2

Encryption Algorithms

2.1. TRANSPOSITION CIPHERS

Transposition ciphers rearrange characters according to some scheme. This rearrangement was classically done with the aid of some type of geometric figure. Encipherment proceeded in two steps as shown next:

$$\begin{tikzcd}\text{plaintext} \arrow[r, "\text{write -- in}"'] & \text{figure} \arrow[r, "\text{take -- off}"'] & \text{ciphertext}\end{tikzcd}$$

First, the plaintext was written into the figure according to some "write-in" path. Second, the ciphertext was taken off the figure according to some "take-off" path. The key consisted of the figure together with the write-in and take-off paths.

The geometrical figure was often a 2-dimensional array (matrix). In **columnar transposition** the plaintext was written into a matrix by rows. The ciphertext was obtained by taking off the columns in some order.

Example:
Suppose that the plaintext RENAISSANCE is written into a 3 × 4 matrix by rows as follows:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline \mathrm{R} & \mathrm{E} & \mathrm{N} & \mathrm{A} \\ \mathrm{I} & \mathrm{S} & \mathrm{S} & \mathrm{A} \\ \mathrm{N} & \mathrm{C} & \mathrm{E} & \end{array}$$

If the columns are taken off in the order 2–4–1–3, the resulting ciphertext is ESCAARINNSE. ■

(See Mellen [Mell73] for a generalization of this technique to n-dimensional arrays.)

Many transposition ciphers permute the characters of the plaintext with a **fixed period** d. Let $\mathbf{Z}_d$ be the integers 1 through d, and let $f{:}\mathbf{Z}_d \rightarrow \mathbf{Z}_d$ be a permutation over $\mathbf{Z}_d$. The key for the cipher is given by the pair $K = (d, f)$. Successive blocks of d characters are enciphered by permuting the characters according to f. Thus, a plaintext message

$$M = m_1 \dots m_d \; m_{d+1} \dots m_{2d} \dots$$

is enciphered as

$$E_K(M) = m_{f(1)} \dots m_{f(d)} \; m_{d+f(1)} \dots m_{d+f(d)} \dots .$$

Decipherment uses the inverse permutation.

Example:

Suppose that $d = 4$ and f gives the permutation:

$$\begin{array}{rl} i: & 1\;2\;3\;4 \\ f(i): & 2\;4\;1\;3\;; \end{array}$$

thus, the first plaintext character is moved to the third position in the ciphertext, the second plaintext character to the first position, and so forth. The plaintext RENAISSANCE is enciphered as:

$$\begin{aligned} M &= \text{RENA ISSA NCE} \\ E_K(M) &= \text{EARN SAIS CNE} . \end{aligned}$$

The preceding ciphertext is broken into groups of four letters only for clarity; the actual ciphertext would be transmitted as a continuous stream of characters to hide the period. The short block at the end is enciphered by moving the characters to their relative positions in the permutation. ■

Like columnar transposition, periodic permutation ciphers can be viewed as transpositions of the columns of a matrix in which the plaintext is written in by rows. With periodic permutations, however, the ciphertext is also taken off by rows. This is more efficient for computer applications, because each row (block) can be enciphered and deciphered independently. With columnar transposition, the entire matrix must be generated for encipherment and decipherment.

The cryptanalyst can easily recognize whether a cipher is a transposition cipher because the relative frequencies of the letters in the ciphertext will closely match the expected frequencies for plaintext. The ciphers are broken by **anagramming**—the process of restoring a disarranged set of letters into their original positions (e.g., see [Sink66,Gain56]). This is facilitated with tables of frequency distributions for digrams (double-letter combinations) and for trigrams (triple-letter combinations). Figure 2.1 shows the frequency distribution of digrams in a file containing 67,375 letters (all letters were converted to uppercase). The file contained the preface of this book and other documents†. Note that the digrams TH and HE, which occur in the word THE, are in the highest group.

†Six ACM President's Letters and a grant proposal.

FIGURE 2.1 Frequency distribution of digrams.

	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
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	•			•					•			•														•

Maximum digram frequency = 2.31 % of the digrams

Key: * High: more than 1.15 % of the digrams
+ Medium: more than 0.46 % of the digrams
– Low: more than 0.12 % of the digrams
· Rare: more than 0.00 % of the digrams
blank: no occurrences

To determine the expected number of characters required to break a permutation cipher with period d, observe that there are $d!$ possible arrangements of d characters. Assuming all keys (i.e., arrangements) are equally likely, the entropy of the key is thus $H(K) = \log_2 d!$ Using Eq. (1.4), the unicity distance is thus:

$$N = \frac{H(K)}{D} = \frac{\log_2 d!}{D} .$$

Using Sterling's approximation for $d!$, we get

$$N \cong \frac{d \log_2\left(\frac{d}{e}\right)}{3.2}$$

$$= .3d \log_2\left(\frac{d}{e}\right)$$

taking $D = 3.2$ bits/letter as the redundancy of English.

Example:
If the period is $d = 27$, then $d/e \cong 10$ and $\log_2 (d/e) \cong 3.2$, so $N \cong 27$. ■

2.2 SIMPLE SUBSTITUTION CIPHERS

There are four types of substitution ciphers: simple substitution, homophonic substitution, polyalphabetic substitution, and polygram substitution. Simple substitution ciphers replace each character of plaintext with a corresponding character of ciphertext; a single one-to-one mapping from plaintext to ciphertext characters is used to encipher an entire message. Homophonic substitution ciphers are similar, except the mapping is one-to-many, and each plaintext character is enciphered with a variety of ciphertext characters. Polyalphabetic substitution ciphers use multiple mappings from plaintext to ciphertext characters; the mappings are usually one-to-one as in simple substitution. Polygram substitution ciphers are the most general, permitting arbitrary substitutions for groups of characters.

A **simple substitution cipher** replaces each character of an ordered **plaintext alphabet**, denoted $\mathcal{A}$, with the corresponding character of an ordered **cipher alphabet**, denoted $\mathcal{C}$ (typically $\mathcal{C}$ is a simple rearrangement of the lexicographic order of the characters in $\mathcal{A}$). Let $\mathcal{A}$ be an n-character alphabet $\{a_0, a_1, \ldots, a_{n-1}\}$. Then $\mathcal{C}$ is an n-character alphabet $\{f(a_0), f(a_1), \ldots, f(a_{n-1})\}$, where $f\colon \mathcal{A} \rightarrow \mathcal{C}$ is a one-to-one mapping of each character of $\mathcal{A}$ to the corresponding character of $\mathcal{C}$. The key to the cipher is given by $\mathcal{C}$ or, equivalently, by the function f.

To encipher, a plaintext message $M = m_1m_2 \ldots$ is written as the ciphertext message:

$$E_K(M) = f(m_1)f(m_2) \ldots .$$

Example:
Suppose that f maps the standard English alphabet $\mathcal{A} = \{\text{A, B}, \ldots, \text{Z}\}$ into the cipher alphabet $\mathcal{C}$ shown next:

$\mathcal{A}$: ABCDEFGHIJKLMNOPQRSTUVWXYZ
$\mathcal{C}$: HARPSICODBEFGJKLMNQTUVWXYZ .

Then the plaintext RENAISSANCE is enciphered as:

$$\begin{aligned} M &= \text{RENAISSANCE} \\ E_K(M) &= \text{NSJHDQQHJRS} . \quad \blacksquare \end{aligned}$$

The preceding example uses a **keyword mixed alphabet**: the cipher alphabet is constructed by first listing the letters of a keyword (in this case HARPSICHORD), omitting duplicates, and then listing the remaining letters of the alphabet in order.

Ciphers based on **shifted alphabets** (sometimes called "direct standard alphabets" [Sink66]) shift the letters of the alphabet to the right by k positions, modulo the size of the alphabet. Formally,

$$f(a) = (a + k) \bmod n ,$$

where n is the size of the alphabet $\mathcal{A}$, and "a" denotes both a letter of $\mathcal{A}$ and its position in $\mathcal{A}$. For the standard English alphabet, $n = 26$ and the positions of the letters are as follows:

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

We noted in Chapter 1 that this type of cipher is called a **Caesar cipher**, because Julius Caesar used it with $k = 3$. Under the Caesar cipher, our plaintext RENAISSANCE is enciphered as

$$\begin{aligned} M &= \text{RENAISSANCE} \\ E_K(M) &= \text{UHQDLVVDQFH} . \end{aligned}$$

More complex transformations of the plaintext alphabet are possible. Ciphers based on multiplications (sometimes called "decimations" [Sink66]) of the standard alphabet multiply each character by a key k; that is,

$$f(a) = ak \bmod n ,$$

where k and n are relatively prime so that the letters of the alphabet produce a complete set of residues (see Section 1.6.2).

Example:

If $k = 9$ and $\mathcal{A}$ is the standard English alphabet, we have

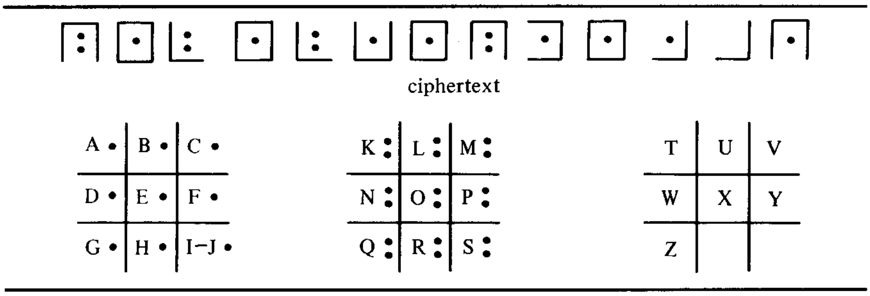
$\mathcal{A}$ = ABC DEF GHI JKL MNO PQR STU VWX YZ
$\mathcal{C}$ = AJS BKT CLU DMV ENW FOX GPY HQZ IR ,

whence

$$\begin{aligned} M &= \text{RENAISSANCE} \\ E_K(M) &= \text{XKNAUGGANSK} . \quad \blacksquare \end{aligned}$$

Previous Chapter Previous Page Home Next Page Next Chapter

FIGURE 2.2 Churchyard cipher.



If k and n are not relatively prime, several plaintext letters will encipher to the same ciphertext letter, and not all letters will appear in the ciphertext alphabet. For example, if $n = 26$ and $k = 13$,

$$f(\mathrm{A}) = f(\mathrm{C}) = f(\mathrm{E}) = \cdots = f(\mathrm{Y}) = \mathrm{A}\ (0)$$
$$f(\mathrm{B}) = f(\mathrm{D}) = f(\mathrm{F}) = \cdots = f(\mathrm{Z}) = \mathrm{N}\ (13)\ .$$

Addition (shifting) and multiplication can be combined to give an **affine transformation** (linear plus a constant):

$$f(a) = (ak_1 + k_0) \bmod n\ ,$$

where k_1 and n are relatively prime.

Higher-order transformations are obtained with **polynomial transformations** of degree t:

$$f(a) = (a^t k_t + a^{t-1} k_{t-1} + \cdots + ak_1 + k_0) \bmod n\ .$$

Caesar ciphers are polynomial transformations of degree 0, while affine transformations are of degree 1.

Some substitution ciphers use nonstandard ciphertext alphabets. For example, the Churchyard cipher shown in Figure 2.2 was engraved on a tombstone in Trinity Churchyard, New York, in 1794 [Kruh77]. The key to the cipher is given by the "tic-tac-toe" diagrams. The solution is left as an exercise to the reader. A similar cipher was also engraved on a tombstone in St. Paul's Churchyard, New York, in 1796. The first published solution to the ciphers appeared in the *New York Herald* in 1896—over 100 years later.

Messages have also been encoded in musical symbols [Sam79]. A common method was a simple substitution of individual notes for letters. Several composers used such schemes to encode the names of people into their works, and then build themes around the encoding. Bach, for example, incorporated his own name (which in German usage can be written B$^\flat$-A-C-B) into the "Musical Offering" and the "Art of the Fugue".

Let us now calculate the number of letters needed to break general substitution alphabets of size n. Observe that the number of possible keys is $n!$—the number of ways of arranging the n letters of the alphabet. If all keys are equally likely, the unicity distance is

$$N \cong \frac{H(K)}{D} = \frac{\log_2 n!}{D} .$$

For English, $N \cong \log_2 26!/3.2 \cong 88.4/3.2 \cong 27.6$. Thus, approximately 27 or 28 letters are needed to break these ciphers by frequency analysis. This explains the difficulty in solving the Churchyard ciphers, which contained only 15 characters. In practice, ciphers with at least 25 characters representing a "sensible" message in English can be readily solved [Frie67].

Ciphers based on polynomial transformations have smaller unicity distances and are easier to solve. For shifted alphabets, the number of possible keys is only 26; the unicity distance, derived in Section 1.4.3; is:

$$N \cong \frac{\log_2 26}{3.2} \cong 1.5 .$$

FIGURE 2.3 Frequency distribution of letters.

