: $y_i = x_i + k_i \mod m$ $1 \le i \le r$

$y_i = x_i + x_{i-r} \mod m$ $r < i \le n$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = k = (k_1, ..., k_i, ..., k_r)$ $k_i \in Z_m$, $1 \le i \le r$, $1 \le r$

and, T^{-1}: $x_i = y_i - k_i \mod m$ $1 \le i \le r$

$x_i = y_i - x_{i-r} \mod m$ $r < i \le n$

For a keyword with a length of r characters there are m^r keys for the Autokey cipher (plaintext version).

This method of encryption uses the keyword to encrypt the first r letters of the plaintext, and then uses the first r letters of the plaintext as the key for the next r letters of the plaintext and so on.

The precursor to this method was invented by Gerelmo Cardano an Italian mathematician, who was the first person to publish solutions to the cubic and quartic equations (Boyer and Merzbach,1989). Cardano did not use an independent keyword to start the encipherment process, instead he used the first word of the plaintext as a keyword and repeated this for each word of plaintext. Bellaso improved on this, but it fell to Vigenère to devise the system used in Definition 3.8, however, he only used a key of length one (Kahn, 1967).

Definition 3.9 The Autokey Cipher (ciphertext version)
The Autokey cipher (ciphertext version) is a polyalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = k = (k_1, ..., k_i, ..., k_r) \quad k_i \in Z_m, \ 1 \le i \le r, \quad 1 \le r$

and, T: $y_i = x_i + k_i \mod m \qquad 1 \le i \le r$

$y_i = x_i + y_{i-r} \mod m \qquad r < i \le n$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = k = (k_1, ..., k_i, ..., k_r) \quad k_i \in Z_m, \ 1 \le i \le r, \quad 1 \le r$

and, T^{-1}: $x_i = y_i - k_i \mod m \qquad 1 \le i \le r$

$x_i = y_i - y_{i-r} \mod m \qquad r < i \le n$

For a keyword with a length of r characters there are m^r keys for the Autokey cipher (ciphertext version).

This method of encryption uses the keyword to encrypt the first r letters of the plaintext, and then uses the first r letters of the ciphertext as the key for the next r letters of the plaintext and so on.

This cipher was also devised by Vigenère, however, again he only used a keyword of length one (Kahn, 1967).

Exercise 3.5

1. Encipher "*send more ammunition*" using the Autokey cipher (plaintext version) and the keyword "*help*".

2. Decipher "*GLGEFPBHWSZMRRQSZPFWKITAS*" using the Autokey cipher (plaintext version) and the keyword "*friend*".

3. Encipher "*send more ammunition*" using the Autokey cipher (ciphertext version) and the keyword "*hawk*".

4. Decipher "*XUIBMICKFTIXODHCODKVPSEEBVWH*" using the Autokey cipher (ciphertext version) and the keyword "*enemy*".

5. Give the equations that define an Autokey cipher (plaintext version) that uses a permutation p of the alphabet A.

6. Encipher "*send more ammunition*" using the cipher defined in the previous question, using the permutation p = *thislepgywnomarkdbfcjquvxz* and the keyword "*turtle*".

7. Give the equations that define an Autokey cipher (ciphertext version) that uses a permutation p of the alphabet A.

8. Encipher "*send more ammunition*" using the cipher defined in the previous question, using the permutation p = *thislepgywnomarkdbfcjquvxz* and the keyword "*tortoise*".

9. Put the permutation p = *thislepgywnomarkdbfcjquvxz* into cyclic form.

3.3 Transposition Ciphers

Transposition ciphers are the class of ciphers that interchange the position of letters in the plaintext to produce a ciphertext.

Definition 3.10 The Columnar Transposition Cipher

The Columnar Transposition cipher is a permutation cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = (\pi, k)$

$k = (k_1, ..., k_i, ..., k_r)$ $\quad k_i \in Z_m, \ 1 \le i \le r, \quad 2 \le r$

π is a permutation of k that puts it into alphabetical order.

and, T: $y_i = x_{\pi(((i-(\lceil \frac{i}{r} \rceil - 1))r-1)q+\lceil \frac{i}{r} \rceil)} \quad 1 \le i \le n$

where

$$q = \left\lceil \frac{n}{r} \right\rceil$$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = (\pi, k)$

$k = (k_1, ..., k_i, ..., k_r)$ $\quad k_i \in Z_m, \ 1 \le i \le r, \quad 2 \le r$

π is a permutation of k that puts it into alphabetical order.

and, T^{-1}: $x_i = y_{\pi(((i-(\lceil \frac{i}{q} \rceil - 1))q-1)r+\lceil \frac{i}{r} \rceil)} \quad 1 \le i \le n$

For a keyword with a length of r characters there are $r!$ keys for the Columnar Transposition cipher.

In practice, the plaintext is put into a table (r columns and q rows) row by row and read from the table columns by columns in the order indicated by the keyword. Note that if $q \times r \neq n$ the plaintext does not fill the last row of the table. In this case random text is added at the end until the table is full.

Definition 3.11 The General Permutation Cipher
The General Permutation cipher uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = \pi$

π is a permutation of the integers $(1,2,...,r)$ $2 \le r$

and, T: $y_{i \bmod r} = x_{\pi(i \bmod r)}$ $1 \le i \le n$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = \pi^{-1}$

π^{-1} is a permutation of the integers $(1,2,...,r)$ $2 \le r$

and, T^{-1}: $x_{i \bmod r} = y_{\pi^{-1}(i \bmod r)}$ $1 \le i \le n$

and where π^{-1} is the inverse permutation of π.

For a keyword with a length of r there are $r!$ keys for the General Permutation cipher. Note that if n is not a multiple of r then random text will need to be added at the end of the plaintext to achieve this.

Note that all transposition ciphers can be regarded as special cases of the General Permutation cipher (with $r = n$).

Exercise 3.6

1. Encipher "*send more ammunition*" using the Columnar Transposition cipher with the permutation p = 5 2 4 1 3. Fill any blank space with random text.

2. For the permutation $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 2 & 6 & 1 & 5 \end{pmatrix}$ calculate the inverse permutation π^{-1}. Then, put both into one-line notation, and then both into cyclic notation.

3. For the permutation $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 9 & 5 & 8 & 1 & 6 & 3 & 2 & 4 \end{pmatrix}$ calculate the inverse permutation π^{-1}. Then, put both into one-line notation, and then both into cyclic notation.

4. Encipher "*send more ammunition*" using the General Permutation cipher with the permutation p = 5 2 4 1 3. Fill any blank space with random text.

5. Decipher "*PHFEO LUNYL AOLLW RIVES LEDEA*" , which has been enciphered using the General Permutation cipher with the permutation p = 2 5 1 4 3.

6. Use the keyword phrase "*ship wrecked on the reef*" to produce a keyword and hence a permutation. Use this to encrypt the plaintext "*send more ammunition*" using the General Permutation cipher.

3.4 Polygraphic Ciphers

Polygraphic ciphers are ciphers that map two characters (or more) in the plaintext to two characters (or more) in the ciphertext.

Definition 3.12 The Playfair Cipher

The Playfair cipher is a polygraphic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = k = (k_1, ..., k_i, ..., k_r)$ $k_i \in Z_m$ $r = 25$

and, T: $y_i y_{i+1} = f(x_i x_{i+1})$ $i = 2j - 1$, $1 \le j \le \frac{n}{2}$

The keyword k is a permutation of the 25 letter alphabet (normally, there is no letter *j*, which is replaced by *i*, alternatively, the letter *q* is dropped) and this is used to form a grid of 25 letters:

k_1	k_2	k_3	k_4	k_5
k_6	k_7	k_8	k_9	k_{10}
k_{11}	k_{12}	k_{13}	k_{14}	k_{15}
k_{16}	k_{17}	k_{18}	k_{19}	k_{20}
k_{21}	k_{22}	k_{23}	k_{24}	k_{25}

Table 3.7 Playfair Grid

The function f is defined by as follows:

If x_i and x_{i+1} lie in the same column of the Playfair grid

$$y_i = LetterBelow(x_i)$$
$$y_{i+1} = LetterBelow(x_{i+1})$$

If x_i and x_{i+1} lie in the same row of the Playfair grid

$$y_i = LetterRightOf(x_i)$$
$$y_{i+1} = LetterRightOf(x_{i+1})$$

If x_i and x_{i+1} are not in the same column or row of the Playfair grid

$$y_i = LettersIntheRowOf(x_i) \cap LettersIntheColumnOf(x_{i+1})$$
$$y_{i+1} = LettersIntheRowOf(x_{i+1}) \cap LettersIntheColumnOf(x_i)$$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $\quad d_k = k = (k_1,...,k_i,...,k_r) \qquad k_i \in Z_m \quad r = 25$

and, $\quad T^{-1}: \quad x_i x_{i+1} = f^{-1}(y_i y_{i+1}) \qquad i = 2j-1 \ , \ 1 \le j \le \frac{n}{2}$

The function f^{-1} is defined as following:

If y_i and y_{i+1} lie in the same column of the Playfair grid

$$x_i = LetterAbove(y_i)$$
$$x_{i+1} = LetterAbove(y_{i+1})$$

If y_i and y_{i+1} lie in the same row of the Playfair grid

$$x_i = LetterLeftOf(y_i)$$
$$x_{i+1} = LetterLeftOf(y_{i+1})$$

If y_i and y_{i+1} are not in the same column or row of the Playfair grid

$$x_i = LettersIntheRowOf(y_i) \cap LettersIntheColumnOf(y_{i+1})$$
$$x_{i+1} = LettersIntheRowOf(y_{i+1}) \cap LettersIntheColumnOf(y_i)$$

Note that there is an additional rule that any pair of plaintext letters above $x_i x_{i+1}$ that are identical must be separated by a letter ***x*** before encipherment (in some variations, the letter *q* is used instead of *x*). An extra letter ***x*** may also need to be added at the end of the plaintext to ensure there are an even number of letters.

The Playfair cipher was invented by the English physicist and inventor Charles Wheatstone 1854. He also invented the Wheatstone bridge and the Telegraph (before Morse). The cipher is called the Playfair cipher after Lyon Playfair who was a scientist, Baron, Postmaster General and Speaker of the House of Commons, although not all at the same time! Playfair was responsible for promoting the cipher to the British government of the day and as a result his name became attached to it (Kahn, 1967).

The advantage of the Playfair cipher is that it operates on 2-grams (hence it is polygraphic) and as a consequence it hides much of the information that a frequency analysis of the cipher text letters would otherwise provide. However, it does little to conceal the 2-gram frequencies of the plaintext. This is because it can be regarded a simple substitution cipher, but one that operates on the alphabet, $A \times A$, that is, the alphabet of pairs of letters.

Exercise 3.7

1. Encipher "*send more ammunition*" using the Playfair cipher with the keyword "*wheatstone*".

3.5 References

Boyer, C. B., and Merzbach, U. C., (1989), "*A History of Mathematics*", 2nd edition, John Wiley and Sons, London.

Kahn, D., (1967), "*The Codebreakers: The Story of Secret Writing*", Macmillan, New York.

Smith, D., (1943), "*Cryptography: The Science of Secret Writing*", W. W. Norton & Co., USA.

Thomson, A., Revised by Forrester, T., (1909), "*The Lives of the Twelve Caesars*", Translation of "*De Vita XII Caesarum*" by *Suetonius Tranquillus*, George Bell & Sons, London.

Chapter 4 – Early Modern Ciphers

At the end of the 19th Century people began to think of using simple mechanical devices to encipher messages and at the start of the 20th Century people started to use advanced mathematical concepts in cryptography.

4.1 The Hill Cipher

The Hill cipher was invented by the American mathematician Lester Hill in 1928 (Hill, 1929). This was the first serious application of algebra to cryptography. In his paper, Hill did not present his cipher in the form below, but instead in the equivalent form of a set of simultaneous equations.

Definition 4.1 The Hill Cipher
The Hill cipher is a polygraphic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = (L, \pi)$ π a permutation of A

$$L = \begin{pmatrix} a_{11} & \dots & a_{1r} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ a_{r1} & \dots & a_{rr} \end{pmatrix}$$

and, T:

$$(y_i, y_{i+1}, ..., y_{i+r-1}) = (\pi(x_i), \pi(x_{i+1}), ..., \pi(x_{i+r-1})) \begin{pmatrix} a_{11} & \dots & a_{1r} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ a_{r1} & \dots & a_{rr} \end{pmatrix} \bmod m$$

$$, \quad i = r(j-1)+1, \; 1 \le j \le \frac{n}{r}$$

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = (L^{-1}, \pi^{-1})$ π^{-1} a permutation of A

$$L^{-1} = \begin{pmatrix} b_{11} & \dots & b_{1r} \\ . & & . \\ . & & . \\ b_{r1} & \dots & b_{rr} \end{pmatrix}$$

and, T^{-1}:

$$(x_i, x_{i+1}, \dots, x_{i+r-1}) = \pi^{-1}\left((y_i, y_{i+1}, \dots, y_{i+r-1}) \begin{pmatrix} b_{11} & \dots & b_{1r} \\ . & & . \\ . & & . \\ b_{r1} & \dots & b_{rr} \end{pmatrix} \bmod m \right), \quad i = r(j-1)+1,\ 1 \le j \le \frac{n}{r}$$

Obviously, decipherment can only take place if the inverse matrix exists.

For an $r \times r$ matrix there are m^{r^2} potential keys for the Hill cipher. However, this is reduced by the constraint that the inverse matrix must exist, which will occur iff the determinant of *L* is non zero and coprime to *m*.

$$|L| \neq 0 \quad \& \quad \gcd(|L|, m) = 1$$

There is a potential difficulty with this cipher, in that, if the matrix selected for the cipher does not have an inverse, the ciphertext cannot be deciphered. One way to avoid this would be to generate a matrix and test to see if its inverse exist, and reject it if it does not. However, this could result in many matrices being examined until a suitable one is found.

To avoid this, Hill developed a theory that allowed him to construct matrices that had the required property modulo 26, which was the size of the alphabet that he used. This considerably reduces the size of the key space, however, due to the enormous number of potential keys (m^{r^2}) when *r* is even 5 or 6 there will still be a very large number left.

Using a small matrix (2×2) the techniques is more time consuming and less secure than say the Playfair cipher, but as the matrix size increases the method becomes more secure. To overcome the difficulties of handling sets of simultaneous equations or matrix multiplication, Hill, working with Louis Weisner, patented a machine that performed the encryption process. The cipher machine was called the Message Protector (see Figure 4.1) and it used gears and a bicycle chain to encipher up to 6-

grams at a time (Weisner and Hill, 1932). This was one of the earliest instances of a mechanical implementation of a complex cipher and was a sign of things to come.

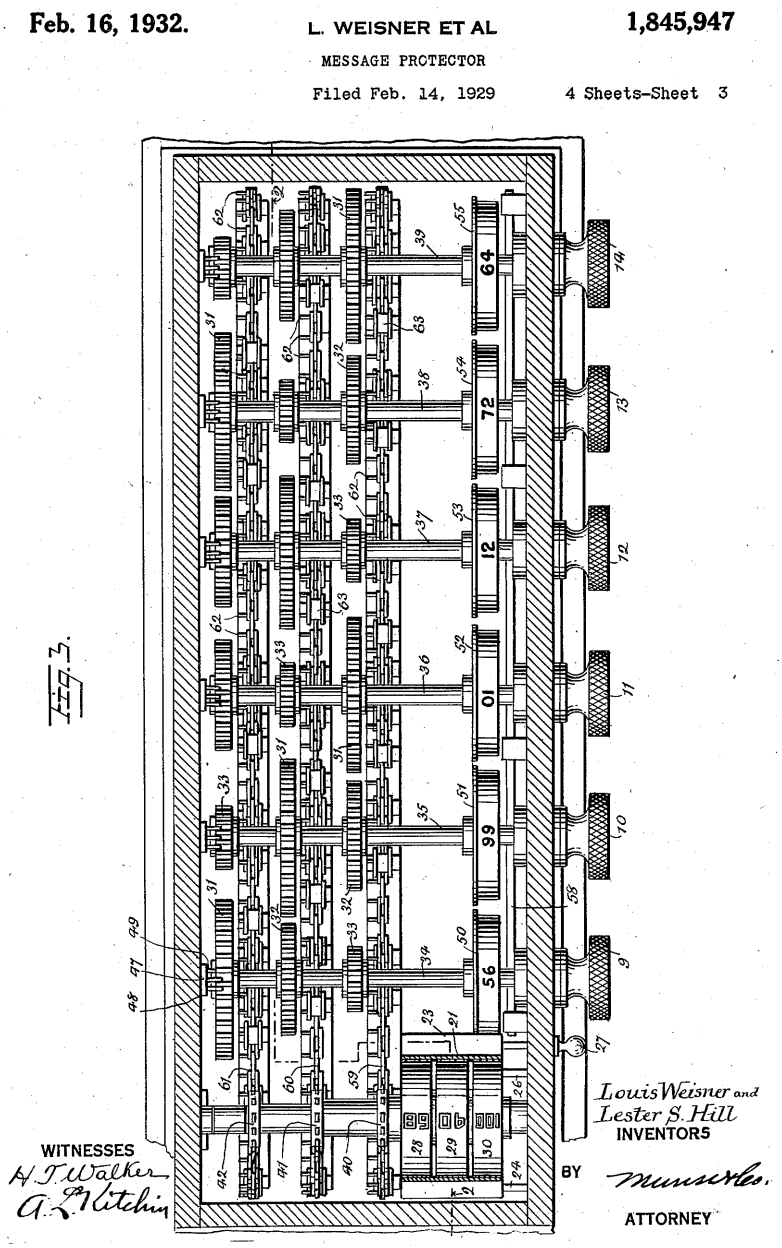

Figure 4.1 The Weisner and Hill Encipherment Machine, the "Message Protector" (from Weisner and Hill, 1932)

In 1931, Hill published a generalisation of the technique that replaced the 1-grams ($x_i, y_i,$ and a_i) with matrices of plaintext, ciphertext and keyword coefficients (Hill, 1929).

The Message Protector was not a commercial success and the Hill cipher was not extensively used. This was probably due to two factors, first, the complexity of using it before the advent of computers, and second, the fact that it is a linear transformation permits certain types of attacks (given enough ciphertext the equations can be solved to find the plaintext).

Despite this, the Hill cipher has had an enormous impact on cryptology. It led the way for the use of algebra in the design of modern ciphers. In addition, many modern ciphers use Hill's idea of matrix transformations as one stage of a more complex process. This is because of its properties of diffusion – a small change in the plaintext results in a widespread change in the ciphertext.

Exercise 4.1

1. Encipher "*send more ammunition*" using the Hill cipher with the key $e_k=(L,\pi)$ where $\pi = I$ the identity permutation and

$$L = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}.$$

2. Encipher "*pay more money*" using the Hill cipher with the key $e_k=(L,\pi)$ where $\pi = I$ the identity permutation and

$$L = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}.$$

3. Decipher the ciphertext from question 2 using the Hill cipher with the key $D_k = (L^{-1},\pi)$ where $\pi = I$ the identity permutation and

$$L^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}.$$

4.1.1 Involutory Matrices

An involutory matrix L is a matrix that is its own inverse, that is, $L = L^{-1}$,

$$L = L^{-1}$$
$$LL = LL^{-1}$$
$$L^2 = I$$

Consider the 2×2 matrix

$$L = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

The involutory condition $L^2 = I$ gives:

$$\begin{pmatrix} a^2 + bc & ab + bd \\ ac + cd & bc + d^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$\therefore$

$ab + bd = 0$

&

$ac + cd = 0$

so

$b(a + d) = 0$

&

$c(a + d) = 0$

Which have solutions:

$$b = 0, \quad d = -a \quad \& \quad c = 0, \quad d = -a$$

For $b = 0 \quad \& \quad c = 0$

$$L^2 = \begin{pmatrix} a^2 & 0 \\ 0 & d^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$\therefore$

$a = \pm 1 \quad \& \quad d = \pm 1$

so

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

For $b=0$ & $d=-a$

$$L^2=\begin{pmatrix} a^2 & 0 \\ 0 & a^2 \end{pmatrix}=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$\therefore$

$$a=\pm 1$$

so

$$L=\begin{pmatrix} 1 & 0 \\ c & -1 \end{pmatrix},\begin{pmatrix} -1 & 0 \\ c & 1 \end{pmatrix}$$

For $c=0$ & $d=-a$

$$L^2=\begin{pmatrix} a^2 & 0 \\ 0 & a^2 \end{pmatrix}=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$\therefore$

$$a=\pm 1$$

so

$$L=\begin{pmatrix} 1 & b \\ 0 & -1 \end{pmatrix},\begin{pmatrix} -1 & b \\ 0 & 1 \end{pmatrix}$$

For $d=-a$ & $d=-a$

$$L^2=\begin{pmatrix} a^2+bc & 0 \\ 0 & a^2+bc \end{pmatrix}=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$\therefore$

$$a^2+bc=1$$

so either

$$c=\frac{1-a^2}{b}$$

&

$$L=\begin{pmatrix} a & b \\ \frac{1-a^2}{b} & -a \end{pmatrix} \qquad b\neq 0$$

or

$$b=\frac{1-a^2}{c}$$

&

$$L=\begin{pmatrix} a & \frac{1-a^2}{c} \\ c & -a \end{pmatrix} \qquad c\neq 0$$

So, the complete set of involutory 2×2 matrices is:

$$L=\begin{pmatrix}\pm 1 & 0\\ 0 & \pm 1\end{pmatrix},\begin{pmatrix}1 & 0\\ c & -1\end{pmatrix},\begin{pmatrix}-1 & 0\\ c & 1\end{pmatrix},\begin{pmatrix}1 & b\\ 0 & -1\end{pmatrix},\begin{pmatrix}-1 & b\\ 0 & 1\end{pmatrix},\begin{pmatrix}a & b\\ \frac{1-a^2}{b} & -a\end{pmatrix},\begin{pmatrix}a & \frac{1-a^2}{c}\\ c & -a\end{pmatrix}$$

which reduces to

$$L=\begin{pmatrix}\pm 1 & 0\\ 0 & \pm 1\end{pmatrix},\begin{pmatrix}a & b\\ \frac{1-a^2}{b} & -a\end{pmatrix},\begin{pmatrix}a & \frac{1-a^2}{c}\\ c & -a\end{pmatrix} \qquad b,c\neq 0$$

as the others are special cases of the last two matrices.

4.1.2 Involutory Matrices for a Hill Cipher

For a Hill cipher $a,b,c,d\in Z_m$, which reduces the number of involutory matrices substantially. It also introduces extra constraints on the division by *b* and *c* in the last two matrices above, this is only possible if they have a reciprocal, which will only occur if they are coprime to *m*. For the case *m* = 26, the following integers have reciprocals mod *m*.

Number	1	3	5	7	9	11	15	17	19	21	23	25
Reciprocal	1	9	21	15	3	19	7	23	11	5	17	25

Table 4.1 Reciprocals Modulus 26

Note the fact that an involutory matrix has an inverse that is the same as itself was extremely useful given the limited ability of mechanical devices of the time.

4.1.3 Hill's Method

Other matrices other than involutory ones exist that can be easily constructed. Hill (1929) generated an $n\times n$ matrix L and its inverse L^{-1} in the following way.
The matrix

$$L=\begin{pmatrix}1 & 0 & & & 0\\ 0 & 1 & & & \\ & 0 & . & & \\ & & . & 1 & 0\\ 0 & & & 0 & \alpha\end{pmatrix}$$

has a determinant $|L|=\alpha$.

Hence, if α is chosen to be coprime to *m*, α^{-1} will exist and therefore L^{-1} will exist and be equal to:

$$L^{-1} = \begin{pmatrix} 1 & 0 & & & 0 \\ 0 & 1 & & & \\ & 0 & . & & \\ & & . & 1 & 0 \\ 0 & & & 0 & \alpha^{-1} \end{pmatrix}$$

Other matrices that have an inverse modulus *m* can now be created by applying row manipulation operations to the matrix L.
Operations that can be applied to a matrix *L* that do not change its determinant $|L| = \alpha$ are:

1. Interchange columns and rows
2. Add a multiply $(\in Z_m)$ of any row to any other, similarly for columns.
3. Interchange any two rows (only alters the sign of the determinant), similarly for columns.
4. The sign of all values in a row are changed (only alters the sign of the determinant), similarly for columns.
5. The elements of a row are multiplied by a value β that is coprime to m and another row is multiplied by the reciprocal of this value β^{-1}, similarly for columns.

As application of any or all of these rules to a matrix *L* will produce a matrix with the same determinant as *L,* its inverse will have the same determinant as L^{-1} and therefore exist mod *m*.

Exercise 4.2

1. Use different values of *a*, *b* and *c* to construct 5 involutory 2×2 matrices. Check that they are their own inverse.

2. Use Hill's method to construct a 3×3 matrix L and hence its inverse L^{-1}, with non zero values off the main diagonal.

3. Use Hill's method to construct a 5×5 matrix L, with non zero values off the main diagonal

4.2 The One Time Pad Cipher

The development of the One Time Pad can be traced back to 1917 and the invention by Gilbert Vernam of an encryption device while working for AT&T. Vernam and others had been set the task of investigating the security of the newly invented teletypewriter. This was a device that turned keystroke into electrical signals that could be sent over telegraph wires and received by a teleprinter at the other end, which turned them back into letters and printed them. The teletypewriter machines

used the Baudot-Murry code (see Table 4.2), which encode letters, digits and other symbols into 5-bit numbers and these could be sent as electrical impulses (1 for a pulse and 0 for no pulse). It was soon realised that the teleprinter machines offered no security (Kahn, 1967).

Vernam now came up with an ingenious idea; he suggested that paper tapes, which were already being used to record messages in Baudot-Murry code, could also be used to hold an encryption key. This key could then be added bit by bit (modulo 2) to the code of the message (Vernam, 1926). This system also has the advantage that the encipherment key is also the decipherment key. The initial system uses a long loop of tape to hold the key. He patented his device in 1918 (Vernam, 1919).

Vernam's cipher machine was submitted to the US Army. Here cryptanalyst Major Joseph Mauborgne came into contact with the system and was impressed with its efficiency, in particular, with the way in which it integrated encryption and decryption seamlessly into the communication process. However, he immediately saw the system's weakness. The initial loop of tape was relatively short, which essential meant that Vernam's cipher was equivalent to a Vigenère cipher. Kahn (1967) believes that Mauborgne was one of the team at the US Army's Signal School that concluded that the only safe running key was one that was the same length as the message. Mauborgne now brought the ideas together; the key must be random and non repeating.

Definition 4.2 The One Time Pad Cipher
The One Time Pad cipher is a polyalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext *n*-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext *n*-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = k = (k_1, ..., k_i, ..., k_n) \qquad k_i \in Z_m, \ 1 \le i \le n$

and, $T: \quad y_i = x_i + k_i \mod m \qquad 1 \le i \le n$

where,

the set of k_i consists of random variables that are Identical and Independently Distributed (i.i.d.) from a uniform distribution.

This means that each k_i is a random variable that can take on any value from the alphabet *A* and with equal probability, ie,

$$P(k_i = k_1) = P(k_i = k_2) = ... = P(k_i = k_m) = \frac{1}{m}$$

Note that the key $k = (k_1, k_2, k_3, ..., k_n)$ is an *n*-gram, i.e., the keyword is the same length as the plaintext message.

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = k = (k_1, ..., k_i, ..., k_n)$ $k_i \in Z_m$, $1 \le i \le n$

and, T^{-1}: $x_i = y_i - k_i \mod m$ $1 \le i \le n$

For a keyword with a length of *n* characters there are m^n keys for the One Time Pad cipher.

The One Time Pad cipher can be viewed as the Vigenère cipher with a key length that is the same size as the plaintext message.

The One Time Pad gets its name from the pads consisting of pages of random cipher keys that were used once and then destroyed.

4.2.1 The Security of the One Time Pad

When used correctly, the One Time Pad is completely and utterly unbreakable.

This raises some interesting questions.

> Given that the Vigenère cipher is not regarded as a very secure cipher, why is the One Time Pad unbreakable?
>
> Given that the One Time Pad cipher is unbreakable, why isn't it the only cipher ever used?
>
> How do we generate the random key?
>
> What are the operational procedures needed for the One Time Pad to be secure?

As the One Time Pad uses a key that is random and the same length as the message it is a polyalphabetic cipher with a random alphabet for each letter of the plaintext. This eliminates the possibility of using frequency analysis to uncover any information about the message.

The other possible means of attach is to take a brute force approach and to try every possible key. This is guaranteed to eventually uncover the plaintext. However, this approach will also uncover every single plaintext message of the same length that has

ever been written or ever will be written and there is no way of distinguishing the correct plaintext message.

Example 4.1

Consider a brute force attack on the ciphertext of a single three-letter word:

XTC

This will eventually uncover the following:

cat
dog
top
tap
tip
pit
wig
dip
etc

amongst many, many other words. And we have no way of knowing which one is correct!

What are the operational procedures needed for the One Time Pad to be secure?

In addition to normal security requirement, the One Time Pad needs:

the key to be the same length as the message;
the key must be random;
the key must never be reused.

Normal operational procedure is to destroy the key once it has been used.

Code	Letter shift	Figure Shift
00000	null	null
00100	space	space
11101	Q	1
11001	W	2
10000	E	3
01010	R	4
00001	T	5
10101	Y	6
11100	U	7
01100	I	8
00011	O	9
01101	P	0
11000	A	-
10100	S	bell
10010	D	$
10110	F	!
01011	G	&
00101	H	#
11010	J	'
11110	K	(
01001	L	)
10001	Z	"
10111	X	/
01110	C	:
01111	V	;
10011	B	?
00110	N	,
00111	M	.
00010	carriage return	carriage return
01000	line feed	line feed
11011	shift to figures	
11111		shift to letters

Table 4.2 The Baudot-Murry code

Exercise 4.3

1. Encipher the message "*the shipment arrives tonight*" using the One Time Pad cipher with key "*xapgyjdnebbyfbploiszaxiem*".

2. Encrypt the plaintext binary number 10010 with the key 10101 by adding bitwise modulo 2. Then encrypt the ciphertext again using the key 10101.

3. Put the plaintext "*I will meet you at 10 o'clock*" into Baudot-Murry code.

4. Create a short message (maximum of 20 characters) put it into Baudot-Murry code then swap this with someone and decode each others message.

5. Encrypt the Baudot-Murry code from question three using the Vernam cipher using the key

   11101 11001 01010 10100 10111 11111 00010 01010 00110 10000
   01101 11001 11010 00100 00111 01111 10010 11011 10110 00000
   00111 11010 00101 11110 01000 00101 00001 10010 01100 01111

6. Attempt to answer the question "How do we generate the random key?" by giving 5 methods of generating random numbers or letters for a key.

7. Attempt to answer the question: "Given that the One Time Pad cipher is unbreakable, why isn't it the only cipher ever used?" by pointing out the disadvantages of the method.

8. Mr Venona has used a One Time Pad to encrypt a message giving a cipher text "*ukjsjhflfbgwsehtovpabvfsbmdkv*". Unfortunately, he has reused his pad having already used it to encipher another message, namely, "*ukjgzjveuucgdlqkvgxasodfktdp*". Can you uncover these messages?

4.3 References

Hill, L. S., (1929), "*Cryptography in an Algebraic Alphabet*", American Mathematical Monthly, vol. 36: 6, pp. 306-312.

Hill, L. S., (1931), "*Concerning Certain Linear Transformation Apparatus of Cryptography*", American Mathematical Monthly, vol. 38: 3, pp. 135-154.

Weisner, L., and Hill, L.S., (1932), US Patent 1,845,947 "*Message Protector*", United States Patent Office.

Vernam, G. S., (1919), US Patent 1,310,719 "*Secret Signalling System*", United States Patent Office.

Vernam, G. S., (1926), "*Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications*", Journal American Institute of Electrical Engineer, vol. 45, pp. 109-115.

Weisner, L., and Hill, L.S., (1932), US Patent 1,845,947 "*Message Protector*", United States Patent Office.

Chapter 5 – Rotor-machine Ciphers

It was not long after the advent of electro-mechanical devices that people started to consider their use to encipher messages. The first electro-mechanical devices to be of any real cryptographical sophistication were the rotor machines. The first of these were invented independently by several people in different countries at around the same time. The exact order of invention is disputed, but there is good evidence on what was published and when (which is not necessarily the same thing as when invented).

5.1 The First Rotor-machines

5.1.1 Hebern

The American Edward Hebern appears to have been the first person to have built a rotor-machine in 1917. He filed a patent for a Rotor-machine cipher with a single rotor in 1921 (Hebern, 1924). See Figures 5.1 and 5.2. This machine was equivalent to a Vigenère cipher, and apart from the advantage of automation and the consequent reduction in human error it was of no advantage over the pen and paper version. However, he developed his machine over the next few years to include five rotors (Hebern, 1928), and other cryptographic features (Hebern, 1932, 1945).

5.1.2 Damm/Hagelin

The Swede Arvid Damm filed a patent for a Rotor-machine cipher in 1919 in Sweden and in the United States in 1920 (Damm, 1924). In 1925, Boris Hagelin took over the management of the company that had been set up to exploit Damm's invention, and after Damm's death in 1927 he became the company's main inventor (Hagelin, 1928, 1958).

At the beginning of WWII Hagelin moved the company from Sweden to Switzerland where it remains to this day as Crypto AG. During WWII the company made the C-38 cipher machine for the US military (called M-209 by the army, CSP-1500 by the Navy) over 140,000 of these were made in the period up until the Korean War. According to Kahn (1967), Hagelin is the only cipher machine maker to become a millionaire. Hagelin and Kahn (1994) give a brief history up to the mid 1950s of the development of the Hagelin machine.

5.1.3 Scherbius/Koch

The German Arthur Scherbius filed a patent for a Rotor-machine cipher in Germany in 1918 and in the United States in 1922 (Scherbius, 1925). The Dutch inventor Hugo Koch, filed a Dutch patent for a Rotor-machine cipher in 1919 and a corresponding United States patent in 1920 (Koch, 1925). Scherbius bought Koch's patent in 1927 (Kahn, 1967). This has led some to believe that Koch had priority, however, as Bauer (1999) points out, Scherbius probably bought the Koch patent to protect his own already existing patent.

5.1.4 Spengler/Hengel

There is some recent evidence (de Leeuw, 2003), that two Dutch naval officers, R. Spengler and Theo van Hengel, invented a Rotor-machine cipher around 1915. However, no patents were awarded (possible due to an intervention by the Dutch government) and drawings and detailed descriptions have since been lost.

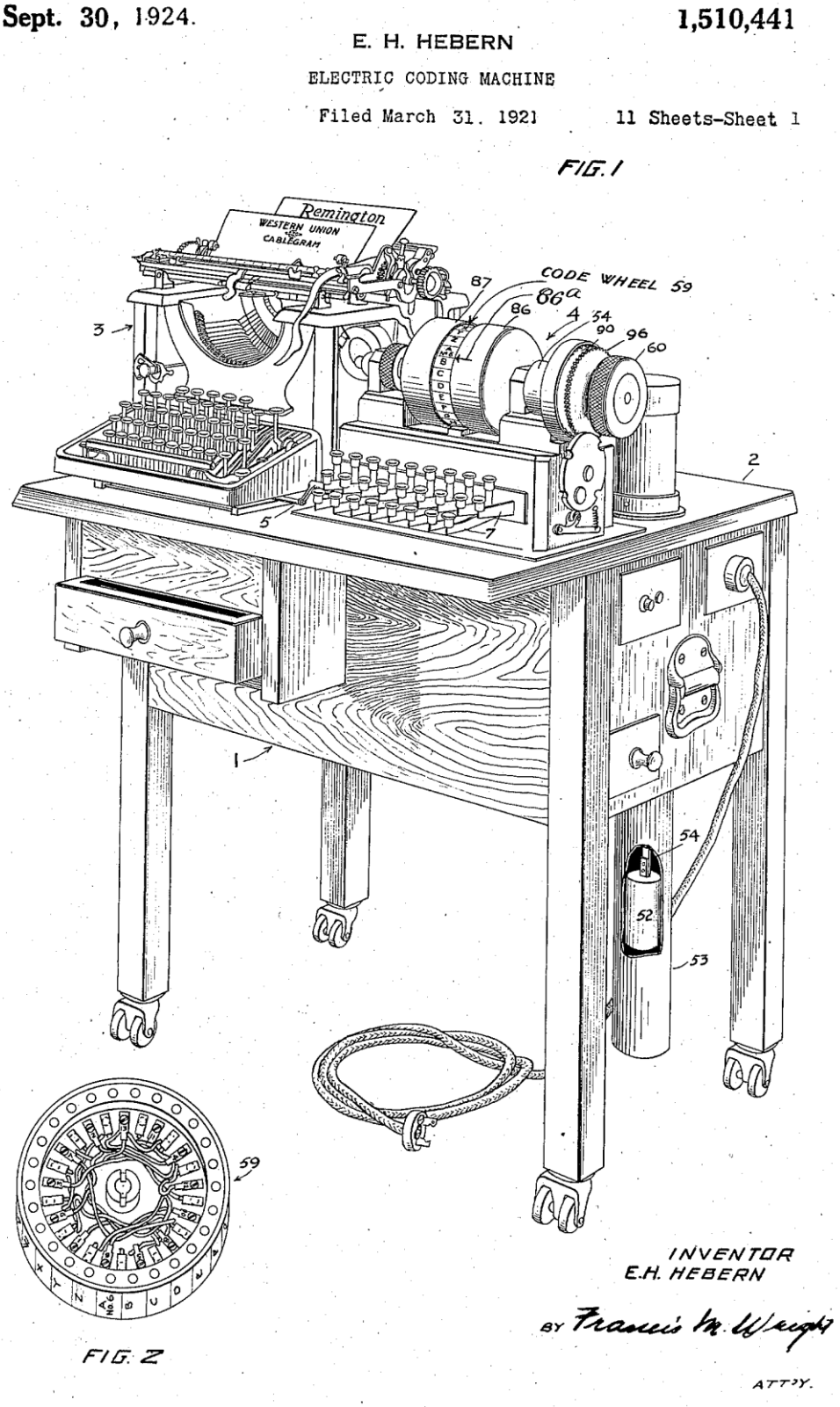

Figure 5.1 Hebern's "Electric Coding Machine" (from Hebern, 1924)

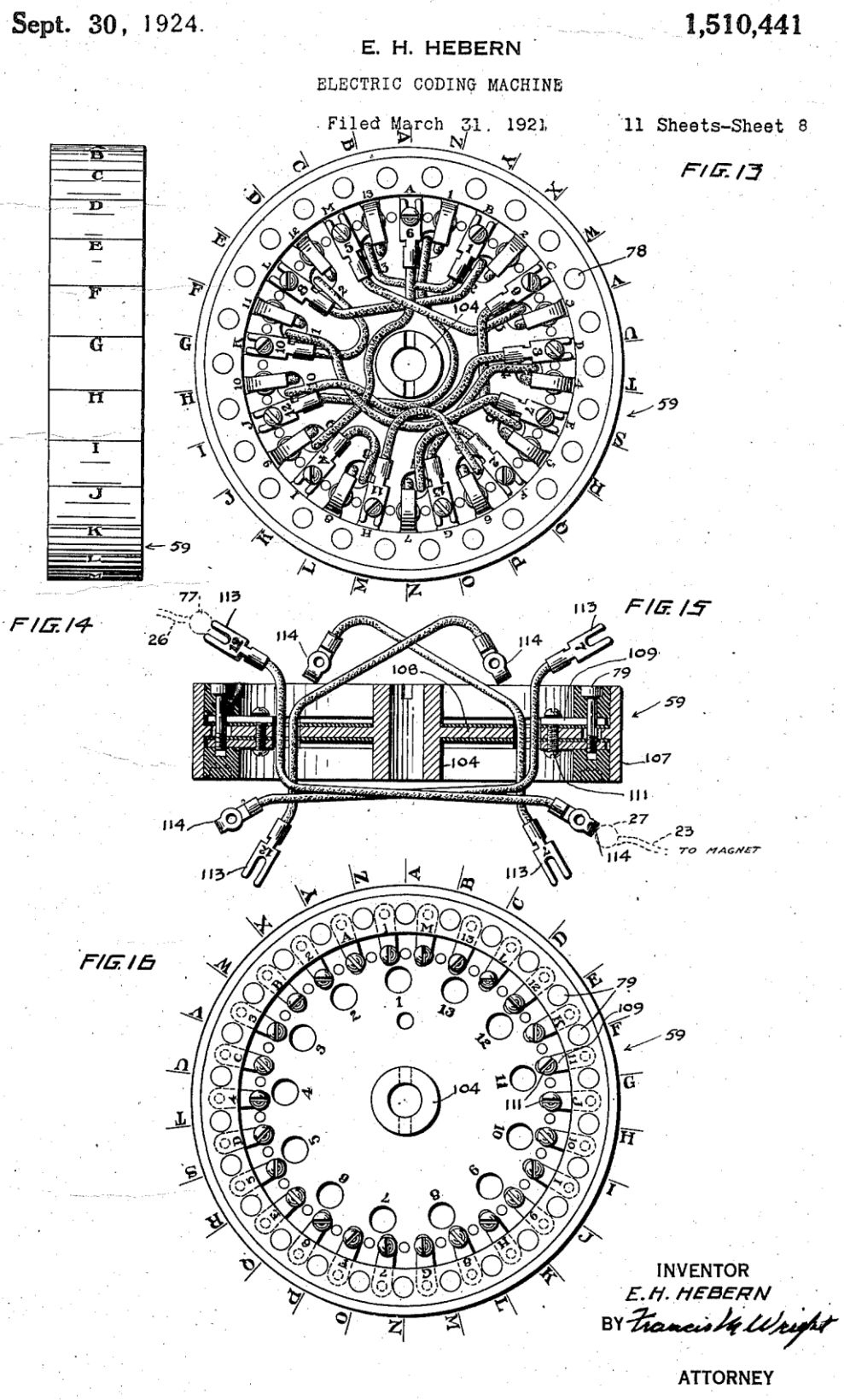

Figure 5.2 Cipher Wheel used in Hebern's Rotor Machine (from Hebern, 1924)

5.2 The One-rotor Machine Cipher

When designing a rotor cipher there are several arbitrary decisions to be made: whether the signals go from left to right or right to left; whether the letters are arranged around the disc clockwise or anticlockwise; and whether the disc rotates clockwise or anticlockwise. Whichever decisions are made, and various designers have made different decisions, essentially the same cipher is produced. Figure 5.3 shows a One-rotor machine in which the plaintext enters from the right hand side and passes through the rotor to produce the ciphertext at the left hand side.

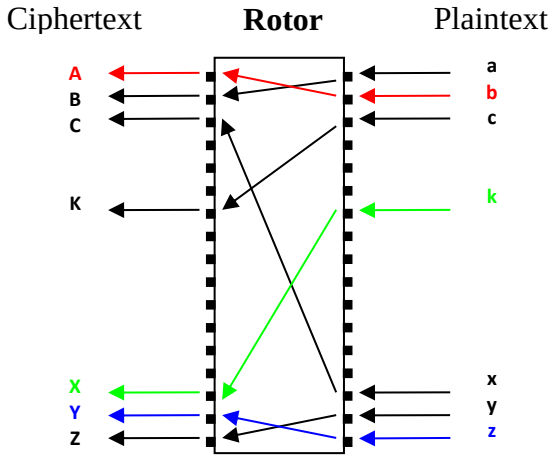

Figure 5.3 A One-rotor Machine Cipher

Definition 5.1 The One-rotor Machine Cipher

The One-rotor Machine cipher is a polyalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = (k, s, \pi)$ π a permutation of A, ie, the rotor wiring
$1 \le s \le 25$, ie, the rotor stepping
$k \in Z_m$, ie, the initial rotor setting

and, T : $y_i = \pi(x_i + c_i) - c_i \mod m \quad 1 \le i \le n$
$c_i = k - s \times (i - 1) \mod m \quad 1 \le i \le n$

where c_i is the cumulative shift of the rotor after the encipherment of i letters.

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = (k, s, \pi^{-1})$ π^{-1} a permutation of A, ie, the rotor wiring
$1 \le s \le 25$, ie, the rotor stepping
$k \in Z_m$, ie, the initial rotor setting

and, T^{-1}: $x_i = \pi^{-1}(y_i + c_i) - c_i \mod m \quad 1 \le i \le n$
$c_i = k - s \times (i-1) \mod m \quad 1 \le i \le n$

Note that Hebern's one rotor machine would have had s and π set at the manufacturing stage.

There are $m \times m \times m!$ keys for the One-rotor Machine cipher.

A One-rotor Machine cipher with a stepping of $s = 1$ is equivalent to a Vigenère cipher. If the rotor is wired to give an identity substitution (π = I), this is equivalent to a Vigenère with a repeat key of "*A*", ie, the first row of the tabular recta in Table 3.1. If the rotor is wired to give another substitution, then this is equivalent to the General Vigenère cipher. For example, when the rotor is wired for the substitution *p* = *thislepgywnomarkdbfcjquvxz*, then the tabular recta in Table 3.2 can be used, and s = 1 is equivalent to using the running key "*ABC…XYZ*".

Exercise 5.1

1. For the One-rotor Cipher, show that the encryption and decryption transformations given in Definition 5.1 are inverse functions of one another.

2. Hebern's Rotor cipher machine had one rotor with in-effect a hard-wired substitution cipher, which rotated one position after enciphering a letter – ie, a One-rotor Machine cipher with s = 1. For a rotor wired to give an identity substitution (π = I), encipher the plaintext "*the more I learnt about ciphers the more confused I became*". Remove spaces and punctuation marks first.
   (i) Use the key "*A*", that is, start the rotor at *A*.
   (ii) Use the key "*B*", that is, start the rotor at *B*.

3. For a rotor wired to give an identity substitution (π = I), simplify the equations given in Definition 5.1.

4. For a wheel wired in the following way:
   $\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ s\ l\ e\ p\ g\ y\ w\ n\ o\ m\ a\ r\ k\ d\ b\ f\ c\ j\ q\ u\ v\ x\ z \end{pmatrix}$,
   and with s = 1, use the equations in Definition 5.1 with the key k = "*B*" to encipher the plaintext "tomorrow".

5.3 The Simulated Rotor Wheel

Figure 5.4 shows a simulation of a One-rotor machine using two disks (partially completed). The letters on the outer disc represents the physical keys on the machine's typewriter or keyboard. This is where the plaintext is input. The letters on the inner disc represent the letters found around the outside of a rotor (see Figure 5.2). The lines on the inner disc represent the wiring of the rotor. After encipherment the letters on the outer disc now represent the ciphertext output, i.e., the letter to be displayed, printed or transmitted by the rotor machine.

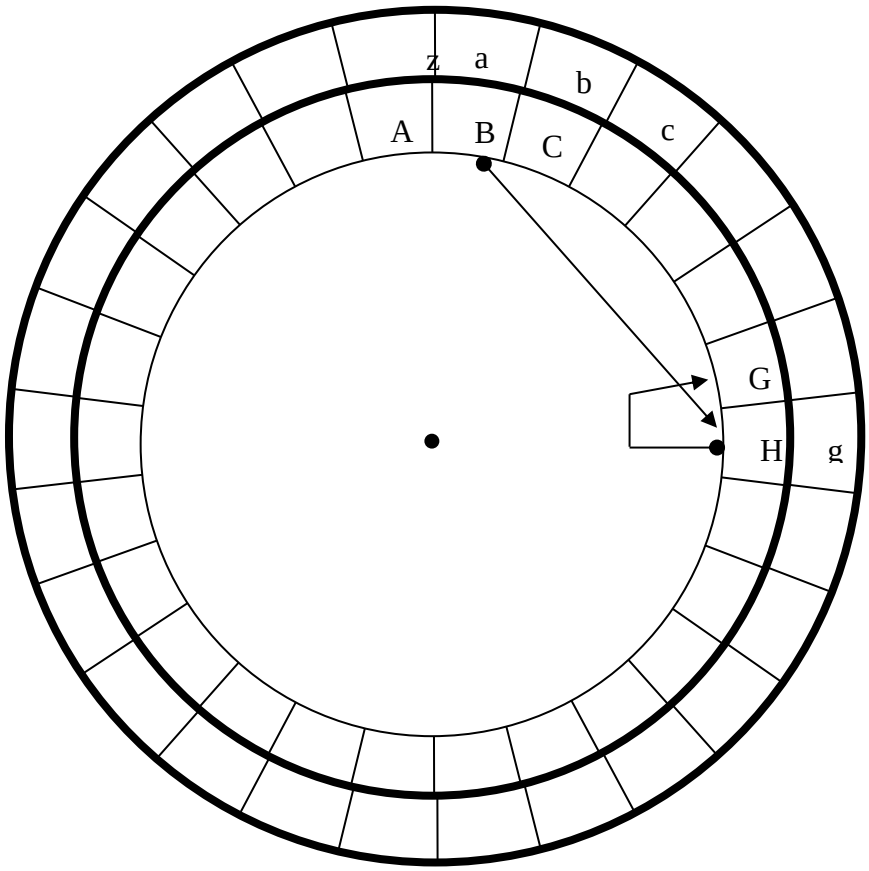

Figure 5.4 The Simulated Rotor Machine

Encipherment using the simulated rotor wheel

First set the key, by aligning the key letter on the inner disc with the "*a*" on the outer disk. To encipher the first letter of the plaintext find it on the outer disc, then move to the adjacent letter on the inner disc, then follow the arrow from tail to head to the letter on the inner disc, then move to the adjacent letter on the outer disc, this is the first ciphertext letter. Now to simulate the rotor stepping, increment the disc by rotating the inner disc clockwise *s* places. Now encipher the second letter of the plaintext, etc.

Example 5.1

Encipher "*today*" using the key "*K*", using a rotor stepping of 1, with a rotor wiring given by the permutation:

$$\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ \ j\ k\ l\ m\ n\ o\ p\ q\ r\ \ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ s\ l\ e\ \ p\ g\ y\ w\ n\ o\ m\ a\ r\ \ k\ d\ b\ f\ \ c\ j\ q\ u\ \ v\ x\ z \end{pmatrix}$$

First align the "*K*" on the inner disc with "*a*" on the outer disc.

Plaintext "*t*" on the outer disc is adjacent to "*D*" on the inner disc. This is connected along the arrow to "*S*" on the inner disc, which is adjacent to "*i*" on the outer disc this gives the ciphertext "*I*".

Next rotate the inner disc one place clockwise.

Now encipher "*o*" to "*M*" and rotate the inner disc again.
Now encipher "*d*" to "*G*" and rotate the inner disc again.
Now encipher "*a*" to "*Z*" and rotate the inner disc again.
Now encipher "*y*" to "*F*" and rotate the inner disc again.

So, "*today*" becomes "*IMGZF*".

Decipherment using the simulated rotor wheel

First set the key, by aligning the key letter on the inner disc with the "*a*" on the outer disk. To decipher the first letter of the ciphertext find it on the outer disc, then move to the adjacent letter on the inner disc, then follow the arrow from head to tail to the letter on the inner disc, then move to the adjacent letter on the outer disc, this is the first plaintext letter. Now to simulate the rotor stepping, increment the disc by rotating the inner disc clockwise *s* places. Now decipher the second letter of the plaintext, etc.

Example 5.2

Decipher "*IMGZF*" using the key "*K*", using a rotor stepping of 1, with a rotor wiring given by the permutation:

$$\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ \ j\ k\ l\ m\ n\ o\ p\ q\ r\ \ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ s\ l\ e\ \ p\ g\ y\ w\ n\ o\ m\ a\ r\ \ k\ d\ b\ f\ \ c\ j\ q\ u\ \ v\ x\ z \end{pmatrix}$$

First align the "*K*" on the inner disc with "*a*" on the outer disc.

Ciphertext "*I*" on the outer disc is adjacent to "*S*" on the inner disc. This is connected backwards along the arrow to "*D*" on the inner disc, which is adjacent to "*t*" on the outer disc this gives the plaintext "*t*".

Next rotate the inner disc one place clockwise.

Now decipher "*M*" to "*o*" and rotate the inner disc again.
Now decipher "*G*" to "*d*" and rotate the inner disc again.
Now decipher "*Z*" to "*a*" and rotate the inner disc again.
Now decipher "*F*" to "*y*" and rotate the inner disc again.

So, "*IMGZF*" becomes "*today*"

Exercise 5.2

1. Using the simulated rotor machine, redo question 4 from Exercise 5.1.

2. Use a One-rotor Machine cipher wired in the following way:
   $\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ s\ l\ e\ p\ g\ y\ w\ n\ o\ m\ a\ r\ k\ d\ b\ f\ c\ j\ q\ u\ v\ x\ z \end{pmatrix}$
   with $s = 3$, and key "*G*" to decipher the ciphertext "*LRURU QBBQB KRYUM R*".

3. Use a One-rotor Machine cipher wired in the following way:
   $\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ s\ l\ e\ p\ g\ y\ w\ n\ o\ m\ a\ r\ k\ d\ b\ f\ c\ j\ q\ u\ v\ x\ z \end{pmatrix}$
   to encipher the plaintext "*the more I learnt about ciphers the more confused I became*". Remove spaces and punctuation marks first.

   (i) Use $s = 1$, and key "*A*".
   (ii) Use $s = 1$, and key "*B*".
   (iii) Use $s = 2$, and key "*A*".
   (iv) Use $s = 3$, and key "*A*".

4. How long before the key repeats in each of the situations in question 3?

5. What methods can be used to decrypt the One-rotor Machine ciphers in question 3?

5.4 Rotor-machines at War

Rotor-machines have played a very significant part in the history of cryptology. In fact they have played a very important part in the course of history in general. The use of rotor-machines by various countries and their cryptanalysis by other countries has almost certainly changed the outcome of battles and possibly of wars.

5.4.1 The Enigma

The most famous rotor cipher was the Enigma machine. This was based on the Scherbius rotor-machine, and had been initially targeted at the commercial market. However, it was the German military that eventually adopted the machine and used it through to the end of WWII; first the Navy in 1926, then the Army in 1928 and finally the Air Force in 1935 (Stripp, 1993).

The Enigma, in various forms, was used by the Germany Army, Navy, Air Force, other areas of the military and the Diplomatic services. In all, around 100,000 machines were deployed by the Germans (Baur, 1997)

There were several versions of the Enigma used, the most common being the one used by the German Army. This was essentially a 3-rotor machine that had a few additional features designed to make cryptanalysis much more difficult. This Enigma came with three removable rotors (later five) that could be placed in the machine in any order.

Figure 5.5 shows a Navy Enigma, which was a 4-rotor machine. Notice the plugboard located at the bottom of the machine, which was used to add a simple substitution cipher that greatly increased the Enigma's keys pace. This was a critical addition that greatly increased the machines cryptographic strength.

Figure 5.6 shows the three rotors of an Army Enigma in their storage case. On the right hand side of the rotors can be seen the metal wheel used by the operator to turn the rotor to its initial setting. On the left hand side can be seen the alphabetic ring that the operator used to set the rotor to the required key.

Figure 5.7 shows the right and left hand sides of an Enigma rotor. On both sides can be seen the brass contacts through which the electrical signals passed. On the right hand side rotor can be seen the ratchet notches that controlled the rotors stepping.

In normal operation, the Enigma needed three operators: one to type in the plaintext message, one to read off the ciphertext and one to send the message by Morse code.

Figure 5.5 A German Navy Enigma Machine

Figure 5.6 Three Enigma Rotors in their Storage Case

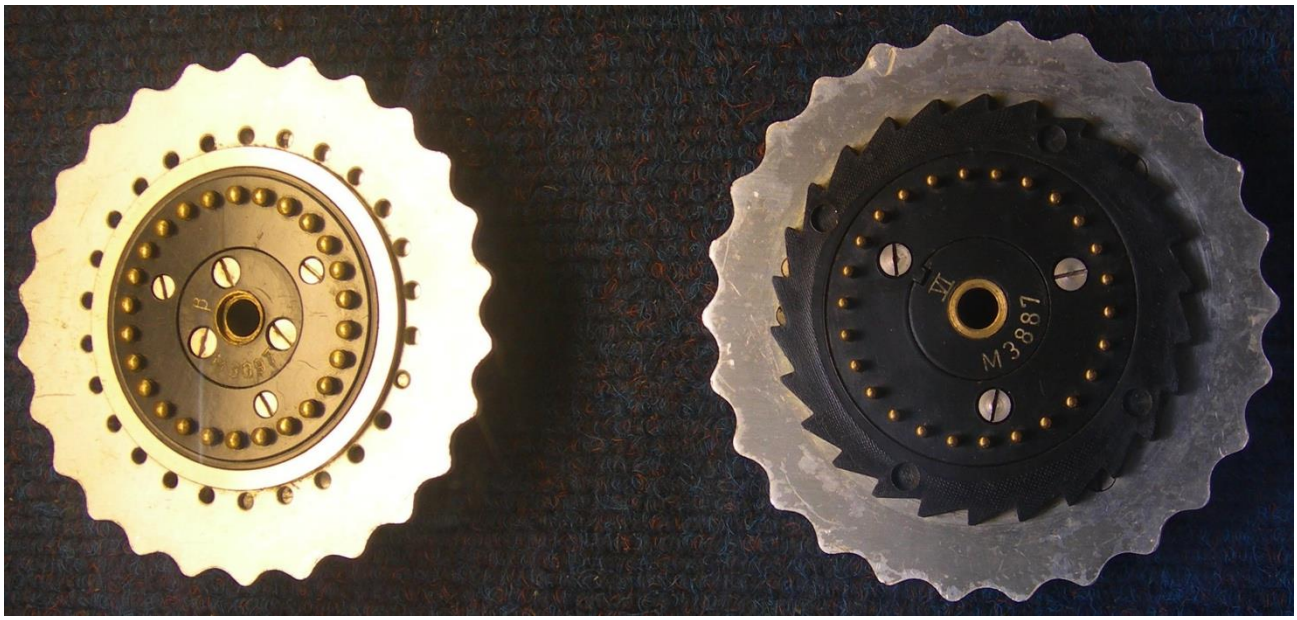

Figure 5.7 Left and Right Hand Sides of an Enigma Rotor

5.4.2 Cryptanalysis of the Enigma

The story of how the British broke the Enigma (based on an earlier break by the Poles) and exploited its messages was kept completely secret for over 30 years. It was not until 1974 when F.W Winterbotham published "The Ultra Secret" (Winterbotham, 1974) that details started to emerge. Yet even today some aspects are still classified.

After the German Army introduced the Enigma the Polish cryptographic service was unable to read their messages, so in 1931 they hired three mathematicians Marian Rejewski, Jerzy Różycki and Henryk Zygalski to analyse the machine. By 1932 the Poles were able to read messages of this pre-war version of the Army Enigma. They were aided in this by the French intelligence officer Gustave Bertrand who had provided them with the Enigma operating manual and key settings that he had obtained by espionage (Lewin, 1978).

One of the most impressive features was Rejewski's use of group theory to discover the wiring of the rotors in this version of the Enigma (Rejewski, 1980), which was the first application of higher-algebra to cryptanalysis (Kahn, 1991).

The Poles made further advances including the invention of Zygalski sheets (see Figure 5.8). These were sheets of card with holes punched in them that corresponded to the 676 possible starting positions of the first and second rotor. A set of 26 sheets were superimposed on top of each other and moved in sequence until a single hole was visible through all the sheets. This could then be used to identify the Enigma's key. A set of 26 sheets was needed for each of the six possible ways of inserting the three rotors into the Enigma.

Another breakthrough by Poles was Rejewski's invention of the Bomba, which was an electro-mechanical device that did the calculations equivalent to 6 Enigma machines at the same time. These were used to conduct a brute force search of the keys on all six of the possible rotor arrangements simultaneously.

However, just before the outbreak of World War II changes to the Army Enigma such as the introduction of two extra rotors giving 60 possible combinations, together with improved operating procedures had made it unreadable again to the Poles.

In 1937, the British cryptographer Dilly Knox had some success in breaking the Italian Navy messages, which used the cryptographically weak commercial Enigma, but had made little progress on the on the German version (Kahn, 1991). After the start of the war the Polish passed on their work to the British and French.

The British employed mathematicians such as Alan Turing and Gordon Welchman, who brought a fresh approach to cryptanalysis, which built on the work previously done and resulted in breakthroughs that changed the outcome of the war.

In 1939, Turing designed the Bombe (Smith, 1998), a development of the Polish Bomba. The first one was completed in early 1940. Figure 5.9 shows a working reconstruction of a Bombe at Bletchley Park. Each set of three vertical drums correspond to the three rotors of an Enigma and it was able to simulate the operation of 36 Enigma machines. Due to a mathematical short cut, this was enough to simulate all 60 possible combinations. The three extra drums on the right hand side of the

middle section are the indicator drums, which when the Bombe stops gives the possible solution.

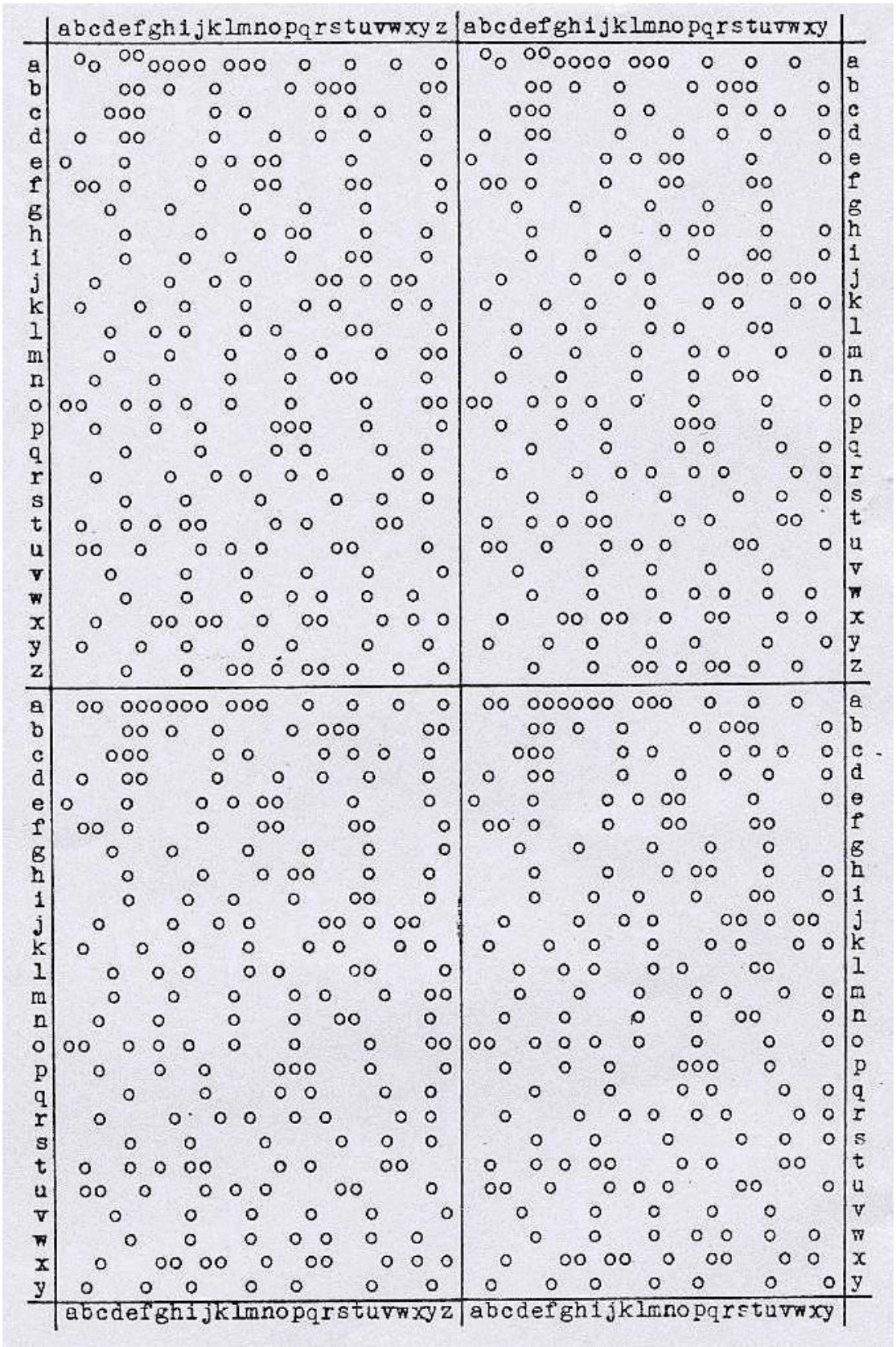

Figure 5.8 A Diagram of a Zygalski Sheet (from Rejewski, 1980)

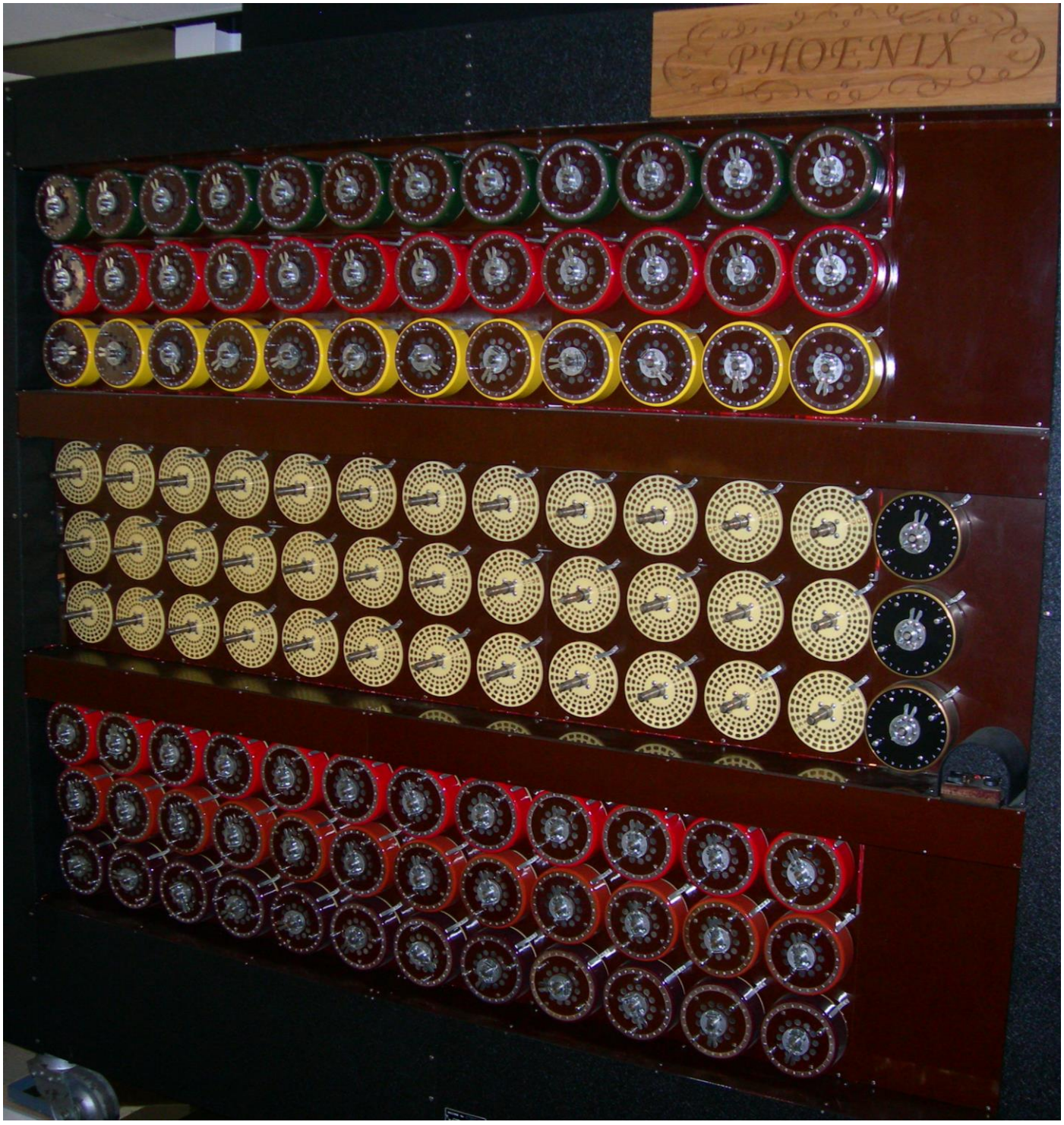

Figure 5.9 A Working Reconstruction of a Bombe at Bletchley Park

In 1940, Welchman refined the design by adding the "diagonal board", and the second Bombe was completed later that year (see Figure 5.10). This substantially increased the Bombe's efficiency (Welchman, 1982).

By the end of the War there were 152 3-Rotor Bombes and 180 4-Rotor Bombes (Alexander, 1945). The later were largely made by the Americans.

As the volume and importance of the decrypted Enigma messages increased in 1940 it was given the codename Ultra (Winterbotham, 1985). At the start of the War, 90 people were transferred to Bletchley Park, the wartime location of the Government Code and Cipher School; by the end of the War there were 9,000 people working there (Hill, 2004).

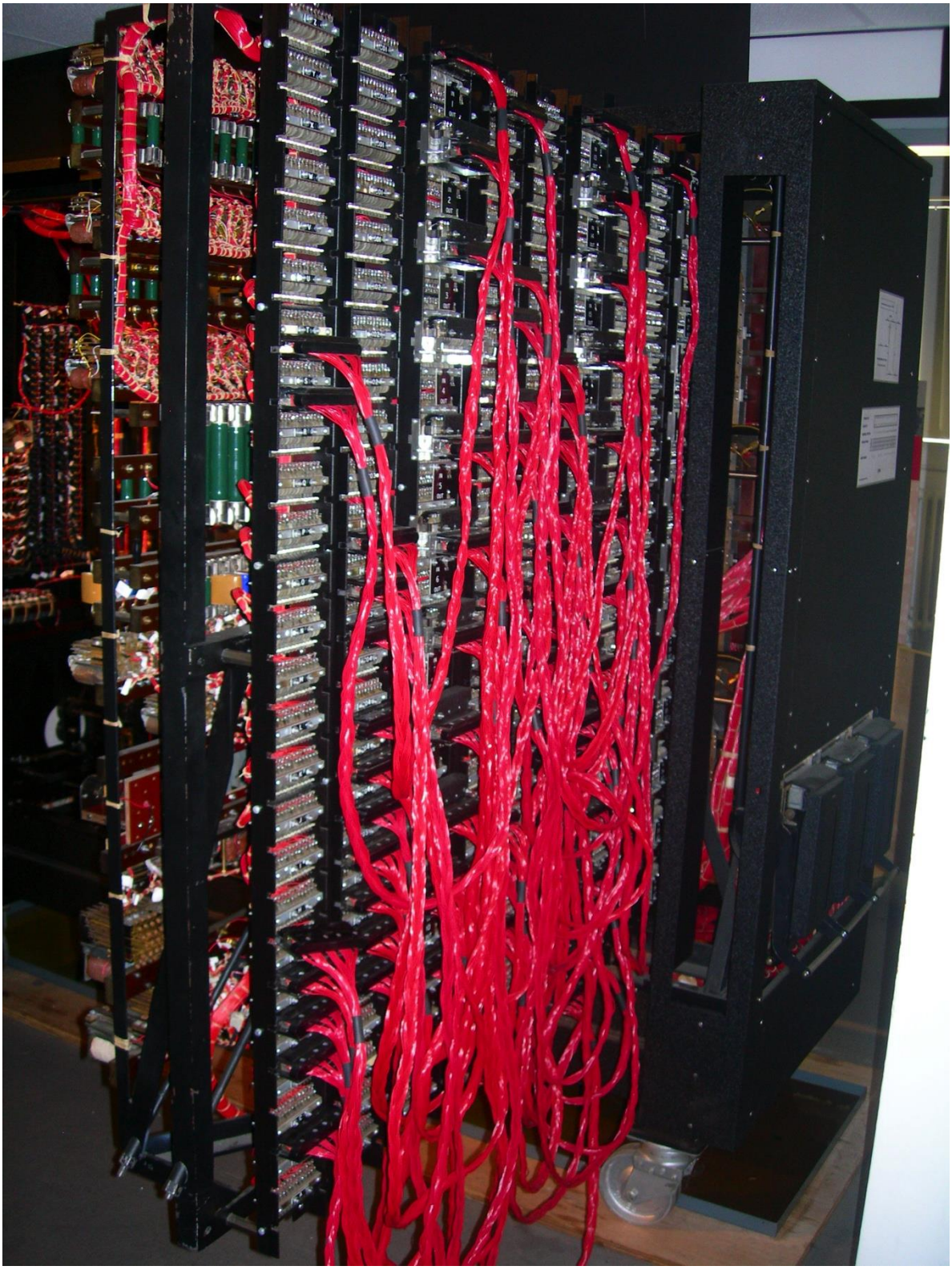

Figure 5.10 The Diagonal Board of a Bombe

5.4.3 The TypeX machine

The British rotor-machine during WWII was the TypeX machine, which was based on the commercial Enigma. It had 5 rotors, although the first 2 did not move during encipherment of the plaintext.

The TypeX needed only one operator, who typed in the plaintext, which was then enciphered and transmitted automatically. However, it did have the drawback that it weighed 120 lbs. There was also a more portable version around the size of a typewriter (this version did not transmit messages automatically).