Char	Expected	Actual %
A	8.0	7.5 ***************
B	1.5	1.4 ***
C	3.0	4.1 ********
D	4.0	3.2 ******
E	13.0	12.7 *************************
F	2.0	2.3 *****
G	1.5	1.9 ****
H	6.0	3.8 ********
I	6.5	7.7 ***************
J	0.5	0.2
K	0.5	0.4 *
L	3.5	3.8 ********
M	3.0	3.0 ******
N	7.0	7.0 **************
O	8.0	7.5 ***************
P	2.0	3.0 ******
Q	0.2	0.2
R	6.5	6.7 *************
S	6.0	7.3 ***************
T	9.0	9.2 ******************
U	3.0	2.8 ******
V	1.0	1.0 **
W	1.5	1.4 ***
X	0.5	0.3 *
Y	2.0	1.6 ***
Z	0.2	0.1

Each * represents 0.5 percent.
Number of Letters = 67375

2.2.1 Single-Letter Frequency Analysis

Simple substitution ciphers are generally easy to break in a ciphertext-only attack using single-letter frequency distributions. Figure 2.3 shows the frequency distribution of the letters in the file used previously for Figure 2.1. The letters had a slightly different distribution than expected based on other sources; for comparison, the percentages in a commonly used table described in Kahn [Kahn67] are shown alongside the actual percentages.

Kahn partitions the letters into subsets of high, medium, low, and rare frequency of usage as shown in Figure 2.4. Other sources give slightly different frequency distributions, but the letters fall into these same general categories. Except for H (which was too low) and P (which was too high), our letters also fell into these categories.

Computer files may have a frequency distribution quite different from that of English text. Figure 2.5 shows the distribution of ASCII characters in the Pascal program that computed the frequency distributions. Note the large number of blanks.

By comparing the letter frequencies in a given ciphertext with the expected frequencies, a cryptanalyst can match the ciphertext letters with the plaintext letters. Digram and trigram distributions are also helpful.

Ciphers based on shifted alphabets are usually easy to solve, because each ciphertext letter is a constant distance from its corresponding plaintext letter.

Ciphers based on affine transformations of the form

$$f(a) = (ak_1 + k_0) \bmod n$$

are somewhat trickier [Sink66]. If a set of t correspondences (or suspected correspondences) between plaintext letters m_i and ciphertext letters c_i $(1 \leq i \leq t)$, then it may be possible to determine the multiplier k_1 and shift factor k_0 by solving the system of equations:

$$\begin{aligned}(m_1 k_1 + k_0) &\bmod n = c_1\\ &\vdots\\ (m_t k_1 + k_0) &\bmod n = c_t\,.\end{aligned}$$

Example:

Consider the following plaintext-ciphertext letter correspondences (the numbers in parentheses are the numerical equivalents):

FIGURE 2.4 Partitioning of letters by frequency.

high:	E T A O N I R S H
medium:	D L U C M
low:	P F Y W G B V
rare:	J K Q X Z

Plaintext	Ciphertext
E (4)	K (10)
J (9)	T (19)
N (13)	U (20)

This gives the three equations:

$$(4k_1 + k_0) \bmod 26 = 10 \quad (1)$$
$$(9k_1 + k_0) \bmod 26 = 19 \quad (2)$$
$$(13k_1 + k_0) \bmod 26 = 20 \quad (3)$$

Subtracting Eq. (1) from Eq. (2), we get

$$5k_1 \bmod 26 = 9 \ .$$

We can solve for k_1 using Eq. (1.6) (see Section 1.6.2), getting

$$\begin{aligned} k_1 &= [9 * inv(5, 26)] \bmod 26 \\ &= (9 * 21) \bmod 26 \\ &= 7 \ . \end{aligned}$$

Substituting in Eq. (1) gives

$$(28 + k_0) \bmod 26 = 10 \ ,$$

whence

$$k_0 = -18 \bmod 26 = 8 \ .$$

Note that we did not need Eq. (3) to solve for k_1 and k_0. We must, however, check that the solution satisfies Eq. (3). If it does not, then at least one of the three plaintext-ciphertext correspondences is incorrect (or else the cipher is not an affine transformation). In this case, the key does satisfy Eq. (3). ■

In general, we may need more than two equations to solve for k_0 and k_1. This is because equations of the form "ak mod 26 = c" have multiple solutions when a divides 26 (see Section 1.6.2).

Peleg and Rosenfeld [Pele79] describe a relaxation algorithm for solving general substitution ciphers. For each plaintext letter a and ciphertext letter b, an initial probability $Pr[f(a) = b]$ is computed based on single-letter frequencies, where $Pr[f(a) = b]$ is the probability that plaintext letter a is enciphered as b. These probabilities are then iteratively updated using trigram frequencies until the algorithm converges to a solution (set of high probability pairings).

2.3 HOMOPHONIC SUBSTITUTION CIPHERS

A **homophonic substitution cipher** maps each character a of the plaintext alphabet into a set of ciphertext elements $f(a)$ called **homophones.** Thus the mapping f from plaintext to ciphertext is of the form f: $\mathcal{A} \longrightarrow 2^{\mathcal{C}}$. A plaintext message $M = m_1 m_2 \ldots$ is enciphered as $C = c_1 c_2 \ldots$, where each c_i is picked at random from the set of homophones $f(m_i)$.

FIGURE 2.5 Distribution of ASCII characters in frequency analysis program.

```
Char                         Percent

        33.9  ********************************************************************
  !      0.0
  "      0.0
  #      0.0
  $      0.0
  %      0.0
  &      0.0
  '      1.2  **
  (      1.9  ****
  )      1.9  ****
  *      2.2  ****
  +      0.2
  ,      0.8  **
  -      0.2
  /      0.1
  0      0.5  *
  1      0.6  *
  2      0.2
  3      0.0
  4      0.0
  5      0.1
  6      0.0
  7      0.0
  8      0.1
  9      0.0
  :      1.6  ***
  ;      2.5  *****
  <      0.0
         1.1  **
  >      0.0
  ?      0.0
  @      0.0
  A      0.0
  B      0.0
  C      0.1
  D      0.0
  E      0.0
  F      0.0
  G      0.0
  H      0.0
  I      0.0
  J      0.0
  K      0.0
  L      0.0
  M      0.0
  N      0.0
  O      0.0
  P      0.0
```

```
Q       0.0
R       0.0
S       0.1
T       0.0
U       0.0
V       0.0
W       0.0
X       0.0
Y       0.0
Z       0.0
[       0.7   *
        0.0
]       0.7   *
^       0.0
        0.0
'       0.0
a       2.6   *****
b       0.6   *
c       3.3   *******
d       1.7   ***
e       6.4   *************
f       2.3   *****
g       0.7   *
h       1.9   ****
i       4.7   *********
j       0.1
k       0.0
l       2.0   ****
m       0.4   *
n       4.4   *********
o       2.8   ******
p       0.9   **
q       0.4   *
r       4.7   *********
s       1.6   ***
t       4.0   ********
u       0.9   **
v       0.1
w       1.1   **
x       0.4   *
y       0.4   *
z       0.1
{       0.0
|       0.0
}       0.0
~       0.0

Number of Characters in Sequence = 2449

Index of Coincidence = .13393
```

Example:

Suppose that the English letters are enciphered as integers between 0 and 99, where the number of integers assigned to a letter is proportional to the relative frequency of the letter, and no integer is assigned to more than one letter. The following illustrates a possible assignment of integers to the letters in the message PLAIN PILOT (for brevity, integer assignments for the remaining letters of the alphabet are not given):

Letter	Homophones
A	17 19 34 41 56 60 67 83
I	08 22 53 65 88 90
L	03 44 76
N	02 09 15 27 32 40 59
O	01 11 23 28 42 54 70 80
P	33 91
T	05 10 20 29 45 58 64 78 99

One possible encipherment of the message is:

$$\begin{array}{llllllllllll} M = & P & L & A & I & N & P & I & L & O & T & \\ C = & 91 & 44 & 56 & 65 & 59 & 33 & 08 & 76 & 28 & 78 & . \end{array}$$ ■

Kahn [Kahn67] notes the first known Western use of a homophonic cipher appears in correspondence between the Duchy of Mantua and Simeone de Crema in 1401. Multiple substitutions were assigned only to vowels; consonants were assigned single substitutions.

Homophonic ciphers can be much more difficult to solve than simple substitution ciphers, especially when the number of homophones assigned to a letter is proportional to the relative frequency of the letter. This is because the relative distribution of the ciphertext symbols will be nearly flat, confounding frequency analysis. A homophonic cipher may still be breakable, however, if other statistical properties of the plaintext (e.g., digram distributions) are apparent in the ciphertext (e.g., see [Prat42,Stah73]).

Clearly, the more ciphertext symbols available to distribute among the plaintext letters, the easier it is to construct a strong cipher. In the limiting case where each letter of plaintext enciphers into a unique ciphertext symbol, the cipher can be unbreakable.

2.3.1 Beale Ciphers

For over a century amateur cryptanalysts have been trying to solve a cipher purported to point to a treasure buried in Virginia around 1820 by a party of adventurers led by Thomas Jefferson Beale. The cipher is the first of three ciphers left by Beale. The second cipher, solved by James Ward [Ward85] in the 1880s, describes the alleged treasure and states that the first cipher contains directions for finding it. The treasure consists of gold, silver, and jewels worth millions of

FIGURE 2.6 Declaration of Independence (first 107 words).

(1) When, in the course of human events, it becomes necessary
(11) for one people to dissolve the political bands which have
(21) connected them with another, and to assume among the Powers
(31) of the earth the separate and equal station to which
(41) the Laws of Nature and of Nature's God entitle them,
(51) a decent respect to the opinions of mankind requires that
(61) they should declare the causes which impel them to the
(71) separation. We hold these truths to be self-evident; that
(81) all men are created equal, that they are endowed by
(91) their Creator with certain unalienable rights; that among
(99) these are Life, Liberty, and the pursuit of Happiness.

dollars today. The third cipher is supposed to list the next of kin of the adventurers.

The second cipher (B2) is an interesting example of a homophonic substitution cipher. The key is the Declaration of Independence (DOI). The words of the DOI are numbered consecutively as shown in Figure 2.6.

Beale enciphered each letter in the plaintext message by substituting the number of some word which started with that letter. The letter W, for example, was enciphered with the numbers 1, 19, 40, 66, 72, 290, and 459. The cipher begins

115 73 24 818 37 52 49 17 31 62 657 22 7 15 ,

which deciphers to "I have deposited . . .".

The first cipher (B1) has been "solved"—allegedly—by several persons, using techniques for solving homophonics plus anagramming to make the "solution" fit. No one admits having found the treasure, however, and there is considerable speculation that the whole thing is a hoax. Gillogly [Gill80] found a strange anomaly in B1, which he believes supports the hoax hypothesis. He deciphered B1 using the initial letters of the DOI, and discovered the sequence

ABFDEFGHIIJKLMMNOHPP

in the middle of the plaintext. Gillogly observed that the first F is encrypted as 195 and that word 194 begins with a C; similarly, the last H is encrypted as 301 and word 302 begins with an O. Hammer [Hamm79] found 23 encrypting errors in B1, so it is possible the F (for C) and H (for O) were "typos", and the original plaintext sequence was ABCDEFGHIIJKLMMNOOPP. Perhaps the sequence is a clue that the DOI is the key—not to abundant riches—but to a gigantic practical joke.

Meanwhile, members of the Beale Cipher Association pool their findings, hoping to either locate the treasure or simply solve the riddle. More information about these fascinating ciphers may also be found in [Beal78,Hamm71].

2.3.2 Higher-Order Homophonics

Given enough ciphertext C, most ciphers are theoretically breakable (in the Shannon sense—doing so may be computationally infeasible). This is because there will be a single key K that deciphers C into meaningful plaintext; all other keys will produce meaningless sequences of letters. Hammer [Hamm81] shows it is possible to construct higher-order homophonic ciphers such that any intercepted ciphertext will decipher into more than one meaningful message under different keys. For example, a ciphertext could decipher into the following two messages under different keys:

THE TREASURE IS BURIED IN GOOSE CREEK

THE BEALE CIPHERS ARE A GIGANTIC HOAX.