Figure 5.11 A TypeX Machine

5.4.4 The M209 Machine

The M209 was a 6-rotor machine used by America during WWII. It was a very portable and robust machine and it was still in service until the Vietnam War. The M209 was the Hagelin C-38 made under licence in the USA.

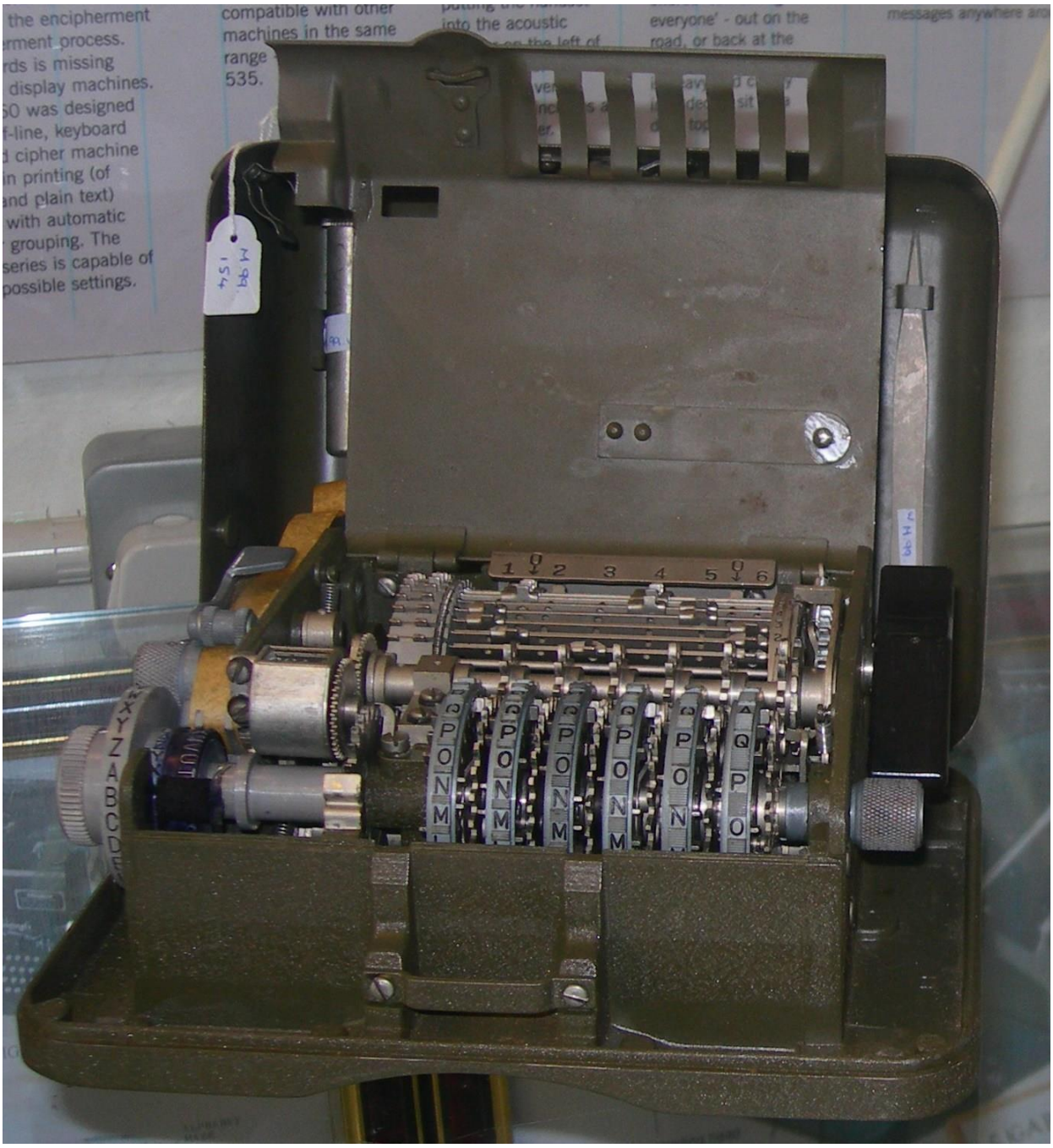

Figure 5.12 The M209 Rotor-machine

5.4.5 The Fialka Rotor-machine

The Fialka was a Soviet rotor machine that was used in the Cold War and was still in use until the mid 1980s. The machine had 10 rotors, and interestingly, each rotor rotated in the opposite direction to the previous one in the sequence. The Fialka came with two sets of rotors, one of which was rewireable. Each rotor had 30 contact points ($m = 30$). This was to accommodate the Cyrillic alphabet, which has 33 characters (3 were not used). Figure 5.13 shows a Fialka with the cover off and with its second rotor set.

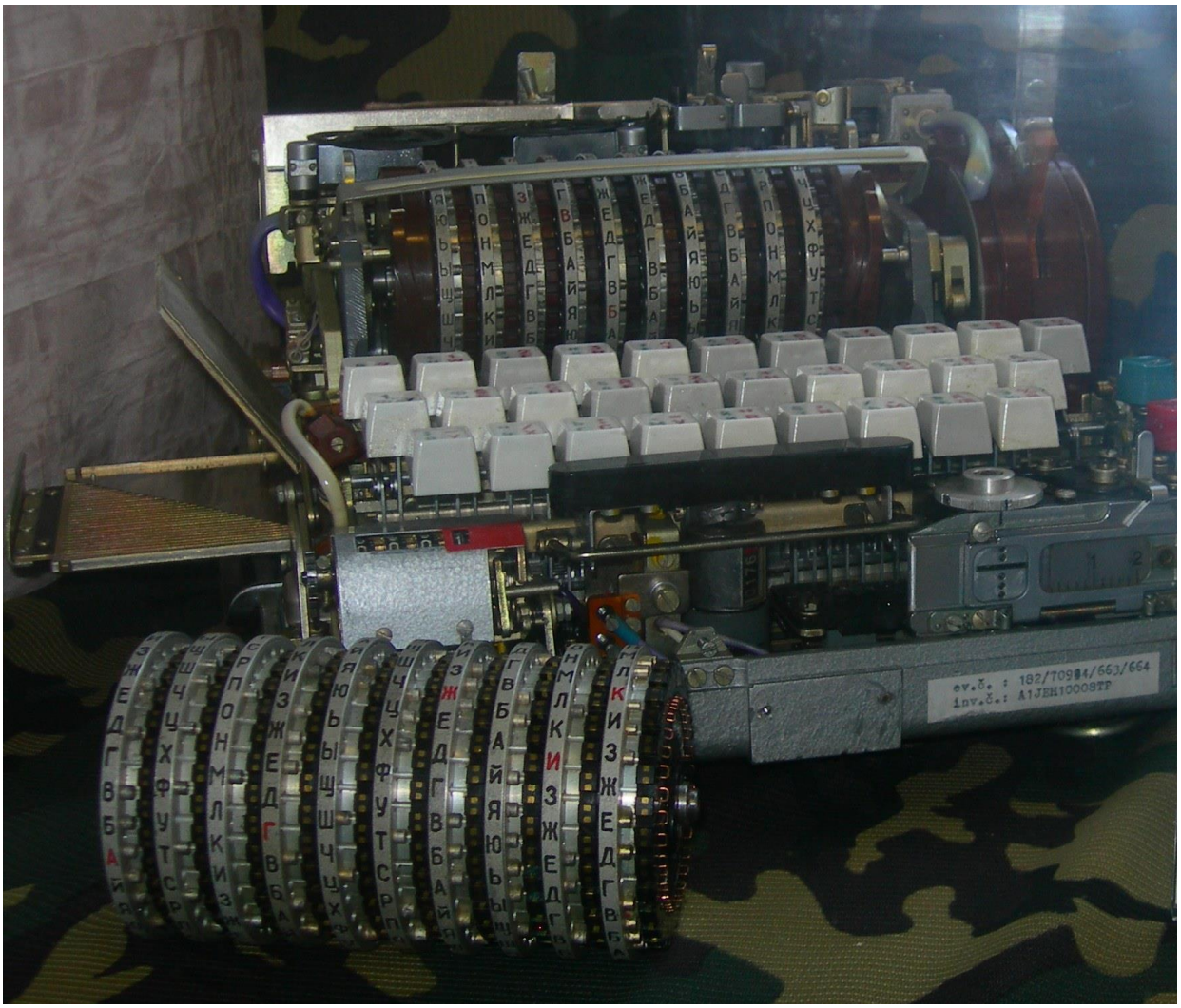

Figure 5.13 A Fialka Cipher Machine

5.5 Machines with Multiple Rotors

Since the One-rotor Machine cipher is very insecure, machines with more than one rotor were quickly developed. Typically, these machines had between three and five rotors. Later machines with up to 15 rotors were constructed, such as the American SIGABA (with 15), the British Mercury (with 10) and the Russian Fialka (with 10).

In a typical multi-rotor machine, the plaintext is enciphered one letter at a time by rotor 1, then this output is enciphered by rotor 2, and the output of rotor 2 is enciphered by rotor 3, and so on. Figure 5.14 shows the partial wiring for a Three-rotor Machine cipher. Note that the plaintext letter "*b*" enciphers to "*Z*", while the "*k*" enciphers to "*Y*" and "*z*" enciphers to "*B*".

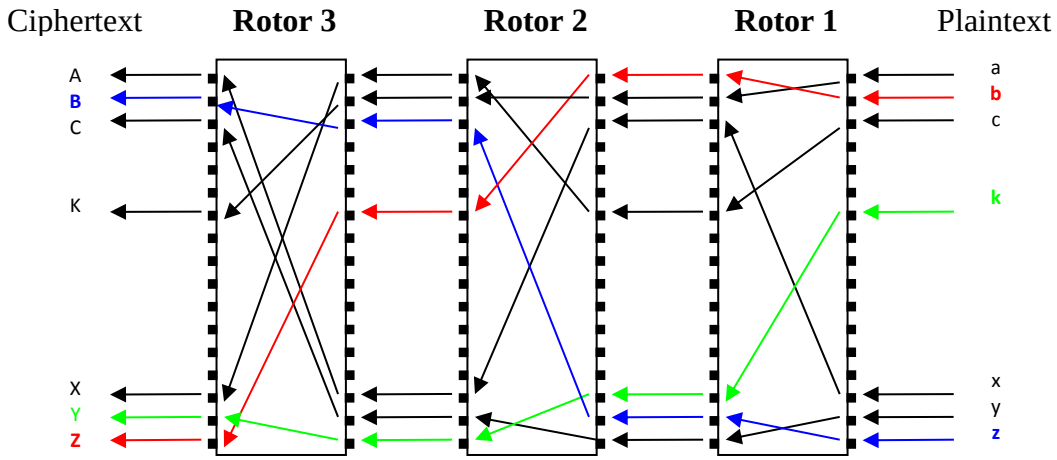

Figure 5.14 A Three-rotor Machine Cipher

After each plaintext letter is enciphered the first rotor rotates. Thus the subsequent letters will be enciphered by a different alphabet until the rotor gets back to its starting position. However, once the first rotor has completed a full rotation it will cause the second rotor, which has been stationary until now, to rotate and introduce a new alphabet. After the second rotor has completed a full rotation it will in turn cause the third rotor to rotate, and so on.

As a result the ciphertext will not be enciphered by the same alphabet until all rotors have completed a full rotation.

Note that rotor two moves when rotor one returns to the base setting (or passes it in a single move), that is when "*A*" returns to "*a*". Similarly, rotor three rotates when rotor two returns to or passes the base setting.

Definition 5.2 The N-rotor Machine Cipher

The N-rotor Machine cipher is a polyalphabetic cipher, it uses an encipherment key e_k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, e_k)$$

where, $e_k = (w_1, ..., w_j, ...w_N)$

$w_j = (k_j, s_j, \pi_j)$ the key settings for rotor j

π_j a permutation of A, ie, the rotor wiring for rotor j

$1 \le s_j \le 25$, ie, the rotor stepping

$k_j \in Z_m$, ie, the initial rotor setting

and, T: $y_{j,i} = \pi_j(x_{j,i} + c_{j,i}) - c_{j,i} \mod m \quad 1 \le i \le n$

$$c_{j,i} = \begin{cases} k_j - s_j \times (i-1) \mod m & 1 \le i \le n \quad j = 1 \\ k_j - s_j \times \left\lfloor \frac{c_{j-1,i}}{m} \right\rfloor \mod m & 1 \le i \le n \quad 2 \le j \le N \end{cases}$$

where $c_{j,i}$ is the cumulative shift of the rotor j after the encipherment of i letters.

Decipherment using decipherment key d_k is accomplished by:

$$x = T^{-1}(y, d_k)$$

where, $d_k = (w_1, ..., w_j, ...w_N)$

$w_j = (k_j, s_j, \pi_j)$ the key settings for rotor j

π_j a permutation of A, ie, the rotor wiring for rotor j

$1 \le s_j \le 25$, ie, the rotor stepping

$k_j \in Z_m$, ie, the initial rotor setting

and, T^{-1}: $x_{j,i} = \pi_j^{-1}(y_{j,i} + c_{j,i}) - c_{j,i} \mod m \quad 1 \le i \le n$

$$c_{j,i} = \begin{cases} k_j - s_j \times (i-1) \mod m & 1 \le i \le n \quad j = 1 \\ k_j - s_j \times \left\lfloor \frac{c_{j-1,i}}{m} \right\rfloor \mod m & 1 \le i \le n \quad 2 \le j \le N \end{cases}$$

Note that, typical, most early rotor machine would have had π_j and s_j set at the manufacturing stage.

There are $(m \times m \times m!)^N$ keys for the N-rotor Machine cipher.

Example 5.3

Encipher "*today is the first day of the rest of your life*" using the key "*D*", and a rotor stepping of 7 on the first rotor, and the key "*A*", and a rotor stepping of 2 on the second rotor. Both rotors use the wiring given by the permutation:

$$\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ \ j\ k\ l\ m\ n\ o\ p\ q\ r\ \ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ \ s\ l\ \ e\ \ p\ g\ y\ w\ n\ o\ m\ a\ r\ \ k\ d\ b\ \ f\ \ c\ \ j\ q\ u\ \ v\ x\ z \end{pmatrix}$$

On the first Rotor
First align the "*D*" on the inner disc with "*a*" on the outer disc.
Plaintext "*t*" on the outer disc is adjacent to "*W*" on the inner disc. This is connected to "*U*" on the inner disc, which is adjacent to "*r*" on the outer Disc this gives the ciphertext "*R*".

Next rotate the inner disc 7 places clockwise.

The "*D*" on the inner disc is now aligned with "*h*" on the outer disc. Note that during this rotation "*A*" on the inner disc has passed "*a*" on the outer disc. This will trigger the second rotor to turn at this point. Plaintext "*o*" on the outer disc is adjacent to "*K*" on the inner disc. This is connected to "*N*" on the inner disc, which is adjacent to "*r*" on the outer Disc this gives the ciphertext "*R*".

Next rotate the inner disc 7 places clockwise.

Now encipher "*d*" to "*Q*" and rotate the inner disc again.
Now encipher "*a*" to "*Q*" and rotate the inner disc again.
ETC.

So, the plaintext is enciphered by the first rotor as:

t o d a y i s t h e f i r s t d a y o f t h e r e s t o f y o u r l i f e
R R Q Q Y O R T Q C S I U I Z C V Y J E T Q R M Z Y R R U H J X Y V H D Z

The second rotor's turnover points have been underlined.

Note that the second rotor will encipher the underlined letter after it has rotated.

On the second Rotor
First align the "*A*" on the inner disc with "*a*" on the outer disc.
Ciphertext "*r*" on the outer disc is adjacent to "*R*" on the inner disc. This is connected to "*B*" on the inner disc, which is adjacent to "*b*" on the outer Disc this gives the ciphertext "*B*".

The second rotor now turns 2 places due to the first rotor.

Ciphertext "*r*" on the outer disc is adjacent to "*P*" on the inner disc. This is connected to "*K*" on the inner disc, which is adjacent to "*m*" on the outer Disc this gives the ciphertext "*M*".

Without rotating the second rotor,
encipher "*q*" to "*T*", then
encipher "*q*" to "*T*", then
encipher "*y*" to "*W*".

Now the second rotor turns 2 places due to the first rotor.
Now, encipher "*q*" to "*T*", etc.

So, the plaintext is enciphered by the first rotor and then the second as:

t o d a y i s t h e f i r s t d a y o f t h e r e s t o f y o u r l i f e
R R Q Q Y O R T Q C S I U I Z C V Y J E T Q R M Z Y R R U H J X Y V H D Z
B M T T W R E O T A S O U B J R Y B J R S X Q L C B G X B R S W H K U Q Y

The positions where the second rotor turns are: 2, 6, 9, 13, 17, 20, 24, 28, 32 and 35.

This gives the following pattern for the number of letters enciphered by each position of the second rotor: 1, 4, 3, 4, 4, 3, 4, 4, 4, and 3.

Exercise 5.3

1. Use a Two-rotor Machine cipher with each rotor wired in the following way
$$\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ s\ l\ e\ p\ g\ y\ w\ n\ o\ m\ a\ r\ k\ d\ b\ f\ c\ j\ q\ u\ v\ x\ z \end{pmatrix}$$
to encipher the plaintext "*the more I learnt about ciphers the more confused I became*". Remove spaces and punctuation marks first.

   (i) Use $s_1 = 1$, $k_1 =$ "*A*", and $s_2 = 1$, $k_2 =$ "*A*".
   (ii) Use $s_1 = 1$, $k_1 =$ "*A*", and $s_2 = 1$, $k_2 =$ "*B*".
   (iii) Use $s_1 = 2$, $k_1 =$ "*A*", and $s_2 = 2$, $k_2 =$ "*K*".
   (iv) Use $s_1 = 3$, $k_1 =$ "*A*", and $s_2 = 3$, $k_2 =$ "*K*".

2. What methods can be used to decrypt the Two-rotor Machine cipher above?

3. Use a Three-rotor Machine cipher with each rotor wired in the following way
$$\pi = \begin{pmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z \\ t\ h\ i\ s\ l\ e\ p\ g\ y\ w\ n\ o\ m\ a\ r\ k\ d\ b\ f\ c\ j\ q\ u\ v\ x\ z \end{pmatrix}$$
to encipher the plaintext "*the more I learnt about ciphers the more confused I became*". Remove spaces and punctuation marks first.

   Use $s_1 = 5$, $k_1 =$ "*C*", $s_2 = 7$, $k_2 =$ "*K*", and $s_3 = 11$, $k_3 =$ "*W*".

4. What methods can be used to decrypt the Three-rotor Machine cipher above if the rotor wirings are known?

5. What happens in the case of question 4 when a simple substitution cipher is used:

   (i) before encipherment on the machine?
   (ii) after encipherment on the machine?

6. Would a 100-rotor Machine cipher be more secure than a 10-rotor Machine cipher?

Exercise 5.4

1. A One-rotor Machine cipher is wired in the following way:

$$\text{Rotor } 2 = \begin{pmatrix} a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\ z & y & x & w & v & u & t & s & r & q & p & o & n & m & l & k & j & i & h & g & f & e & d & c & b & a \end{pmatrix}$$

   (i) Use the rotor machine with the key: $s_1 = 2$, $k_1 =$ "*F*" to encipher the plaintext "*the more I learnt about ciphers the more confused I became*". Remove spaces and punctuation marks first.

   (ii) Use the rotor machine with the key: $s_1 = 5$, $k_1 =$ "*Z*" to decipher the ciphertext "*IENNH HJBZL GDYYX IMWVY IDAZT EX*".

2. A Two-rotor Machine cipher has Rotor 1 wired in the following way:

$$\text{Rotor } 1 = \begin{pmatrix} a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\ t & h & i & s & l & e & p & g & y & w & n & o & m & a & r & k & d & b & f & c & j & q & u & v & x & z \end{pmatrix}$$

   and Rotor 2 wired in the following way:

$$\text{Rotor } 2 = \begin{pmatrix} a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\ z & y & x & w & v & u & t & s & r & q & p & o & n & m & l & k & j & i & h & g & f & e & d & c & b & a \end{pmatrix}$$

   (i) Use the rotor machine with the key: $s_1 = 3$, $k_1 =$ "*K*", and $s_2 = 5$, $k_2 =$ "*P*" to encipher the plaintext "*the more I learnt about ciphers the more confused I became*". Remove spaces and punctuation marks first.

   (ii) Use the rotor machine with the key: $s_1 = 4$, $k_1 =$ "*C*", and $s_2 = 3$, $k_2 =$ "*D*" to decipher the ciphertext "*KNOLH JCMVU EKOFB BEUAK RDKAU KPQLL XPMGM AAXUV WMWQF SYXLV VENLV ESRF*".

5.6 References

Alexander, C., H., O'D., (1945), "*Cryptographic History of Work on the German Naval Enigma*", The National Archives, Kew, Reference HW 25/1, UK.

Baur, F., L., (1997), "*Decrypted Secrets: Methods and Maxims of Cryptology*", Springer, Berlin.

Bauer, F. L., (1999), "*An Error in the History of Rotor Encryption Devices*", Cryptologia, vol. 23: 3, pp. 206 - 210.

Damm, A. G., (1924), US Patent 1,502,376 "*Production of Ciphers*", United States Patent Office.

de Leeuw, K, (2003), "*The Dutch Invention of the Rotor Machine, 1915-1923*", Cryptologia, vol. 27: 1, pp. 73 - 94.

Hagelin, B., (1928), US Patent 1,846,105 "*Ciphering Apparatus*", United States Patent Office.

Hagelin, B., (1958), US Patent 2,851,794 "*Cryptogrammic Coding and Decoding Apparatus*", United States Patent Office.

Hagelin, B., and Kahn, D., (1994), "*The Story of the Hagelin Cryptos*", Cryptologia, vol. 18:3, pp. 204 - 242.

Hebern, E. H., (1924), US Patent 1,510,441 "*Electric Coding Machine*", United States Patent Office.

Hebern, E. H., (1928), US Patent 1,683,072 "*Electric Code Machine*", United States Patent Office.

Hebern, E. H., (1932), US Patent 1,861,857 "*Cryptographic Machine*", United States Patent Office.

Hebern, E. H., (1945), US Patent 2,373,890 "*Cipher Machine*", United States Patent Office.

Hill, M., (2004), "*Bletchley Park People: Churchill's Geese that Never Cackled*", Sutton Publishing, UK

Kahn, D., (1991), "*Seizing the Enigma: the Race to Break the German U-boat codes, 1939-43*", Houghton Mifflin Company, Boston, USA.

Koch, H.A., (1925), US Patent 1,533,252 "*Coding and Decoding Machine*", United States Patent Office.

Lewin, R., (1978), "*Ultra Goes to War: The Secret Story*", Hutchinson & Co., London.

Rejewski, M., (1980), "*An Application of the Theory of Permutations to Breaking the Enigma Cipher*", Applicaciones Mathematicae, Vol. 16, No. 4, Warsaw.

Scherbius, A., (1925), US Patent 1,556,964 "*Electric Ciphering Apparatus*", United States Patent Office.

Strip, A., (1993), "*The Enigma Machine: Its Mechanism and Use*" in Hinsley, F., H., and Stripp, A., (Editors), "*Code Breakers: The inside Story of Bletchley Park*", Oxford University Press.

Smith, M., (1998), "*Station X: The Codebreakers of Bletchley Park*", Pan McMillan Ltd, London.

Welchman, G., (1982), "*The Hut Six Story: Breaking the Enigma Codes*", McGraw Hill, USA.

Winterbotham, F., (1974), "*The Ultra Secret*", Weidenfeld and Nicolson, London.

Winterbotham, F., (1985), "*The Ultra Spy*", Macmillan, London.

Chapter 6 – Bayesian Cryptanalysis

6.1 Bayes' Theorem

This theorem is named after the Reverend Thomas Bayes who was the first person to discover it. It was published posthumously in 1764.

Assume that two events *A* and *B* can occur and that event *A* is from a discrete probability distribution and that event *B* is from another discrete probability distribution.

P(*A*) is the probability that the event *A* occurs, this is called the *unconditional probability*, as there is no condition imposed on or by the event *B* occurring or not occurring. It is also called the *prior probability*, as it does not assume any *prior* knowledge of event *B*.

Similarly, *P*(*B*) is the *unconditional probability* of the event *B* occurring.

P(*A*/*B*) is the *conditional probability* of event *A* occurring given that *B* has already occurred. It is defined as

$$P(A/B) = \frac{P(A \text{ and } B)}{P(B)}$$

where $P(A \text{ and } B)$ is the probability that both *A* and *B* occur.

Note, $P(A \text{ and } B)$ is the *joint probability* of events *A* and *B* occurring. It is often written as $P(A \cap B)$ and also as $P(A,B)$.

Similarly, *P*(*B*/*A*) is the *conditional probability* of event *B* occurring given that *A* has already occurred.

Given these definitions, Bayes' theorem states that

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

Note that *P*(*A*) and *P*(*B*) are also called the *marginal probabilities* of events *A* and *B* respectively. This is comes from viewing *A* and *B* as the two dimensions of a single probability distribution and *P*(*A*) is then the sum over all events *B* for a particular event *A*. (The name comes from the probability table of all possible pairs of events *A* and *B* and the fact that the sums were written in the margins for each row and each column – hence marginal probabilities.)

Using the fact that probabilities sum to 1

$$P(B) = P(B/A)P(A) + P(B/A^c)P(A^c)$$

where A^c is the complement of A, (ie, not A).

Using this, Bayes' theorem is often given in the following form (DeGroot, 1986)

$$P(A/B) = \frac{P(B/A)P(A)}{P(B/A)P(A) + P(B/A^c)P(A^c)}$$

It can also be given in the form

$$P(A/B) = \frac{P(B/A)P(A)}{\sum_{i=1}^{k} P(B/A_i)P(A_i)}$$

where there are k possible events A_i

Exercise 6.1

1. In a small town 60% of drivers are male and 40% female. All female drivers drive blue cars. Of male driver, 80% drive red cars and 20% drive blue cars. A speed camera has recorded a blue car as breaking the speed limit. What is the probability that the driver is male?

2. A bag contains two coins, one is a normal coin the other has two heads. A coin is taken out of the bag and tossed and lands with a heads up.

   (i) What is the probability that this is the fair coin?

   (ii) The coin is tossed a second time and again gives a head again. What is the probability that this is the fair coin?

3. One in 50,000 people have a particular type of cancer. There exists a test for this cancer; if someone has this cancer the test will give a positive result 99% of the time and a negative result 1% of the time (false negative); if someone does not have this cancer the test will give a negative result 95% of the time and a positive result 5% of the time (false positive).

   (i) A random individual has taken the test and has a positive result. What is the probability that he has the cancer?

   (ii) What would happen if a mass screening programme for this cancer was implemented using this test?

4. Prove Bayes' theorem.

6.2 Cryptanalysis by the Bayesian Opponent

Bayesian analysis was first used in the cryptanalysis of ciphers by Alan Turing during WWII, and between 1940 and 1941 he developed both Bayesian analysis and its application to cryptanalysis (Good, 1979, 2000). This use of Bayesian analysis was not published and seems not to have directly influenced the development of statistics in the wider world. Other workers such as Wald (1944) who were not working in such a secretive field were left to independently develop similar approaches applied to other field. Turing, with the help of I. J. Good, who was his main statistical assistant, developed a range of other statistical methods for use in cryptanalysis, some of which reached a wider audience (Good, 1953). Turing died in 1954, but Good went on to become a major force in the development of the Bayesian approach to statistics.

6.2.1 Deterministic Decision Functions

Definition 6.1 A Deterministic Decision Function
A deterministic decision function δ is a sequence of *n* transformations $\{\delta^{(n)}\}$, one for each of the *n*-gram approximations of the text:

$$\delta = \delta^{(n)} : Z_{m,n} \to Z_{m,n} \quad n = 1,2,3...$$
$$\delta = \delta^{(n)} : y \to \delta^{(n)}(y)$$

where $Z_{m,n}$ is the set of *n*-grams with all *n* letters from Z_m

Definition 6.2 The Family of Deterministic Decision Functions
Define Δ as the family of deterministic decision functions.

For a given ciphertext *y*, the opponent's decision $\delta(y)$ will be the same for all pairs of plaintext *x* and key *k* that satisfies $y = T(x,k)$. A decision function $\delta(y)$ will produce an incorrect estimate of the plaintext for all pairs of *x* and *k* that satisfy:

$$\delta(y) \neq x$$
$$\delta(T(x,k)) \neq x$$

Definition 6.3 Loss Function
A loss function L_δ can be defined

$$L_\delta(x,y) = \begin{cases} 1 & \text{if } \delta(y) \neq x \\ 0 & \text{if } \delta(y) = x \end{cases}$$

The loss function $L_\delta(x,T(x,k))$ will give a zero loss to a correct decision and a loss of one to an incorrect decision.

The average loss $I_{n,\delta}$ of the deterministic function δ on *n*-grams is the expectation of the random variable $L_\delta(X,Y)$ with respect to the joint probability distribution of the *n*-gram pairs (*X*,*Y*).

$$I_{n,\delta} = E\{L_\delta(x,y)\} = \sum_{x,y \in Z_{m,n}} P(x,y) L_\delta(x,y)$$

Definition 6.4 Optimal Deterministic Decision Function
A deterministic decision function δ' is optimal if

$$I_{n,\delta'} \le I_{n,\delta} \quad \forall \delta \in \Delta, \forall n \in N$$

Definition 6.5 Bayesian Deterministic Decision Functions
A deterministic decision function δ_B is Bayesian if

$$P(\delta_B(y)/y) = \max_{x \in Z_{m,n}} (P(x/y))$$
$$\delta_B(y) = x \text{ only if } P(x/y) = \max_{x \in Z_{m,n}} (P(x/y))$$

The decision function $\delta_B(y)$ is defined for ciphertext *y* that satisfy $P(y) > 0$

Note that there may be more than one Bayesian decision function.

6.2.2 Properties of Deterministic Decision Functions

Theorem 6.1 A deterministic decision function δ is optimal iff it is Bayesian.

Proof
The expected loss of δ on *n*-grams is

$$\begin{aligned} I_{n,\delta} &= E\{L_\delta(x,y)\} \\ &= \sum_{x,y \in Z_{m,n}} P(x,y) L_\delta(x,y) \\ &= \sum_{x,y \in Z_{m,n}} P(y) P(x/y) L_\delta(x,y) \\ &= \sum_{y \in Z_{m,n}} P(y) \sum_{x \in Z_{m,n}} P(x/y) L_\delta(x,y) \end{aligned}$$

Now from Definition 6.3, $L_\delta(x,y) = 1$ iff $\delta(y) \neq x$

$$I_{n,\delta} = \sum_{y \in Z_{m,n}} P(y) \sum_{\substack{x \in Z_{m,n} \\ \text{with } \delta(y) \neq x}} P(x/y) L_\delta(x,y)$$
$$= \sum_{y \in Z_{m,n}} P(y) \sum_{\substack{x \in Z_{m,n} \\ \text{with } \delta(y) \neq x}} P(x/y)$$

Using the fact that probabilities sum to 1

$$I_{n,\delta} = \sum_{y \in Z_{m,n}} P(y) \left(1 - \sum_{x \in Z_{m,n} \text{ with } \delta(y) = x} P(x/y) \right)$$
$$= \sum_{y \in Z_{m,n}} P(y) \left(1 - P(\delta^n(y)/y) \right)$$
$$= \sum_{y \in Z_{m,n}} P(y) - \sum_{y \in Z_{m,n}} P(y) P(\delta^n(y)/y)$$

IF Case

The decision function δ is optimal if $I_{n,\delta}$ is a minimum (from Definition 6.4).

Now, $I_{n,\delta}$ will be a minimum when

$$\sum_{y \in Z_{m,n}} P(y) P(\delta^n(y)/y) \quad \text{is a maximum.}$$

Since, $P(y)$ is independent of $P(\delta^n(y)/y)$ this will be when

$$\sum_{y \in Z_{m,n}} P(\delta^n(y)/y) \text{ is a maximum for all } n\text{-grams of } y.$$

which will be when

$$P(\delta^n(y)/y) = \max_{x \in Z_{m,n}} \left(P(x/y) \right).$$

Therefore, it is Bayesian.

Only If Case

Now, from Definition 6.5, δ is Bayesian if

$$P(\delta(y)/y) = \max_{x \in Z_{m,n}} \left(P(x/y) \right)$$
$$\delta(y) = x \text{ only if } P(x/y) = \max_{x \in Z_{m,n}} \left(P(x/y) \right)$$

Hence, $\sum_{y \in Z_{m,n}} \max_{x \in Z_{m,n}} \left(P(x/y) \right) = \sum_{y \in Z_{m,n}} P(\delta^n(y)/y)$ is a maximum over all n-grams of y.

Since, $P(y)$ is independent of $P(\delta^n(y)/y)$ this will be when

$$\sum_{y \in Z_{m,n}} P(y)P(\delta^n(y)/y) \quad \text{is a maximum.}$$

Which will make $I_{n,\delta}$ a minimum.

Therefore, δ is optimal.

QED.

Example 6.1

A plaintext x = "switchradiofrequency" is enciphered using the Shift cipher with k = 17, giving the ciphertext y = "jnzktyiruzfwivhlvetp".

Table 6.1 lists the conditional probabilities $P(x/\text{JNZKTY})$ for all plaintext 6-grams.

Trial Key	**Candidate Plaintext**	**1-gram Source (Definition 1.3)**	**2-gram Source (Definition 1.4)**	**Markov Source (Definition 1.5)**
k	***x = y-k***	δ_B	δ_B	δ_B
1	IMYJSX	0.0000184908	0.0000000000	0.0000000000
2	HLXIRW	0.0018728672	0.0001041171	0.0000000000
3	GKWHQV	0.0000084968	0.0000000000	0.0000000000
4	FJVGPU	0.0000128209	0.0000000000	0.0000000000
5	EIUFOT	0.1982319583	0.0582251272	0.0092158755
6	DHTENS	**0.6712556817**	0.1098981653	0.1365403054
7	CGSDMR	0.0110109485	0.0000000000	0.0000000000
*8	BFRCLQ	0.0001335152	0.0000000266	0.0000000349
9	AEQBKP	0.0000888545	0.0000000000	0.0000000000
10	ZDPAJO	0.0000233168	0.0001523178	0.0000015296
11	YCOZIN	0.0008221755	0.0035232504	0.0141513866
12	XBNYHM	0.0002458563	0.0000000000	0.0000000000
13	WAMXGL	0.0002859070	0.0000000000	0.0000000000
14	VZLWFK	0.0000064915	0.0000000000	0.0000000000
15	UYKVEJ	0.0000227984	0.0000000000	0.0000000000
16	TXJUDI	0.0000697960	0.0000000000	0.0000000000
17	**SWITCH**	0.0678027125	**0.7443798521**	**0.6092668497**
18	RVHSBG	0.0036375906	0.0000178996	0.0000000088
19	QUGRAF	0.0003471186	0.0571690723	0.2159648691
20	PTFQZE	0.0000245593	0.0000028319	0.0000000000
21	OSEPYD	0.0396330589	0.0241474700	0.0139689763
22	NRDOXC	0.0030692745	0.0016610339	0.0008238823
23	MQCNWB	0.0000799441	0.0000000000	0.0000000000
24	LPBMVA	0.0010898358	0.0000720453	0.0000000112
25	KOALUZ	0.0002021462	0.0006467905	0.0000662706
26	JNZKTY	0.0000037841	0.0000000000	0.0000000000

Table 6.1 The Conditional Probability of Plaintext, Given Ciphertext "JNZKTY"

Under the assumption that all keys are equally likely to have been used to encipher the plaintext $P(k)=1/26$

And the only *x* that can be enciphered into JNZKTY are those that satisfy

$$x = y - k \bmod m \qquad \forall k \in Z_m$$

So,

$$P(x,y)=\begin{cases}(1/26)P(y-k) & \text{if } x = y\text{-}k \bmod m \text{ for some } k \in Z_m \\ 0 & \text{otherwise}\end{cases}$$

and

$$\begin{aligned}P(y) &= \sum_x P(x,y) \\ &= \sum_{0\le i\le 25} P(y-i,y) \\ &= \sum_{0\le i\le 25} (1/26)P(y-i)\end{aligned}$$

giving

$$\begin{aligned}P(y-k/y) &= P(x/y)) \\ &= \frac{P(x,y)}{P(y)} \\ &= \frac{(1/26)P(y-k)}{\sum_{0\le i\le 25}(1/26)P(y-i)} \\ &= \frac{P(y-k)}{\sum_{0\le i\le 25}P(y-i)}\end{aligned}$$

As can be seen from Table 6.1, with the 1-gram model the Bayesian decision function returns the incorrect plaintext "DHTENS". The second choice of the decision function is "EIUFOT", while the correct plaintext "SWITCH" is only the third choice.