To construct a second-order homophonic cipher (one in which each ciphertext has two possible plaintext messages), the numbers 1 through n^2 are randomly inserted into an $n \times n$ matrix K whose rows and columns correspond to the characters of the plaintext alphabet $\mathcal{A}$. For each plaintext character a, row a of K defines one set of homophones $f_1(a)$ and column a defines another set of homophones $f_2(a)$. The set of rows thus corresponds to one key (mapping) f_1 and the set of columns to a second key f_2. A plaintext message $M = m_1m_2 \ldots$ is enciphered along with a dummy message $X = x_1x_2 \ldots$ to get ciphertext $C = c_1c_2 \ \ldots$, where

$$c_i = K[m_i, x_i] \quad i = 1, 2, \ldots .$$

Each ciphertext element c_i is thus selected from the intersection of the sets $f_1(m_i)$ and $f_2(x_i)$ and, therefore, deciphers to either m_i (under key f_1) or x_i (under f_2). A cryptanalyst cannot deduce the correct message from the ciphertext C because both M and X are equally likely. The intended recipient of the message can, however, decipher C knowing K.

Example:

Let $n = 5$. The following illustrates a 5×5 matrix for the plaintext alphabet {E, I, L, M, S}:

$$\begin{array}{c|ccccc}
 & \text{E} & \text{I} & \text{L} & \text{M} & \text{S} \\
\hline
\text{E} & 10 & 22 & 18 & 02 & 11 \\
\text{I} & 12 & 01 & 25 & 05 & 20 \\
\text{L} & 19 & 06 & 23 & 13 & 07 \\
\text{M} & 03 & 16 & 08 & 24 & 15 \\
\text{S} & 17 & 09 & 21 & 14 & 04
\end{array}$$

The message SMILE is enciphered with the dummy message LIMES as follows:

$$\begin{array}{lcccccc}
M = & \text{S} & \text{M} & \text{I} & \text{L} & \text{E} & \\
X = & \text{L} & \text{I} & \text{M} & \text{E} & \text{S} & \\
C = & 21 & 16 & 05 & 19 & 11 & . \ \blacksquare
\end{array}$$

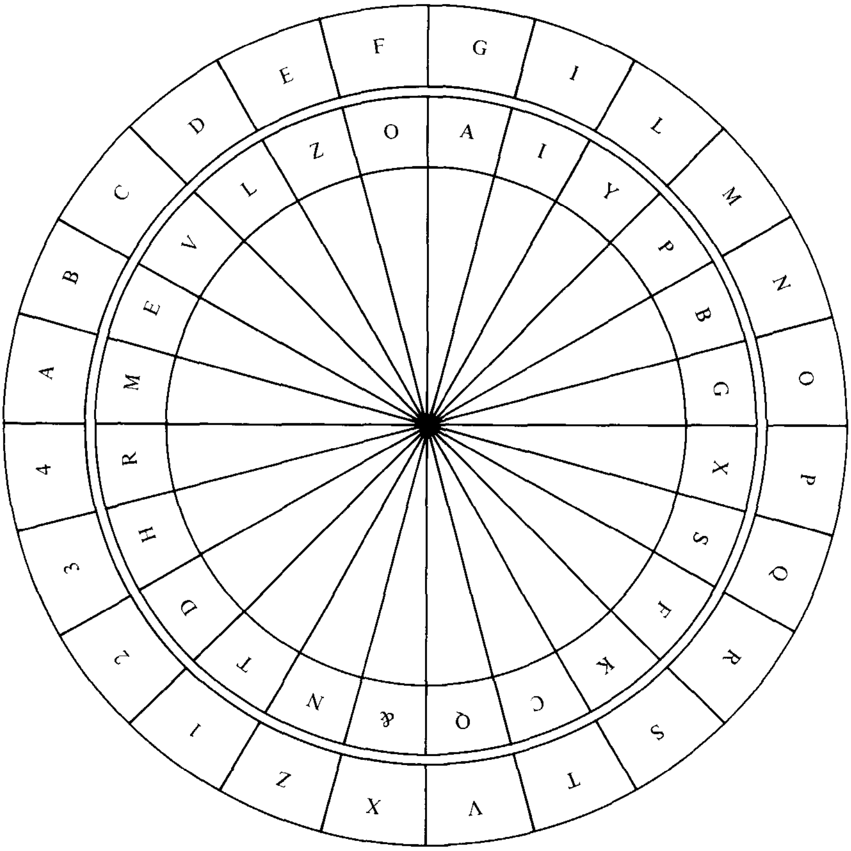
Previous Chapter Previous Page Home Next Page Next Chapter

Hammer investigated the possibility of B1 being a second-order homophonic. Beale might have enciphered messages M and X using two documents—for example, the DOI for M and the Magna Carta (MC) for X. Each ciphertext element c_i would then be some number j such that the jth word of the DOI begins with m_i and the jth word of the MC begins with x_i. Hammer deduced, however, that Beale had probably not used this method.

2.4 POLYALPHABETIC SUBSTITUTION CIPHERS

Because simple substitution ciphers use a single mapping from plaintext to ciphertext letters, the single-letter frequency distribution of the plaintext letters is preserved in the ciphertext. Homophonic substitutions conceal this distribution by defining multiple ciphertext elements for each plaintext letter. **Polyalphabetic substitution ciphers** conceal it by using multiple substitutions.

FIGURE 2.7 Cipher disk.



The development of polyalphabetic ciphers began with Leon Battista Alberti, the father of Western cryptography [Kahn67]. In 1568, Alberti published a manuscript describing a cipher disk that defined multiple substitutions (see Figure 2.7). In the outer circle Alberti placed 20 plaintext letters (H, K, and Y were not used and J, U, and W were not part of the Latin alphabet) and the numbers 1–4 (for special codes). In the movable inner circle he randomly placed the letters of the Latin alphabet plus "&". The disk thus defined 24 possible substitutions from the plaintext letters in the outer ring to the ciphertext letters in the inner ring, depending on the position of the disks. Alberti's important insight was his realization that the substitution could be changed during encipherment by turning the inner disk.

Most polyalphabetic ciphers are **periodic substitution ciphers** based on a period d. Given d cipher alphabets $\mathcal{C}_1, \ldots, \mathcal{C}_d$, let $f_i{:}\mathcal{A} \rightarrow \mathcal{C}_i$ be a mapping from the plaintext alphabet $\mathcal{A}$ to the ith cipher alphabet $\mathcal{C}_i$ $(1 \le i \le d)$. A plaintext message

$$M = m_1 \ldots m_d \, m_{d+1} \ldots m_{2d} \ldots$$

is enciphered by repeating the sequence of mappings $f_1, \ldots, f_d$ every d characters:

$$E_K(M) = f_1(m_1) \ldots f_d(m_d) \, f_1(m_{d+1}) \ldots f_d(m_{2d}) \ldots .$$

For the special case $d = 1$, the cipher is **monoalphabetic** and equivalent to simple substitution.

2.4.1 Vigenère and Beaufort Ciphers

A popular form of periodic substitution cipher based on shifted alphabets is the **Vigenère cipher**. As noted by Kahn [Kahn67], this cipher has been falsely attributed to the 16th Century French cryptologist Blaise de Vigenère. The key K is specified by a sequence of letters:

$$K = k_1 \ldots k_d \, ,$$

where k_i $(i = 1, \ldots, d)$ gives the amount of shift in the ith alphabet; that is,

$$f_i(a) = (a + k_i) \bmod n \; .$$

Example:

The encipherment of the word RENAISSANCE under the key BAND is shown next:

$$\begin{aligned} M &= \text{RENA ISSA NCE} \\ K &= \text{BAND BAND BAN} \\ E_K(M) &= \text{SEAD JSFD OCR} \; . \end{aligned}$$

In this example, the first letter of each four-letter group is shifted (mod 26) by 1, the second by 0, the third by 13, and the fourth by 3. ■

TABLE 2.1 Vigenère Tableau.

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

The Vigenère Tableau facilitated encryption and decryption (see Table 2.1). For plaintext letter a and key letter k, the ciphertext c is the letter in column a of row k. For ciphertext c, the plaintext a is the column containing c in row k.

The **Beaufort cipher** is similar, using the substitution

$$f_i(a) = (k_i - a) \bmod n.$$

Note that the same function can be used to decipher; that is, for ciphertext letter c,

$$f_i^{-1}(c) = (k_i - c) \bmod n\,.$$

The Beaufort cipher reverses the letters in the alphabet, and then shifts them to the right by $(k_i + 1)$ positions. This can be seen by rewriting f_i as follows:

$$f_i(a) = [(n - 1) - a + (k_i + 1)] \bmod n\,.$$

Example:

If k_i = D, the mapping from plaintext to ciphertext letters is given by $f_i(a)$ = (D $- a$) mod 26 as shown next:

$\mathcal{A}$: 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

$\mathcal{C}$: D C B A Z Y X W V U T S R Q P O N M L K J I H G F E . ■

The Vigenère Tableau can be used to encipher and decipher. For plaintext letter a, the ciphertext letter c is the row containing the key k in column a. For ciphertext letter c, the plaintext letter a is the column containing k in row c. The Beaufort cipher is named after the English admiral Sir Francis Beaufort, although it was first proposed by the Italian Giovanni Sestri in 1710 [Kahn67].

The **Variant Beaufort cipher** uses the substitution

$$f_i(a) = (a - k_i) \bmod n\,.$$

Because

$$(a - k_i) \bmod n = \big(a + (n - k_i)\big) \bmod n\,,$$

the Variant Beaufort cipher is equivalent to a Vigenère cipher with key character $(n - k_i)$.

The Variant Beaufort cipher is also the inverse of the Vigenère cipher; thus if one is used to encipher, the other is used to decipher.

The unicity distance for periodic substitution ciphers is easy to calculate from the individual substitution ciphers. If there are s possible keys for each simple substitution, then there are s^d possible keys if d substitutions are used. The unicity distance is thus

$$N \simeq \frac{H(K)}{D} = \frac{\log_2(s^d)}{D} = \frac{\log_2 s}{D}\, d\,.$$

If N ciphertext characters are required to break the individual substitution ciphers, then dN characters are required to break the complete cipher. For a Vigenère cipher with period d, $s = 26$, giving

$$N \cong \frac{4.7}{3.2} d \cong 1.5d \, .$$

To solve a periodic substitution cipher, the cryptanalyst must first determine the period of the cipher. Two tools are helpful: the index of coincidence and the Kasiski method.

2.4.2 Index of Coincidence

The **index of coincidence** (*IC*), introduced in the 1920s by William Friedman [Frie20] (see also [Kahn67]), measures the variation in the frequencies of the letters in the ciphertext. If the period of the cipher is 1 (i.e., simple substitution is used), there will be considerable variation in the letter frequencies and the *IC* will be high. As the period increases, the variation is gradually eliminated (due to diffusion) and the *IC* will be low.

Following Sinkov [Sink66], we shall derive the *IC* by first defining a **measure of roughness** (*MR*), which gives the variation of the frequencies of individual characters relative to a uniform distribution:

$$MR = \sum_{i=0}^{n-1} \left(p_i - \frac{1}{n}\right)^2 ,$$

where p_i is the probability that an arbitrarily chosen character in a random ciphertext is the ith character a_i in the alphabet ($i = 0, \ldots, n - 1$), and

$$\sum_{i=0}^{n-1} p_i = 1 \, .$$

For the English letters we have

$$\begin{aligned} MR &= \sum_{i=0}^{25} \left(p_i - \frac{1}{26}\right)^2 \\ &= \sum_{i=0}^{25} p_i^2 - \frac{2}{26} \sum_{i=0}^{25} p_i + 26\left(\frac{1}{26}\right)^2 \\ &= \sum_{i=0}^{25} p_i^2 - \frac{2}{26} + \frac{1}{26} \\ &= \sum_{i=0}^{25} p_i^2 - .038 \, . \end{aligned}$$

Because the period and, therefore, the probabilities are unknown, it is not possible to compute MR. But it is possible to estimate MR using the distribution of letter frequencies in the ciphertext. Observe that

$$MR + .038 = \sum_{i=0}^{25} p_i^2$$

is the probability that two arbitrarily chosen letters from a random ciphertext are the same. Now, the total number of pairs of letters that can be chosen from a given ciphertext of length N is $\binom{N}{2} = N(N-1)/2$. Let F_i be the frequency of the ith letter of English ($i = 0, \ldots, 25$) in the ciphertext; thus, $\sum_{i=0}^{25} F_i = N$. The number of pairs containing just the ith letter is

$$\frac{F_i(F_i - 1)}{2} \, .$$

The IC is defined to be the probability that two letters chosen at random from the given ciphertext are alike:

$$IC = \frac{\sum_{i=0}^{25} F_i(F_i - 1)}{N(N-1)} \, .$$

Because this is an estimate of $\sum_{i=0}^{25} p_i^2$, the IC is an estimate of $MR + .038$. But unlike MR, IC can be computed from the ciphertext.

Now, MR ranges from 0 for a flat distribution (infinite period) to .028 for English and ciphers with period 1. Thus, IC varies from .038 for an infinite period to .066 for a period of 1. For a cipher of period d, the expected value of IC is:

$$\left(\frac{1}{d}\right)\frac{N-d}{N-1}(.066) + \left(\frac{d-1}{d}\right)\frac{N}{N-1}(.038) \, .$$

Table 2.2 shows the expected value of IC for several values of d.

To estimate the period of a given cipher, the cryptanalyst measures the frequencies of the letters in the ciphertext, computes IC using Eq. (2.1), and

TABLE 2.2 Expected index of coincidence.

d	IC
1	.066
2	.052
3	.047
4	.045
5	.044
10	.041
large	.038

Previous Chapter Previous Page Home Next Page Next Chapter

finally compares this with the expected values shown in Table 2.2. Because *IC* is statistical in nature, it does not necessarily reveal the period exactly. Nevertheless, it may provide a clue as to whether the cipher is monoalphabetic, polyalphabetic with small period, or polyalphabetic with large period.

2.4.3 Kasiski Method

The **Kasiski method**, introduced in 1863 by the Prussian military officer Friedrich W. Kasiski [Kasi63], analyzes repetitions in the ciphertext to determine the exact period (see also [Kahn67,Gain56]). For example, suppose the plaintext TO BE OR NOT TO BE is enciphered with a Vigenère cipher using key HAM as shown next:

$$\begin{aligned} M &= \text{T O B E O R N O T T O B E} \\ K &= \text{H A M H A M H A M H A M H} \\ E_K(M) &= \underline{\text{A O N L}}\ \text{O D U O F}\ \underline{\text{A O N L}} \end{aligned}$$

Notice that the ciphertext contains two occurrences of the cipher sequence AONL, 9 characters apart.

Repetitions in the ciphertext occur when a plaintext pattern repeats at a distance equal to a multiple of the key length. Repetitions more than two cipher characters long are unlikely to occur by pure chance. If m ciphertext repetitions are found at intervals I_j $(1 \le j \le m)$, the period is likely to be some number that divides most of the m intervals. The preceding example has an interval $I_l = 9$, suggesting a period of 1, 3, or 9 (in fact it is 3).

The *IC* is useful for confirming a period d found by the Kasiski method or by exhaustive search (whereas this would be tedious and time-consuming by hand, a computer can systematically try $d = 1, 2, 3, \ldots$ until the period is found). Letting $c_l c_2 \ldots$ denote the ciphertext, the *IC* is computed for each of the sequences:

$$\begin{array}{l} 1\text{: } c_1\ c_{d+1}\ \ c_{2d+1}\ \ \ldots \\ 2\text{: } c_2\ c_{d+2}\ \ c_{2d+2}\ \ \ldots \\ \quad\ \cdot \\ \quad\ \cdot \\ \quad\ \cdot \\ d\text{: } c_d\ c_{2d}\ \ \ c_{3d}\ \ \ \ \ldots \end{array}$$

If each sequence is enciphered with one key, each will have an *IC* close to .066. Once the period is determined, each cipher alphabet is separately analyzed as for simple substitution.

Example:
We shall now apply the index of coincidence and Kasiski method to analyze the ciphertext shown in Figure 2.8. Figure 2.9 shows that the frequency distribution of the letters in the ciphertext is flatter than normal, and that the *IC* is .0434; this suggests a polyalphabetic substitution cipher with a period of about 5.

FIGURE 2.8 Sample ciphertext.

```
ZHYME ZVELK OJUBW CEYIN CUSML RAVSR YARNH CEARI UJPGP VARDU
QZCGR NNCAW JALUH GJPJR YGEGQ FULUS QFFPV EYEDQ GOLKA LVOSJ
TFRTR YEJZS RVNCI HYJNM ZDCRO DKHCR MMLNR FFLFN QGOLK ALVOS
JWMIK QKUBP SAYOJ RRQYI NRNYC YQZSY EDNCA LEILX RCHUG IEBKO
YTHGV VCKHC JEQGO LKALV OSJED WEAKS GJHYC LLFTY IGSVT FVPMZ
NRZOL CYUZS FKOQR YRYAR ZFGKI QKRSV IRCEY USKVT MKHCR MYQIL
XRCRL GQARZ OLKHY KSNFN RRNCZ TWUOC JNMKC MDEZP IRJEJ W
```

Previous Chapter Previous Page Home Next Page Next Chapter

FIGURE 2.9 Frequency distribution of ciphertext in Figure 2.8.

Char	Percent
A	4.0 ********
B	0.9 **
C	6.1 ************
D	2.0 ****
E	4.9 **********
F	3.5 *******
G	4.0 ********
H	3.2 ******
I	3.5 *******
J	4.6 *********
K	5.2 **********
L	5.8 ************
M	3.2 ******
N	4.6 *********
O	4.0 ********
P	2.0 ****
Q	3.8 ********
R	8.7 *****************
S	4.3 *********
T	2.0 ****
U	3.5 *******
V	4.0 ********
W	1.7 ***
X	0.6 *
Y	6.1 ************
Z	3.8 ********

Number of Characters = 346
Index of Coincidence = .0434

To determine the exact period, we observe there are three occurrences of the 11-character sequence QGOLKALVOSJ (see Figure 2.8), the first two separated by a distance of 51 characters, and the second two by 72 characters. Because 3 is the only common divisor of 51 and 72, the period is almost certain to be 3, even though the *IC* predicted a somewhat larger period.

Figure 2.10 shows the *IC* and frequency distribution for each of the sequences.

1: $c_1\ c_4\ c_7\ \ldots$
2: $c_2\ c_5\ c_8\ \ldots$
3: $c_3\ c_6\ c_9\ \ldots$.

Each ciphertext alphabet has an *IC* near or above .66, confirming the hypothesis that the period is 3.

FIGURE 2.10 Frequency distributions for separate sequences.

```
     Frequency Analysis for Sequence  1

Char  Percent

 A      0.0
 B      0.0
 C      6.0    ************
 D      1.7    ***
 E      4.3    *********
 F      5.2    **********
 G      2.6    *****
 H      0.9    **
 I      5.2    **********
 J      8.6    *****************
 K     13.8    ****************************
 L      1.7    ***
 M      0.9    **
 N      0.9    **
 O      0.0
 P      2.6    *****
 Q      1.7    ***
 R      9.5    *******************
 S      0.0
 T      0.9    **
 U      5.2    **********
 V     11.2    **********************
 W      1.7    ***
 X      1.7    ***
 Y      4.3    *********
 Z      9.5    *******************

Number of Characters in Sequence = 116

Index of Coincidence =  .06747
```

```
     Frequency Analysis for Sequence  2

Char  Percent

 A      9.6    *******************
 B      0.9    **
 C      1.7    ***
 D      2.6    *****
 E     10.4    *********************
 F      2.6    *****
 G      2.6    *****
 H      8.7    *****************
 I      5.2    **********
 J      0.0
 K      0.9    **
 L      1.7    ***
 M      1.7    ***
 N      9.6    *******************
 O     12.2    ************************
 P      1.7    ***
 Q      0.0
 R      4.3    *********
 S      8.7    *****************
 T      5.2    **********
 U      4.3    *********
 V      0.9    **
 W      0.9    **
 X      0.0
 Y      3.5    *******
 Z      0.0

Number of Characters in Sequence = 115

Index of Coincidence =  .06499
```

```
     Frequency Analysis for Sequence  3

Char  Percent

 A      2.6    *****
 B      1.7    ***
 C     10.4    *********************
 D      1.7    ***
 E      0.0
 F      2.6    *****
 G      7.0    **************
 H      0.0
 I      0.0
 J      5.2    **********
 K      0.9    **
 L     13.9    ****************************
 M      7.0    **************
 N      3.5    *******
 O      0.0
 P      1.7    ***
 Q      9.6    *******************
 R     12.2    ************************
 S      4.3    *********
 T      0.0
 U      0.9    **
 V      0.0
 W      2.6    *****
 X      0.0
 Y     10.4    *********************
 Z      1.7    ***

Number of Characters in Sequence = 115

Index of Coincidence =  .07597
```

To determine the type of cipher, we might first consider a simple Vigenère or Beaufort cipher. Looking at the second alphabet, we see a surprising similarity between this distribution and the expected distribution for English letters (see Figure 2.3). This suggests a Vigenère cipher with an A as the second key character, whence the letters c_2, c_5, . . . are already in plaintext. We leave finding the first and third characters and deciphering the text as an exercise for the reader. The plaintext is one of Raymond Smullyan's† gems. ■

2.4.4 Running-Key Ciphers

The security of a substitution cipher generally increases with the key length. In a **running-key cipher**, the key is as long as the plaintext message, foiling a Kasiski attack (assuming the key does not repeat). One method uses the text in a book (or any other document) as a key sequence in a substitution cipher based on shifted alphabets (i.e., a nonperiodic Vigenère cipher). The key is specified by the title of the book and starting position (section, paragraph, etc.).

Example:
Given the starting key: "*Cryptography and Data Security,* Section 2.3.1, paragraph 2," the plaintext message "THE TREASURE IS BURIED . . ." is enciphered as

$$\begin{aligned} M &= \text{THETREASUREISBURIED}\ldots \\ K &= \text{THESECONDCIPHERISAN}\ldots \\ E_K(M) &= \text{MOILVGOFXTMXZFLZAEQ}\ldots \end{aligned}$$

(the string "(B2)" in the text was omitted from the key sequence because some of the characters are not letters). ■

Because perfect secrecy is possible using key sequences as long as the messages they encipher, one might expect a running key cipher to be unbreakable. This is not so. If the key has redundancy (as with English text), the cipher may be breakable using Friedman's methods [Frie18] (see also [Kahn67,Diff79]). Friedman's approach is based on the observation that a large proportion of letters in the ciphertext will correspond to encipherments where both the plaintext and key letters fall in the high frequency category (see Figure 2.4).

Example:
In our earlier example, 12 of the 19 ciphertext letters came from such pairs, as noted next:

```
    ↓↓↓↓↓ ↓↓  ↓ ↓  ↓↓↓
M = THETREASUREISBURIED ...
K = THESECONDCIPHERISAN ... .
```

†R. Smullyan, *This Book Needs No Title,* Prentice-Hall, Englewood Cliffs, N.J., 1980.

Of the remaining 7 pairs, either the plaintext or key letter belongs to the high frequency category in 6 of the pairs, and both belong to the medium category in the remaining pair. ■

Friedman recommends starting with the assumption that all ciphertext letters correspond to high frequency pairs, thus reducing the number of initial possibilities for each plaintext and key letter. The initial guesses are then related to digram and trigram distributions (and possibly probable words) to determine the actual pairings.

Example:
Consider the first three ciphertext letters MOI in the preceding example. There are 26 possible plaintext-key letter pairings for each ciphertext letter (assuming a shifted substitution). Examining the possible pairs for M, we see there are only three pairs where both the plaintext and key character fall in the high frequency category:

					↓				↓											↓						
plaintext letter:	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
key letter:	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
ciphertext letter:	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M

The high frequency pairs for all three letters are:

M	O	I
E–I	A–O	A–I
I–E	O–A	I–A
T–T	H–H	E–E
		R–R

Now, there are 3 * 3 * 4 = 36 possible combinations of the preceding pairs:

plaintext:	EAA	EAI	. . .	THE	. . .	THR
key:	IOI	IOA	. . .	THE	. . .	THR
ciphertext:	MOI	MOI	. . .	MOI	. . .	MOI .

Many of the trigrams for either the plaintext or key are very unlikely (e.g., EAA and IOI in the first combination). The trigram THE occurring simultaneously in both the plaintext and key is the most likely, and a cryptanalyst making this choice will have correctly guessed the first word of both the plaintext and key. ■

2.4.5 Rotor and Hagelin Machines

Rotor and Hagelin machines implement polyalphabetic substitution ciphers with a long period.

A rotor machine consists of a bank of t rotors or wired wheels. The perimeter of each rotor R_i has 26 electrical contacts (one for each letter of the alphabet) on both its front and rear faces. Each contact on the front face is wired to a contact on the rear face to implement a mapping f_i from plaintext to ciphertext letters. R_i can rotate into 26 positions, and each position alters the mapping. When R_i is in position j_i, the mapping is defined by

$$F_i(a) = (f_i(a - j_i) \bmod 26 + j_i) \bmod 26 \ .$$

A plaintext letter (signal) enters the bank of rotors at one end, travels through the rotors in succession, and emerges as ciphertext at the other end. A machine with t rotors effectively implements a substitution cipher composed of $F_1, \ldots, F_t$. The ith plaintext letter m_i of a message $M = m_1 m_2 \ldots$ is enciphered as

$$E_{k_i}(m_i) = F_t \circ \ldots \circ F_1(a) \ ,$$

where k_i consists of the wired mappings $f_1, \ldots, f_t$ and the positions $j_1, \ldots, j_t$ of the rotors.