This is due to the simplistic nature of the model, which gets misled by the high frequency letters (E,T,N,S,H) and (E,T,I) in choices 1 and 2 respectively. The correct plaintext has the high frequency letters (T,I,S,H), which are sufficient to make it a clear third choice but no better.

When the more sophisticated 2-gram and Markov models are used the decision function returns the correct plaintext "SWITCH".

Plaintext Source Model	δ_B (JNZKTY)	k
1-gram	DHTENS	6
2-gram	SWITCH	17
Markov	SWITCH	17

Table 6.2 The Plaintext Recovered by the Bayesian Decision Function

In the case of three plaintext source models the Bayesian decision function becomes more accurate for longer *n*-grams. Table 6.3 shows the results for the Bayesian decision function δ_B(JNZKTYIRUZFW). All three models now identify the correct plaintext.

Trial Key	Candidate Plaintext	1-gram Source (Definition 1.3)	2-gram Source (Definition 1.4)	Markov Source (Definition 1.5)
k	$x = y\text{-}k$	δ_B	δ_B	δ_B
1	IMYJSXHQTYEV	0.0000011098	0.0000000000	0.0000000000
2	HLXIRWGPSXDU	0.0000350045	0.0000000000	0.0000000000
3	GKWHQVFORWCT	0.0000208471	0.0000000000	0.0000000000
4	FJVGPUENQVBS	0.0000005304	0.0000000000	0.0000000000
5	EIUFOTDMPUAR	0.2168125007	0.0001783388	0.0000005089
6	DHTENSCLOTZQ	0.0021832583	0.0000000000	0.0000000000
7	CGSDMRBKNSYP	0.0006225138	0.0000000000	0.0000000000
*8	BFRCLQAJMRXO	0.0000009934	0.0000000000	0.0000000000
9	AEQBKPZILQWN	0.0000001650	0.0000000000	0.0000000000
10	ZDPAJOYHKPVM	0.0000003010	0.0000000000	0.0000000000
11	YCOZINXGJOUL	0.0000011048	0.0000000024	0.0000000034
12	XBNYHMWFINTK	0.0001489158	0.0000000000	0.0000000000
13	WAMXGLVEHMSJ	0.0000176353	0.0000000000	0.0000000000
14	VZLWFKUDGLRI	0.0000114898	0.0000000000	0.0000000000
15	UYKVEJTCFKQH	0.0000002359	0.0000000000	0.0000000000
16	TXJUDISBEJPG	0.0000014139	0.0000000000	0.0000000000
17	**SWITCHRADIOF**	**0.7796793285**	**0.9998216363**	**0.9999994873**
18	RVHSBGQZCHNE	0.0000237800	0.0000000000	0.0000000000
19	QUGRAFPYBGMD	0.0000149607	0.0000000058	0.0000000004
20	PTFQZEOXAFLC	0.0000028677	0.0000000000	0.0000000000
21	OSEPYDNWZEKB	0.0003021930	0.0000000166	0.0000000000
22	NRDOXCMVYDJA	0.0000238015	0.0000000001	0.0000000000
23	MQCNWBLUXCIZ	0.0000001195	0.0000000000	0.0000000000
24	LPBMVAKTWBHY	0.0000946625	0.0000000000	0.0000000000
25	KOALUZJSVAGX	0.0000001701	0.0000000000	0.0000000000
26	JNZKTYIRUZFW	0.0000000967	0.0000000000	0.0000000000

Table 6.3 The Conditional Probability of Plaintext, Given Ciphertext "JNZKTYIRUZFW"

Exercise 6.2

1. A plaintext has been enciphered using the Shift cipher, giving the ciphertext "ODCTHXMXHIWTIPGVTIPG". Use the Bayesian opponent method to discover the plaintext and the key used. Use the following plaintext models.

   (i) 1-gram model with $n = 4$.
   (ii) 1-gram model with $n = 8$.
   (iii) 2-gram model with $n = 4$.
   (iv) Markov model with $n = 4$.

6.3 Good-Turing Frequency Estimation Method

There are many situations where it useful to know the probability that a word (or *n*-gram) will occur in a plaintext. However, when taking a sample of words from some language there will be gaps due to words not occurring in the sample. This will lead to incorrect estimates for the probabilities of words occurring in a plaintext.

Example 6.1

Consider a bag that contains an unknown number of balls. All the balls are coloured, but the number of colours is unknown. A sample of 50 balls are drawn from the bag, one at a time, and replaced. In the sample, 7 balls are red, 5 balls are blue, 5 are green, 4 are violet, 4 are turquoise, 3 are indigo, 3 are bronze, 3 are copper, 2 are yellow, 2 are grey, 2 purple, 2 are gold, 2 are silver, and there are 1 of each of the following orange, brown, lime, magenta, cyan and pink.

Given this information, the normal approach used to estimate the probability of a particular coloured ball appearing is to divide the number of times a colour occurs in the sample by the sample size (this is the Maximum Likelihood Estimate - MLE). This gives the probabilities of the colours as:
red = 0.14,
blue = green = 0.10, violet = turquoise = 0.08,
indigo = bronze = copper = 0.06,
yellow = grey = purple = gold = silver = 0.04,
orange = brown = lime = magenta = cyan = pink = 0.02.

However, it also gives the probabilities of black, white, etc., as zero. That is, all colours which have not been seen are predicted to never occur. Of course, the probabilities derived from any sample are never expected to be perfect, but the predictions for balls that have yet to be seen are particularly inaccurate.

The probabilities of 1-grams in Table 1.1 and 2-grams in Table 1.2 have been calculated this way. This does not cause a problem for the 1-grams presented since the sample size used was very large and all letters occurred many times in the sample. However, despite the size of the sample some of the 2-grams never occurred, which may be due to the fact that they are very rare and did not occur in the sample or that, they don't exist in English and never will appear.

If an analysis, similar to that given in Example 6.1, was conducted on a sample of English texts it would give a probability of zero for all words not occurring in the sample, which is clearly not realistic. However, in Chapter 7 a frequency analysis of *Treasure Island* is given, which has a sample size of 70,419 words with 5,868 unique words in the book. Figure 7.1 shows the relationship between a word's rank and its frequency, and fitting a line to the data and extrapolating it implies that words not in the book will occur with lower, but non zero, probabilities in a predictable way.

Turing (Good, 1953) came up with a way of estimating the probabilities for words that have yet to be seen, and also of improving the estimate for a word that had already been seen. Turing was breaking the Enigma cipher by using a Bayesian method to discover the daily key. However, before he could start this analysis it was

necessary to uncover a second key randomly selected by the operator – one for each message. This second key was unknown to the receiver so it was encoded by a much simpler second cipher and sent as part of the message. It was this second key setting and the simple cipher that Turing was conducting a frequency analysis of, not the German language itself (Good, 2000).

6.3.1 Turing's Method

Let N be the sample size of words (or depending on the situation: 2-grams, 3-grams, or code groups, etc.), and n_r be the number of different words seen exactly r times in the sample, so that

$$N = \sum_{r=1}^{\infty} r n_r$$

Note that the sample gives us the values $n_1, n_2, \ldots$, but not n_0. Also note that this sum is finite, since for most vales of r (and all values of r after a certain point) n_r will be zero.

Then the population frequency p_r of a word that appear r times in the sample is given by.

$$p_r = \frac{r^*}{N}$$

Where, r^* is the estimate for the number of words that would occur in a perfect sample that actually occur r times in the sample provided.

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

so,

$$p_r = \frac{(r+1)}{N}\frac{n_{r+1}}{n_r}$$

It follows from this that

$$p_0 = \frac{n_1}{N}$$

Example 6.2

Applying Turing's method to the data from Example 6.1 gives the results in Table 6.4. From this it can be seen that the probability of a ball occurring that has not been seen before (such as white or black) is now given as 0.12. It can also be seen that the probabilities for most of the other colours have been reduced (because some of the probability distribution is now used for the unseen colours). It can also be seen that

there are some problems introduced by the method, i.e., the probability for the red ball and the probability for those balls occurring 6 times.

Number of times seen	Number of types of ball seen *r* times	Names of the colours seen *r* times	Probability of a colour appearing *r* times
r	n_r		$p_r = \frac{(r+1)}{N}\frac{n_{r+1}}{n_r}$
0	-	???	$\frac{n_1}{N} = \frac{6}{50} = 0.120$
1	6	orange, brown, lime, magenta, cyan, pink	$\frac{(1+1)}{50}\frac{5}{6} = \frac{10}{300} = 0.033$
2	5	yellow, grey, purple, gold, silver	$\frac{(2+1)}{50}\frac{3}{5} = \frac{9}{250} = 0.036$
3	3	indigo, bronze, copper	$\frac{(3+1)}{50}\frac{2}{3} = \frac{8}{150} = 0.053$
4	2	violet, turquoise	$\frac{(4+1)}{50}\frac{2}{2} = \frac{10}{100} = 0.100$
5	2	blue, green	$\frac{(5+1)}{50}\frac{0}{2} = 0$
6	0	-	$\frac{(6+1)}{50}\frac{1}{0} = \infty$
7	1	red	$\frac{(7+1)}{50}\frac{0}{1} = 0$
8	0	-	$\frac{(8+1)}{50}\frac{0}{0} = 0$
9	0	-	

Table 6.4 Turing's Bayesian Estimation Method

6.3.2 Good's Refinement

Good (1953) improved on Turing's equation for *r** by introducing a smoothing function *S*, (Good proposed several different types of smoothing functions that could be used).

$$r^* = (r+1)\frac{S(n_{r+1})}{S(n_r)}$$

A range of other properties can be estimated using this approach, such as the variance of word frequencies, and the probability that all words of a particular frequency in a language will occur in a sample (Good, 1953; Good & Toulmin, 1956).

The introduction of a smoothing function, made the technique more robust and accurate, but significantly increased the amount of computation required. Less

computationally intensive versions of the test have since been developed (Gale & Sampson, 1995).

Exercise 6.3

1. Interception of 44 coded military call signs has collected the following codewords: oak 6 times, ash 5 times, yew 5 times, alder 4 times, birch 3 times, beach 3 times; spruce, juniper, larch and pine twice each; and willow, laylandii, holly, elm, aspen, hawthorn, sycamore, maple, rowan and chestnut once each.

   (i) For each codeword use the Maximum Likelihood Estimate to calculate the probability that it will be the next one received.

   What is the probability of a new call sign being received?

   (ii) Use the Good-Turning method recalculate the answer to part (i)

6.4 References

Degroot, Morris H., (1986) "*Probability and Statistics*", 2nd Edition, Addison Wesley Publishing, London.

Gale, W. A., and Sampson, G., (1995) "*Good–Turing Frequency Estimation Without Tears*", Journal of Quantitative Linguistics, vol. 2, pp. 217–37.

Good, I. J., (1953), "*The Population Frequencies of Species and the Estimation of Population Parameters*", Biometrika, vol. 40: 16, pp. 237-264.

Good, I. J., and Toulmin, G. H., (1956), "*The Number of New Species, and the Increase of Population Coverage, when a Sample is Increased*", Biometrika, vol. 43: 16, pp. 45-63.

Good, I. J., (1979), "*Studies in the History of Probability and Statistics. XXXVII: A. M. Turing's Statistical Work in World War II*", Biometrika, vol. 66: 2, pp. 393-6.

Good, I. J., (2000), "*Turing's anticipation of Empirical Bayes in Connection with the Cryptanalysis of the Naval Enigma*", Journal of Statistical Computation and Simulation, vol. 66:2, pp. 101-112.

Wald, A. (1944). "*On Cumulative Sums of Random Variables*". Ann. Math. Statistics. vol. 15, pp. 283-96.

Chapter 7 – Unicity and Entropy

7.1 Unicity and Entropy

7.1.1 Unicity Distance

One approach to breaking a cipher is to apply every possible decryption key until the plaintext appears. For some ciphers this is possible to do by hand, for others this is feasible to do by computer and for others it will take longer than the lifetime of the universe even when using all the computers in the world. In principle, all ciphers can be attacked in this way to recover the plaintext message given sufficient time. However, this raises the possibility that for some decryption key a plaintext will be recovered that appears to makes sense, but which is not the original one.

Claude Shannon (1949) introduced the unicity distance as a measure of how much ciphertext is required to ensure that the decrypted text can be recognised as the original plaintext.

If D is the redundancy of the plaintext in bits and $H(k)$ is the entropy of the key space of the key k, then the unicity distance is defined by:

$$U = \frac{H(k)}{D}$$

Shannon (1948) had already introduced the concept of entropy into the field of communications (sometimes called Shannon entropy to distinguish it from entropy in physics, although they are related concepts). Entropy can be thought of as a measure of the unpredictability or randomness of a message and hence the information that it carries. For example, a two headed coin will always give heads, so there is no randomness and it is completely predictable and another flip of the coin provides no more information – this has entropy of 0. On the other hand, a fair coin will give a heads or a tails with equal likelihood, it is completely random and unpredictable and each flip of the coin provides new information – this has entropy of 1. So, in a message if the next character is completely predictable based on the characters already known then it provides no extra information.

Based on Boltzmann's definition of entropy in thermodynamics, Shannon (1948), defined entropy for a random variable X that can take on the values $(x_1, x_2, ..., x_n)$ as:

$$H(X) = -\sum_{i=1}^{n} P(x_i)\log_2(P(x_i))$$

7.1.2 Redundancy

Normal English text consists of more characters than are needed to read the text and understand its meaning. In other words English text contains redundant information. For example, the "u" in "qu" is redundant since "q" is always followed by "u". The following sentence has had 29% of the characters deleted from it.

"ths sntnc cntns n vwls nly cnsnnts"

As an experiment, Shannon gave the following text, in which all vowels and spaces have been deleted, to subjects and asked them to reconstruct the original.

"fctsstrngrthnfctn"

Despite the 41% deletion rate, six subjects reconstructed 93% of the deletions and several subjects reconstructed the full text

Shannon (1951) estimates that printed English (of 100 characters length) is 75% redundant, that is, 75% of the characters in a message could be deleted and it would still be possible to extract the meaning. This is not to say that 75% of the characters in a message could be selected at random and thrown away, but rather, that with the right choices 75% of the characters could be deleted without loss of meaning.

7.1.3 Zipf's Law

Based on empirical results, the American linguist George Zipf proposed the following relationship between the frequency f_n of a word in a language and its rank order *n* in the language.

$$f_n = \frac{c}{n}$$

Where *c* is a constant that depend on the language source (Zipf, 1932, 1949).

An analysis of the 70,419 words from Robert Louis Stevenson's *Treasure Island* (Stevenson, 1883), shows that there are 5,868 unique words in the book. Of these, "the" is the most common word (ie, it ranks 1st) and it occurs 4,412 times, which is 6.3% of the entire book. The other four most common word are, "and" (2nd, 2,889 times, 4.1%); "I" (3rd, 1968 times, 2.8%); "a" (4th, 1,756 times, 2.5%); "of" (5th, 1,685 times, 2.4%). So these 5 words account for 18.1% of the book. Figure 7.1 gives a plot of word relative frequency against word rank for the 2,000 most common words in the book.

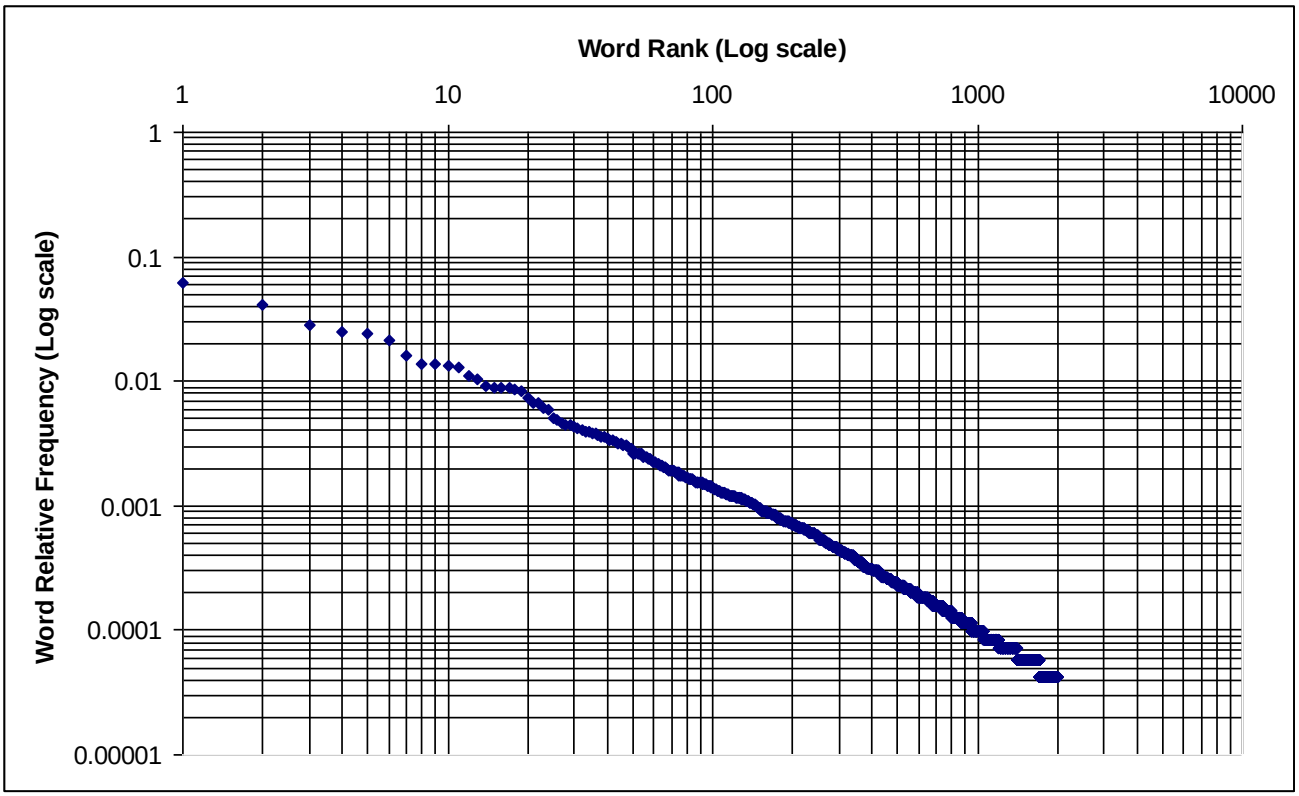

Figure 7.1 Word Relative Frequency against Word Rank, *Treasure Island* (Stevenson, 1883)

Exercise 7.1

1. If a PC can test a decryption key in a billionth of a second and you have a million PCs at your disposal, how long, on average, will it take using a brute force search of the key space to decrypt the following?

   (i) A General Substitution cipher.
   (ii) A Vigenère cipher with a key of length 10.
   (iii) A Vigenère cipher with a key of length 25.
   (iv) An Autokey cipher with a key of length 10.

   Assume that $m = 26$ in all cases.

2. Using the formula for entropy verify the following.

   (i) For a toss of a fair coin $H(X) = 1$.
   (ii) For a toss of a two headed coin $H(X) = 0$.

3. What is the entropy of a roll of a die?

4. What is the entropy of the toss of two fair coins? Hint, enumerate all the cases.

5. What is the entropy of a roll of two dice?

6. What is the entropy of a fair coin tossed four times?

7. A letter is picked at random from a large book written in English. What is the entropy of this process?

8. One copy of each letter of the alphabet is put into a bag and one letter is draw at random. What is the entropy of this process?

9. Based on the answers two questions 8 and 7, estimate the redundancy of English.

10. Use the answer to question 9 to estimate the Unicity distance for the Caesar cipher.

7.2 Properties of the Entropy Function

The entropy function H has some useful properties that can be used to simplify calculations.

Theorem 7.1

For independent random variables X_1 and X_2, ie, $P(x_i x_j) = P(x_i)P(x_j)$.

$$H(X_1, X_2) = H(X_1) + H(X_2)$$

Proof

$$H(X_1, X_2) = -\sum_{i,j=1}^{n,m} P(x_i x_j) \log_2(P(x_i x_j))$$

Using the independence of the two events gives

$$\begin{aligned}
&= -\sum_{i,j=1}^{n,m} P(x_i)P(x_j) \log_2(P(x_i)P(x_j)) \\
&= -\sum_{i,j=1}^{n,m} P(x_i)P(x_j)(\log_2(P(x_i)) + \log_2(P(x_j))) \\
&= -\sum_{i,j=1}^{n,m} (P(x_i)P(x_j) \log_2(P(x_i)) + P(x_i)P(x_j) \log_2(P(x_j))) \\
&= -\sum_{i,j=1}^{n,m} P(x_i)P(x_j) \log_2(P(x_i)) - \sum_{i,j=1}^{n,m} P(x_i)P(x_j) \log_2(P(x_j)) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)P(x_j) \log_2(P(x_i)) - \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)P(x_j) \log_2(P(x_j)) \\
&= -\sum_{i=1}^{n} P(x_i) \log_2(P(x_i)) \sum_{j=1}^{m} P(x_j) - \sum_{j=1}^{m} P(x_j) \log_2(P(x_j)) \sum_{i=1}^{n} P(x_i)
\end{aligned}$$

Using the fact that probabilities sum to 1 gives

$$\begin{aligned}
&= -\sum_{i=1}^{n} P(x_i) \log_2(P(x_i)) - \sum_{i=1}^{n} P(x_j) \log_2(P(x_j)) \\
&= H(X_1) + H(X_2)
\end{aligned}$$

QED.

Theorem 7.2

For $X_1, X_2, \ldots, X_n$ independent and identically distributed (i.i.d.) random variables.

$$H(X_1, X_2, ..., X_n) = nH(X_1)$$

Proof

Since the random variables are independent, repeated application of Theorem 7.1 gives:

$$H(X_1, X_2, ..., X_n) = H(X_1) + H(X_2) + ... + H(X_n)$$

As $X_1, X_2, \ldots, X_n$ are identically distributed:

$$H(X_1) = H(X_2) = ... = H(X_n)$$

Therefore,

$$H(X_1, X_2, ..., X_n) = H(X_1) + H(X_1) + ... + H(X_1)$$
$$= nH(X_1)$$

QED.

7.3 Plaintext Source Entropy

7.3.1 Entropy of the 1-gram Source (see Definition 1.3)

Consider the entropy of a plaintext source that is *n* characters long, which is generated by 1-gram independent and identical trials, that is,

$$P\{(x_1, x_2, ..., x_n)\} = P(x_1)P(x_2)...P(x_n).$$

This has entropy:

$$H(X) = H((X_1, X_2, ..., X_n))$$
$$= nH(X_1) \quad \text{by Theorem 7.2}$$

therefore,

$$= -n\sum_{i=1}^{m} P(x_i)\log_2(P(x_i))$$

using the probabilities from Table 1.1 gives:

$$= 4.166n \quad \text{bits/message}$$

or in other words it has an entropy value of 4.166 bits per character.

If the probabilities of all 26 characters occurring were equal then the entropy per character would be:

$$H(X) = -\sum_{i=1}^{26} \tfrac{1}{26}\log_2(\tfrac{1}{26})$$
$$= \log_2(26)$$
$$= 4.700 \quad \text{bits/character}$$

Therefore, the redundancy for the 1-gram source is:

$$D = 4.700 - 4.166$$
$$= 0.534 \quad \text{bits/character}$$

7.3.2 Entropy of the 2-gram Source (see Definition 1.4)

Consider the entropy of a plaintext source that is 2*n* characters long, which is generated by 2-gram independent and identical trials, that is,

$$P\{(x_1, x_2, x_3,..., x_{2n})\} = P(x_1, x_2)P(x_3x_4)...P(x_{2n-1}, x_{2n})$$

This has entropy

$$H(X) = H((X_1, X_2,..., X_n))$$

Where $H(X_i)$ is the entropy for the 2-gram (x_{2i-1},x_{2i})

$$H(X) = nH(X_1) \text{ by Theorem 7.2}$$

Summing over all $m \times m$ values of the 2-gram gives

$$= -n\sum_{i,j=1}^{m} P(x_i x_j)\log_2(P(x_i x_j))$$

using the probabilities from Table 1.2 gives:

$$= 7.407n \quad \text{bits/message}$$

or in other words it has an entropy value of 3.703 bits per character.

Therefore, the redundancy per character for the 2-gram source is

$$D = 4.700 - 3.703$$
$$= 0.997 \quad \text{bits/character}$$

7.3.3 Entropy of the Markov Chain Source (see Definition 1.5)

Consider the entropy of a plaintext source that is *n* characters long, which is generated (1-gram at a time) by a Markov chain with a transition matrix *P* and equilibrium distribution π, that is,

$$P\{(x_1, x_2, x_3,..., x_n)\} = \pi(x_1)P(x_2 / x_1)P(x_3 / x_2)...P(x_n / x_{n-1})$$

This has entropy

$$H_n = H(X) = H((X_1, X_2,..., X_n))$$
$$= -\sum P\{(x_1, x_2, x_3,..., x_n)\}\log_2(P\{(x_1, x_2, x_3,..., x_n)\})$$

For $n = 1$

$$H_1 = -\sum P\{(x_1)\}\log_2\left(P\{(x_1)\}\right)$$

$$= -\sum_{i=1}^{m} P(x_i)\log_2 P(x_i)$$

$$= -\sum_{i=1}^{m} \pi(x_i)\log_2 \pi(x_i)$$

Using the probabilities from Table 1.4 gives:

$H_1 = 4.071$ bits/character

Therefore, the redundancy per character for the Markov source, using the H_1 approximation is:

$$D = 4.700 - 4.071$$
$$= 0.629 \quad \text{bits/character}$$

For $n = 2$

$$H_2 = -\sum P\{(x_1, x_2)\}\log_2\left(P\{(x_1, x_2)\}\right)$$

$$= -\sum_{i,j=1}^{m} P(x_i, x_j)\log_2\left(P(x_i, x_j)\right)$$

$$= -\sum_{i,j=1}^{m} \pi(x_i)P(x_j / x_i)\log_2\left(\pi(x_i)P(x_j / x_i)\right)$$

using the probabilities from Tables 1.3 and 1.4 gives:

$H_2 = 7.411$ bits/message (2 characters long)

or in other words it has an entropy value of 3.705 bits per character.

Therefore, the redundancy per character for the Markov source, using the H_2 approximation is:

$$D = 4.700 - 3.705$$
$$= 0.995 \quad \text{bits/character}$$

Table 7.1 shows how the entropy per character converges for large n for the Markov source.

n	H_n/n
1	4.071
2	3.705
3	3.583
4	3.522
5	3.486
…	…
∞	3.340

Table 7.1 The Entropy per Character for a Markov Source

7.3.4 Entropy of English

Shannon (1948) initial used the statistical properties of up to eight letters, ie, 8-grams, to estimate the entropy of English as 2.3 bits per character, giving a redundancy of 2.4 bits/character or approximately 50%. In doing this, Shannon had to make several approximations as he did not have available any statistics on *n*-grams larger than 3-grams and even here they were not in the most useful form.

From Figure 7.1 it can be seen that words in English can be modelled by the equation

$$P_n = \frac{0.1}{n}$$

as predicted by Zipf.

Unfortunately,

$$\sum_{1}^{\infty} \frac{0.1}{n} = \infty$$

So Shannon summed the series to the number N until the probability equalled one.

$$\sum_{1}^{N} \frac{0.1}{n} = 1$$

This gave a value of N = 8,727, which gives an entropy of

$$\begin{aligned} H &= \sum_{1}^{8,727} P_n \log_2(P_n) \\ &= \sum_{1}^{8,727} \frac{0.1}{n} \log_2\left(\frac{0.1}{n}\right) \\ &= 11.82 \quad \text{bits/word} \end{aligned}$$

Since the average word length of English is 4.5 letters,

H = 11.82/4.5 = 2.63 bits/character.

Shannon (1951) used an experiment in which humans predicted the next letter in a sentence to produce tables that enabled him to estimate the probabilities of *n*-grams up to 100-gram. This information was then used to estimate the entropy of English as 0.987 bit/character, giving a redundancy of 3.713 bit/character or approximately 75%.

This means that the unicity distance for a code can be calculated against English and also against a range of models of the plaintext source.

The unicity distance does not give information on how easy it is to break a cipher, but rather on how much text is necessary to be able to uniquely recognise the plaintext after it has been deciphers. In practice, this recognition may well be against one of the plaintext sources (i.e., when conducted by a computer program) and not against English (i.e., when conducted by a human cryptanalyst).

Exercise 7.2

1. What is the redundancy in bits per character for a long plaintext from a Markov source?

2. For $m = 26$ and $r = 20$, calculate the entropy in bits for the General Substitution cipher, General Vigenère cipher, Autokey cipher, General Permutation cipher and the Hill cipher.

3. Construct a table of unicity distances for ciphers against plaintext sources. Use the ciphers from question 2 as the rows and use the following plaintext sources, 1-gram, 2-gram, Markov (use the value in the limit as n goes to infinity) and English (use Shannon's estimate of D =3.713) as the columns.

4. For a message length of 20 characters, calculate the One Time Pad's unicity distance. To which of the ciphers in question 3 does this match? Compare these two ciphers.

   (i) Does having the same unicity distance mean that the ciphers are equivalent in strength for these parameters settings?
   (ii) What happens as the parameters vary?

5. For a random variable with two possible outcomes, which occur with probabilities p and q, show that the entropy is maximised when $p = q$.

7.4 Perfect Secrecy

A cipher has perfect secrecy if a cryptanalyst with infinity computational power can determine nothing about the plaintext from the ciphertext with the possible exception of its length. Shannon reasoned that if receiving a copy of the ciphertext *y* provided no more information about the plaintext *x* than not receiving it, the cryptanalyst is in the same position as not receiving the ciphertext *y*. In which case, the cryptanalyst is in a position equivalent to guessing what the plaintext message is before it is enciphered (or even before it is written). That is, the *a posteriori* probability of the message occurring is the same as the *a priori* probability of it occurring.

Definition 7.1 Perfect Secrecy
A cipher T: $y = T(x)$ has perfect secrecy iff

$$P(x/y) = P(x) \qquad \forall x, y$$

where,

$P(x)$ - is the *a priori* probability of plaintext *x* occurring.

$P(x/y)$ - is the *a posteriori* probability that *x* is the plaintext given ciphertext *y* has been intercepted.

Theorem 7.3 (Shannon)
A necessary and sufficient condition for prefect secrecy is that

$$P(y/x) = P(y) \qquad \forall x, y$$

where,

$P(y)$ - is the probability of ciphertext *y* occurring under any circumstance. That is, from the encipherment of any plaintext under any key.

$P(y/x)$ - is the conditional probability of ciphertext *y* occurring given that the plaintext is *x*. This is the sum of all the probabilities of all the keys such that *x* is enciphered to *y*.

Proof

From Definition 7.1, a cipher *T* has perfect secrecy iff

$$P(x/y) = P(x) \qquad \forall x, y$$

Bayes' theorem gives:

$$P(x/y) = \frac{P(y/x)P(x)}{P(y)}$$

Therefore,

$$P(x) = \frac{P(y/x)P(x)}{P(y)} \qquad \forall x, y$$

For this to hold either

$$P(x) = 0$$

(which can be excluded as it does not apply for all x)

or

$$P(y/x) = P(y) \qquad \forall x, y$$

Conversely, if

$$P(y/x) = P(y) \qquad \forall x, y$$

substitution into Bayes' theorem gives,

$$P(x/y) = \frac{P(y)P(x)}{P(y)} \qquad \forall x, y$$

therefore,

$$P(x/y) = P(x) \qquad \forall x, y$$

QED.

Theorem 7.4 (Shannon) - The One Time Pad has perfect secrecy
The One Time Pad cipher uses an encipherment key k to encipher the plaintext n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext n-gram $y = (y_1, y_2, y_3, ..., y_n)$, using:

$$y = T(x, k)$$

where, $\quad k = (k_1, ..., k_i, ..., k_n) \qquad k_i \in Z_m, \;\; 1 \le i \le n$

and, $\quad T: \quad y_i = x_i + k_i \mod m \qquad 1 \le i \le n$

where, the set of k_i consists of random variables that are i.i.d. from a uniform distribution. Note that the key $k = (k_1, k_2, k_3, ..., k_n)$ is an n-gram, ie, the keyword is the same length as the plaintext message.

For the OTP, each k_i is a random variable that can take on any value from the alphabet *A* of size *m* and with equal probability, ie,

$$P(k_i = A_1) = P(k_i = A_2) = ... = P(k_i = A_m) = \frac{1}{m}$$

So, probability of the key is:

$$P(k) = \left(\frac{1}{m}\right)^n = m^{-n}$$

From Bayes' theorem

$$\begin{aligned} P(y/x) &= \frac{P(x/y)P(y)}{\sum_{k \in K} P(x/y)P(y)} \\ &= \frac{P(T_k^{-1}(y)/y)P(y)}{\sum_{k \in K} P(T_k^{-1}(y)/y)P(y)} \\ &\frac{P((y-k \bmod m)/y)P(y)}{\sum_{k \in K} P((y-k \bmod m)/y)P(y)} \\ &= \frac{m^{-n}P(y)}{\sum_{k \in K} m^{-n}P(y)} \\ &= \frac{P(y)}{\sum_{k \in K} P(y)} \\ &= P(y) \end{aligned}$$

So by Theorem 7.3 the OTP has perfect secrecy

QED.

7.5 References

Shannon, C.E., (1948), "*A Mathematical Theory of Communication*", Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October.

Shannon, C.E., (1949), "*Communication Theory of Secrecy Systems*", Bell System Technical Journal, vol. 28, pp. 658-715.

Shannon, C.E., (1951), "*Prediction and Entropy of Printed English*", Bell System Technical Journal, vol. 30.

Stevenson, R.L., (1883), "*Treasure Island*", Cassell & Company, London.

Zipf, G.K., (1932), "*Selective Studies and the Principle of Relative Frequency in Language"*, Harvard University Press.

Zipf, G.K., (1949), "*Human Behaviour and the Principle of Least Effort"*, Addison-Wesley Press, London.

Chapter 8 – Modern Ciphers

8.1 Principles of Cipher Design

The security of the first ciphers depended upon the secrecy of the method and the inability of the opponent to bring cryptanalytic techniques to bear on the cipher. Once elementary cryptanalysis was in common use cryptographers tried to make their cipher invulnerable to this by making their ciphers more complex. However, complexity on its own does not guarantee a cipher's security, but could easily make it too difficult to use reliably.

8.1.1 Kerckhoffs' Principles

The first person to expound principles for a good cryptographic system was Auguste Kerckhoffs in *La Cryptographie Militaire* (1883, p12). In this article he gave the following six criteria that are required of a cipher system to be used under military conditions:

1. The system must be practically, if not mathematically, indecipherable;

2. The system must not rely on secrecy, and will not cause inconvenience if it falls into the hands of the enemy;

3. The key must be communicated and retained without the use of written notes, and be changed or modified by the correspondents;

4. It must be applicable to Telegraphic communication;

5. The system must be portable, and its usage and function must not require the assistance of several people;

6. The system must be simple to use, with no long series of calculations.

8.1.2 Shannon's Criteria

Shannon (1949) gave the following five criteria to assess the value of a cipher system:

1. Amount of Secrecy
   "There are some systems that are perfect—the enemy is no better off after intercepting any amount of material than before. Other systems, although giving him some information, do not yield a unique "solution" to intercepted cryptograms. Among the uniquely solvable systems, there are wide variations in the amount of labor [*sic*] required to effect this solution and in the amount of material that must be intercepted to make the solution unique."

2. Size of Key
   "The key must be transmitted by non-interceptible [*sic*] means from transmitting to receiving points. Sometimes it must be memorized. It is therefore desirable to have the key as small as possible."

3. Complexity of Encipherment and Decipherment Operations
   "Enciphering and deciphering should, of course, be as simple as possible. If they are done manually, complexity leads to loss of time, errors, etc. If done mechanically, complexity leads to large expensive machines."

4. Propagation of Errors
   "In certain types of ciphers an error of one letter in enciphering or transmission leads to a large number of errors in the deciphered text. The errors are spread out by the deciphering operation, causing the loss of much information and frequent need for repetition of the cryptogram. It is naturally desirable to minimize this error expansion."

5. Expansion of Messages
   "In some types of secrecy systems the size of the message is increased by the enciphering process. This undesirable effect may be seen in systems where one attempts to swamp out message statistics by the addition of many nulls, or where multiple substitutes are used. It also occurs in many "concealment" types of systems (which are not usually secrecy systems in the sense of our definition)."

Shannon observed that these five criteria are mutually exclusive, and also that relaxing any one of them allowed the other four to be achieved.

8.2 Product Ciphers

An obvious way to make a message more secure is to encrypt it using more than one cipher. This idea goes back to the superencipherment of codes, which was still in use in WWII. Plaintext words were first replaced by their equivalent codes words, and then the code words were enciphered, usually with a relatively simple cipher.