The wirings plus initial positions of the rotors determine the starting key. As each plaintext letter is enciphered, one or more of the rotors moves to a new position, changing the key. A machine with t rotors does not return to its starting position until after 26^t successive encipherments; a machine with 4 rotors, for example, thus has a period of $26^4 = 456{,}976$ letters; one with 5 rotors has a period of 11,881,376 letters. The Enigma, invented by Arthur Scherbius and used by the Germans up through World War II, uses an odometer rotor motion. The first rotor advances to the next position after each character is enciphered. After it has made a complete rotation, the second rotor advances to its next position. Similarly, after the second rotor has made a complete rotation, the third advances, and so on until all rotors have moved through all positions. (See [Diff79,Kahn67,Konh81, Deav80a,Deav80b] for additional information about these machines, including methods used to break them.)

The Hagelin machines, invented by Boris Hagelin [Kahn67] in the 1920s and 1930s, use keywheels with pins. There are t wheels, and each wheel has p_i pins ($1 \le i \le t$), where the p_i are relatively prime. The Hagelin C-48, for example, has 6 wheels with 17, 19, 21, 23, 25, and 26 pins, respectively. The pins can be pushed either left or right, and the combined setting of the pins and positions of the wheels determine a key. Letting k_i be the key when the ith plaintext character m_i is enciphered, m_i is enciphered as in a Beaufort substitution:

$$E_{k_i}(m_i) = (k_i - m_i) \bmod 26$$

(deciphering is the same). After each ciphertext character is enciphered, all t wheels are rotated one position forward. Because the p_i do not have any common factors, the wheels do not return to their starting position until after $\prod_{i=1}^{t} p_i$ encipherments. The C-48, for example, has a period of over 100 million. Hagelin machines were used extensively during World War II, and are still used today. (See also [Diff79,Kahn67,Bark77,Rive81]).

Although rotor and Hagelin machines generate long key streams, they are not as random as they might seem, and the machines are vulnerable to cryptanalysis (see preceding references). Techniques for generating key streams that seem less vulnerable are described in Chapter 3.

2.4.6 Vernam Cipher and One-Time Pads

If the key to a substitution cipher is a random sequence of characters and is not repeated, there is not enough information to break the cipher. Such a cipher is called a **one-time pad**, as it is only used once.

The implementation of one-time pads in computer systems is based on an ingenious device designed by Gilbert Vernam in 1917 [Kahn67]. An employee of American Telephone and Telegraph Company (A. T. & T.), Vernam designed a cryptographic device for telegraphic communications based on the 32-character Baudot code of the new teletypewriters developed at A. T. & T. Each character is represented as a combination of five marks and spaces, corresponding to the bits 1 and 0 in digital computers. A nonrepeating random sequence of key characters is punched on paper tape, and each plaintext bit is added modulo 2 to the next key bit. Letting $M = m_1 m_2 \ldots$ denote a plaintext bit stream and $K = k_1 k_2 \ldots$ a key bit stream, the **Vernam cipher** generates a ciphertext bit stream $C = E_K(M) = c_1 c_2, \ldots$, where

$$c_i = (m_i + k_i) \bmod 2 \ , \quad i = 1, 2, \ldots \ .$$

The cipher is thus like a Vigenère cipher over the binary alphabet $\{0, 1\}$.

The Vernam cipher is efficiently implemented in microelectronics by taking the "exclusive-or" of each plaintext/key pair (see Section 1.6.3):

$$c_i = m_i \oplus k_i \ .$$

Because $k_i \oplus k_i = 0$ for $k_i = 0$ or 1, deciphering is performed with the same operation:

$$\begin{aligned} c_i \oplus k_i &= m_i \oplus k_i \oplus k_i \\ &= m_i \ . \end{aligned}$$

Example:

If the plaintext character A (11000 in Baudot) is added to the key character D (10010 in Baudot), the result is:

$$\begin{aligned} M &= 1\ 1\ 0\ 0\ 0 \\ K &= 1\ 0\ 0\ 1\ 0 \\ E_K(M) &= 0\ 1\ 0\ 1\ 0\ . \ \blacksquare \end{aligned}$$

Army cryptologist Major Joseph Mauborgne suggested the one-time use of each key tape, and thus the most formidable cipher of all was born. The only drawback of the cipher is that it requires a long key sequence; we shall return to this problem in the next chapter.

If the key to a Vernam cipher is repeated, the cipher is equivalent to a running-key cipher with a text as key. To see why, suppose two plaintext streams M and M' are enciphered with a key stream K, generating ciphertext streams C and C', respectively. Then

$$\begin{aligned} c_i &= m_i \oplus k_i, \\ c_i' &= m_i' \oplus k_i, \end{aligned}$$

for $i = 1, 2, \ldots$. Let C'' be a stream formed by taking the $\oplus$ of C and C'; then

$$\begin{aligned} c_i'' &= c_i \oplus c_i' \\ &= m_i \oplus k_i \oplus m_i' \oplus k_i \\ &= m_i \oplus m_i' , \end{aligned}$$

for $i = 1, 2, \ldots$. The stream C'' is thus equivalent to a stream generated by the encipherment of message M with message (key) M' and is no longer unbreakable. Note that this ciphertext is broken by finding the plaintext rather than the key. Of course, once the plaintext is known, the key is easily computed (each $k_i = c_i \oplus m_i$).

2.5 POLYGRAM SUBSTITUTION CIPHERS

All of the preceding substitution ciphers encipher a single letter of plaintext at a time. By enciphering larger blocks of letters, **polygram substitution ciphers** make cryptanalysis harder by destroying the significance of single-letter frequencies.

2.5.1 Playfair Cipher

The **Playfair cipher** is a digram substitution cipher named after the English scientist Lyon Playfair; the cipher was actually invented in 1854 by Playfair's friend, Charles Wheatstone, and was used by the British during World War I [Kahn67]. The key is given by a 5 × 5 matrix of 25 letters (J was not used), such as the one shown in Figure 2.11. Each pair of plaintext letters $m_1 m_2$ is enciphered according to the following rules:

1. If m_1 and m_2 are in the same row, then c_1 and c_2 are the two characters to the right of m_1 and m_2, respectively, where the first column is considered to be to the right of the last column.
2. If m_1 and m_2 are in the same column, then c_1 and c_2 are the two characters

FIGURE 2.11 Key for Playfair cipher.

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

below m_1 and m_2, respectively, where the first row is considered to be below the last row.

3. If m_1 and m_2 are in different rows and columns, then c_1 and c_2 are the other two corners of the rectangle having m_1 and m_2 as corners, where c_1 is in m_1's row and c_2 is in m_2's row.
4. If $m_1 = m_2$, a null letter (e.g., X) is inserted into the plaintext between m_1 and m_2 to eliminate the double.
5. If the plaintext has an odd number of characters, a null letter is appended to the end of the plaintext.

Example:

To encipher the first two letters of RENAISSANCE, observe that R and E are two corners of the rectangle

$$\begin{matrix} \mathrm{H} & \mathrm{A} & \mathrm{R} \\ \mathrm{I} & \mathrm{C} & \mathrm{O} \\ \mathrm{E} & \mathrm{F} & \mathrm{G} \end{matrix} .$$

They are thus enciphered as the other two corners, H and G. The entire plaintext is enciphered as:

$$\begin{array}{lllllll} M & = \mathrm{RE} & \mathrm{NA} & \mathrm{IS} & \mathrm{SA} & \mathrm{NC} & \mathrm{EX} \\ E_K(M) & = \mathrm{HG} & \mathrm{WC} & \mathrm{BH} & \mathrm{HR} & \mathrm{WF} & \mathrm{GV} \, . \end{array} \quad \blacksquare$$

2.5.2 Hill Cipher

The **Hill cipher** [Hill29] performs a linear transformation on d plaintext characters to get d ciphertext characters. Suppose $d = 2$, and let $M = m_1m_2$. M is enciphered as $\mathrm{C} = E_K(M) = c_1c_2$, where

$$\begin{aligned} c_1 &= (k_{11}m_1 + k_{12}m_2) \bmod n \\ c_2 &= (k_{21}m_1 + k_{22}m_2) \bmod n \, . \end{aligned}$$

Expressing M and C as the column vectors $M = (m_1, m_2)$ and $C = (c_1, c_2)$, this can be written as

$$C = E_K(M) = KM \bmod n \, ,$$

where K is the matrix of coefficients:

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} . \quad \blacksquare$$

That is,

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \bmod n \, .$$

Deciphering is done using the inverse matrix K^{-1}:

$$\begin{aligned} D_K(C) &= K^{-1}C \bmod n \\ &= K^{-1}KM \bmod n \\ &= M \ , \end{aligned}$$

where $KK^{-1} \bmod n = I$, and I is the 2×2 identity matrix.

Example:
Let n be 26 and let K and K^{-1} be as follows:

$$\overset{K}{\begin{pmatrix} 3 & 2 \\ 3 & 5 \end{pmatrix}} \overset{K^{-1}}{\begin{pmatrix} 15 & 20 \\ 17 & 9 \end{pmatrix}} \bmod 26 = \overset{I}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} .$$

Suppose we wish to encipher the plaintext message EG, which corresponds to the column vector (4, 6). We compute

$$\begin{aligned} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} &= \begin{pmatrix} 3 & 2 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} 4 \\ 6 \end{pmatrix} \bmod 26 \\ &= \begin{pmatrix} 24 \\ 16 \end{pmatrix} , \end{aligned}$$

getting the ciphertext $C = (24, 16)$ or YQ. To decipher, we compute

$$\begin{pmatrix} 15 & 20 \\ 17 & 9 \end{pmatrix} \begin{pmatrix} 24 \\ 16 \end{pmatrix} \bmod 26 = \begin{pmatrix} 4 \\ 6 \end{pmatrix} . \ \blacksquare$$

Because the enciphering transformation is linear, it may be possible to solve for the key with just a few characters of known plaintext. Suppose $d = 2$, and the cryptanalyst has the following correspondences of plaintext and ciphertext:

$$\begin{aligned} M_1 &= (m_1, m_2),\ C_1 = (c_1, c_2) \\ M_2 &= (m_3, m_4),\ C_2 = (c_3, c_4) \ . \end{aligned}$$

Let M and C be the following matrices of column vectors:

$$\begin{aligned} M &= (M_1, M_2) = \begin{pmatrix} m_1 & m_3 \\ m_2 & m_4 \end{pmatrix} \\ C &= (C_1, C_2) = \begin{pmatrix} c_1 & c_3 \\ c_2 & c_4 \end{pmatrix} . \end{aligned}$$

We have

$$C = KM \bmod n \ .$$

If M is nonsingular, the cryptanalyst can compute its inverse and solve for K:

$$K = CM^{-1} \bmod n \ .$$

This strategy can be applied for any d, and requires only $O(d^3)$ operations to compute the inverse.

2.6 PRODUCT CIPHERS

A **product cipher** E is the composition of t functions (ciphers) $F_1, \ldots, F_t$, where each F_i may be a substitution or transposition. Rotor machines are product ciphers, where F_i is implemented by rotor R_i ($1 \leq i \leq t$).

2.6.1 Substitution-Permutation Ciphers

Shannon [Shan49] proposed composing different kinds of functions to create "mixing transformations", which randomly distribute the meaningful messages uniformly over the set of all possible ciphertext messages. Mixing transformations could be created, for example, by applying a transposition followed by an alternating sequence of substitutions and simple linear operations.

This approach is embodied in the LUCIFER cipher, designed at IBM by Feistel [Feis73]. LUCIFER uses a transformation that alternately applies substitutions and transpositions.

Figure 2.12 illustrates how the basic principle is applied to 12-bit blocks (in practice, longer blocks should be used). The cipher alternatively applies substitutions S_i and permutations P_i, giving

$$C = E_K(M) = S_t \circ P_{t-1} \circ \ldots \circ S_2 \circ P_1 \circ S_1(M) \ ,$$

where each S_i is a function of the key K. The substitutions S_i are broken into 4 smaller substitutions $S_{i1}, \ldots, S_{i4}$, each operating on a 3-bit subblock to reduce the complexity of the microelectronic circuits. Because the permutations P_i shuffle all

FIGURE 2.12 Substitution-permutation cipher.

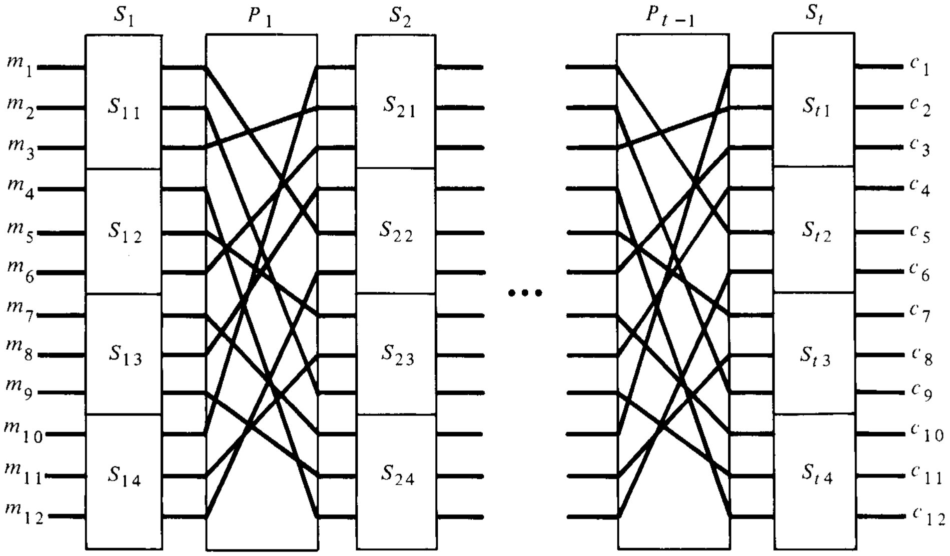
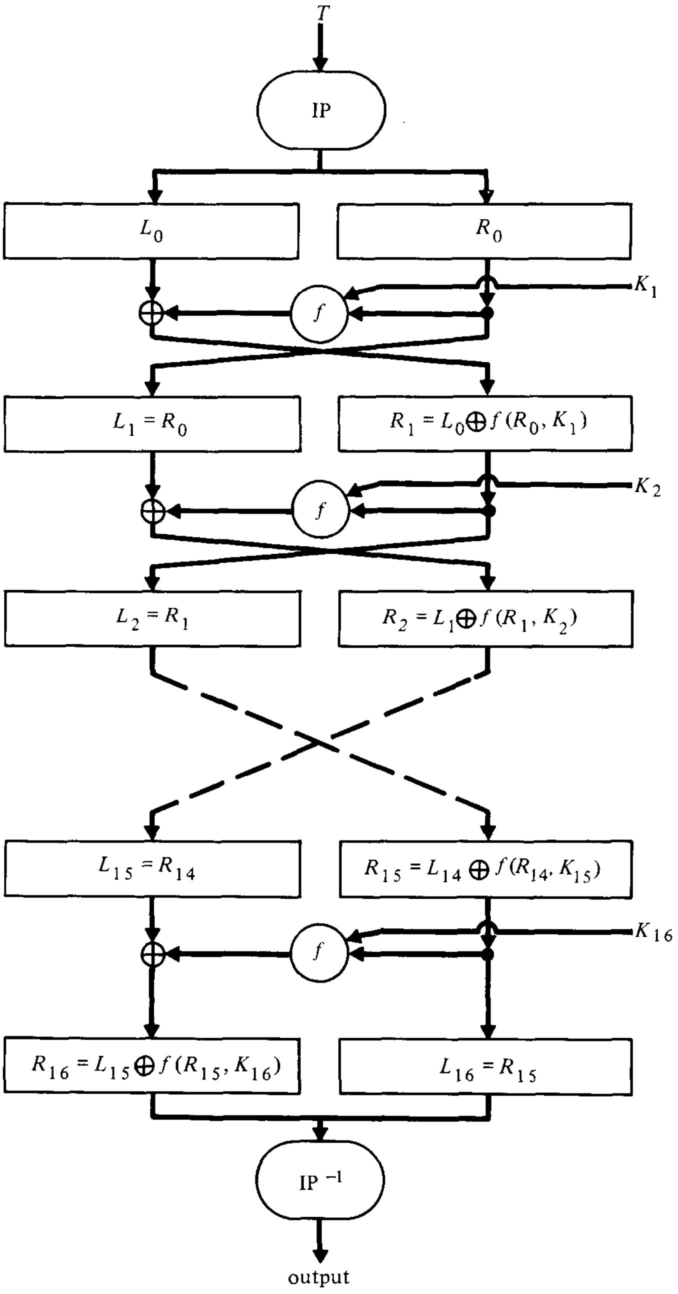


FIGURE 2.13 DES enciphering algorithm.


T
IP
L_0
R_0
K_1
f
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0, K_1)$
K_2
f
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1, K_2)$
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
f
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
$L_{16} = R_{15}$
IP^{-1}
output

12-bits, each bit of plaintext can conceivably affect each bit of ciphertext. (See [Feis70,Feis75,Kam78] for details on the design of substitution-permutation ciphers.)

2.6.2 The Data Encryption Standard (DES)

In 1977 the National Bureau of Standards announced a Data Encryption Standard to be used in unclassified U.S. Government applications [NBS77]. The encryption algorithm was developed at IBM, and was the outgrowth of LUCIFER.

DES enciphers 64-bit blocks of data with a 56-bit key. There is disagreement over whether a 56-bit key is sufficiently strong; we shall discuss this controversy after describing the algorithm.

The algorithm—which is used both to encipher and to decipher—is summarized in Figure 2.13. An input block T is first transposed under an initial permutation IP, giving $T_0 = \mathrm{IP}(T)$. After it has passed through 16 iterations of a function f, it is transposed under the inverse permutation IP^{-1} to give the final result. The permutations IP and IP^{-1} are given in Tables 2.3(a) and 2.3(b), respectively. These tables (as well as the other permutation tables described later) should be read left-to-right, top-to-bottom. For example, IP transposes $T = t_1 t_2 \ldots t_{64}$ into $T_0 = t_{58} t_{50} \ldots t_7$. All tables are fixed.

Between the initial and final transpositions, the algorithm performs 16 iterations of a function f that combines substitution and transposition. Let T_i denote the result of the ith iteration, and let L_i and R_i denote the left and right halves of T_i, respectively; that is, $T_i = L_i R_i$, where

$$L_i = t_1 \ldots t_{32}$$
$$R_i = t_{33} \ldots t_{64} \ .$$

Then

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

where "$\oplus$" is the exclusive-or operation and K_i is a 48-bit key described later. Note that after the last iteration, the left and right halves are not exchanged; instead the

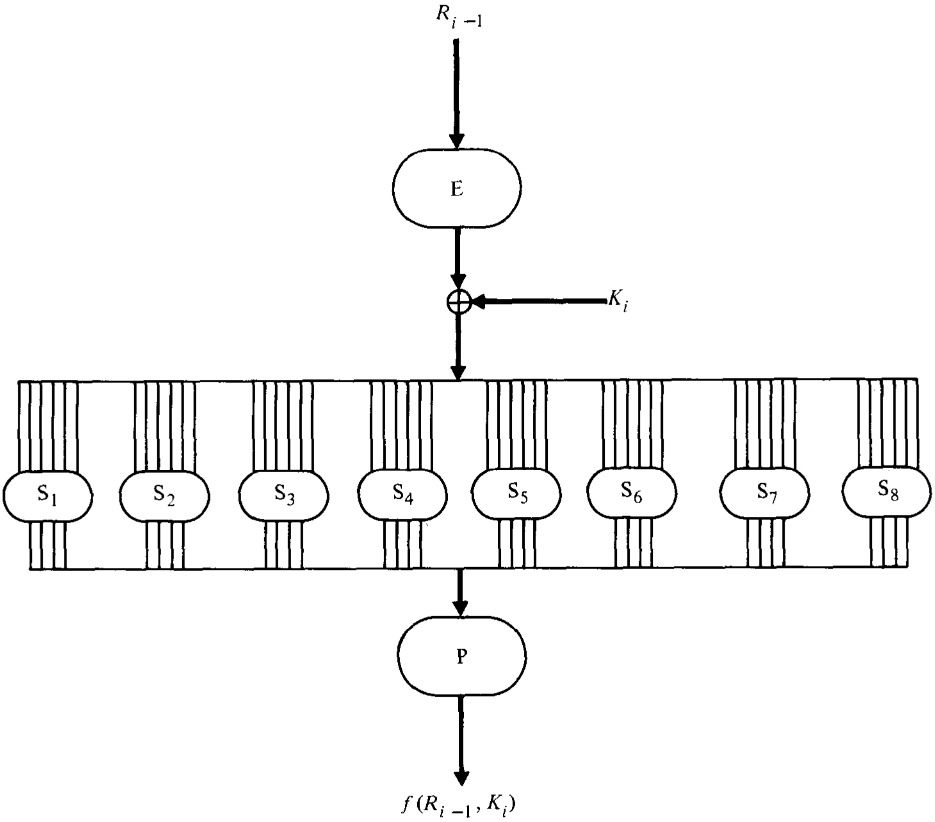
TABLE 2.3(a) Initial permutation IP.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

TABLE 2.3(b) Final permutation IP^{-1}.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

FIGURE 2.14 Calculation of $f(R_{i-1}, K_i)$.



concatenated block $R_{16}L_{16}$ is input to the final permutation IP^{-1}. This is necessary in order that the algorithm can be used both to encipher and to decipher.

The function f *and* S-*boxes.* Figure 2.14 shows a sketch of the function $f(R_{i-1}, K_i)$. First, R_{i-1} is expanded to a 48-bit block $\text{E}(R_{i-1})$ using the bit-selection table E

TABLE 2.4 Bit-selection Table E.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

TABLE 2.5 Permutation P.

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Previous Chapter Previous Page Home Next Page Next Chapter

shown in Table 2.4. This table is used in the same way as the permutation tables, except that some bits of R_{i-1} are selected more than once; thus, given $R_{i-1} = r_1 r_2 \dots r_{32}$, $\mathrm{E}(R_{i-1}) = r_{32} r_1 r_2 \dots r_{32} r_1$.

Next, the exclusive-or of $\mathrm{E}(R_{i-1})$ and K_i is calculated and the result broken into eight 6-bit blocks $B_1, \dots, B_8$, where

$$\mathrm{E}(R_{i-1}) \oplus K_i = B_1 B_2 \dots B_8 \ .$$

TABLE 2.6 Selection functions (S-boxes).

	Column																
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Each 6-bit block B_j is then used as input to a selection (substitution) function (**S-box**) S_j, which returns a 4-bit block $S_j(B_j)$. These blocks are concatenated together, and the resulting 32-bit block is transposed by the permutation P shown in Table 2.5. Thus, the block returned by $f(R_{i-1}, K_i)$ is

$$P(S_1(B_1) \dots S_8(B_8)) \ .$$

Each S-box S_j maps a 6-bit block $B_j = b_1b_2b_3b_4b_5b_6$ into a 4-bit block as defined in

FIGURE 2.15 Key schedule calculation.

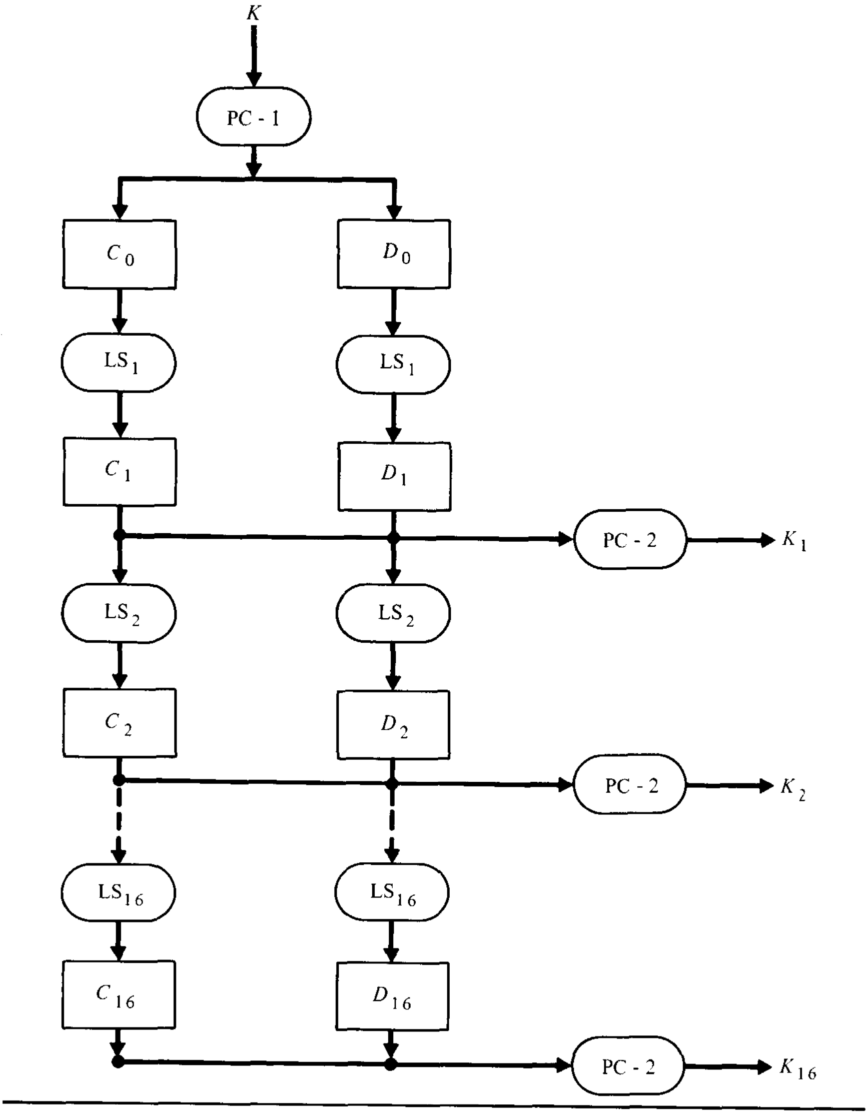


Table 2.6. This is done as follows: The integer corresponding to b_1b_6 selects a row in the table, while the integer corresponding to $b_2b_3b_4b_5$ selects a column. The value of $S_j(B_j)$ is then the 4-bit representation of the integer in that row and column.