However, encrypting a plaintext twice using the same cipher using two different keys does not necessarily give much, if any, extra security. This could even reduce the level of security.

Example 8.1

Consider the encipherment of a message x using a Shift cipher with a key k followed by a Shift cipher using a key $-k$.

$$y = T(T(x,k),-k)$$

therefore,

$$y_i = ((x_i + k) \bmod m - k) \bmod m \qquad 1 \le i \le n$$

$$y_i = (x_i + k - k) \bmod m \qquad 1 \le i \le n$$

$$y_i = x_i \bmod m \qquad 1 \le i \le n$$

so,

$$y = x$$

So, clearly, in the worst case double encryption can reduce security.
The theory behind the modern use of multiple ciphers in combination can be traced back to Claude Shannon's groundbreaking paper "*Communication Theory of Secrecy Systems*" (Shannon, 1949).

Exercise 8.1

1. Show that double encryption by the following ciphers using two independent keys is equivalent to encryption by the same cipher with a third key. Hence there is no extra security.
   (i) Shift cipher.
   (ii) Affine cipher.

8.2.1 Group Properties of Ciphers

Some ciphers can be shown to be groups. A group $(G,\circ)$ is a set G with an associated operator $\circ$, that satisfies the four group axioms, namely, closure, associativity, identity element, and inverse element.

The cipher,

$$y = T(x, e_k)$$

can be considered as the set of ciphers T with elements T_k, $T_{k'}$, $T_{k''}$, $T_{k'''}$, ... identified by the keys $k,\ k',\ k'',\ k''',\ ... \in K$, where,

$$T(x, e_k) \equiv T_{e_k}$$

For T to be a group, it needs to satisfy the following axioms.

Closure
For all, T_k, $T_{k'} \in T$

$$T_k \circ T_{k'} \in T$$

Associativity
For all, T_k, $T_{k'}$, $T_{k''} \in T$

$$T_k \circ (T_{k'} \circ T_{k''}) = (T_k \circ T_{k'}) \circ T_{k''}$$

Identity element
There exists an identity element $T_1 \in T$, such that, for all $T_k \in T$

$$T_1 \circ T_k = T_k \circ T_1 = T_k$$

Inverse element
For all, $T_k \in T$ there exists an inverse element $T_k^{-1} \in T$, such that,

$$T_k^{-1} \circ T_k = T_k \circ T_k^{-1} = T_1$$

8.2.2 Diffusion and Confusion

Shannon (1949) also introduced the concepts of *diffusion* and *confusion.*

Diffusion is the process whereby the statistical properties of the plaintext that result from its redundancy are stretched out or spread throughout the ciphertext. This means that more data has to be collected by the cryptanalyst and the statistical analysis will be more difficult to accomplish.

Confusion is the idea that the link between the cipher key and the ciphertext should be as complex as possible. This means that even given some partial knowledge about the key it will be difficult to use this to decrypt the ciphertext.

Shannon (1949) stated that a good cipher should have both diffusion and confusion properties. One way to achieve is by the use of a combination of substitution ciphers (confusion) and transposition ciphers (diffusion).

8.3 Stream Ciphers and Block Ciphers

Modern ciphers are often classified as either stream ciphers or block ciphers. A stream cipher is one which operates on the plaintext one letter (or symbol) at a time producing the corresponding ciphertext letter, before moving onto the next letter to be enciphered. Block ciphers operate on a block of several letters (or symbols) at the same time to produce the corresponding block of ciphertext.

Although the terms stream cipher and block cipher were introduced long after classic ciphers were invented all ciphers can be classified this way.

Examples of stream ciphers include: the Shift cipher, Vigenère cipher, Autokey cipher and Rotor ciphers such as the Enigma.

Examples of block ciphers include: the Hill cipher and Playfair cipher.

Note that the Playfair cipher has a block size of two, which is very small by modern standards. The Playfair cipher is usually regarded as a di-graph cipher, in other words, a cipher that operates on the 2-gram alphabet. Using the same reasoning any block cipher that operates on *n* symbols at a time can be regarded as a stream cipher operating on the *n*-gram alphabet.

Modern Stream and Block ciphers operate directly on bits instead of characters. Characters are represented by the American Standard Code for Information Interchange (ASCII), see Table 8.1.

Long strings of binary numbers can be hard to type-in without making errors and they are hard to read from documentation. To overcome this they are often represented as hexadecimal (hex) numbers – one hexadecimal character represents four binary bits, see Table 8.2. Note that computers will still operate in binary; hex is used merely for the convenience of humans.

ASCII	Binary	Char	ASCII	Binary	Char	ASCII	Binary	Char
32	00100000	space	64	01000000	@	96	01100000	`
33	00100001	!	65	01000001	A	97	01100001	a
34	00100010	"	66	01000010	B	98	01100010	b
35	00100011	#	67	01000011	C	99	01100011	c
36	00100100	$	68	01000100	D	100	01100100	d
37	00100101	%	69	01000101	E	101	01100101	e
38	00100110	&	70	01000110	F	102	01100110	f
39	00100111	'	71	01000111	G	103	01100111	g
40	00101000	(	72	01001000	H	104	01101000	h
41	00101001	)	73	01001001	I	105	01101001	i
42	00101010	*	74	01001010	J	106	01101010	j
43	00101011	+	75	01001011	K	107	01101011	k
44	00101100	,	76	01001100	L	108	01101100	l
45	00101101	-	77	01001101	M	109	01101101	m
46	00101110	.	78	01001110	N	110	01101110	n
47	00101111	/	79	01001111	O	111	01101111	o
48	00110000	0	80	01010000	P	112	01110000	p
49	00110001	1	81	01010001	Q	113	01110001	q
50	00110010	2	82	01010010	R	114	01110010	r
51	00110011	3	83	01010011	S	115	01110011	s
52	00110100	4	84	01010100	T	116	01110100	t
53	00110101	5	85	01010101	U	117	01110101	u
54	00110110	6	86	01010110	V	118	01110110	v
55	00110111	7	87	01010111	W	119	01110111	w
56	00111000	8	88	01011000	X	120	01111000	x
57	00111001	9	89	01011001	Y	121	01111001	y
58	00111010	:	90	01011010	Z	122	01111010	z
59	00111011	;	91	01011011	[	123	01111011	{
60	00111100	<	92	01011100	\	124	01111100	|
61	00111101	=	93	01011101	]	125	01111101	}
62	00111110	>	94	01011110	^	126	01111110	~
63	00111111	?	95	01011111	_	127	01111111	
Note, ASCII codes 0-31 (not shown) and 127 are non printable codes								

Table 8.1 The American Standard Code for Information Interchange - ASCII

Decimal	Binary	Hex	Decimal	Binary	Hex
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

Table 8.2 Binary Values and Corresponding Hex Values

Exercise 8.2

1. Show that under repeated encipherment the Shift cipher forms a group.

2. Show that the Affine cipher forms a group.

3. Show that the General Substitution cipher forms a group.

4. What other ciphers have this property and which do not?

5. Convert the following hex numbers to binary.
   (i) AB
   (ii) E9F
   (iii) ABCD

6. Convert the following binary numbers to hex.
   (i) 11010111
   (ii) 01011101
   (iii) 1001111011111011

8.4 References

Kerckhoffs, A., (1883), "*La Cryptographie Militaire*", Journal des Sciences Militaires, Vol. IX, pp. 5-38.

Chapter 9 – Stream Ciphers

Most modern stream ciphers are based on the principles of the Vernam cipher (Vernam, 1926). The Vernam cipher relies on a random key that is as long as the message, and provided the key is used only once gives perfect security. However, due to the difficulty in producing and securely transporting the keys this system is impractical in most situations.

Consequently, attempts were made to develop mathematical algorithm that generated a key that appeared to be random. The hope was that the algorithm would produce numbers "random enough" to prevent cryptanalysis of the ciphertext.

9.1 Pseudo Random Numbers

A sequence of numbers $R_1, R_2, R_3, \ldots$ generated by any algorithm cannot actually be random, since, if you know the algorithm you can predict the next number with complete accuracy. An algorithm that produces numbers that appear to be random is called a Pseudo Random Number Generator (PRNG).

Depending on the amount of storage that is allocated to hold a single number, there will be a maximum of different numbers *M* that can be represented. Today, in most computers the CPU allocates 32 bits called a "word" to hold a single number. Therefore, a word can represent $M = 2^{32} = 4{,}294{,}967{,}296$ different numbers, typically, from 0 to $2^{32}-1$. As a consequence, any algorithm that uses the current number to generate the next number in a sequence must eventually repeat in a cycle; the cycle length is called the period with the maximum possible period being *M*.

Note that some computers use 64 bits per word, and also that it is possible via software to have a larger size word on any computer (although this comes at the expense of execution time). This will increase the size of *M*, but the sequence will still repeat with a maximum period of *M*. Similarly, some algorithms will use two (or more) numbers in the sequence to generate the next pseudo random number, but the sequence will still repeat eventually with a maximum period of M^2.

There are two statistical properties of real random numbers that we a pseudo random number generator needs to exhibit. These are Uniformity and Independence.

Uniformity
Uniformity means that all numbers are equally likely to occur. Therefore, the probability of a random number occurring in any sub-interval of (0, *M*-1) will be proportional to the relative size of that sub-interval.

Independence
Independence means that the next random number is not related to the previous random numbers. Therefore, the probability of a random number being in a specific interval is independent of all previous random numbers.

9.1.1 Design Principles for Pseudo-Random Numbers Generation

Pseudo-random number generators (PRNGs) are used in a range of applications in addition to cryptology, such as, simulation, gambling and random sampling. These are general requirements for good PRNGs:

1. The algorithm should produce a sequence of pseudo-random numbers with a very long period. For modern cryptographic purposes the period should be at least a few billion.

2. The algorithm should be scalable. That is sub-streams (ie, subsets of the numbers) should also be pseudo-random.

3. The algorithm should be consistent with respect to its start value (seed).

4. The algorithm should produce numbers that pass statistical tests for uniformity.

5. The algorithm should produce numbers that pass statistical tests for independence.

6. The algorithm should produce numbers that are unpredictable. That is, given a sequence of outputs from the algorithm the next number should not be predictable.

7. The algorithm should be very fast.

8. The algorithm should be easy to implement in a range of programming languages and computer hardware.

9. The algorithm should be parallelisable. That is, it should be possible to use it to generate separate PRN streams on a parallel computer (or networked computers).

9.1.2 Algorithms for Generating Pseudo Random Numbers

There are a wide range of algorithms for generating PRNs. Many of these do not fully meet the design principles given above. The Linear Congruential Algorithm is the most widely used PRNG in general use, although not cryptographic use. Another important PRNG is the Linear Feedback Shift Register algorithm.

9.1.3 The Linear Congruential Generator

D. H. Lehmer was the first to propose the Linear Congruential Generator (LCG) (Lehmer, 1949). The LCG uses the following algorithm to generate a sequence of integers $R_1, R_2, R_3, \ldots$ in the range 0 to M-1:

$$R_i = (aR_{i-1} + c) \bmod M \quad 1 \leq i$$

where,

a is the constant multiplier,
c is the increment,
M is the modulus,
and R_0 is the initial value, which is called the seed;

Note that, in the case, $c = 0$, the algorithm reduces to the multiplicative congruential generator.

Exercise 9.1

1. Consider the PRNG given by $R_{i+1} = i \mod M \quad 0 \leq i$ with $M = 8$. Does this meet the Uniformity and or the Independence criteria?

2. The first pseudo random number generator was proposed by John von Neumann in the early 1940s. The algorithm is as follows:

   (1) take a 4-digit number;
   (2) square it;
   (3) if necessary, add zeros to the left of the number to make it 8-digits long;
   (4) extract the middle 4-digits of the number;
   (5) divide this by 10,000 to form a random number from the uniform distribution;
   (6) take the 4-digit number from step (4) and go to step (2).

   Use this generator, starting from the initial seed of 7182 to produce a sequence of 20 numbers.

3. What is the period for the linear congruential generator with $M = 23$, $a = 0$, $c = 2$ and $R_0 = 1$?

4. For the linear congruential generator with $M = 15$, $a = 4$, $c = 1$ and $R_0 = 3$, generate the sequence R_i of random integers and calculate the period.

5. For the linear congruential generator with $M = 16$, by experimentation find values of *a* and *c* that ensure a maximum cycle length. Generate the sequence R_i of random integers.

9.1.4 Theory Behind The Linear Congruential Generator

The values chosen for *a*, *c* and *M* will dictate the statistical properties such as the mean, variance, and cycle length.

- The numbers generated from the example can only assume values from the set $I = \{0, 1, 2, ..., (M\text{-}1)\}$. If *M* is very large, it is less of a problem. Values of $M = 2^{31} - 1$ and $M = 2^{48}$ are in common use.

- To achieve maximum density for a given range, proper choice of *a*, *c*, and *M* is very important. Maximal period can be achieved by some proven selection of these values.

- For *M* a power of 2, i.e. $M = 2^b$, $b \neq 0$, the longest possible period is $P = M = 2^b$, when *c* is relatively prime to *M* and $a = 1 + 4k$ where *k* is an integer.

- For *M* a power of 2, i.e. $M = 2^b$, and $c = 0$, the longest possible period $P = M/4 = 2^{b-2}$, when R_0 is odd and the multiplier, *a* is given by $a = 3 + 8k$ or $a = 5 + 8k$ where *k* is an integer.

- For *M* a prime number and $c = 0$, the longest possible period is $P = M - 1$ when *a* satisfies the property that the smallest *k* such that $a^k - 1$ is divisible by *M* is $k = M - 1$.

For example, we choose $M = 7$ and $a = 3$, the above conditions satisfy. Here *k* has to be 6.

- when k = 6, $a^k - 1 = 728$ which is divisible by *M*
- when k = 5, $a^k - 1 = 242$ which is not divisible by *M*
- when k = 4, $a^k - 1 = 80$ which is not divisible by *M*
- when k = 3, $a^k - 1 = 26$ which is not divisible by *M*

Of course, the longest possible period here is 6, which is of no practical use. But the example shows how the conditions can be checked.

Definition 9.1 The LCG Stream Cipher
The Linear Congruential Generator Stream cipher is a polyalphabetic cipher, it enciphers the plaintext *n*-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext *n*-gram $y = (y_1, y_2, y_3, ..., y_n)$ using an encipherment key $e_k = (a, c, R_0, M)$ where:

$$y_i = x_i + R_i \mod m \quad 1 \le i \le n$$

where,

$$R_i = aR_{i-1} + c \mod M \quad 1 \le i \le n$$

Decipherment using decipherment key d_k is accomplished by:

$$x_i = y_i - R_i \mod m \quad 1 \le i \le n$$

The LCG Stream cipher has M^3 keys for any particular value of M.

Exercise 9.2

1. Use the LCG Stream cipher with values $M = 16$, $a = 5$, $c = 3$ and $R_0 = 7$ to encipher "This is not really a one time pad".

2. Decipher the output of question 1.

9.2 Statistical Tests for Pseudo Random Numbers Generators

Although a PRNG may have good theoretical properties, such as a long period, this does not guarantee that the numbers that it produces will be satisfactory. Before a PSRNG is used it should pass a series of statistical tests. For example, the US National Institute of Standards and Technology specifies that a PRNG should pass a battery of 15 tests (NIST SP 800-22, 2010).

When devising a PRNG the two properties of true random numbers that it is aimed to emulate are uniformity and independence.

9.2.1 Tests for Uniformity

The *null hypothesis* H_0, is that numbers drawn from the PRNG are uniformly distributed. The *alternative hypothesis* H_a, is that numbers drawn from the PRNG are not uniformly distributed.

Then based on some statistical test we either *reject* H_0 or *fail to reject* H_0.

If, based on a statistical test of a sample of numbers from a PRNG, we *reject* H_0 when it is in fact uniform this is a *Type I Error*. On the other hand, if we *fail to reject* H_0 when the PRNG is not uniform this is a *Type II Error*.

The level of statistical significance α for a test is the probability of rejecting H_0 when it is true, in other words the probability of making a *Type I Error*.

$$\alpha = P(\text{rejecting } H_0 | H_0 \text{ is true})$$

The smaller the value of α the less likely it will be that a *Type I Error* will be made, however, this will increase the chance that a *Type II Error* will be made.

9.2.2 Frequency test

This test compares the distribution of a sample of numbers drawn from the PRNG against the theoretical uniform distribution. It does this by counting the frequencies of pseudo random numbers in various classes against the frequencies expected from the uniform distribution.

There are two common ways of implementing a frequency test the chi-square test and the Kolmogorov-Smirnov test.

9.2.3 Chi-squared Test

The chi-squared test compares the observed frequencies O_i in a set of n classes with the frequencies to be expected E_i from the uniform distribution. It uses the test statistic:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

If the test statistic χ^2 is greater than the critical value χ^2_α for a particular level of confidence α then H_0 is rejected. Table 9.1 gives the critical values for the χ^2 distribution. For the test to be reliable all the expected frequencies need to be 5 or greater.

Note that, if all *n* classes are the same size and the sample size is *N*.

$$E_i = \frac{N}{n}$$

This would give (*n*-1) degrees of freedom.

Critical Values for the χ^2 distribution									
Degrees of Freedom	**Percentage points of the distribution**								
	99	**98**	**95**	**90**	**10**	**5**	**2**	**1**	**0.1**
1	0.00016	0.00063	0.0039	0.016	2.71	3.84	5.41	6.64	10.83
2	0.02	0.04	0.10	0.21	4.60	5.99	7.82	9.21	13.82
3	0.12	0.18	0.35	0.58	6.25	7.82	9.84	11.34	16.27
4	0.30	0.43	0.71	1.06	7.78	9.49	11.67	13.28	18.46
5	0.55	0.75	1.14	1.61	9.24	11.07	13.39	15.09	20.52
6	0.87	1.13	1.64	2.20	10.64	12.59	15.03	16.81	22.46
7	1.24	1.56	2.17	2.83	12.02	14.07	16.62	18.48	24.32
8	1.65	2.03	2.73	3.49	13.36	15.51	18.17	20.09	26.12
9	2.09	2.53	3.32	4.17	14.68	16.92	19.68	21.67	27.88
10	2.56	3.06	3.94	4.86	15.99	18.31	21.16	23.21	29.59
11	3.05	3.61	4.58	5.58	17.28	19.68	22.62	24.72	31.26
12	3.57	4.18	5.23	6.30	18.55	21.03	24.05	26.22	32.91
13	4.11	4.76	5.89	7.04	19.81	22.36	25.47	27.69	34.53
14	4.66	5.37	6.57	7.79	21.06	23.68	26.87	29.14	36.12
15	5.23	5.98	7.26	8.55	22.31	25.00	28.26	30.58	37.70
16	5.81	6.61	7.96	9.31	23.54	26.30	29.63	32.00	39.29
17	6.41	7.26	8.67	10.08	24.77	27.59	31.00	33.41	40.75
18	7.02	7.91	9.39	10.86	25.99	28.87	32.35	34.80	42.31
19	7.63	8.57	10.12	11.65	27.20	30.14	33.69	36.19	43.82
20	8.26	9.24	10.85	12.44	28.41	31.41	35.02	37.57	45.32
21	8.90	9.92	11.59	13.24	29.62	32.67	36.34	38.93	46.80
22	9.54	10.60	12.34	14.04	30.81	33.92	37.66	40.29	48.27
23	10.20	11.29	13.09	14.85	32.01	35.17	38.97	41.64	49.73
24	10.86	11.99	13.85	15.66	33.20	36.42	40.27	42.98	51.18
25	11.52	12.70	14.61	16.47	34.38	37.65	41.57	44.31	52.62
26	12.20	13.41	15.38	17.29	35.56	38.88	42.86	45.64	54.05
27	12.88	14.12	16.15	18.11	36.74	40.11	44.14	46.96	55.48
28	13.56	14.85	16.93	18.94	37.92	41.34	45.42	48.28	56.89
29	14.26	15.57	17.71	19.77	39.09	42.56	46.69	49.59	58.30
30	14.95	16.31	18.49	20.60	40.26	43.77	47.96	50.89	59.70

Table 9.1 The Critical Values for the χ^2 distribution

Exercise 9.3

1. A new advance in random number generation based on Digital Indicator of Entropy (DIE) technology is used to generate 36 integers in the range 1 to 6:

   1, 1, 3, 5, 6, 3, 4, 2, 1, 3, 6, 5, 2, 2, 5, 3, 2, 5, 1, 4, 3, 2, 6, 6, 1, 4, 4, 5, 2, 3, 2, 5, 1, 3, 4, 6.

   Use the chi-squared test to test for uniformity. Test at the 5% significance level.

2. Use the Linear Congruential Generator with parameters M=16, a=7, c=3 and seed R_0=11 to generate 20 pseudo-random numbers.

3. Test the LCG in question 2 using the chi-squared test. Use 4 intervals of equal size. Test at the 0.1% significance level.

9.2.4 Pseudo-Random Numbers in the range (0, 1)

In many applications it useful to have the pseudo-random numbers defined as decimals in the range (0, 1) instead of integers in the range (0, *M*-1). This can be achieved by the use of:

$$U_i = \frac{R_i}{M}$$

Note that U_i can only take on *M* values, that is, $U_i \in (0, 1/M, 2/M, ..., (M-1)/M)$. For large values of *M* the gaps between the decimals becomes smaller.

9.2.5 The Kolmogorov-Smirnov test

This test compares the cumulative distribution function (cdf) of the uniform distribution *F*(*X*):

$$F(X) = x \quad 0 \le x \le 1$$

to the empirical cdf $S_N(X)$ of the sample of *N* observations, where

$$S_N(X) = \frac{\text{number of } U_1, U_2, ..., U_N \le x}{N}$$

The Kolmogorov-Smirnov test is based on the statistic $D = \max|F(X) - S_N(X)|$ that is, the absolute value of the differences between *F*(*x*) the cdf of the uniform distribution and $S_N(X)$ the empirical cdf.

As *N* becomes larger, $S_N(X)$ should be close to *F*(*X*).

Procedure

Sort the numbers U_i into ascending order then compute.

Compute $D = \max(D^+, D^-)$

where

$$D^+ = \max_{1 \le i \le N} \left| \frac{i}{N} - U_i \right|$$

$$D^- = \max_{1 \le i \le N} \left| U_i - \frac{i-1}{N} \right|$$

Here *D* is a random variable; its sampling distribution is tabulated in the Table 9.2.

The Kolmogorov-Smirnov test, tests for uniformity.

H_0: The U_i are uniform ($F(X)=S_N(X)$)

H_a: $F(X) \neq S_N(x)$

This is a one-tailed test.

Determine the critical value, D_α, from the Table 9.2 for the specified significance level α and the given sample size N.

If the sample statistic D is greater than the critical value D_α, the null hypothesis that the sample data is from a uniform distribution is rejected; if $D \leq D_\alpha$, then there is no evidence to reject it.

SAMPLE SIZE (N)	LEVEL OF SIGNIFICANCE FOR D = MAXIMUM [$F_n(X)$ - $S_n(X)$]				
	0.2	0.15	0.1	0.05	0.01
1	0.900	0.925	0.950	0.975	0.995
2	0.684	0.726	0.776	0.842	0.929
3	0.565	0.597	0.642	0.708	0.828
4	0.494	0.525	0.564	0.624	0.733
5	0.446	0.474	0.510	0.565	0.669
6	0.410	0.436	0.470	0.521	0.618
7	0.381	0.405	0.438	0.486	0.577
8	0.358	0.381	0.411	0.457	0.543
9	0.339	0.360	0.388	0.432	0.514
10	0.322	0.342	0.368	0.410	0.490
11	0.307	0.326	0.352	0.391	0.468
12	0.295	0.313	0.338	0.375	0.450
13	0.284	0.302	0.325	0.361	0.433
14	0.274	0.292	0.314	0.349	0.418
15	0.266	0.283	0.304	0.338	0.404
16	0.258	0.274	0.295	0.328	0.392
17	0.250	0.266	0.286	0.318	0.381
18	0.244	0.259	0.278	0.309	0.371
19	0.237	0.252	0.272	0.301	0.363
20	0.231	0.246	0.264	0.294	0.356
25	0.210	0.220	0.240	0.270	0.320
30	0.190	0.200	0.220	0.240	0.290
35	0.180	0.190	0.210	0.230	0.270
OVER 35	$\frac{1.07}{\sqrt{n}}$	$\frac{1.14}{\sqrt{n}}$	$\frac{1.22}{\sqrt{n}}$	$\frac{1.36}{\sqrt{n}}$	$\frac{1.63}{\sqrt{n}}$

Table 9.2 The Critical Values for the Kolmogorov-Smirnov test

Exercise 9.4

1. Use the LCG with parameters $M = 2^{16}$, $a = 59$, $c = 97$ and seed $R_0 = 7$ to generate 20 pseudo-random numbers from the uniform distribution.

   (i) Test the LCG using the chi-squared test. Use 4 intervals of equal size.
   (ii) Test the LCG using the chi-squared test. Use 10 intervals of equal size.
   (iii) Test the LCG using the Kolmogorov-Smirnov test.

   Test at the 1% significance level.

2. Use the LCG with parameters $M = 51$, $a = 36$, $c = 33$ and seed $R_0 = 48$ to generate 20 pseudo-random numbers from the uniform distribution.

   (i) Test the LCG using the chi-squared test. Use 4 intervals of equal size.
   (ii) Test the LCG using the chi-squared test. Use 10 intervals of equal size.
   (iii) Test the LCG using the Kolmogorov-Smirnov test.

   Test at the 1% significance level.

9.2.6 Tests for Independence

There are many test for independence that can be used to evaluate a PRNG, including:

1. Runs test
2. Gap test
3. Poker test
4. Autocorrelation test

9.2.7 Runs Tests

The runs test examines the arrangement of numbers in a sequence to test the hypothesis of independence. A run is defined as a succession of similar events followed by a different event.

E.g., in a sequence of tosses of a coin, we may have

H T T H H T T T H T

The first toss is preceded by and the last toss is followed by a "no event". This sequence has six runs, first with a length of one, second and third with length two, fourth length three, fifth and sixth length one.
A few features of a run

- Two characteristics: number of runs and the length of run.
- An up run is a sequence of numbers each of which is succeeded by a larger number; a down run is a sequence of numbers each of which is succeeded by a smaller number.

1. Total number of runs up and down

If a sequence of numbers have too few runs, it is unlikely to be a real random sequence. E.g. 8, 8, 23, 36, 42, 55, 63, 72, 89, 91, the sequence has one run, an up run. It is not likely that this is a random sequence.

If a sequence of numbers have too many runs, it is unlikely to be a real random sequence. E.g. 8, 93, 15, 96, 26, 84, 28, 79, 36, 57. It has nine runs, five up and four down. It is not likely that this is a random sequence.

If a is the total number of runs in a truly random sequence, the mean and variance of a is given by

$$\mu_a = \frac{2N-1}{3}$$

and

$$\sigma^2 = \frac{16N-29}{90}$$

For $N > 20$, the distribution of a is reasonably approximated by a normal distribution, $N(\mu_a, \sigma^2)$. Converting it to a standardised normal distribution by

$$Z_a = \frac{a - \mu_a}{\sigma_a}$$

that is

$$Z_a = \frac{a - \left(\frac{2N-1}{3}\right)}{\sqrt{\frac{16N-29}{90}}}$$

This is a two-tailed test.

Failure to reject the hypothesis of independence occurs when $-Z_{\alpha/2} \le Z_a \le Z_{\alpha/2}$, where α is the level of significance.

2. Runs above and below the mean.

The previous test for up runs and down runs is important. But it is not adequate to assure that the sequence is random.

Let n_1 and n_2 be the number of individual observations above and below the mean, let b the total number of runs of numbers above the mean.

For a given n_1 and n_2, the mean and variance of b can be expressed as

$$\mu_b = \frac{2n_1n_2}{N} + \frac{1}{2}$$

and

$$\sigma_b^2 = \frac{2n_1n_2(2n_1n_2 - N)}{N^2(N-1)}$$

For either n_1 or n_2, greater than 20, b is approximately normally distributed

$$Z_b = \frac{b - (2n_1n_2/N) - 1/2}{\left(\frac{2n_1n_2(2n_1n_2 - N)}{N^2(N-1)}\right)^{\frac{1}{2}}}$$

Failure to reject the hypothesis of independence occurs $-Z_{\alpha/2} \le Z_b \le Z_{\alpha/2}$, where α is the level of significance.

3. Runs test: length of runs up and down.

Example 9.1

0.16, 0.27, 0.58, 0.63, 0.45, 0.21, 0.72, 0.87, 0.27, 0.15, 0.92, 0.85, ...

If the same pattern continues, two numbers below average, two numbers above average, it is unlikely a random number sequence. But this sequence will pass other tests.

We need to test the randomness of the length of runs.

Let Y_i be the number of runs of length i in a sequence of N numbers. E.g. if the above sequence stopped at 12 numbers ($N = 12$), then $Y_1 = Y_3 = Y_4 = \ldots = Y_{11} = 0$ and $Y_2 = 6$

Obviously Y_i is a random variable. Among various runs, the expected value for runs up and down is given by

$$E(Y_i) = \frac{2}{(i+3)!}\left[N(i^2 + 3i + 1) - (i^3 + 3i^2 - i - 1)\right] \quad \text{for } i \le N - 2$$

and

$$E(Y_i) = \frac{2}{N!} \quad \text{for } i = N - 1$$

4. Runs test: length of runs above and below the mean.

The number of runs above and below the mean, also random variables, the expected value of Y_i is approximated by

$$E(Y_i) = \frac{Nw_i}{E(I)} \quad \text{for } N > 20$$

where $E(I)$ the approximate expected length of a run w_i is the approximate probability of length i. w_i is given by:

$$w_i = \left(\frac{n_1}{N}\right)^i \left(\frac{n_2}{N}\right) + \left(\frac{n_1}{N}\right)\left(\frac{n_2}{N}\right)^i$$

$E(I)$ is given by:

$$E(I) = \frac{n_1}{n_2} + \frac{n_2}{n_1} \qquad \text{for } N > 20$$

The approximate expected total number of runs (of all length) in a sequence of length N is given by

$$E(A) = \frac{N}{E(I)} \qquad \text{for} \quad N > 20$$

(total number divided by expected run length).

The appropriate test is the chi-squared test with O_i being the observed number of runs of length i

$$\chi^2 = \sum_{i=1}^{L} \frac{(O_i - E(Y_i))^2}{E(Y_i)}$$

where $L = N - 1$ for runs up and down, $L = N$ for runs above and below the mean.

Exercise 9.5

1. A Pseudo random number generator has produced the following numbers:

   17, 22, 25, 26, 30, 21, 18, 14, 13, 2, 1, 6, 9, 10, 1,
   2, 5, 6, 9, 10, 14, 21, 22, 25, 29, 5, 13, 18, 26, 29.

   Check whether the PRNG is producing independent numbers. Use the *total number of runs* version of the test.
   (i) Test at the 1% significance level.
   (ii) Test at the 5% significance level.

2. A Pseudo random number generator has produced the following numbers:

   17, 2, 22, 5, 25, 6, 26, 9, 30, 10, 21, 14, 18, 13, 14,
   2, 6, 1, 21, 9, 10, 1, 22, 5, 25, 29, 13, 26, 18, 29.

   Check whether the PRNG is producing independent numbers. Use the *total number of runs* version of the test.
   (i) Test at the 1% significance level.
   (ii) Test at the 5% significance level.

3. A Pseudo random number generator has produced the following numbers:

   2, 25, 8, 15, 14, 5, 20, 27, 26, 17, 0, 7, 6, 29, 12,
   19, 18, 9, 24, 31, 30, 21, 4, 11, 10, 1, 16, 23, 22, 13.

   Check whether the PRNG is producing independent numbers. Use the *total number of runs* version of the test.
   (i) Test at the 1% significance level.
   (ii) Test at the 5% significance level.

4. Check whether the PRNGs in questions 1, 2 and 3 are producing independent numbers. Use the *number of runs above and below the mean* version of the test. Test at both the 1% and 5% significance levels.
   (i) Check the numbers in question 1.
   (ii) Check the numbers in question 2.
   (iii) Check the numbers in question 3.

5. Check whether the PRNGs in questions 1, 2 and 3 are producing independent numbers. Use the *length of runs* version of the test and check the length of runs above and below the mean. Test at both the 1% and 5% significance levels.
   (i) Check the numbers in question 1.
   (ii) Check the numbers in question 2.
   (iii) Check the numbers in question 3.

Z	Phi(Z)	Z	Phi(Z)	Z	Phi(Z)	Z	Phi(Z)	Z	Phi(Z)	Z	Phi(Z)	Z	Phi(Z)
0.00	0.50000	0.50	0.69146	1.00	0.84134	1.50	0.93319	2.00	0.97725	2.50	0.99379	3.00	0.99865
0.01	0.50399	0.51	0.69497	1.01	0.84375	1.51	0.93448	2.01	0.97778	2.51	0.99396	3.01	0.99869
0.02	0.50798	0.52	0.69847	1.02	0.84614	1.52	0.93574	2.02	0.97831	2.52	0.99413	3.02	0.99874
0.03	0.51197	0.53	0.70194	1.03	0.84849	1.53	0.93699	2.03	0.97882	2.53	0.99430	3.03	0.99878
0.04	0.51595	0.54	0.70540	1.04	0.85083	1.54	0.93822	2.04	0.97932	2.54	0.99446	3.04	0.99882
0.05	0.51994	0.55	0.70884	1.05	0.85314	1.55	0.93943	2.05	0.97982	2.55	0.99461	3.05	0.99886
0.06	0.52392	0.56	0.71226	1.06	0.85543	1.56	0.94062	2.06	0.98030	2.56	0.99477	3.06	0.99889
0.07	0.52790	0.57	0.71566	1.07	0.85769	1.57	0.94179	2.07	0.98077	2.57	0.99492	3.07	0.99893
0.08	0.53188	0.58	0.71904	1.08	0.85993	1.58	0.94295	2.08	0.98124	2.58	0.99506	3.08	0.99896
0.09	0.53586	0.59	0.72240	1.09	0.86214	1.59	0.94408	2.09	0.98169	2.59	0.99520	3.09	0.99900
0.10	0.53983	0.60	0.72575	1.10	0.86433	1.60	0.94520	2.10	0.98214	2.60	0.99534	3.10	0.99903
0.11	0.54380	0.61	0.72907	1.11	0.86650	1.61	0.94630	2.11	0.98257	2.61	0.99547	3.11	0.99906
0.12	0.54776	0.62	0.73237	1.12	0.86864	1.62	0.94738	2.12	0.98300	2.62	0.99560	3.12	0.99910
0.13	0.55172	0.63	0.73565	1.13	0.87076	1.63	0.94845	2.13	0.98341	2.63	0.99573	3.13	0.99913
0.14	0.55567	0.64	0.73891	1.14	0.87286	1.64	0.94950	2.14	0.98382	2.64	0.99585	3.14	0.99916
0.15	0.55962	0.65	0.74215	1.15	0.87493	1.65	0.95053	2.15	0.98422	2.65	0.99598	3.15	0.99918
0.16	0.56356	0.66	0.74537	1.16	0.87698	1.66	0.95154	2.16	0.98461	2.66	0.99609	3.16	0.99921
0.17	0.56749	0.67	0.74857	1.17	0.87900	1.67	0.95254	2.17	0.98500	2.67	0.99621	3.17	0.99924
0.18	0.57142	0.68	0.75175	1.18	0.88100	1.68	0.95352	2.18	0.98537	2.68	0.99632	3.18	0.99926
0.19	0.57535	0.69	0.75490	1.19	0.88298	1.69	0.95449	2.19	0.98574	2.69	0.99643	3.19	0.99929
0.20	0.57926	0.70	0.75804	1.20	0.88493	1.70	0.95543	2.20	0.98610	2.70	0.99653	3.20	0.99931
0.21	0.58317	0.71	0.76115	1.21	0.88686	1.71	0.95637	2.21	0.98645	2.71	0.99664	3.21	0.99934
0.22	0.58706	0.72	0.76424	1.22	0.88877	1.72	0.95728	2.22	0.98679	2.72	0.99674	3.22	0.99936
0.23	0.59095	0.73	0.76730	1.23	0.89065	1.73	0.95818	2.23	0.98713	2.73	0.99683	3.23	0.99938
0.24	0.59483	0.74	0.77035	1.24	0.89251	1.74	0.95907	2.24	0.98745	2.74	0.99693	3.24	0.99940
0.25	0.59871	0.75	0.77337	1.25	0.89435	1.75	0.95994	2.25	0.98778	2.75	0.99702	3.25	0.99942
0.26	0.60257	0.76	0.77637	1.26	0.89617	1.76	0.96080	2.26	0.98809	2.76	0.99711	3.26	0.99944
0.27	0.60642	0.77	0.77935	1.27	0.89796	1.77	0.96164	2.27	0.98840	2.77	0.99720	3.27	0.99946
0.28	0.61026	0.78	0.78230	1.28	0.89973	1.78	0.96246	2.28	0.98870	2.78	0.99728	3.28	0.99948
0.29	0.61409	0.79	0.78524	1.29	0.90147	1.79	0.96327	2.29	0.98899	2.79	0.99736	3.29	0.99950
0.30	0.61791	0.80	0.78814	1.30	0.90320	1.80	0.96407	2.30	0.98928	2.80	0.99744	3.30	0.99952
0.31	0.62172	0.81	0.79103	1.31	0.90490	1.81	0.96485	2.31	0.98956	2.81	0.99752	3.31	0.99953
0.32	0.62552	0.82	0.79389	1.32	0.90658	1.82	0.96562	2.32	0.98983	2.82	0.99760	3.32	0.99955
0.33	0.62930	0.83	0.79673	1.33	0.90824	1.83	0.96638	2.33	0.99010	2.83	0.99767	3.33	0.99957
0.34	0.63307	0.84	0.79955	1.34	0.90988	1.84	0.96712	2.34	0.99036	2.84	0.99774	3.34	0.99958
0.35	0.63683	0.85	0.80234	1.35	0.91149	1.85	0.96784	2.35	0.99061	2.85	0.99781	3.35	0.99960
0.36	0.64058	0.86	0.80511	1.36	0.91308	1.86	0.96856	2.36	0.99086	2.86	0.99788	3.36	0.99961
0.37	0.64431	0.87	0.80785	1.37	0.91466	1.87	0.96926	2.37	0.99111	2.87	0.99795	3.37	0.99962
0.38	0.64803	0.88	0.81057	1.38	0.91621	1.88	0.96995	2.38	0.99134	2.88	0.99801	3.38	0.99964
0.39	0.65173	0.89	0.81327	1.39	0.91774	1.89	0.97062	2.39	0.99158	2.89	0.99807	3.39	0.99965
0.40	0.65542	0.90	0.81594	1.40	0.91924	1.90	0.97128	2.40	0.99180	2.90	0.99813	3.40	0.99966
0.41	0.65910	0.91	0.81859	1.41	0.92073	1.91	0.97193	2.41	0.99202	2.91	0.99819	3.41	0.99968
0.42	0.66276	0.92	0.82121	1.42	0.92220	1.92	0.97257	2.42	0.99224	2.92	0.99825	3.42	0.99969
0.43	0.66640	0.93	0.82381	1.43	0.92364	1.93	0.97320	2.43	0.99245	2.93	0.99831	3.43	0.99970
0.44	0.67003	0.94	0.82639	1.44	0.92507	1.94	0.97381	2.44	0.99266	2.94	0.99836	3.44	0.99971
0.45	0.67364	0.95	0.82894	1.45	0.92647	1.95	0.97441	2.45	0.99286	2.95	0.99841	3.45	0.99972
0.46	0.67724	0.96	0.83147	1.46	0.92785	1.96	0.97500	2.46	0.99305	2.96	0.99846	3.46	0.99973
0.47	0.68082	0.97	0.83398	1.47	0.92922	1.97	0.97558	2.47	0.99324	2.97	0.99851	3.47	0.99974
0.48	0.68439	0.98	0.83646	1.48	0.93056	1.98	0.97615	2.48	0.99343	2.98	0.99856	3.48	0.99975
0.49	0.68793	0.99	0.83891	1.49	0.93189	1.99	0.97670	2.49	0.99361	2.99	0.99861	3.49	0.99976

Table 9.3 The Standard Normal Distribution

9.2.8 Gap Test

The gap test is used to determine the significance of the interval between recurrences of the same digit.