Example:
If B_1 = 010011, then S_1 returns the value in row 1, column 9; this is 6, which is represented as 0110. ■

Key calculation. Each iteration i uses a different 48-bit key K_i derived from the initial key K. Figure 2.15 illustrates how this is done. K is input as a 64-bit block, with 8 parity bits in positions 8, 16, . . . , 64. The permutation PC-1 (permuted choice 1) discards the parity bits and transposes the remaining 56 bits as shown in Table 2.7. The result PC-1(K) is then split into two halves C and D of 28 bits each. The blocks C and D are then successively shifted left to derive each key K_i. Letting C_i and D_i denote the values of C and D used to derive K_i, we have

$$C_i = \mathrm{LS}_i(C_{i-1}),\ D_i = \mathrm{LS}_i(D_{i-1}),$$

where LS_i is a left circular shift by the number of positions shown in Table 2.8, and C_0 and D_0 are the initial values of C and D. Key K_i is then given by

$$K_i = \text{PC-2}(C_i\, D_i), \text{ where}$$

PC-2 is the permutation shown in Table 2.9.

Deciphering. Deciphering is performed using the same algorithm, except that K_{16} is used in the first iteration, K_{15} in the second, and so on, with K_1 used in the

TABLE 2.7 Key permutation PC-1.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

TABLE 2.8 Key schedule of left shifts LS.

Iteration i	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

TABLE 2.9 Key permutation PC-2.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

16th iteration. This is because the final permutation IP^{-1} is the inverse of the initial permutation IP, and

$$\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f(L_i, K_i) \, . \end{aligned}$$

Note that whereas the order of the keys is reversed, the algorithm itself is not.

Implementation. DES has been implemented both in software and in hardware. Hardware implementations achieve encryption rates of several million bps (bits per second).

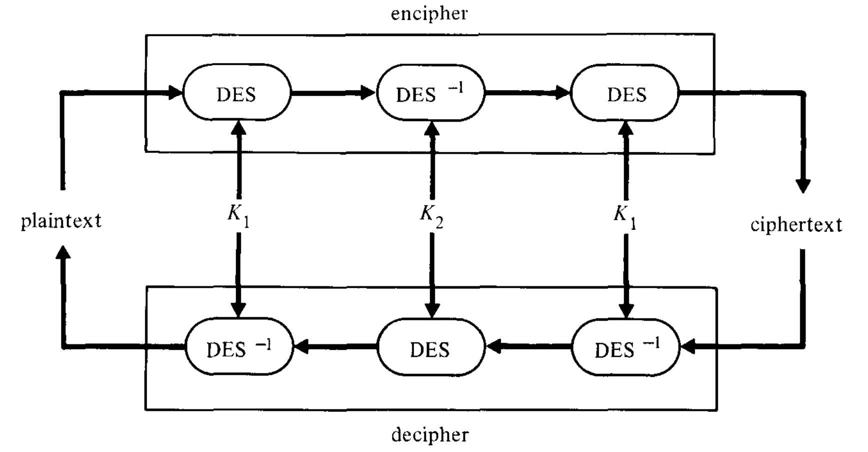
In 1976 (before DES was adopted), the National Bureau of Standards held two workshops to evaluate the proposed standard. At that time, Diffie and Hellman, and others, were concerned about possible weaknesses (see [Hell76, Diff77]. After the second workshop, Morris, Sloane, and Wyner [Morr77] reported that they believed the DES had two major weaknesses:

1. *Key size:* 56 bits may not provide adequate security.
2. *S-boxes:* The S-boxes may have hidden trapdoors.

Diffie and Hellman argue that with 56-bit keys, DES may be broken under a known-plaintext attack by exhaustive search. In [Diff77] they show that a special-purpose machine consisting of a million LSI chips could try all $2^{56} \cong 7 \times 10^{16}$ keys in 1 day. Each chip would check one key per μsec or 8.64×10^{10} keys per day. Whereas it would take almost 10^6 days for one chip to check all keys, 10^6 chips can check the entire key space in 1 day. The cost of such a machine would be about $20 million. Amortized over 5 years, the cost per day would be about $10,000. Because on the average only half the key space would have to be searched, the average search time would be half a day, making the cost per solution only $5,000. More recently, Diffie [Diff81] has increased this estimate to a 2-day average search time on a $50M machine (using 1980 technology). But he and Hellman [Hell79] both predict the cost of building a special-purpose DES search machine will drop substantially by 1990.

Hellman [Hell80] has also shown that it is possible to speed up the searching process by trading time for memory in a chosen-plaintext attack (see following section). The cost per solution would be $10 on a $5M machine. Because the

FIGURE 2.16 Multiple encipherment with DES.



attack can be thwarted with techniques such as cipher block chaining (see Chapter 3), the search strategy itself does not pose a real threat.

Hellman and others argue that the key size should be doubled (112 bits). Tuchman [Tuch79] claims that the same level of security can be obtained with 56-bit keys, using a multiple encryption scheme invented by Matyas and Meyer. Let DES_K denote the encipering transformation for key K, and DES_K^{-1} the corresponding deciphering transformation. A plaintext message M is enciphered as

$$C = \mathrm{DES}_{K_1}\left(\mathrm{DES}_{K_2}^{-1}(\mathrm{DES}_{K_1}(M))\right);$$

that is, it is successively enciphered, deciphered, and then enciphered again, using one key (K_1) for the encipherments and another (K_2) for the decipherment (see Figure 2.16). The message M is restored by reversing the process and applying the inverse transformations:

$$\mathrm{DES}_{K_1}^{-1}\left(\mathrm{DES}_{K_2}(\mathrm{DES}_{K_1}^{-1}(C))\right) = M.$$

Merkle and Hellman [Merk81] believe this approach may be less secure than using a single 112-bit key (or even using 3 separate keys in a triple-encryption scheme), showing that it can be theoretically broken under a chosen-plaintext attack using about 2^{56} operations and 2^{56} keys stored on 4 billion tapes.

Hellman and others have also questioned whether the S-boxes are secure (the analysis behind their design is presently classified). They believe that the design should be unclassified so that it may be publicly evaluated. (See [Suga79, Hell79,Davi79,Bran79,Tuch79] for a debate on these issues.)

2.6.3 Time-Memory Tradeoff

There are two naive approaches to breaking a cipher with n keys: exhaustive search and table lookup. **Exhaustive search** uses a known-plaintext attack: Given ciphertext C, the known plaintext M is enciphered with each key K until $E_K(M) = C$. The time complexity is $T = O(n)$ and the space complexity is $S = O(1)$. We saw earlier that if one key can be tested in 1 μsec on a special-purpose chip, then $n = 2^{56} \cong 7 \times 10^{16}$ keys can be checked in $T = 10^6$ days, and 1 million chips can search the entire key space in parallel in approximately 1 day.

Table lookup uses a chosen-plaintext attack: For chosen plaintext M_0, the ciphertexts $C_i = E_{K_i}(M_0)$ are precomputed for $i = 1, \dots, n$. The keys K_i are arranged in a table so that C_i gives the index of K_i. Thus, for a given ciphertext, the corresponding key K can be found in time $T = O(1)$, though the space complexity is $S = O(n)$. For 56-bit keys, the space requirements are $S = 56(2^{56}) \cong 56 \times 7 \times 10^{16} \cong 4 \times 10^{18}$ bits. This would require about 4 billion magnetic tapes (at about 10^9 bits per tape), costing about \$80 billion (at \$20 per tape).

Hellman's **time-memory tradeoff** technique is a hybrid approach that trades time for space in a chosen-plaintext attack [Hell80]. Like table lookup, it requires the precomputation and storage of a table. For the time-memory tradeoff technique, however, the table size is only $S = O(n^{2/3})$. Like exhaustive search, the technique also requires searching. Here again, however, the search time is only $T = O(n^{2/3})$.

Define

$$f(K) = R(E_K(M_0))$$

for chosen plaintext M_0, where R is a function that reduces a 64-bit block to a 56-bit block (by discarding 8 bits). For a given ciphertext $C_0 = E_K(M_0)$, the objective (of cryptanalysis) is to find the inverse K of f; that is,

$$K = f^{-1}(R(C_0)) .$$

For the precomputation, m starting points $SP_1, \dots, SP_m$ are randomly chosen from the key space. The following values are computed for some integer t and $i = 1, \dots, m$:

$$\begin{aligned} X_{i0} &= SP_i \\ X_{ij} &= f(X_{i,j-1}) \quad (1 \le j \le t) \end{aligned}$$

(see Table 2.10). The ending points EP_i of the computation can thus be expressed: $EP_i = f^t(SP_i)$ $(1 \le i \le m)$. The m pairs of values (SP_i, EP_i) are stored in a table sorted by the EP_i. The storage requirements for the table are $S = O(m)$, and the precomputation time is $T_p = O(mt)$.

Given an intercepted ciphertext $C_0 = E_K(M_0)$, the search for K proceeds by first looking for K in column $t - 1$, of the X's (see Table 2.10). If this fails, the remaining columns are searched in the order $t - 2, t - 3, \dots, 0$. Because the X's are not stored in the table, they must be computed from the starting points as described earlier.

To begin, $Y_1 = R(C_0)$ is computed in order to reduce C_0 to 56 bits. Next, if $Y_1 = EP_i$ for some i $(1 \le i \le m)$, then $Y_1 = f(X_{i,t-1})$. This implies that either $K = X_{i,t-1}$ or that f has more than one inverse. This latter event is called a "false alarm." To determine whether $X_{i,t-1}$ is correct, it is used to encipher the chosen plaintext M_0; $K = X_{i,t-1}$ if and only if $E_{X_{i,t-1}}(M_0) = C_0$.

TABLE 2.10 Time-memory tradeoff table.

Starting Points		Ending Points
SP_1	$= X_{10} \xrightarrow{f} X_{11} \xrightarrow{f} X_{12} \xrightarrow{f} \dots \xrightarrow{f} X_{1,t-2} \xrightarrow{f} X_{1,t-1} \xrightarrow{f} X_{1t} =$	EP_1
.		.
.		.
.		.
SP_i	$= X_{i0} \rightarrow X_{i1} \rightarrow X_{i2} \rightarrow \dots \rightarrow X_{i,t-2} \rightarrow X_{i,t-1} \rightarrow X_{it} =$	EP_i
.		.
.		.
.		.
SP_m	$= X_{m0} \rightarrow X_{m1} \rightarrow X_{m2} \rightarrow \dots \rightarrow X_{m,t-2} \rightarrow X_{m,t-1} \rightarrow X_{mt} =$	EP_m

If K is not found in column $t - 1$ of the X's, column $t - 2$ is searched next. This is done by setting $Y_2 = f(Y_1)$. $K = X_{i,t-2}$ for some i if and only if $Y_2 = EP_i$ and $E_{X_{i,t-2}}(M_0) = C_0$. The remaining columns are searched by computing $Y_j = f(Y_{j-1})$ for $j = 3, \dots, t$. $K = X_{i,t-j}$ for some i if and only if $Y_j = EP_i$ and $E_{X_{i,t-j}}(M_0) = C_0$.

Assuming false alarms are rare, the time to search one table is $T = O(t)$. If all mt values of X represented by the table are randomly chosen and different, then the probability of finding a random key K in the table is $p = mt/n$. Allowing for overlap, Hellman shows that if $mt^2 = n$, the probability of success has the approximate lower bound of:

$$p \geq .8 \frac{mt}{n} \cong \frac{mt}{n}.$$

Let $m = t = n^{1/3}$. Then the storage requirements are $S = O(m) = O(n^{1/3})$, the precomputation time is $T_p = O(mt) = O(n^{2/3})$, and the search time is $T_s = O(t) = O(n^{1/3})$. The probability of success, however, is only approximately $n^{-1/3}$. For $n = 10^{17}$, this is a little better than 10^{-6}, which is not good. If $t = n^{1/3}$ tables are precomputed and searched instead, then the probability of success is quite high. The total storage requirements then become $S = O(tm) = O(n^{2/3})$, while the total precomputation time becomes $T_p = O(tmt) = O(n)$ and the total search time becomes $T_s = O(tt) = O(n^{2/3})$. The real time can be reduced by precomputing and searching some (or all) of the tables in parallel.