A gap of length *x* occurs between the recurrence of some digit.

1, 4, 5, 1, 4, 5, 1, 2, 3, 4, 3, 4, 5, 4, 2, 1, 5, 5, 2, 3, 1, 4, 3, 4, 3, 5, 4, 3, 3, 5, 3, 2, 1, 2, 4, 2, 1, 5, 2, 3

There are a total of nine 4's in the list. Thus only eight gaps can occur.

The probability of a particular gap length can be determined by a Bernoulli trial.

$$P(\text{gap of } n) = P(x \neq 4)P(x \neq 4)\ldots P(x \neq 4)$$

If we are only concerned with digits between 1 and 5, then

$$P(\text{gap of } n) = 0.8^n 0.2$$

The theoretical frequency distribution for randomly ordered digits is given by

$$P(\text{gap} \leq x) = F(X) = 0.1\sum_{n=0}^{x} 0.9^n = 1 - 0.9^{x+1}$$

Steps involved in the test.

Step 1.
Specify the cdf for the theoretical frequency distribution based on the selected class interval width.

Step 2.
Arrange the observed sample of gaps in a cumulative distribution with these same classes.

Step 3.
Find *D*, the maximum deviation between $F(X)$ and $S_N(X)$.

Step 4.
Determine the critical value, D_α, from Table 9.2 for the specified value of α and the sample size *N*.

Step 5.
If the calculated value of *D* is greater than the tabulated value of D_α, the null hypothesis of independence is rejected.

9.2.9 Poker Test

The poker test for independence is based on the frequency in which certain digits are repeated in a series of numbers.

For example, 0.255, 0.577, 0.331, 0.414, 0.828, 0.909, 0.303, 0.001... In each case, a pair of like digits appears in the number.

In a three digit number, there are only three possibilities.

1. The individual digits can be all different. Case 1.
2. The individual digits can all be the same. Case 2.
3. There can be one pair of like digits. Case 3.

    - P(case 1) = P(second differ from the first) * P(third differ from the first and second) = 0.9 * 0.8 = 0.72

    - P(case 2) = P(second the same as the first) * P(third same as the first) = 0.1 * 0.1 = 0.01

    - P(case 3) = 1 - 0.72 - 0.01 = 0.27

9.2.10 Tests for Auto-correlation

The tests for auto-correlation are concerned with the dependence between numbers in a sequence.

The list of the 30 numbers: 1, 4, 2, 5, 75, 2, 1, 7, 6, 63, 4, 3, 5, 7, 89, 1, 5, 1, 9, 94, 8, 2, 0, 2, 67, 9, 5, 3, 4, 76 appears to have the property that every 5th number has a very large value. If this is a regular pattern, we can't really say the sequence is random.

The test computes the auto-correlation between every m numbers (*m* is also known as the lag) starting with the i^{th} number.

Thus the autocorrelation ρ_{im} between the following numbers would be of interest.

$$R_i, R_{i+m}, R_{i+2m}, \ldots, R_{i+(M+1)m}$$

The value *M* is the largest integer such that $i + (M + 1) \leq N$ where *N* is the total number of values in the sequence.

E.g. when $N = 17$, $i = 3$, $m = 4$, then the above sequence would be 3, 7, 11, 15 ($M = 2$). The reason we require $M+1$ instead of *M* is that we need to have at least two numbers to test ($M = 0$) the autocorrelation.

Since a non-zero autocorrelation implies a lack of independence, the following test is appropriate

$$H_0: \ \rho_{\text{im}} = 0$$
$$H_a: \ \rho_{\text{im}} \neq 0$$

For large values of M, the distribution of the estimator ρ_{im} , denoted as $\hat{\rho}_{im}$, is approximately normal if the values $R_i, R_{i+m}, R_{i+2m}, \ldots, R_{i+(M+1)m}$ are uncorrelated.

Form the test statistic

$$Z = \frac{\hat{\rho}_{im}}{\sigma_{\hat{\rho}_{im}}}$$

which is distributed normally with a mean of zero and a variance of one.

The formulas for $\hat{\rho}_{im}$ and the standard deviation are

$$\hat{\rho}_{im} = \frac{1}{M+1}\left[\sum_{k=0}^{M} R_{i+km} R_{(k+1)m}\right] - 0.25$$

and

$$\sigma_{\hat{\rho}_{im}} = \frac{\sqrt{13M+7}}{12(M+1)}$$

After computing Z_0 , do not reject the null hypothesis of independence if

$$-z_{\alpha/2} \leq Z \leq z_{\alpha/2}$$

where α is the level of significance.

Exercise 9.6

1. A PRNG has produced the following 100 numbers.

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

   Test the above PRNG for independence using the gap test:
   (i) Test the digit 4 at the 1% significance level and use intervals of size 5.
   (ii) Test the digit 7 at the 5% significance level and use intervals of size 5.
   (iii) Test the digit 0 at the 10% significance level and use intervals of size 4.

2. A PRNG has produced the following 30 3-digit numbers:

   771, 472, 484, 319, 108, 077, 204, 108, 715, 006,
   679, 276, 002, 116, 057, 413, 816, 459, 712, 486,
   599, 380, 009, 547, 058, 039, 463, 903, 121, 247.

   Test the above PRNG for independence using the poker test. Test at the 5% significance level.

3. A PRNG has produced the following 40 3-digit numbers:

   668, 138, 609, 829, 486, 097, 591, 630, 764, 783,
   217, 774, 365, 272, 100, 942, 158, 061, 004, 509,
   503, 717, 840, 545, 425, 760, 124, 481, 217, 109,
   746, 350, 025, 064, 866, 393, 832, 420, 437, 840.

   Test the above PRNG for independence using the poker test. Test at the 1% significance level.

9.3 Linear Feedback Shift Register Ciphers

Another major class of Stream ciphers are the Linear Feedback Shift Register ciphers. These are based on Linear Feedback Shift Register circuits, a common piece of electronics equipment that has many other applications outside of cryptography (for example, counters and scramblers).

Linear Feedback Shift Registers were invented at Bell Laboratories in 1965 to add randomness to a signal (Fracassi & Tammaru, 1981). This was not to make the signal secret, but rather to ensure that the signal was evenly spread over the frequencies that it was to be transmitted over to minimise interference.

9.3.1 Shift Register

A register is a basic computer circuit used to store a binary number. It consists of a series of switches called flip-flop circuits, each one of which is used to represent one bit of the binary number. Figure 9.1 gives a representation of a shift register, which is storing the decimal number 67 in binary. The most significant bit (msb) and the least significant bit (lsb) are shown.

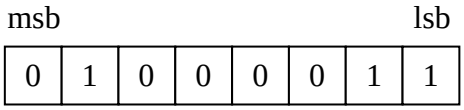

Figure 9.1 Representation of an 8-bit Register Holding the Value 67_{10}

A shift register is a register that will, when sent a signal, move the values held in each switch to the switch to its left (as seen in Figure 9.2). The lsb is replaced by zero while the msb is thrown away (or used for other purposes). This, in effect, results in the multiplication of the number held in the shift register by 2 with no carry, which is the same as multiplying by 2 and taking the modulus of the word size.

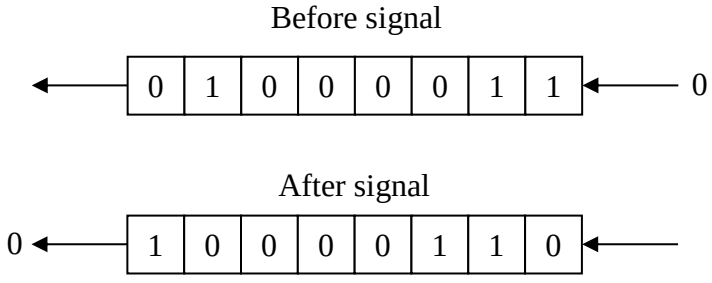

Figure 9.2 Operation of a Shift Register

Note that there is another version of the shift register, which moves the bits in the opposite direction - to the right. In this case, it is the lsb that is discarded. This is equivalent to dividing the number held in the register by 2 with no remainder.

An important feature of these circuits is that the control signal is sent to all parts of the register circuit at the same time, so that they all operate in parallel and no flip-flop circuit is left waiting for another one.

9.3.2 Linear Feedback Shift Register

A Linear Feedback Shift Register (LFSR) is a shift register which takes the values held by some of its switches via connectors called taps and adds these together bitwise. The resulting value is used as the input value to the LFSR (instead of the value zero as in an ordinary shift register).

Figure 9.3 shows an LFSR with taps on switches 1, 3, 5, and 8. The LFSR is currently holding the value 42_{10}. After the LFSR has been signalled to update it will take the values from the switches numbered 1, 3, 5 and 8 and add them together bitwise generating the value 0, which will replace the lsb when all the registers values are shifted to the left. The resulting number will be 84_{10}.

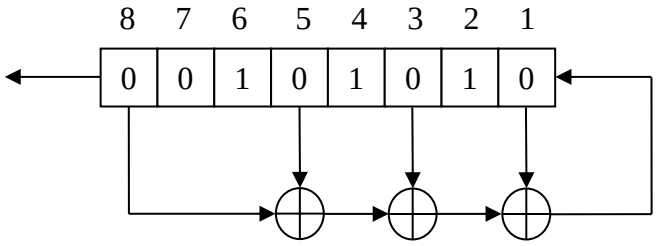

Figure 9.3 An 8-bit Linear Feedback Shift Register

Note that, if the register in Figure 9.3 holds the value 0 then the bitwise addition of the taps will produce a 0 to be input to the lsb, so when the register is updated the next value that it holds will also be 0.

An important feature of an LSFR is the period of the sequence of numbers that it produces. For an *n*-bit register there are 2^n states that it can be in (corresponding to the different number that it can hold). Since the next state of the register is completely determined by the current state, the maximum possible period is at most 2^n. However, as noted for the LSFR in Figure 9.3, if an LSFR hold the value 0 then, irrespective of where the taps are, the next input bit to the lsb will be a sum of zeros mod 2, which leaves the register unchanged. So, if a LSFR starts at 0 or reaches it after a sequence of numbers, it will remain stuck at 0 from then onwards.

Therefore, the maximum possible value for the period of an *n*-bit LSFR is 2^n-1 and for the maximum period to occur the sequence of values held in the register must never include 0.

To achieve this maximal period there must be a tap on the last flip-flop. Since, if this tap is not in place then the initial value held in the n^{th} flip-flop (the msb) has no effect on the input to the lsb and the register has in effect only 2^{n-1} states.

Clearly, even for a small LFSR there are a great many possible configurations for the taps - 2^{n-1} for an n-bit register (assuming there is a tap on the msb).

Figures 9.4 shows a 4-bit LFSR (with seed = 5) that has a sub-maximal period, and Figure 9.5 shows another 4-bit LFSR (with seed = 5) that has a maximal period.

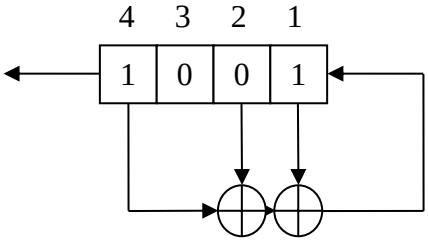

Figure 9.4 A 4-bit Linear Feedback Shift Register with Sub-maximal Period

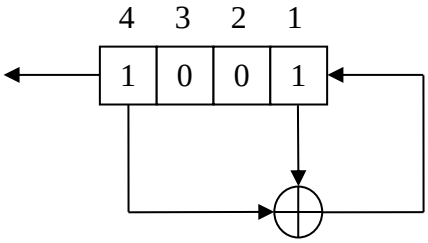

Figure 9.5 A 4-bit Linear Feedback Shift Register with Maximal Period

Exercise 9.7

1. For the 4-bit LFSR in Figure 9.4 with a seed of 1001 calculate the sequence of numbers that it produces. As the sequence produced does not have maximal period, find all the other sequences.

2. For the 4-bit LFSR in Figure 9.5 with a seed of 1001 calculate the sequence of numbers that it produces.

3. Use the LFSR in question 2 to generate 16 numbers and test them for uniformity using the chi-squared test. Use 4 intervals of equal size. Test at the 5% significance level.

9.3.3 Mathematical Representation of the Linear Feedback Shift Register

If the number held by the register at iteration i is R_i and the j^{th} bit of this number is denoted by $R_i(j)$, then

$$R_i(j) = \begin{cases} R_{i-1}(j-1) & \forall j \in (2,n) \\ \sum_{j'=1,n} c_j R_{i-1}(j') \mod 2 & j = 1 \end{cases}$$

and

$$c_j \in (0,1) \quad \forall j$$

Where the c_j's represent the taps in the LSFR circuit; $c_j = 1$ means that a tap exists at switch j, and $c_j = 0$ means that there is no tap at that position.

9.3.4 The Feedback Polynomial of the Linear Feedback Shift Register

The LFSR in Figure 9.3 can be represented by the feedback polynomial (or characteristic polynomial) $z^8 + z^5 + z^3 + z^1 + 1$, where the powers of z are the positions of the switches from which the taps are taken see Figure 9.6, and the 1 corresponds to z^0 and represents the input to the lsb of the LSFR.

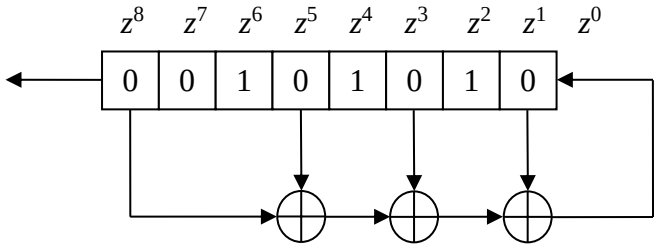

Figure 9.6 A LFSR with Feedback Polynomial $z^8 + z^5 + z^3 + z^1 + 1$

In general, a shift register will have a feedback polynomial of the form:

$$f_n(z) = c_n z^n + c_{n-1} z^{n-1} + \ldots + c_3 z^3 + c_2 z^2 + c_1 z^1 + 1$$

$$c_i \in (0,1) \quad \forall i < n$$
$$c_n = 1$$

where the constants c_i represent whether or not a tap exists at that position in the register.

It can be shown that a LSFR has maximal period if and only if its feedback polynomial is a primitive polynomial over the finite field $GF(2)$ (Golomb, 1967).

A characteristic polynomial $f_n(z)$ is primitive if it has the following properties:

(i) it has no proper non trivial factors,

(ii) $f_n(z)$ does not divide $z^k + 1$ for any $k < 2^n - 1$.

Exercise 9.8

1. Write down the characteristic polynomials for the linear feedback registers in Figure 9.4 and Figure 9.5.

2. Calculate the first 5 states for the shift register represented by the feedback polynomial $z^7 + z^3 + z^1 + 1$ starting with the seed 1011010.

3. Show that the feedback polynomial $z^3 + z^2 + z^1 + 1$ is not primitive.

4. Show that the feedback polynomial $z^4 + z^2 + z^1 + 1$ is not primitive.

9.3.5 The Linear Feedback Shift Register Cipher

The Linear Feedback Shift Register can be used to create a stream cipher. Due to the nature of shift registers and the areas of application, it is natural to work in binary, so in this section $m = 2$, and $x = (x_1, x_2, x_3, ..., x_n)$ and $y = (y_1, y_2, y_3, ..., y_n)$ are binary number of length n.

Definition 9.2 The LFSR Cipher

The Linear Feedback Shift Register cipher is a polyalphabetic cipher, it enciphers the plaintext binary n-gram $x = (x_1, x_2, x_3, ..., x_n)$ to the ciphertext binary n-gram $y = (y_1, y_2, y_3, ..., y_n)$ using an encipherment key $e_k = (R_0, (c_1, .., c_{n'}))$ where R_0 is the register seed and $(c_1, .., c_{n'})$ are the taps of a register of size n'.

$$y_i = x_i + R_i(n') \mod 2 \quad 1 \le i \le n$$

where,

$$R_i(j) = \begin{cases} R_{i-1}(j-1) & \forall j \in (2, n') \\ \sum_{j'=1,n} c_{j'} R_{i-1}(j') \mod 2 & j = 1 \end{cases}$$

and

$$c_j \in (0,1) \quad \forall j$$

Decipherment using decipherment key d_k is accomplished by:

$$y_i = x_i + R_i(n') \mod 2 \quad 1 \le i \le n$$

The LFSR Stream cipher has $2^n - 1$keys for any feedback polynomial $f_n(z)$.

Exercise 9.9

1. Use the LFSR Stream cipher with values characteristic polynomial $z^4 + z^3 + 1$ and seed 1011 to encipher the binary plaintext *n*-gram "1101010011".

2. Decipher the output of question 1.

3. Use the output from the LFSR Stream cipher in Exercise 9.7, Question 2 to encipher the binary plaintext *n*-gram "0111000111".

9.4 Historic Stream Ciphers

9.4.1 The Lorenz Machine

A famous stream cipher was the Lorenz machine that was used by the Germans in WWII. This was based on rotors, but unlike the Enigma machine, which was based on a series of complex substitutions, the Lorenz's rotors were used to generate a pseudo random running key that was added to the plaintext message. Figure 9.7 shows the Lorenz machine with its covers removed, exposing its 12 rotors.

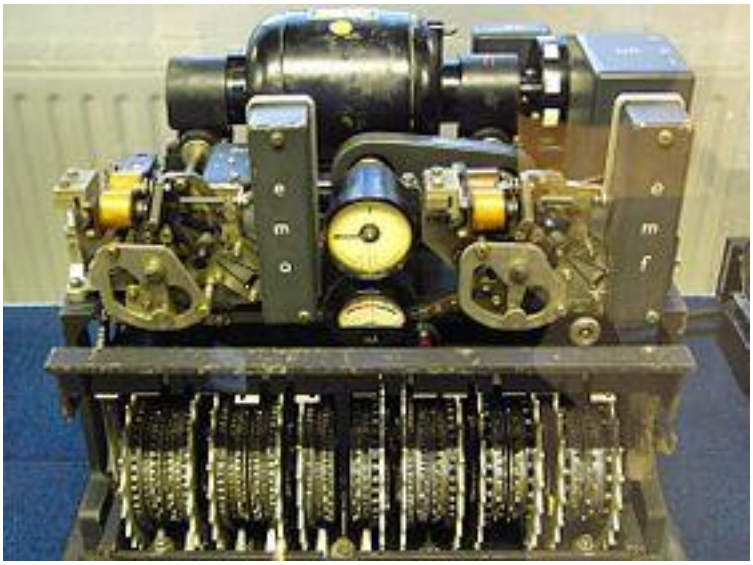

Figure 9.7 A Lorenz SZ42 Cipher Machine

The Lorenz cipher was used for high level communications between the German Supreme High Command and its armies and army groups. Consequently, the importance and value of the information protected by the Lorenz machine was higher than the traffic carried by the Enigma machine (Hinsley et al, 1981).

The breaking of the Lorenz cipher was one of the greatest cryptanalytical achievements of WWII. No Lorenz machine or manuals were ever recovered during WWII and the breaking of the cipher was conducted solely by cryptanalysis. The initial break occurred when against protocol a message was sent twice using the same key, and the second time the message was sent some words were abbreviated. The British cryptanalyst John Tiltman used the two ciphertexts to uncover both plaintexts and the key.

The key was then passed to the mathematician Bill Tutte who uncovered the logical structure of the Lorenz machine. After this it was possible to attack ciphertext messages to uncover their key, however, this could take around six weeks to achieve. To speed up the key recovery procedure that Tutte had devised, machines were developed. First came an electromechanical devise called the Heath Robinson and then the Colossus computer was developed.

9.4.2 The Colossus

The Colossus was the first electronic digital computer, although its programmability was somewhat limited. The colossus Mark I was built in December 1943 and contained 1500 thermionic valves. The Colossus Mark II, with 2,400 valves was built in June 1944. Figure 9.8 shows the reconstruction at Bletchley Park of a fully working Colossus machine. By the end of WWII there were10 Colossus machines in operation.

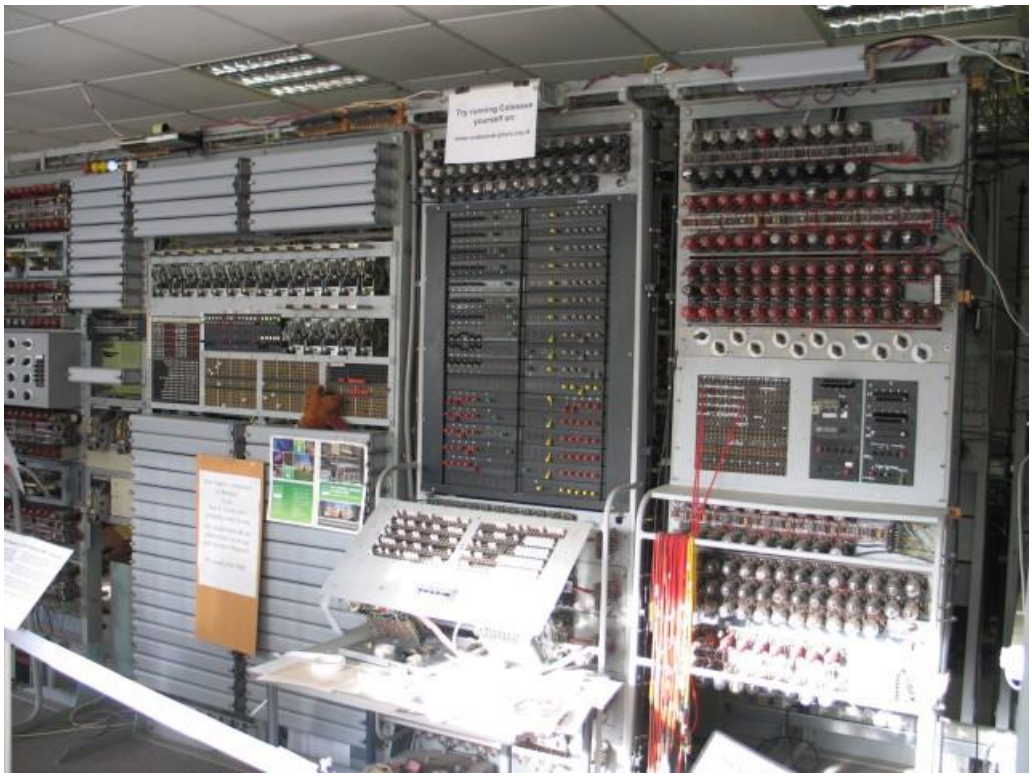

Figure 9.8 The Reconstruction of a Colossus at Bletchley Park

9.5 References

Berlekamp, E., R., (1967), "*Nonbinary BCH Decoding*", International Symposium on Information Theory, San Remo, Italy. McGraw-Hill 1968.

Fracassi, R. D., & Tammaru, T., (1981), US Patent 4,304,962 "*Data Scrambler*", United States Patent Office.

Golomb, S., W., (1967), "*Shift Register Sequences*", Holden-Day Inc, USA.

Goresky, M., and Klapper, A., M., (2012), "*Algebraic Shift Register Sequences*", Cambridge University Press, UK.

Hinsley, F., H., ed, Thomas, E., E., Ransome, C., F., G., and Knight, R., C., (1981), *"British Intelligence in the Second World War, Volume 2: Its Influence on Strategy and Operations"*, Her Majesty's Stationery Office (HMSO), London.

Lehmer, D. H., (1949), "*Mathematical methods in large-scale computing units*", Proc. 2nd symposium on large-scale digital calculating machinery, Annals of the computation laboratory of Harvard University, Vol. 26, pp. 141-14.

Massey, J. L., (1969), "*Shift-Register Synthesis and BCH Decoding*", IEEE Transactions on Information Theory, Vol. IT-15, No. 1

NIST SP 800-22, (2010), "*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*", Special Publication 800-22 Revision 1a, National Institute of Standards and Technology, US Department of Commerce.

Chapter 10 – Block Ciphers

A block cipher operates on a fixed length of bits of data called a block. Typical block sizes used by ciphers are 64 bits, 128 bits and 256 bits. Input data that is longer than the block size is simple cut up into a series of blocks.

Mathematically, each block of plain data is mapped to a block of cipher data. This means that a block cipher is no more than a General Substitution cipher operating on blocks of data. However, in practice, this is never achieved by the use of a look up table, because even for the smallest block size of 64-bits it would contain $2^{64} = 18x10^{18}$ rows.

Plaintext must first be put into a binary format, typically, using ASCII, see Table 8.1. Note that the typical block sizes used by ciphers are all a multiple of 8 bits, due to ASCII being 8 bits long. Thus eight characters of plaintext, i.e. an 8-gram, would correspond to a 64-bit block.

10.1 Lucifer

In the early 1970s Horst Feistel, whilst working for IBM, developed a cipher called Lucifer (Feistel, 1971). Lucifer was a block cipher that operated on bits of information. It was a product cipher that implemented Shannon's ideas of confusion and diffusion by the use of substitutions and permutations. These were repeatedly combined in what became known as a Feistel Network. A Feistel Network consists of a number of rounds, each of which involves a key, a substitution and a permutation.

In the first round, half the block of plaintext is encrypted using a key, this is then added bit by bit modulo 2 to the other half block of plaintext. Note that the adding of a bit to another bit modulo 2 is the same as XOR, which is the Exclusive-OR logic operation. Most computer hardware can perform an XOR command on a word of bits in a single operation. This means that this operation is very fast.

The half blocks are then switched and pass into the second round, where the same process is applied. This process is repeated through each round and finally the two blocks are recombined to produce the ciphertext block.

Figure 10.1 shows the encryption of a 16-bit block of data by a Feistel Network consisting of four rounds. This block is split into two 8-bit blocks one of which in each round is enciphered by function F_i using key K_i before being added to the other half block.

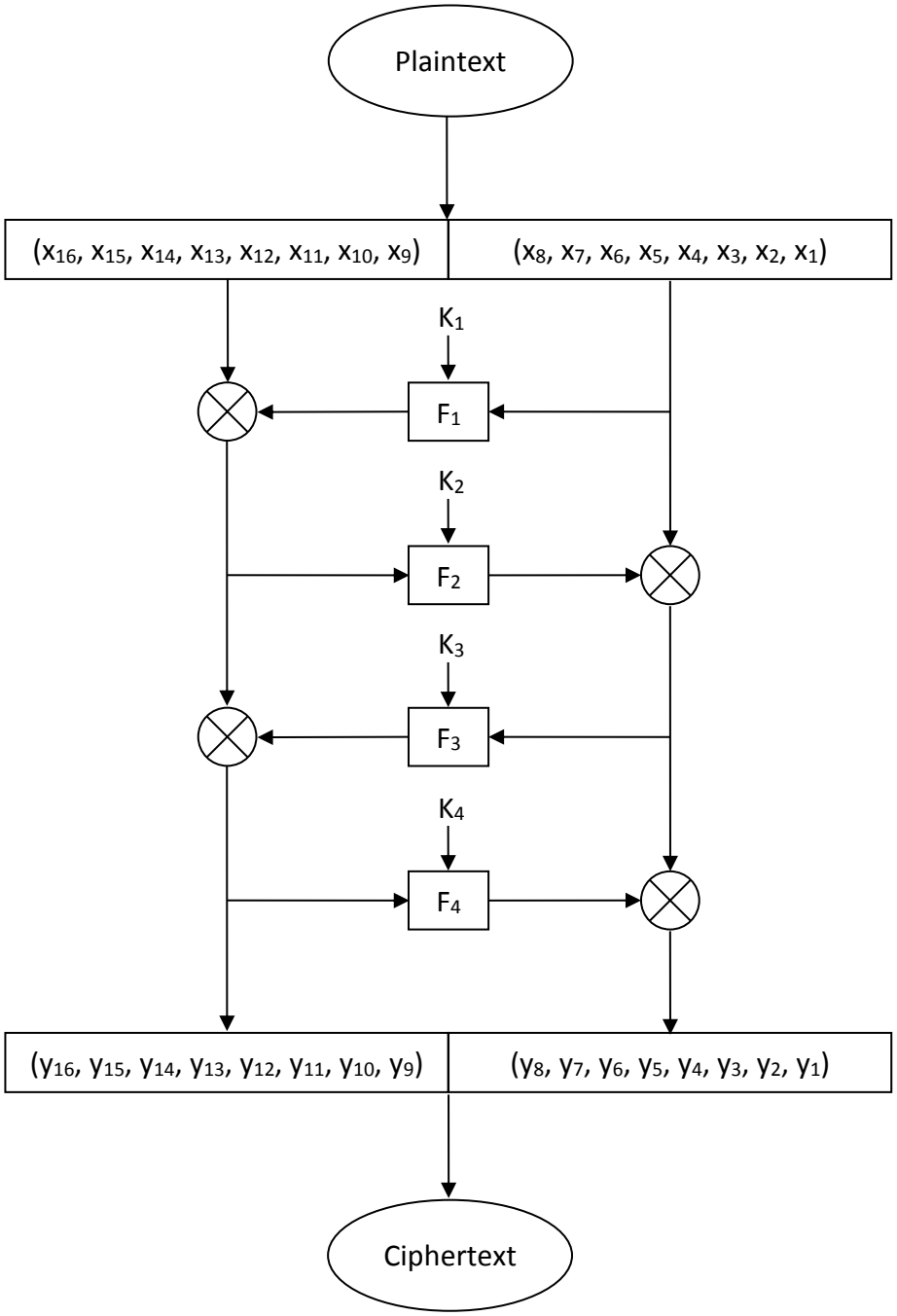

Figure 10.1 Encryption of a 16-bit block by a Four Round Feistel Network

Decryption is accomplished by reversing this process, that is, by following the arrows in Figure 10.1 backwards. This decryption process can be seen in Figure 10.2. Note that the overall structure of the algorithms shown in Figures 10.1 and 10.2 are the same. Further note that if all encryption functions F_i are equal then decryption can be accomplished by the same algorithm (and hardware) simply by reversing the order of the keys K_i employed. Actually, this will hold with the less restrictive condition that

$$F_i = F_{n-i} \qquad 1 \le i \le n$$

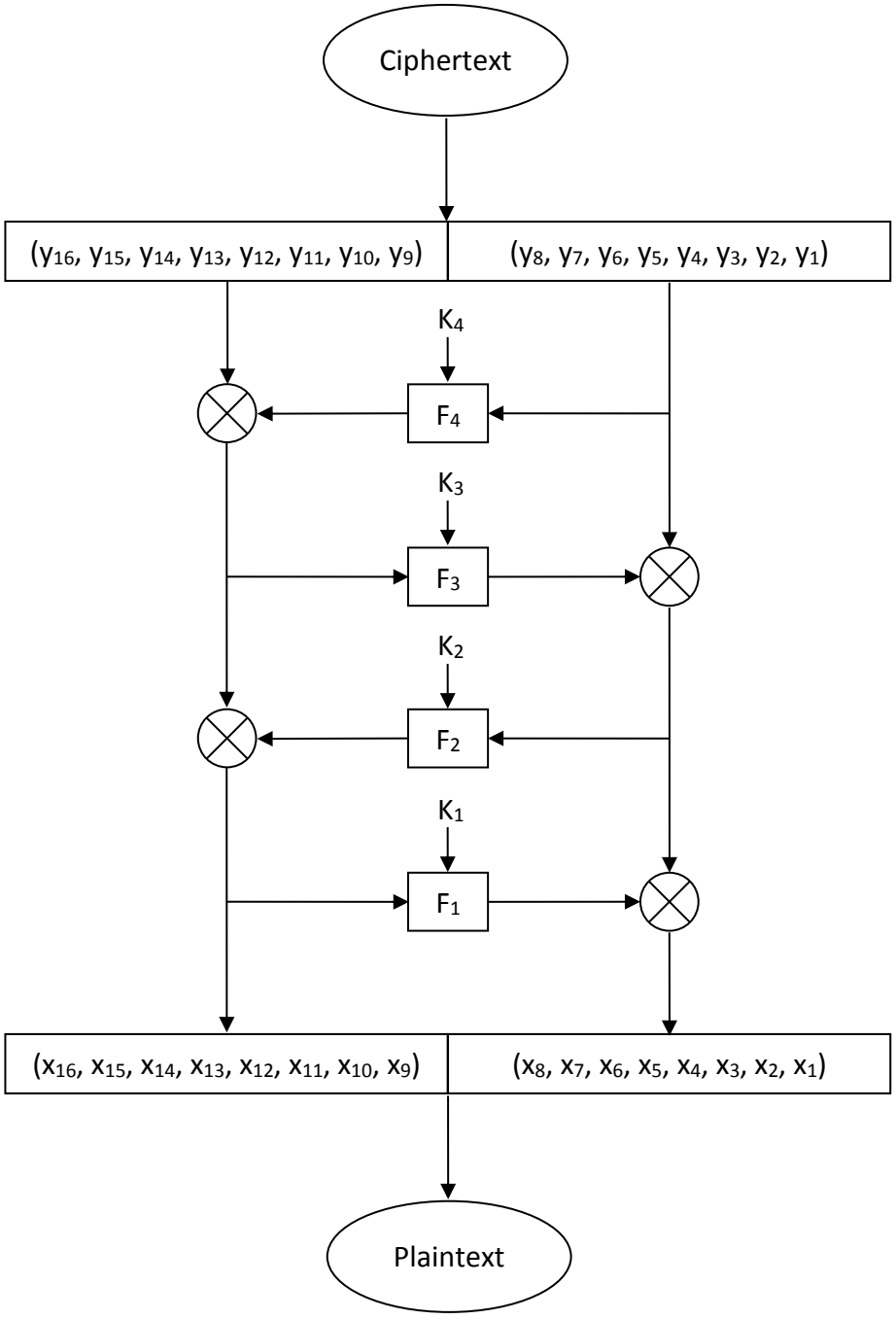

Figure 10.2 Decryption of a 16-bit block by a Four Round Feistel Network

10.2 The Data Encryption Standard

In 1973 the United States National Bureau of Standards (NBS) published a request for submissions for an encryption algorithm, which was to be used as a national standard for government communications. The aim was to provide government agencies with a means of protecting sensitive, but unclassified information. It was also to be published and commercial organisations encouraged to use it to secure their information.

IBM put together a team led by Feistel to further developed Lucifer and submitted it to NBS in 1974. This led to NBS publishing the Data Encryption Standard (DES) in 1977 (FIPS-46, 1977).

Strictly speaking the encryption algorithm is called the Data Encryption Algorithm (DEA). DES is the overarching standard and covers more than the algorithm itself, such as how it is to be implemented. However, DES is the term that is commonly used to describe the cipher.

10.2.1 DEA

The Data Encryption Algorithm encrypts blocks of data 64 bits long, using a 56-bit key. First the data goes through a fixed initial permutation (IP). Then it goes through a 16 round Feistel network, and finally through an inverse of the initial permutation. The keys for each round, called the sub-keys, are 48 bits long and are generated from the 56-bit key by the key schedule. See Figure 10.3 for an outline of how the cipher works.

10.2.2 Sub-key Generation

The sub-keys are generated from the 56-bit key using the algorithm shown in Figure 10.4. Note that the key is actually 64 bits in length. However, 8 of these bits (8, 16, 24, 32, 40, 48, 56 and 64) are used for parity checking and play no part in the cipher, and hence provide no extra security.

First the 56 bits undergo a fixed permutation called Permuted Choice 1 (PC-1), then split into two 28-bit blocks C_0 and D_0 both these are shifted left by 1 position to form C_1 and D_1 respectively. Note when shifting left, the original left most bit wraps around and becomes the right most bit. C_1 and D_1 are now combined to form a 56-bit number. This then undergoes a fixed permutation called Permuted Choice 2 (PC-2) to form a 48-bit sub-key 1, K_1. (Note that 8 bits are not used.)

C_1 and D_1 are now both shifted left by 1 position to form C_2 and D_2. C_2 and D_2 are now combined to form a 56-bit number. This is then permuted by Permuted Choice 2 (PC-2) to form a 48-bit sub-key 1, K_2.

This process is then repeated to produce the other keys. Notice that the shift left operation varies for each of the 16 stages. The shifts for each of the 1 to 16 stages are: 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 respectively.

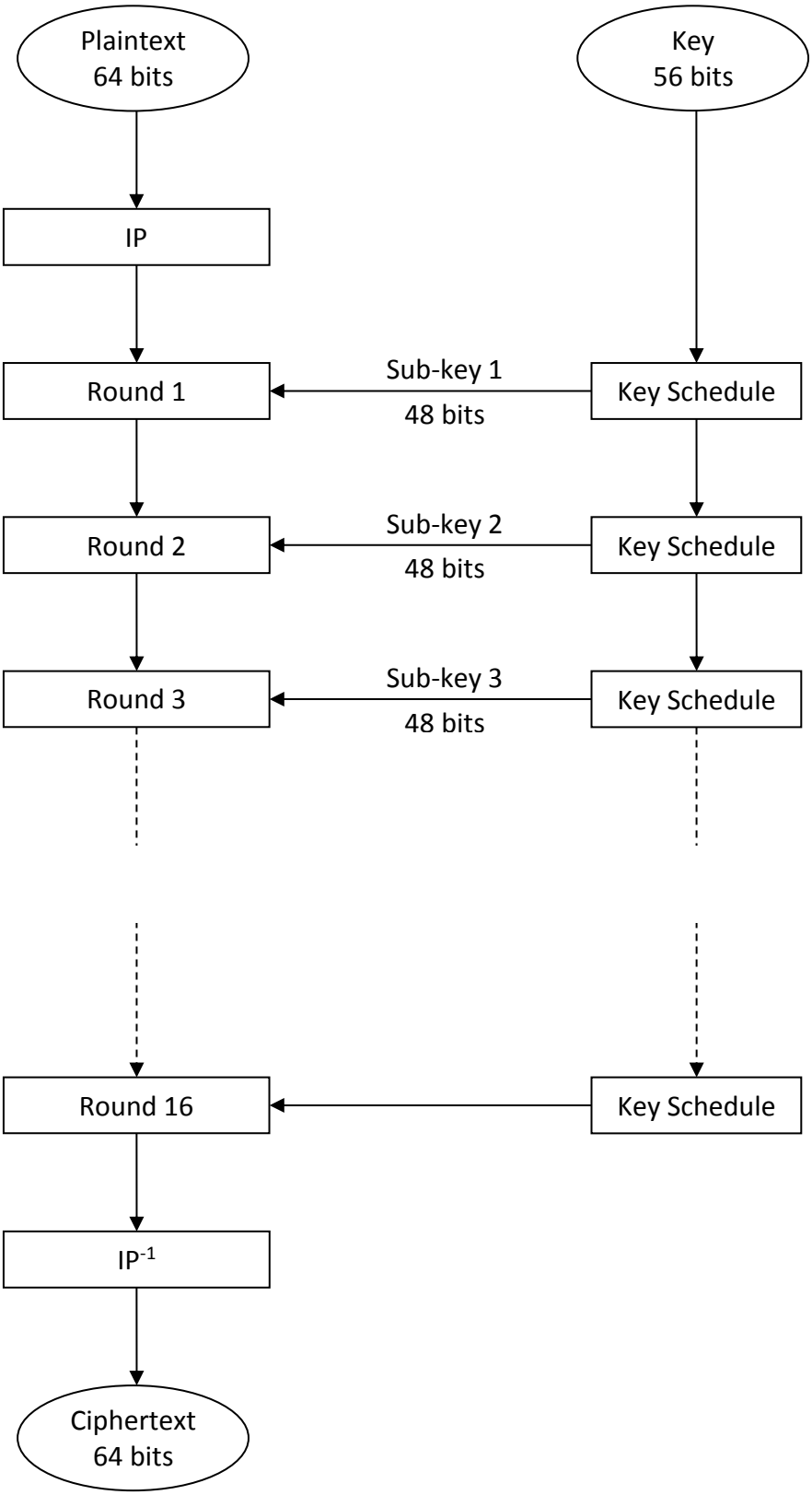

Figure 10.3 Outline of the DES Cipher

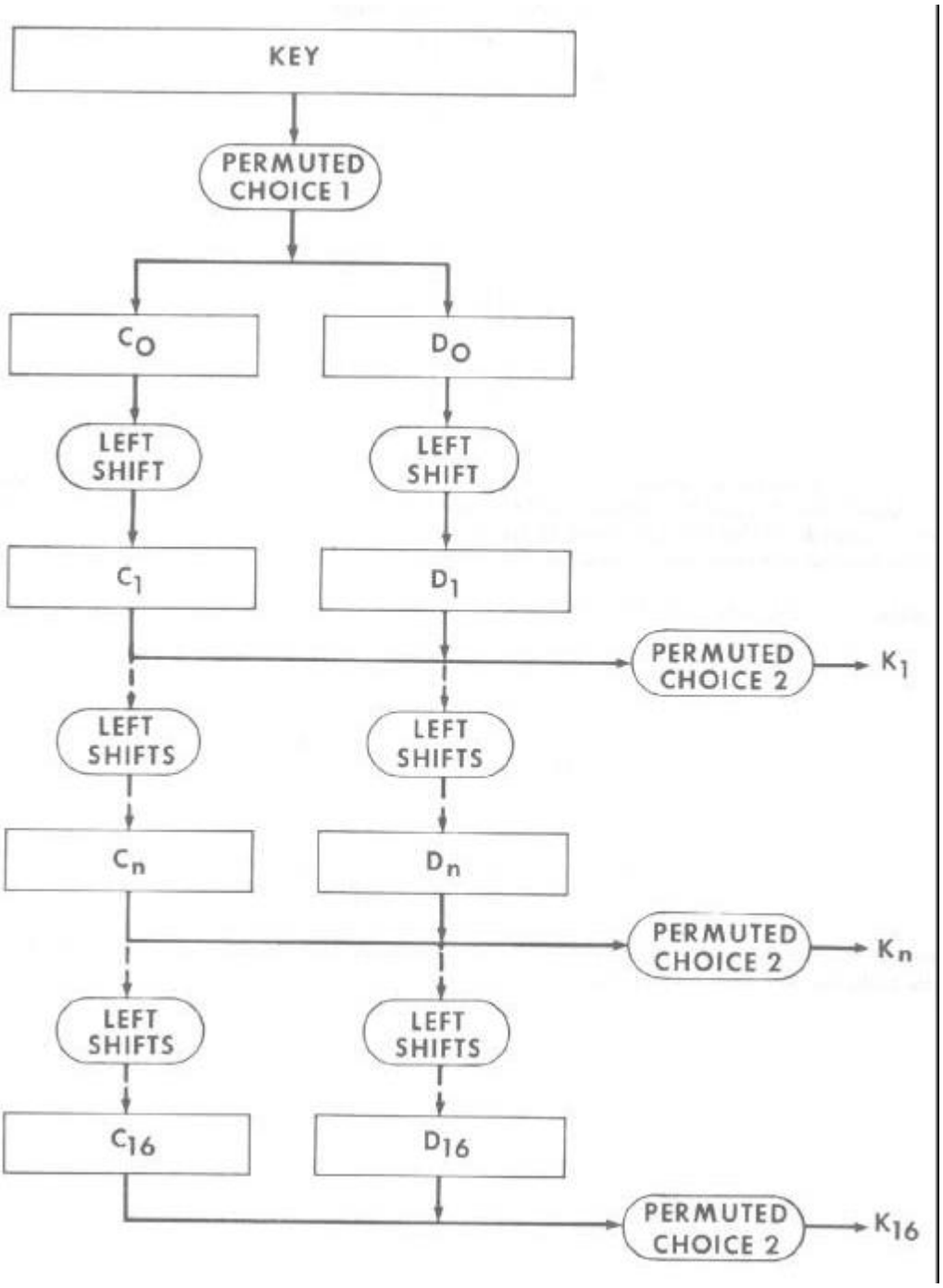

Figure 10.4 DES Key Schedule (from FIPS-46, 1977).

10.2.3 Rounds

After the IP the 64 bits of data are split into two 32-bit blocks L_0 and R_0 (Left and Right). These then enter the Feistel network. In general, the relationship between the input and output left and right blocks of data is given by:

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i + F(R_i, K_{i+1}) \bmod 2$$

and is shown in Figure 10.5.

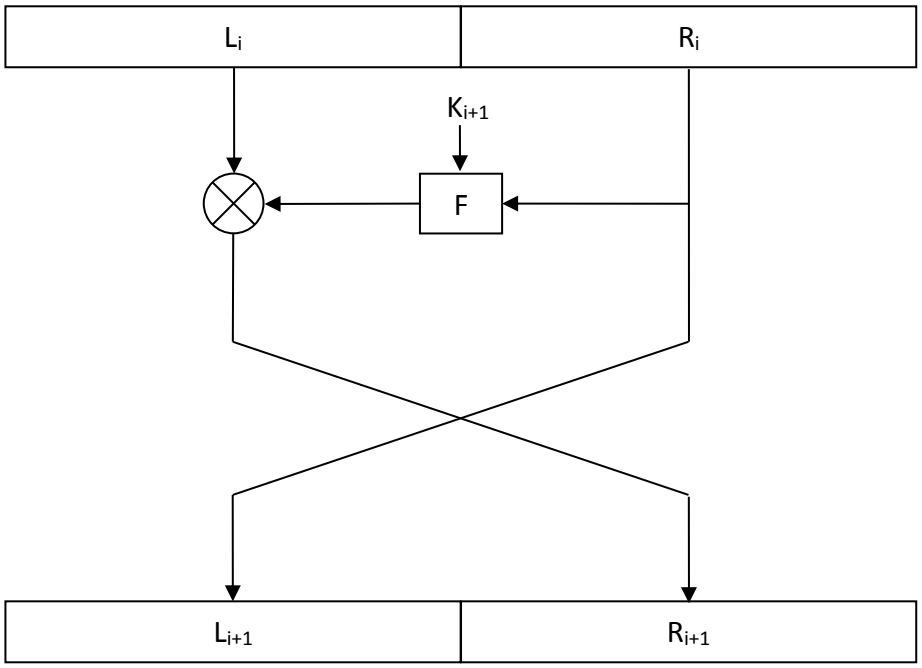

Figure 10.5 One Round of DES's Feistel Network

10.2.4 Encryption Function *F*

The operation of the encryption function *F* is shown in Figure 10.6. The Input data R_i is first expanded from 32 bits to 48 bits by repeating some of the data. Next, it is added to the sub-key K_i. After this, eight 6-bits slices of R_i are sent to the eight S-boxes, which perform a substitution ciphers. Finally, the outputs of the S-boxes are sent to a permutation called the P-box.

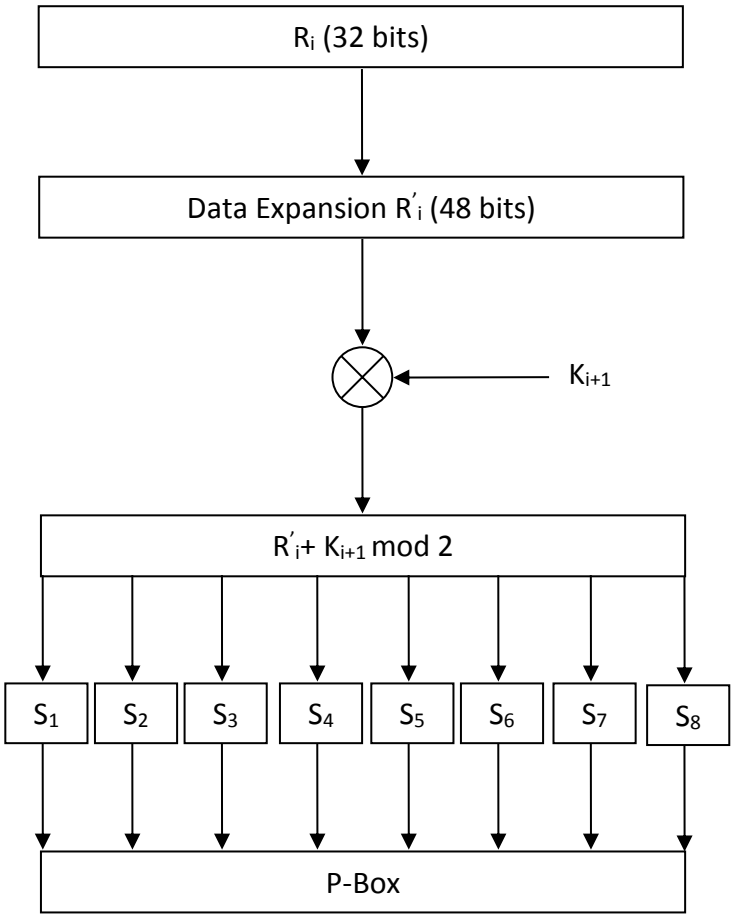

Figure 10.6 DES's Round Encryption Function

10.2.5 The Impact of DES

DES was the first widely available advanced cipher. As such, it was the first cipher to affect the daily lives of the public. Its use was mandatory for sensitive and confidential US government business, it was used by banks to secure their records, it was used to secure the transactions of Automatic Teller Machines (ATMs), and even to encrypt satellite television pictures. For over 30 years it was the dominant cipher in use.

An unexpected consequence of DES was that it led to a great upsurge in cryptography in the academic community. Before DES virtually all professional cryptographers were employed by the cryptographic agencies of governments or companies working for these agencies; after DES a substantial number of cryptographers work in universities.

10.2.6 Controversy over DES

The version of DES that IBM submitted to NBS used a 64-bit key, when the US National Security Agency (NSA) were asked for their opinion of the cipher the NSA asked for the key size to be reduced to 48 bits, although eventually it was agreed that

a 56-bit key should be used. This led to critics to claim that this was done to allow a brute force attack to be possible for an agency such as the NSA that had advanced computational devices.

According to one of the team that developed DES, Alan Konheim (Schneier, 1996), the NSA also changed the values used in the S-boxes, which again led critics to claim that these had been devised by the NSA to contain a trap-door, that is, a secret weakness that would allow the NSA to decipher any message. The idea that DES had been intentionally crippled by the NSA plagued the cipher for its entire operational life.

It is still not fully clear what actually happened, but it does now appear that IBM in developing and testing DES had re-invented differential analysis, a powerful cryptanalysis technique already known to the NSA. Further, it seems that the NSA had changed the S-boxes to make the cipher more secure against this type of attack. However, the NSA stopped IBM from publishing information on differential analysis and also stopped the NBS from publishing information on the design principles used to construct the S-boxes. This was because they feared that the release of the design principles would give away the fact that it had been hardened against an attack using differential analysis and hence allow other to discover this cryptanalysis technique, which it appears the NSA were using against a range of other ciphers at the time.

Finally, in 1994 Don Coppersmith one of the IBM team who developed DES and who was involved in the development of the S-boxes revealed the design criteria used (Coppersmith, 1994). He confirmed that strengthening the S-boxes to be resistant to differential cryptanalysis was one of the important considerations.

If the conspiracy theorist were wrong on the S-boxes, they were probably correct on the reduction of the key size. The reduction from a 64-bit key to a 56-bit key serves no purpose in making the cipher more secure, in fact it makes the cipher 256 times easier to break using a brute force search of the key space. If the NSA's proposal to reduce the key size to 48 bits had been accepted it would have made the cipher 65,536 times weaker against this type of attack.

Diffie and Hellman (1977) pointed out that it was feasible to build a specialist machine at a cost of $20 million that would be able to crack a DES cipher by a brute force search of the key space in one day. This level of spending would have easily have been within the resources of the NSA, which around this time was buying large numbers of Cray-1 supercomputers at $8 million each. In 1998, as predicted by Diffie and Hellman, just such a machine was built that could crack DES by testing all possible keys. The machine cost $250,000 to build and was able to crack a key in 4.5 days on average (EFF, 1998).

10.2.7 Triple DES

As a result of the increasingly successful brute force attacks on the key size of DES, it became clear that its remaining life was limited. In 1999 NIST updated the standard to require the use of Triple-DES, which was essential a triple application of the cipher with three independent keys (NIST FIPS 46-3, 1999). NIST also initiated a programme to look for the successor of DES.

The main purpose of triple enciphering the plaintext with DES was to increase it key size to stop a brute force attack. It also makes the cipher more difficult to break by other means. The obvious advantage of introducing Triple DES was that since DES was already widely implemented in software and hardware systems only very minor changes would be needed to use Triple DES. Figure 10.7 shows an outline of Triple DES- note that the middle use of DES is in decipherment mode. Triple DES uses three standard DES keys (key1, key2, key3), which are called the key bundle. So instead of having a 56-bit key as in DES, Triple DES has a 168-bit key.

Definition 10.1 Triple DES
Given three 56-bit keys K_1, K_2 and K_3, a 64-bit block of data x, and T represents encryption by DES, Triple DES is defined by:

$$y = T_{k_3}(T_{k_2}^{-1}(T_{k_1}(x)))$$

Decryption is accomplished by:

$$x = T_{k_1}^{-1}(T_{k_2}(T_{k_3}^{-1}(y)))$$

Triple DES has three keying options for the key bundle $K = (K_1, K_2, K_3)$ defined as:

Keying Option 1. All three keys are chosen independently, ie, $k_1 \neq k_3 \neq k_3$. This gives a 168-bit key and is the strongest cipher.

Keying Option 2. the first and second keys are chosen independently, and the third key is equal to the first, ie, $k_1 \neq k_2$ and $k_1 = k_3$. This gives a 112-bit key, which gives a weaker cipher than option 1.

Keying Option 3. All three keys are identical, ie, $k_1 = k_3 = k_3$. This gives a 56-bit key and is the weakest cipher. Note that this is exactly equivalent to DES and was provided to be backward compatible with older DES equipment.

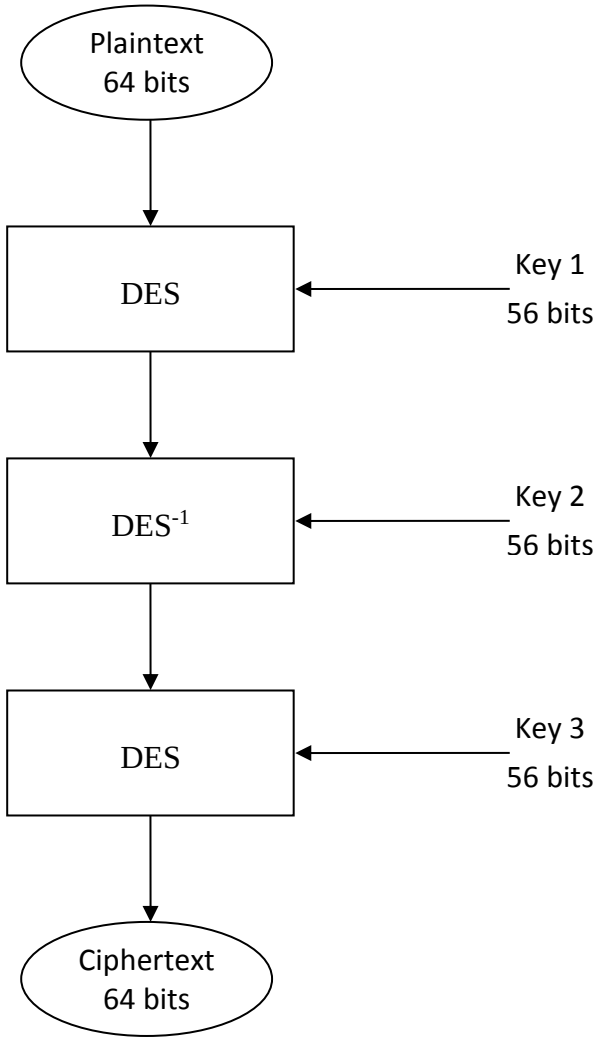

Figure 10.7 Triple DES

Exercise 10.1

1. Left shift the following binary numbers:

   (i) 00110101 by one place.
   (ii) 10101100 by one place.
   (iii) 11111111 by one place.
   (iv) 10101111 by two places.
   (v) 00000000 by two places.
   (vi) 10101010 by two places.

   What does this imply for DES sub-keys and key schedules?

10.3 The Advanced Encryption Standard

The Advanced Encryption Standard (AES) is the replacement for DES. Due to the controversy that surround the development of DES, the NIST decided to organised an open competition for a replacement. In 1997, NIST asked for candidate ciphers to be submitted for evaluation. Fifteen candidates were submitted, and there then followed three rounds of evaluation in which there was public consultation at each stage (Rijem, 2000, Schneier, 2000). This eventually led to the winner Rijndael, which had been developed by the Belgian cryptographers Vincent Rijem and Joan Daemen. This led to the publication of AES in 2001 (NIST FIPS 197, 2001).

AES is a symmetric block cipher. The AES is actually a family of three ciphers AES-128, AES-196 and AES-256, all of which encrypts 128-bit blocks of data. All three versions of AES have essentially the same structure, but have varying key lengths and number of rounds of encipherment (See Table 10.1).

Version	Key size	Rounds
AES-128	128 bits	10
AES-196	196 bits	12
AES-256	256 bits	14

Table 10.1 The Features of AES Variations

Unlike DES, which was based on a Feistel network, AES is based on a Substitution Permutation network (see Figure 10.8 for an example of an S-P network).

10.3.1 AES

AES-128 consists of 10 rounds each of which involves encryption with a 128-bit sub key derived from the main key by the key schedule. See Figure 10.9 for an outline of AES-128.

Round 1
Round 1 is simple the addition modulo 2 of sub-key 1 to the plaintext.

Rounds 2-9
Rounds 2 to 9 are identical in structure and consist of 4 stages. The 128 bits of data is viewed as consisting of 16 blocks of 8 bits (a SubByte) and these 16 SubBytes are viewed as being in a 4 by 4 matrix. The stages are:

1. SubBytes – each byte of data goes into a substitution cipher.
2. ShiftRows – Each row of the 4 by 4 matrix is shifted left by a fixed number of places. Rows 1, 2, 3, and 4 are shifted by 0, 1, 2, and 3 places respectively. This is a permutation.
3. MixColumns – Each column of 4 bytes individually undergoes a linear transformation to produce 4 bytes of output. Each byte of output depends on each byte of input.
4. Addition of the sub-key for that round modulo 2.

Round 10
This is the same as rounds 2 to 9 except that there is no MixColumn step.

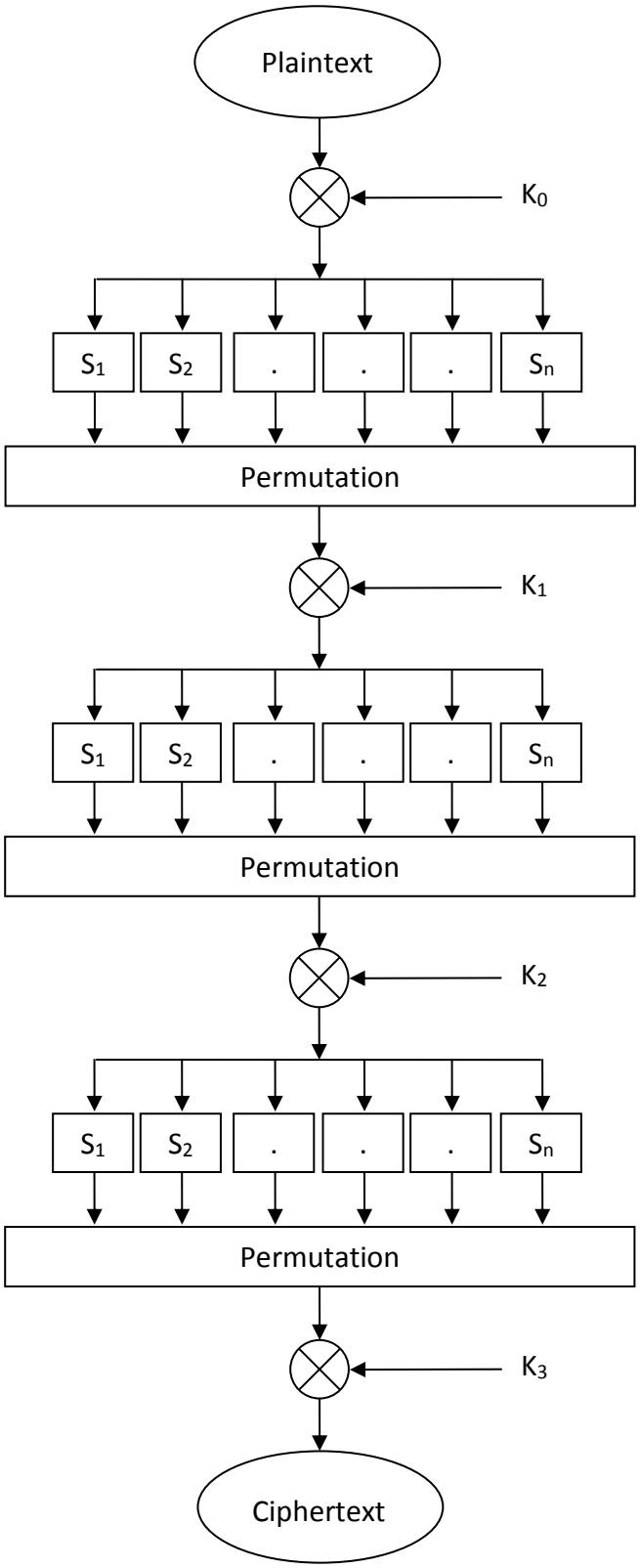

Figure 10.8 A Three Round Substitution Permutation Network

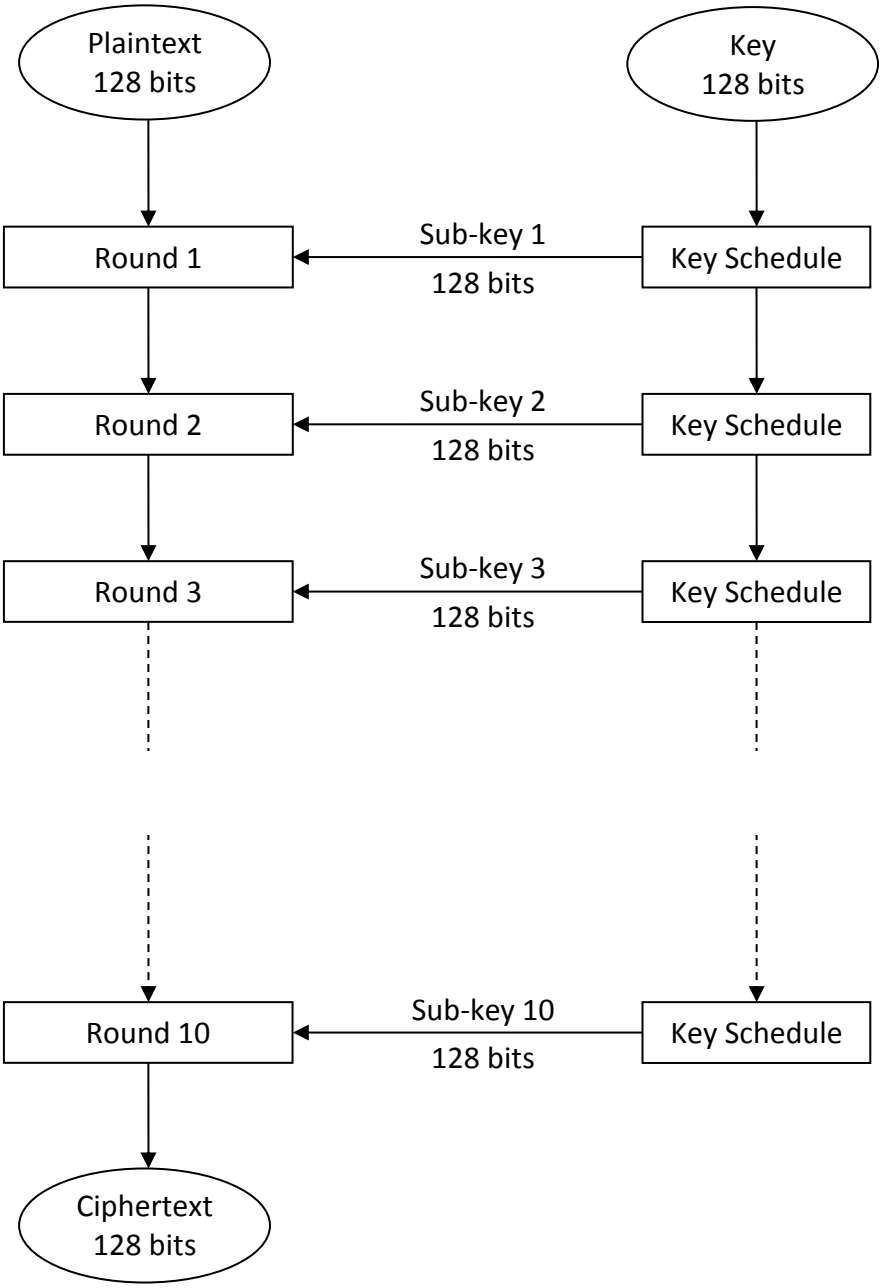

Figure 10.9 Outline of the AES-128 Cipher

10.4 Comparison of DES and AES

AES operates on 128-bit blocks of data while DES works on 64-bit blocks of data. This makes AES more secure against frequency analysis type attacks.

DES has a 56-bit key, but AES has a 128-bit (or 196-bit, or 256-bit). Therefore, AES is more resistant to a brute force attack of the key space. Triple DES has a 168-bit key, which makes it stronger than AES-128, but weaker than AES-196 and AES-256.

DES has 16 rounds, while AES has 10 rounds (or 12 or 14) this should make DES more secure.

AES is simpler with fewer different operations, which makes it easier to implement in a range of situations, such as, smart cards, mobile phones, memory sticks, etc.

AES has a simple algebraic structure, while DES has a more complicated one. This may make DES more difficult to crack.

DES is slower than AES, it has more steps and some of the steps are slower. Triple DES is approximately 3 times slower than DES.

AES is a more parallel algorithm than DES. On hardware that supports parallelism this allows AES implementations to be even faster.

There is still some doubt over DES including an NSA trapdoor. AES was produced independently, using an open and transparent process.

10.5 Modes of Operation of Block Ciphers

Block ciphers can be used in a range of ways depending on a range of factors such as application area and security issues. The modes Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB) were specified in "*DES Modes of Operation*" (NBS FIPS-81, 1980). The Counter (CTR) mode was introduced in "*Recommendation for Block Cipher Modes of Operation*" (NIST SP 800-38A, 2001) and XTS-AES was introduced in 2010 (NIST SP 800-38E, 2010).

In this section a plaintext *x* is regarded as consisting of a series of *n* blocks $x = (x_1, x_2, x_3, ..., x_n)$ that match the block size of the cipher. Similarly, for the ciphertext $y = (y_1, y_2, y_3, ..., y_n)$.

10.5.1 Electronic Code Book Mode

The simplest way to use a block cipher on a plaintext is in Electronic Code Book (ECB) mode. In general, the plaintext will be longer than the block size used by the cipher, so the plaintext is split into a series of blocks and these are enciphered one after another, with each block using the same key. If a plaintext does not consist of an exact multiply of the block size then the last block will need to be padded out with meaningless data. This mode gets its name from the fact that when operated this way the cipher can be regarded as a giant codebook. Figure 10.10 shows a 16-bit block cipher (in hexadecimal form), note that the repeated blocks (1st and 4th) in the plaintext are enciphered to blocks with the same value in the ciphertext.

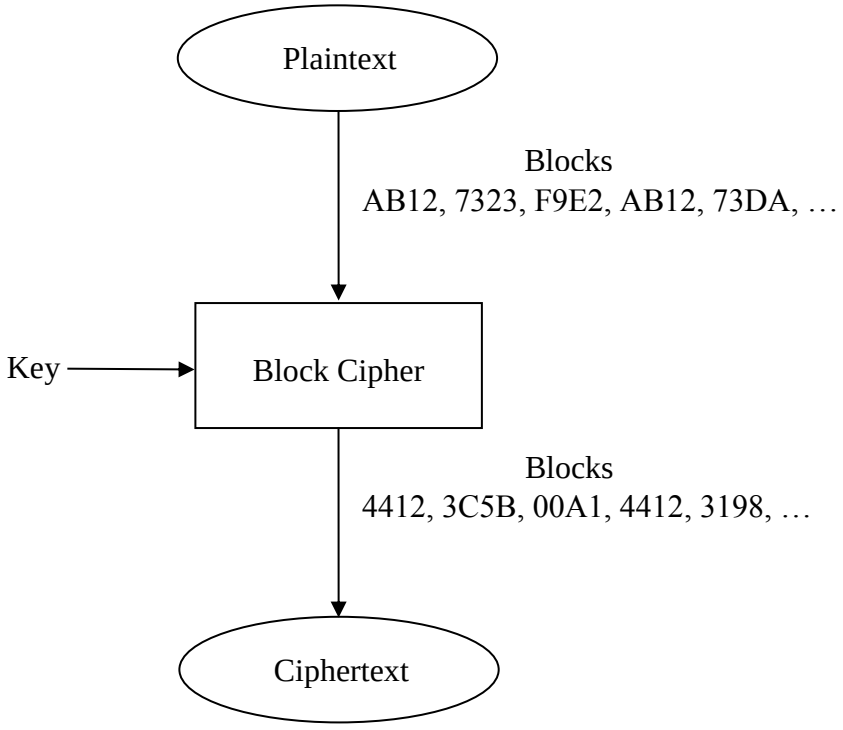

Figure 10.10 A Block Cipher in ECB Mode

Definition 10.2 Block Cipher in ECB Mode
Plaintext $x = (x_1, x_2, x_3, ..., x_n)$ is enciphered to ciphertext $y = (y_1, y_2, y_3, ..., y_n)$, by block cipher *T* in ECB mode and key *k*, using:

$$y_i = T_k(x_i) \qquad 1 \le i \le n$$

Decipherment is accomplished by:

$$x_i = T_k^{-1}(y_i) \qquad 1 \le i \le n$$

10.5.2 Cipher Block Chaining Mode

Cipher Block Chaining (CBC) was invented by IBM (Ehrsam et al, 1978), it was designed to overcome the problems caused by ECB mode encrypting identical blocks to the same ciphertext block and also to provide detection of transmissions errors. It uses an initial vector *IV*, which is the same size as the data block size. This initial vector is added EOR to the first message block, which is then encrypted. The result of the encryption is then added to the next block of data before it is then encrypted and so on. As a result, each cipher block is dependent on the previous cipher block. As a result of this dependency, when decrypting the message any bit errors will result in the message being indecipherable from that point onwards. Note that the initial vector is not secret.

Definition 10.3 Block Cipher in CBC Mode
Plaintext $x = (x_1, x_2, x_3, ..., x_n)$ is enciphered to ciphertext $y = (y_1, y_2, y_3, ..., y_n)$, by block cipher *T* and key *k*, together with an initial vector *IV*, using:

$$y_0 = IV$$
$$y_i = T_k(x_i \oplus y_{i-1}) \qquad 1 \le i \le n$$

Decipherment is accomplished by:

$$y_0 = IV$$
$$x_i = T_k^{-1}(y_i) \oplus y_{i-1} \qquad 1 < i \le n$$

10.5.3 Cipher Feedback Mode

Cipher Feedback mode is similar to CBC mode in that it chains each cipher block to the previous one. As a result, any transmission error causes all blocks from the error onwards to be indecipherable. This mode also requires a non secret initial vector *IV*, which is the same size as the data block size.

Definition 10.4 Block Cipher in CFB Mode
Plaintext $x = (x_1, x_2, x_3, ..., x_n)$ is enciphered to ciphertext $y = (y_1, y_2, y_3, ..., y_n)$, by block cipher *T* and key *k*, together with an initial vector *IV*, using:

$$y_0 = IV$$
$$y_i = T_k(y_{i-1}) \oplus x_i \qquad 1 \le i \le n$$

Decipherment is accomplished by:

$$y_0 = IV$$
$$x_i = T_k(y_{i-1}) \oplus y_i \qquad 1 < i \le n$$

10.5.4 Output Feedback Mode

Output Feedback (OFB) mode uses an initial vector as an input to the block cipher. The output is then used as the input to the block cipher, etc. These outputs can then be used as a stream cipher to be XORed to the plaintext.

Definition 10.5 Block Cipher in OFB Mode

Plaintext $x = (x_1, x_2, x_3, ..., x_n)$ is enciphered to ciphertext $y = (y_1, y_2, y_3, ..., y_n)$, by block cipher *T* and key *k*, together with an initial vector *IV*, using:

$$I_0 = IV$$
$$I_i = T_k(I_{i-1}) \qquad 1 \le i \le n$$
$$y_i = x_i \oplus I_i \qquad 1 \le i \le n$$

Decipherment is accomplished by:

$$I_0 = IV$$
$$I_i = T_k(I_{i-1}) \qquad 1 \le i \le n$$
$$x_i = y_i \oplus I_i \qquad 1 \le i \le n$$

10.5.5 Counter Mode

The Counter (CTR) mode is another method of producing a stream cipher from a block cipher. It uses a sequence of counters as inputs to a block cipher, with the outputs used as a stream cipher to be XORed to the plaintext. The last block of the message may not be complete, in which case, the most significant bits of the last block of output from the stream cipher are used. Note that the counters must not repeat, either within the message or in any message that uses the same key.

Definition 10.6 Block Cipher in CTR Mode

Plaintext $x = (x_1, x_2, x_3, ..., x_n)$ is enciphered to ciphertext $y = (y_1, y_2, y_3, ..., y_n)$, by block cipher *T* and key *k*, together with a sequence of counters $c_1, c_2, c_3, ..., c_n$, using:

$$I_i = T_k(c_i) \qquad 1 \le i \le n$$
$$y_i = x_i \oplus I_i \qquad 1 \le i \le n$$

Decipherment is accomplished by:

$$I_i = T_k(c_i) \qquad 1 \le i \le n$$
$$x_i = y_i \oplus I_i \qquad 1 \le i \le n$$

Note that the counters are usually defined by

$$c_i = c_{i-1} + 1 \qquad 2 \le i \le n$$

where, c_1 is a randomly chosen value that plays a similar role to an initial vector.

Exercise 10.2

1. Show that encipherment and decipherment transformations are inverse functions of one another, for a block cipher operated in each of the following modes.

   (i) ECB mode
   (ii) CBC mode
   (iii) CFB mode
   (iv) OFB mode
   (v) CTR mode

2. A new 4-bit block cipher has been proposed, at a particular key setting, it transforms 4-bit plaintext (x) as given in the following table.

Plaintext	Ciphertext	Plaintext	Ciphertext
0000	0100	1000	0001
0001	0110	1001	0111
0010	1010	1010	0101
0011	1000	1011	1110
0100	1100	1100	0000
0101	0011	1101	1011
0110	1111	1110	0010
0111	1101	1111	1001

   Use this block cipher together with an initial vector of 0110 to encipher the plaintext message 1011 0010 0010 0101 using the following modes:

   (i) ECB mode
   (ii) CBC mode
   (iii) CFB mode
   (iv) OFB mode
   (v) CTR mode

3. Using the block cipher give in question 2 and initial vector 1001, decipher the ciphertext message 0011 1001 1000 0001 using the following modes:

   (i) ECB mode
   (ii) CBC mode
   (iii) CFB mode
   (iv) OFB mode
   (v) CTR mode

4. Using the block cipher give in question 2, encipher the plaintext block 0000 then encipher the result using the same cipher and repeat this process. What happens and what does this imply for the security of the five modes given?

10.6 References

Coppersmith, D., (1994), "*The Data Encryption Standard (DES) and its strength against attacks*", IBM Journal of Research and Development, Vol 38:3

Diffie, W., & Hellman, M., (1977), "*Exhaustive Cryptanalysis of the NBS Data Encryption Standard*", Computer, Vol. 10, no. 6, pp. 74-84.

Ehrsam, W. F., et al, (1978), US Patent 4,074,066, "Message Verification and Transmission Error Detection by Block Chaining", United States Patent Office.

Electronic Frontier Foundation, (1998), "*Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*" O'Reilly & Associates, Inc., CA, USA.

Feistel, H., (1971), US Patent 3,798,359 "*Block Cipher Cryptographic System*", United States Patent Office.

NBS FIPS-46, (1977), "*Data Encryption Standard (DES)*", Federal Information Processing Standards Publications 46, National Bureau of Standards, US Department of Commerce.

NBS FIPS-81, (1980), "*DES Modes of Operation*", Federal Information Processing Standards Publications 81, National Bureau of Standards, US Department of Commerce.

NIST FIPS-46-3, (1999), "*Data Encryption Standard (DES)*", Federal Information Processing Standards Publications 46, National Institute of Standards and Technology, US Department of Commerce.

NIST FIPS 197, (2001), "*Advanced Encryption Standard (AES)*", Federal Information Processing Standards Publications 197, National Institute of Standards and Technology, US Department of Commerce.

NIST SP 800-38A, (2001), *"Recommendation for Block Cipher Modes of Operation*", Special Publication 800-38A, National Institute of Standards and Technology, US Department of Commerce.

NIST SP 800-38E, (2010), "*Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*", Special Publication 800-38E, National Institute of Standards and Technology, US Department of Commerce.

Preneel, B., Rijmen, V., and Bosselaers, A., (1998), "*Principles and Performance of Cryptographic Algorithms*", Dr Dobb's Journal, Vol. 23, No. 12.

Rijem, V., & Daemen, J., (2000), "*Rijndael for AES*", The 3rd AES Conference.

Schneier, B., (1996), "*Applied Cryptography*", 2nd ed., John Wiley & Sons.

Schneier, B., Whiting, D., (2000), "*A performance Comparison of the Five AES Finalists*", AES Submission to NIST, National Institute of Standards and Technology, US Department of Commerce.

Chapter 11 – Advanced Cryptanalysis

11.1 Cryptanalytic Attack Models

When conducting cryptanalysis of cipher there are several general approaches that can be taken, which are sometimes called attack models. These are not specific techniques for solving a particular cipher, but rather overarching methods that can be deployed against a range of ciphers. Often the factor that decides which attack model is to be used will depend on how the cipher is actually employed rather than the cipher itself. Figure 11.1 shows, for various situations, the attack models that are typically used and the information that is aimed to be discovered.

11.1.1 Ciphertext-Only Attack

The cryptanalyst has the ciphertext only and is trying to recover the key and plaintext. This is the most common and most difficult type of attack to make on a cipher.

Very often the cipher system employed is known and cryptanalyst works to uncover the key currently being used. However, this is not always the case, sometime a new cipher is introduced and then additional analysis needs to be undertaken to determine the cipher type.

In most circumstance, the plaintext source, that is, the language is known. In situations where it is not known, this can result in extreme difficulties for the cryptanalyst.

11.1.2 Known-Plaintext Attack

The cryptanalyst has the ciphertext and the corresponding plaintext and is either trying to recover the key from a known cipher or key and cipher. This type of attack may be employed when a ciphertext has been transmitted together with accidental copy of the plaintext. Alternatively the plaintext may be available if it has been transmitted in a second cipher that has already been broken.

11.1.3 Probable-Plaintext Attack

The cryptanalyst has the ciphertext, but does not have the plaintext; however, due to the stylised or predictable nature of the plaintext an educated guess can be made of part of the plaintext. The aim is to recover the key and the full plaintext.

The opening lines of diplomatic and military messages are often very formal and hence predictable. For example, a communiqué to the British ambassador may begin "His Excellence, Sir John Smith, the Ambassador of the United Kingdom of Great Britain and Northern Ireland, Dear Ambassador …".

11.1.4 Chosen-Plaintext Attack

The cryptanalyst has the ability to get a chosen plaintext enciphered and the ciphertext captured. The aim is to uncover the key.

This could occur when a message is given to a foreign ambassador who then has it enciphered and sent back to his government. It could also occur if a spy gets temporary access to a cipher machine (but not its key).

This attack is particularly useful in modern applications of cryptography as these are usually automated, and allow the cryptanalyst to submit plaintexts that would not get past a cipher clerk in a more traditional setting.

11.1.5 Adaptive Chosen-Plaintext Attack

This is a refinement of the chosen plaintext attack in which repeated chosen-plaintexts can be submitted for encipherment and these can be picked in the light of the previous ciphertext produced. This can only be done in an automated setting.

11.1.6 Chosen-Ciphertext Attack

The cryptanalyst has the ability to get a chosen ciphertext deciphered to give the plaintext. This is only feasible under certain systems and protocols. The aim is to uncover the key.

11.1.7 Adaptive Chosen-Ciphertext Attack

This is a refinement of the chosen-ciphertext attack in which repeated chosen-ciphertexts can be submitted for decipherment and these can be picked in the light of the previous plaintexts produced. This can only be done in an automated setting. This is only feasible under certain systems and protocols. The aim is to uncover the key.

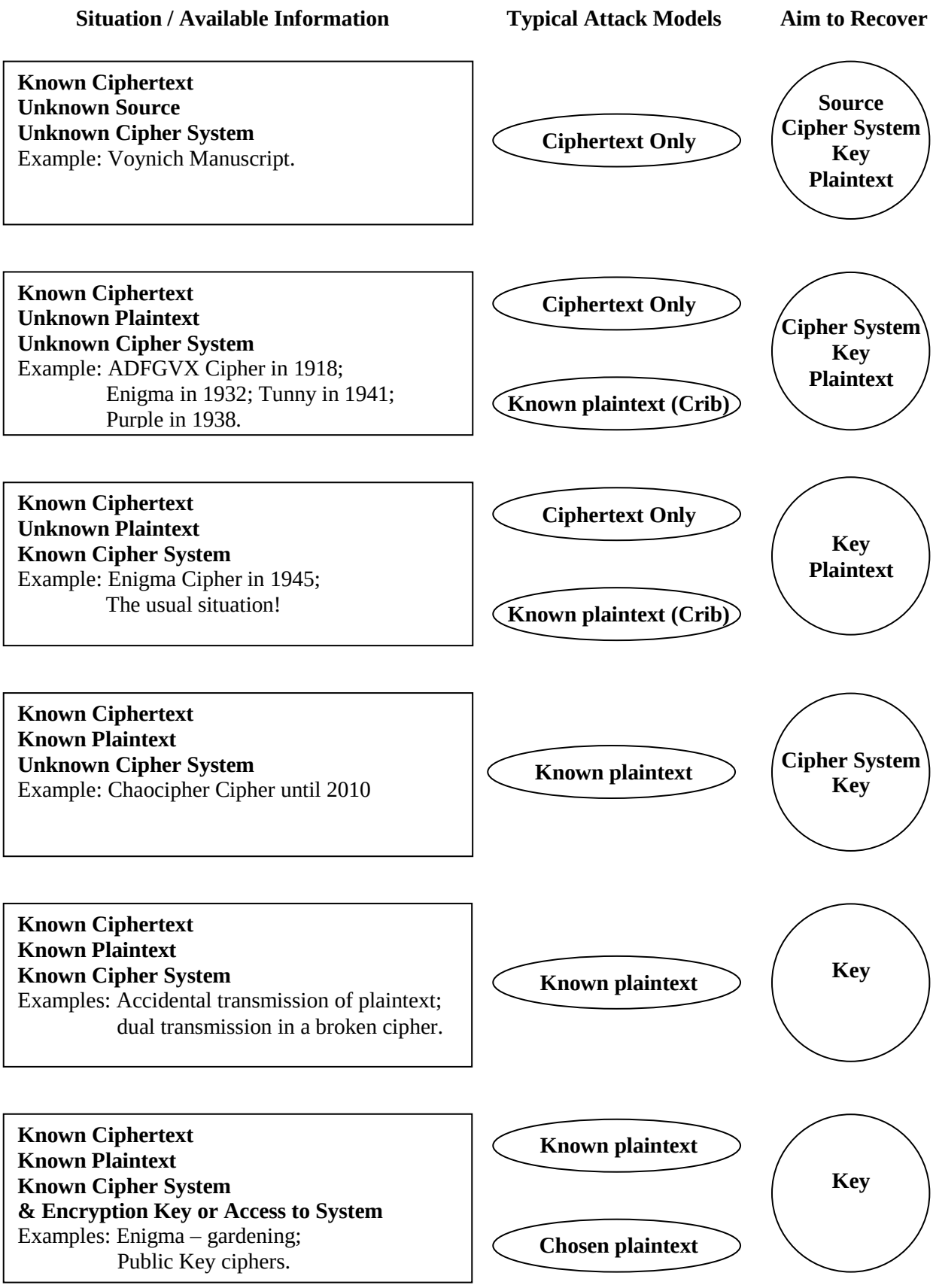

Figure 11.1 Cryptographic Systems: Available Information and Typical Attack Models

11.2 Advanced Techniques

11.2.1 Meet in the Middle Attack

This attack was first published by Diffie and Hellman (1977). It was devised as a means of attacking double encipherment with a block cipher *T* using two independent keys k_1 and k_2. If the key to the block cipher has *n* bits there will be 2^n possible keys. Double encipherment with two independent keys means there will be 2^{2n} possible keys, which is far harder to solve.

This is a known-plaintext attack. So assuming a plaintext and ciphertext pair (*x*,*y*) are known the aim is to discover the keys the k_1 and k_2.

$$y = T_{k_2}(T_{k_1}(x))$$

This can be partially inverted to get the equation:

$$T_{k_2}^{-1}(y) = T_{k_1}(x)$$

First, $T_{k_1}(x)$ is computed for all possible values of the key k_1 of which there are 2^n cases. These are then stored in memory.

Second, $T_{k_2}^{-1}(y)$ is computed for all possible values of the key k_1 of which there are 2^n cases. These are then compared to those in memory – a match means they keys have been uncovered.

Note that this methods only needs $2^n+2^n = 2^{n+1}$ keys to be examined, which is only twice as hard as the single encryption case. One drawback of this approach is that the memory requirements are 2^n x block size of the cipher.

11.2.2 Differential Cryptanalysis

Differential cryptanalysis was "discovered" by Biham and Shamir in 1988, (Biham, 2006), although it later emerged that it was known to IBM, where it was called the T attack, as early as 1974 and by the NSA even before that (Coppersmith, 1994).

This is a known-plaintext attack, so, given a known plaintext ciphertext pair (*x*,*y*) the aim is to discover the unknown key *k*.

Consider the simple block cipher shown in Figure 11.2, which consists only of a series of sub keys being XORed to the message interspersed by a substitution. One approach would be to try to follow *x* through each round of the cipher. Unfortunately, due to the message being XORed with sub keys in each round this becomes impossible. Instead, differential cryptanalysis uses two plaintext messages *x* and x' that differ by a known value Δx given by

$$\Delta x = x \oplus x'$$

It is this difference Δx, called the differential, that is followed through the cipher.

At round *j* the input to the *S* boxes will be, for plaintext *x*

$$x_j \oplus k_j$$

and for plaintext x', it will be

$$x'_j \oplus k_j$$

The XOR of these two inputs will be

$$\begin{aligned}(x'_j \oplus k_j) \oplus (x_j \oplus k_j) &= (x'_j \oplus x_j) \oplus (k_j \oplus k_j) \\ &= (x'_j \oplus x_j) \\ &= \Delta x_j\end{aligned}$$

This means the effect of the key has been removed.

There will many values of *x* and x' that will produce the same differential.

Now, assuming that the S boxes are known, it is easy to calculate how various differentials are transformed by the substitutions. These will not all be equally likely as they depend on the actual values of *x* and x'. However, they will allow statistical evidence to be accumulated that can be used to identify the following sub key.

This process can be repeated to peel away rounds and uncover the keys. Note that this gets more and more difficult with the number of rounds.

Exercise 11.1

1. What does the Meet in the Middle attack imply for the number of secure bits given by double encryption with DES using two independent keys?

2. What does the Meet in the Middle attack imply for the number of secure bits given by encryption with Triple-DES using two independent keys?

3. What does the Meet in the Middle attack imply for the number of secure bits given by encryption with Triple-DES using three independent keys?

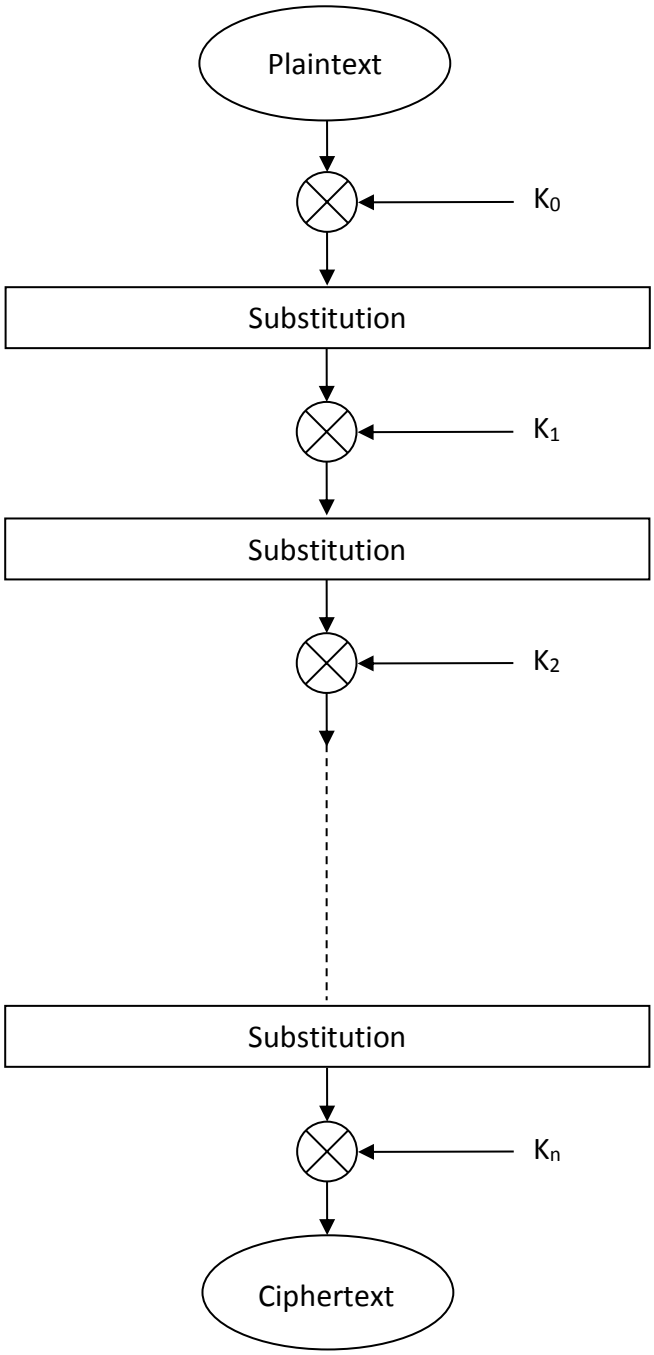

Figure 11.2 A Simple Block Cipher

11.3 Side Channel Attacks

Side channel attacks are attacks that do not try to break the cipher directly; instead they use properties of the physical system that implements the cipher to either side step the cipher entirely or to gain extra information that can aid in its cryptanalysis.

These attacks rely on the attacker having access in some way to the cipher device. As cryptographic equipment has become more common in everyday use, this type of attack has become more feasible due to commercial systems having less access restrictions than military or diplomatic systems.

11.3.1 Electromagnetic Attack

The first side channel attack was the electromagnetic attack. All electrical equipment radiate electromagnetic waves when operated. The strength of this radiated signal will depend on many factors, such as the power of the equipment and the length of wires that the electricity travels along. This radiated signal can be picked up by sensitive monitoring equipment in the same way that a radio picks up a radio signal. The partially declassified, and heavily redacted, NSA document "Tempest: A Signal Problem" (NSA, 1972), shows that the Americans were aware of this problem as far back as 1943.

An example of an electromagnetic attack is when a key on a computer keyboard is pressed and an electrical current is generated that tells the computer which key has been pressed. This current will generate a faint electromagnetic signal that is radiated, which can be picked up by a receiver. Since each key is attached to a slightly different length of wire it radiates a unique signal and hence it is possible to distinguish which key has been pressed. In this way a secret message can be intercepted before it is even encrypted. This type of phenomenon is present in even the most modern equipment, Kuhn (2004) shows how flat-panel displays are vulnerable to having their screens read via the electromagnetic signals that they emit.

TEMPEST is the NATO codeword given to the standard required of military security equipment that can withstand electromagnetic attack. Most details of the standard are classified.

11.3.2 Acoustic Attack

All mechanical equipment generate sound waves when operated. These sounds can be picked up by hidden microphones located nearby. Another way these sounds can be picked up is by laser beams measuring the distance to the window in which the equipment is located. The sounds will cause the window to vibrate slightly and the laser measurement is accurate enough to measure this. Once recorded, the sounds can be analysed using Fast Fourier Transforms (FFT) to detect which keys have been pressed on a keyboard. Peter Write (1987) recounts how in 1956 he used a bugged telephone in the Egyptian Embassy in London to monitor the sounds made by a Hagelin cipher machine as its daily settings were entered and thus enabled its keys to be determined.

11.3.3 Timing Attack

This attack is based on the fact that some ciphers take different times to encipher different data blocks. For example, the time to encipher "00000000" may differ from that taken to encipher "11111111". This is because different operations such as addition, multiplication, exponentiation, and table access all take different times. So, if a particular data block requires more operations that are time consuming it will take longer to encipher. Examining the timings for particular blocks gives information that will help break the cipher. Kocher (1996) gave a general methodology for this attack.

A requirement for this attack is that the enemy has access to the enciphered blocks together with their respective timings. This means the cryptanalyst must either have direct access to the cipher machine or access to live transmission of the ciphertext.

Vulnerability to this attack is an implementation issue rather than a mathematical one. However, as pointed out by Bernstein (2005) some algorithms are inherently vulnerable unless they are implemented in a less efficient way.

11.3.4 Simple Power Analysis

Simple Power Analysis (SPA) is a side channel attack that uses measurements of a crypto system's power consumption to break the cipher. It is similar to the timing attack in that it relies on different blocks of data requiring different amounts of power to encipher them due the underlying operations used by the cipher (Kocher, Jaffe, and Jun, 1998).

A requirement for this attack is that the attacker has access to the enciphered blocks together with their respective power consumption.

11.3.5 Differential Power Analysis

Differential Power Analysis (DPA) is a refinement of SPA, it uses statistical analysis of the power measurements of cipher equipment to extract information about the cipher key (Kocher, Jaffe, and Jun, 1999).

11.3.6 High-Order Differential Power Analysis

High Order Differential Power Analysis (HO-DPA) is a refinement of Differential Power Analysis that combines signals from different sources, and with different temporal offsets and advanced signal processing techniques (Kocher, Jaffe, and Jun, 1998).

11.3.7 Differential Fault Analysis

Differential fault Analysis is a side channel attack that extracts from a cryptographic system such as a smartcard by pushing it beyond the specification it was designed to operating and inducing it to make errors as it encrypts data. Examples of pushing a smartcard beyond its operating specification include: applying a current or voltage outside its operating range, heating it above its normal working temperature, applying radiation, etc. The effect of applying the strains on the smartcard is to alter memory

states in a non symmetrical way, that is, a bit containing a value 1 becomes a 0, but a 0 remains a 0. The initial idea of using faults for cryptanalysis was put forward by Boneh, DeMillo, and Lipton (1997, 2001) and generalised by Biham and Shamir (1997).

11.4 References

Bernstein, D. J., (2005), "*Cache-timing attacks on AES*", Technical Report M/C 249, Department of Mathematics, Statistics, and Computer Science, The University of Illinois at Chicago, USA.

Biham, E., Shamir, A., (1997), "*Differential Fault Analysis of Secret Key Cryptosystems*", in Proc. of Crypto '97, Lecture Notes in Computer Science Vol. 1294, pp. 513–528, Springer-Verlag, Berlin.

Biham, E. (2006), "*History of Differential Cryptanalysis*", FSE 2006, Fast Software Encryption, 13th International Workshop, Graz, Austria.

Boneh, D., DeMillo, R. Lipton, R., (1997), "*On the Importance of Checking Cryptographic Protocols for Faults*", in Proc. of Eurocrypt '97, Lecture Notes in Computer Science Vol. 1233, pp. 37–51Springer-Verlag, Berlin.

Boneh, D., DeMillo, R. Lipton, R., (2001), "*On the Importance of Eliminating Errors Cryptographic Calculations*", Journal of Cryptology, Vol. 14, pp. 101–119, Springer-Verlag, Berlin.

Kocher, P. C., (1995), "*Cryptanalysis of Diffie-Hellman, RSA, DSS, and Other Systems using Timing Attacks* ", Advances in Cryptology: Proceedings of CRYPTO '95, Lecture Notes in Computer Science, Vol. 963, pp. 171-183, Coppersmith, D., (ed), Springer-Verlag.

Kocher, P. C., (1996), "*Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*", Advances in Cryptology: Proceedings of CRYPTO '96, Lecture Notes in Computer Science, Vol. 1109, pp. 104-113, Koblits, N., (ed), Springer-Verlag.

Kocher, P. C., Jaffe, J., Jun, B., (1998), "*Introduction to Differential Power Analysis and Related Attacks*", Technical Paper, Cryptography Research Inc., San Francisco.

Kocher, P. C., Jaffe, J., Jun, B., (1999), "*Differential Power Analysis*", Advances in Cryptology: Proceedings of - Crypto '99, Lecture Notes in Computer Science, Vol. 1666, Wiener, M., ed., Springer-Verlag.

Kuhn, M. G., (2004), "*Electromagnetic Eavesdropping Risks of Flat-Panel Displays*", Computer Laboratory, University of Cambridge, UK.

NSA, (1972), "*TEMPEST: A Signal Problem : The Story of the Discovery of Various Compromising Radiations from Communications and Comsec Equipment*", Cryptologic Spectrum, Vol. 2, No. 3, National Security Agency, Summer 1972, FOAI Case number 51613 – partially declassified 27/09/2007.

Wright, P., (1987), "*Spycatcher: The Candid Autobiography of a Senior Intelligence Officer*", Viking Penguin, New York.

Chapter 12 – Applications of Cryptography

Originally, cryptography was developed to secure secret military and diplomatic messages, however, there are now many other applications of cryptography.

12.1 Automatic Teller Machines

Automatic Teller Machines (ATMs) allows a customer to conduct bank transactions by inserting a plastic bank card and a corresponding 4-digit Personal Identification Number (PIN). The card contains on a magnetic strip details such as the bank account number this information together with the PIN is sent by the ATM to the bank's central computer as evidence of the customer's identity. A thief who steals a card would have to guess the PIN, but after 3 failed attempts the card is retained.

This appears to offer a high level of security. However, the information sent by the ATM could be intercepted by means of tapping the telephone wire, which would allow an eavesdropper to uncover the customer's bank account number and PIN. In the late 1970s IBM introduced the "3624" ATM, which had a Hardware Security Module (HSM) to protect the security of the user's bank details. This encrypted the information before it left the ATM and decrypted any information from the bank's central computer. The HSM used DES to encrypt the information. This approach soon became standard. DES has since been replaced by Triple-DES for added security.

Although bank personnel do not know a customer's PIN, this system is not completely secure since there are only 10,000 possible PINs. A bank employee with access to the HSM could in theory write a program to try all possible PINs against a customer's account – cracking it in, on average, 1 minute. Bond and Zielinski (2003) described how a bank insider could exploit other weaknesses in the system to speed up this PIN cracking to 0.25 second.

12.2 Chip and Pin / Smart Cards

To help counter counterfeiting of cards with magnetic strips, debit and credit card companies introduced Chip and PIN Smart Cards. These cards use the Europay Mastercard and VISA (EMV) standard. The EMV standard uses the Triple-DES, RSA and SHA-1 ciphers to protect the user's PIN (EMVCo, 2008).

12.3 Biometric Passports

Modern biometric passports contain a Radio Frequency Identification (RFI) chip, which can be machine read from a distance. Under international standards (ICAO, 2006) the RFI chip stores personal details of the passport holder together with biometric data such as a digital copy of the passport photograph (mandatory), fingerprints (optional) and iris scans (optional). This biometric data is stored on the chip in an encrypted form - the algorithms used are country specific. In addition, the communication between the chip and the machine reader is encrypted using Triple-DES in CBC mode.

12.4 Hard Disks

Some operating systems provide a facility to encrypt the entire hard disk of a computer. Once this facility is turned on all information stored on the hard disk is encrypted. When programs or data files are moved off the drive they are automatically decrypted. Similarly, when a program is run it is decrypted as it is loaded into memory. Any files transferred to the hard disk are automatically encrypted. The entire encryption decryption process is transparent to the user. Table 12.1 shows some operating systems that have this facility and the cipher algorithm used.

Operating System	Name	Cipher used	Key Size
Microsoft Windows 7	BitLocker	AES in CBC mode	128 bit
Apple OX 10.7 Lion	FileVault 2	AES in XTS-AESW mode	128 bit
Linux	dm-crypt/crypto API	Various - user selectable	Various

Table 12.1 Ciphers used by Common Operating Systems.

12.5 Wi-Fi

Wi-Fi is a wireless technology that is designed to provide network connectivity between computers, laptops, tablets, internet routers, printers and other devices. It has three security protocols WEP, WPA and WPA2. The oldest protocol Wired Equivalent Privacy (WEP) is no longer regarded as secure due to weaknesses in the protocol (not its cipher), but it is still in use for backward compatibility of equipment. When the flaws in WEP were discovered it was quickly replaced by Wi-Fi Protected Access (WPA), which was an intermediate step towards the fully ratified standard Wi-Fi Protected Access 2 (WPA2). The encryption algorithm used by both WEP and WPA is RC4, while WPA2 uses AES (IEEE, 2007).

12.6 Bluetooth

Bluetooth, originally developed by Ericsson, is a wireless technology designed for short range low power communications. Examples of applications include: connecting a mobile phone with a handsfree headset; and connecting a computer with a wireless mouse and keyboard. Data is encrypted using a cipher that is the sum of four LSFR ciphers (Bluetooth, p1075, 2010).

12.7 Mobile Phones

Most mobile phones in the world use the Global System for Mobile Communications (GSM) standard. The first generation of phones had used different standards in different countries. The European Telecommunications Standards Institute (ETSI) introduced the GSM standard when the second generation (2G) of mobile phones was being developed. It has since been updated for third generation (3G) and fourth generation (4G) mobile phones and networks. GSM uses the A5/1 and A5/2 stream ciphers to encrypt the user's conversation as it transmitted from the mobile phone to their phone operator.

12.8 Internet

Security over the internet when using web browsers is provided by the Transport Layer Security (TLS) protocol, which replaced the previous Secure Sockets Layer (SSL) protocol. TSL version 1.1 uses the SHA-1 and MD5 ciphers (NWG, 2006), while TSL version 1.2 uses SHA-256, (NWG, 2006).

12.9 Pay Television

Satellite, cable and digital terrestrial television signals are all used to deliver Pay TV in many countries around the World. There are several systems used to protect these signals from unauthorised users, two of the most popular are: PowerVU, which uses DES or the Common Scrambling Algorithm CSA; VideoGuard, which uses a proprietary cipher algorithm (NDS, 2011).

12.10 Car Remote Central Locking

Originally, car remote central locking was simply an unique radio signal to open the door sent by pressing a button on the car key fob. In the first remote central locking keys, it was assumed that the security rested in the uniqueness of the signal, in the same way that the security of an ordinary key rests in it only being able to open a single lock. However, car thieves soon realised that they could simply record the signal being sent by the key to the car (using a gadget called a grabber) and then play this back at a later time to unlock the car.

The KeeLoq car remote central locking system was designed to overcome the key grabber attack. KeeLoq uses a Non-Linear Feedback Shift Register (NLFSR) encryption algorithm (Bruwer, Smit, and Kuhn, 1996).

12.11 Radio Frequency Identification

Radio Frequency Identification (RFID) is a wireless technology used to identify and track individual products and items. A small wireless circuit called a tag is inserted or attached to each item. A scanner emits a radio signal that is absorbed by electromagnetic induction by the RFID tag and used to power it. The RFID tag can then emit an identification signal. Shops are major users of this technology and use it to monitor stock levels and to reduce shoplifting. Other versions of RFID tags are implanted under the skin of pets so that they can be identified when lost. RFID tags are also used as security cards in buildings – simply wearing the card allows users access to parts of the building. As the technology allows the tags to be read at a distance of several feet the signals are enciphered using stream ciphers to stop illegal recording and reuse of the signals (Evans *et al*, 2008).

12.12 Password Storage

When someone logs onto a computer system they are required to supply a password that is checked against information stored in a file on the system. This provides a potential vulnerability, since this file could be accessed either by someone with admin privileges on the system or by someone who manages to get some access to the

system. To avoid this vulnerability the encrypted password is stored in the file and not the actual password. So accessing the file does not reveal the password. When a password is first created it is encrypted and the encrypted version is stored. Then when access is required the password is entered and then enciphered by the system and compared with the stored version.

12.13 Satellite Navigation

Satellite Navigation is based on the USA's Global Positioning System (GPS), which was designed for military applications. A freely available signal is also generated for Satnav use, however, this can be degraded or turned off in times of conflict. The military signal provides better accuracy and is protected by encryption of unknown type (Navstar, 1996).

12.14 References

Bluetooth, (2010), "*Core System Package*", Specification of the Bluetooth System Version 4, Volume 3, Bluetooth SIG, Washington.

Bond, M., and Zielinski, P., (2003), "Decimalisation Table Attacks for PIN Cracking", Technical Report 560, Computer laboratory, University of Cambridge.

Bruwer, F., J., Smit, W., and Kuhn, G., J., (1996), US Patent 5,517,187 "*Micropchips and Remote Control Devices Comprising Same*", United States Patent Office.

Cisco, (2011), "*Cisco PowerVu Network Centre Version 10.0*", Cisco Systems Inc, San Jose, USA.

Evans, D., Nohl, K., Plotz, H., & Starbug, (2008), "*Reverse Engineering a Cryptographic RFID Tag*", USENIX Security Symposium, 31 July 2008, San Jose, USA.

EMVCo, (2008), "*EMV Integrated Circuit Card Specifications for Payment Systems, Book 2: Security and Key Management*", Version 4.2, EMVCo LLC.

ICAO, (2006), "*Machine Readable Travel Documents: Part 1 Machine Readable Passports*", Document 9303, Sixth Edition, International Civil Aviation Organization.

IEEE, (2007), "*IEEE Standard 802.11-2007*", Revision of IEEE Standard 802.11-1999, IEEE Computer Society, New York.

Navstar, (1996), "*Navstar GPS User Equipment Introduction: Public Release Version*", Navtech Seminars, United States Air Force, Space Systems Division.

NDS, (2011), "*VideoGuard CA/DRM*", NDS Ltd, Staines, UK.

NWG, (2006), "*The Transport Layer Security (TLS) Protocol, Version 1.1*", Network Working Group.

NWG, (2008), "*The Transport Layer Security (TLS) Protocol, Version 1.2*", Network Working Group.