Rivest has observed that the search time can be reduced by forcing each endpoint EP_i to satisfy some easily tested syntactic property (e.g., begins with a fixed number of 0's) that is expected to hold after t encipherments of the starting point SP_i (so the expected number of entries represented by a table of m starting and ending points is still mt). Thus, instead of precomputing $\mathrm{EP}_i = f^t(SP_i)$ for a fixed t, SP_i would be successively reenciphered until it satisfied the chosen property.

Hellman suggests a hardware implementation using $m = 10^5$, $t = 10^6$, and 10^6 tables. Because each table entry contains a 56-bit starting point SP and a 56-

bit ending point *EP*, the total storage requirements are thus $10^{11} \times 112$ bits or about 10^{13} bits, which can be stored on 10,000 tapes (assuming 10^9 bits per tape). These can be read and searched in 1 day with 100 tape drives. Each tape drive would have a 10^7 bit semiconductor memory for storing one table. Each memory would have 100 DES chips searching for 100 different keys in parallel. The machine would also be capable of doing the precomputation; with all 10,000 DES units operating in parallel, this would take about 1.1 years (using a DES chip operating at 4 μsec per encipherment).

The total parts would cost about $3.6M using 1980 technology. Depreciated over 5 years, this is about $2500 per day, or $25 per solution. With decreasing hardware costs, the cost per solution could drop to $1.

To break a cipher using time-memory tradeoff, it must be possible to perform a chosen-plaintext attack. This is possible when plaintext messages are known to contain commonly occurring sequences (e.g., blanks) or standard headers (e.g., "LOGIN:"), and each block is separately enciphered. Techniques such as cipher block chaining (see Section 3.4.1) protect against this type of attack. The proposed Federal standards suggest taking such precautions.

2.7 EXPONENTIATION CIPHERS

In 1978, Pohlig and Hellman [Pohl78a] published an encryption scheme based on computing exponentials over a finite field. At about the same time, Rivest, Shamir, and Adleman [Rive78a] published a similar scheme, but with a slight twist—a twist that gave the MIT group a method for realizing public-key encryption as put forth by Diffie and Hellman [Diff76]. In *Scientific American,* Martin Gardner [Gard77] described the RSA scheme as "A New Kind of Cipher That Would Take Millions of Years to Break." Oddly enough, just 60 years earlier in 1917, the same journal published an article touting the Vigenère ciphers as "impossible of translation" [Kahn67].

The Pohlig-Hellman and RSA schemes both encipher a message block $M \epsilon$ $[0, n - 1]$ by computing the exponential

$$C = M^e \bmod n \ , \tag{2.2}$$

where e and n are the key to the enciphering transformation. M is restored by the same operation, but using a different exponent d for the key:

$$M = C^d \bmod n \ . \tag{2.3}$$

Enciphering and deciphering can be implemented using the fast exponentiation algorithm shown in Figure 1.20 (see Section 1.6.1):

$$C = fastexp(M, e, n)$$
$$M = fastexp(C, d, n) \ .$$

The enciphering and deciphering transformations are based on Euler's generalization of Fermat's Theorem (see Theorem 1.5 in Section 1.6.2), which states that for every M relatively prime to n,

$$M^{\phi(n)} \bmod n = 1 \ .$$

This property implies that if e and d satisfy the relation

$$ed \bmod \phi(n) = 1 \ , \tag{2.4}$$

then Eq. (2.3) is the inverse of Eq. (2.2), so that deciphering restores the original plaintext message. This result is proved in the following theorem:

Theorem 2.1:
Given e and d satisfying Eq. (2.4) and a message $M \epsilon [0, n - 1]$ such that $gcd(M, n) = 1$,

$$(M^e \bmod n)^d \bmod n = M.$$

Proof:
We have

$$(M^e \bmod n)^d \bmod n = M^{ed} \bmod n \ .$$

Now, $ed \bmod \phi(n) = 1$ implies that $ed = t\phi(n) + 1$ for some integer t. Thus,

$$\begin{aligned} M^{ed} \bmod n &= M^{t\phi(n)+1} \bmod n \\ &= MM^{t\phi(n)} \bmod n \\ &= M(M^{t\phi(n)} \bmod n) \bmod n \ , \end{aligned}$$

where:

$$\begin{aligned} M^{t\phi(n)} \bmod n &= (M^{\phi(n)} \bmod n)^t \bmod n \\ &= 1^t \bmod n \\ &= 1 \ . \end{aligned}$$

Thus,

$$\begin{aligned} M^{ed} \bmod n &= (M * 1) \bmod n \\ &= M \ . \ \blacksquare \end{aligned}$$

By symmetry, enciphering and deciphering are commutative and mutual inverses; thus,

$$(M^d \bmod n)^e \bmod n = M^{de} \bmod n = M \ .$$

It is because of this symmetry that the RSA scheme can be used for secrecy and authenticity in a public-key system.

Given $\phi(n)$, it is easy to generate a pair (e, d) satisfying Eq. (2.4). This is done by first choosing d relatively prime to $\phi(n)$, and then using the extended version of Euclid's algorithm (see Figure 1.22 in Section 1.6.2) to compute its inverse:

$$e = inv(d, \phi(n)) \ . \tag{2.5}$$

[Because e and d are symmetric, we could also pick e and compute $d = inv(e, \phi(n))$.]

Given e, it is easy to compute d (or vice versa) if $\phi(n)$ is known. But if e and n can be released without giving away $\phi(n)$ or d, then the deciphering transformation can be kept secret, while the enciphering transformation is made public. It is the ability to hide $\phi(n)$ that distinguishes the RSA scheme from the Pohlig-Hellman scheme.

2.7.1 Pohlig-Hellman Scheme

In the Pohlig-Hellman scheme, the modulus is chosen to be a large prime p. The enciphering and deciphering functions are thus given by

$$C = M^e \bmod p$$
$$M = C^d \bmod p \ ,$$

where all arithmetic is done in the Galois field $\mathbf{GF}(p)$ (see Section 1.6.3). Because p is prime, $\phi(p) = p - 1$ (see Section 1.6.2), which is trivially derived from p. Thus the scheme can only be used for conventional encryption, where e and d are both kept secret.

Example:
Let $p = 11$, whence $\phi(p) = p - 1 = 10$. Choose $d = 7$ and compute $e = inv(7, 10) = 3$. Suppose $M = 5$. Then M is enciphered as:

$$C = M^e \bmod p = 5^3 \bmod 11 = 4 \ .$$

Similarly, C is deciphered as:

$$M = C^d \bmod p = 4^7 \bmod 11 = 5 \ . \quad \blacksquare$$

The security of the scheme rests on the complexity of computing discrete logarithms in $\mathbf{GF}(p)$. This is because under a known-plaintext attack, a cryptanalyst can compute e (and thereby d) given a pair (M, C):

$$e = \log_M C \quad \text{in } \mathbf{GF}(p)$$

(p may be deduced by observing the sizes of plaintext and ciphertext blocks). Pohlig and Hellman show that if $(p - 1)$ has only small prime factors, it is possible to compute the logarithm in $O(\log^2 p)$ time, which is unsatisfactory even for large values of p. They recommend picking $p = 2p' + 1$, where p' is also a large prime.

Now, the fastest known algorithm for computing the discrete logarithm in $\mathbf{GF}(p)$, due to Adleman [Adle79], takes approximately

$$T = \exp\left(\text{sqrt}\left(\ln(p)\ln\left(\ln(p)\right)\right)\right) \qquad \textbf{(2.6)}$$

steps, where "ln" denotes the natural logarithm and "exp" its inverse). If p is 200 bits long, Eq. (2.6) evaluates to

$$T = 2.7 \times 10^{11} \ .$$

Assuming 10^{11} steps can be performed per day (i.e., about 1 step per μsec), the entire computation would take only a few days. But if p is 664 bits long (200 decimal digits),

$$T = 1.2 \times 10^{23},$$

which would take about 10^{12} days or several billion years. Figure 2.17 shows a graph of $\log_{10} T$ as a function of the length of p in bits. Techniques for picking large primes are described in the next section.

Pohlig and Hellman also note that their scheme could be implemented in the Galois Field $\mathbf{GF}(2^n)$, where $2^n - 1$ is a large prime (called a Mersenne prime—e.g., see [Knut69]). Such an implementation would be efficient and have the advantage that all messages would be exactly n bits; furthermore, every element in the range $[1, 2^n - 2]$ could be used as a key.

2.7.2 Rivest-Shamir-Adleman (RSA) Scheme

In the RSA scheme, the modulus n is the product of two large primes p and q:

$$n = pq \ .$$

Thus

$$\phi(n) = (p - 1)\ (q - 1)$$

(see Theorem 1.3 in Section 1.6.2). The enciphering and deciphering functions are given by Eq. (2.2) and (2.3). Rivest, Shamir, and Adleman recommend picking d relatively prime to $\phi(n)$ in the interval $[\max\ (p, q) + 1, n - 1]$ (any prime in the interval will do); e is computed using Eq. (2.5). If $inv(d, \phi(n))$ returns e such that $e < \log_2 n$, then a new value of d should be picked to ensure that every encrypted message undergoes some wrap-around (reduction modulo n).

Example:
Let $p = 5$ and $q = 7$, whence $n = pq = 35$ and $\phi(n) = (5 - 1)\ (7 - 1) = 24$. Pick $d = 11$. Then $e = inv(11, 24) = 11$ (in fact, e and d will always be the same for $p = 5$ and $q = 7$—see exercises at end of chapter). Suppose $M = 2$. Then

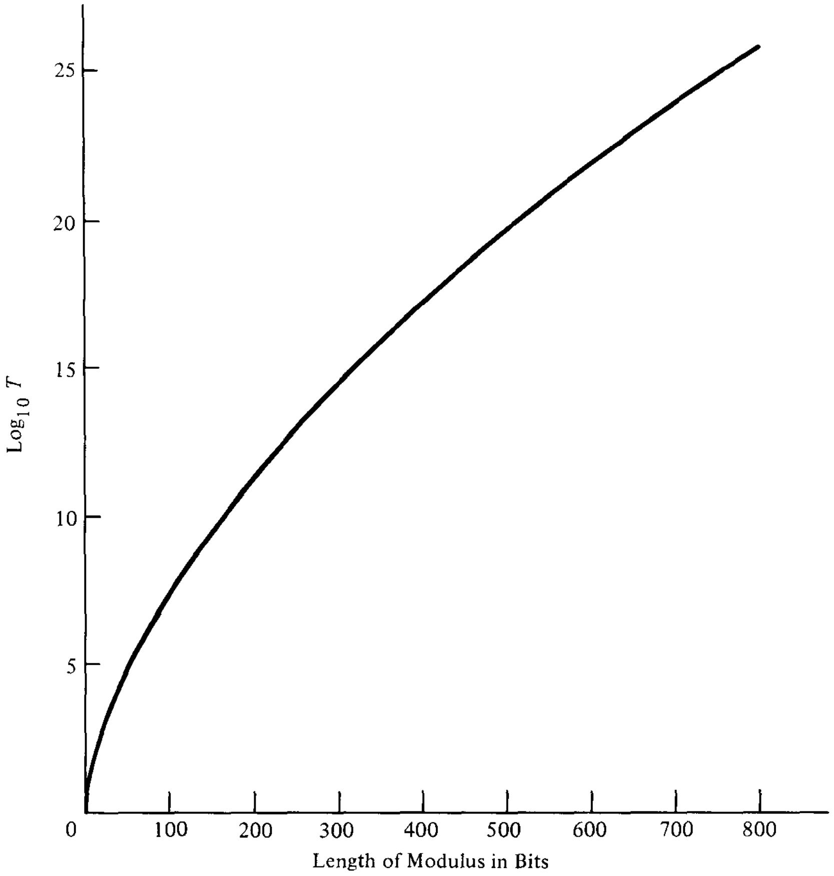
$$C = M^e \bmod n = 2^{11} \bmod 35 = 2048 \bmod 35 = 18\ ,$$

and

$$C^d \bmod n = 18^{11} \bmod 35 = 2 = M\ . \ \blacksquare$$

Example:
Let $p = 53$ and $q = 61$, whence $n = 53 * 61 = 3233$ and $\phi(n) = 52 * 60 = 3120$. Letting $d = 791$, we get $e = 71$. To encipher the message M = RENAISSANCE, we break it into blocks of 4 digits each, where A = 00, B = 01, . . . , Z = 25, and blank = 26 (in practice, characters would be represented by their 8-bit ASCII codes). We thus get

FIGURE 2.17 Time to compute discrete logarithm or to factor.



$$M = \text{R E N A I S S A N C E}$$
$$= 1704\ 1300\ 0818\ 1800\ 1302\ 0426\ .$$

The first block is enciphered as $1704^{71} = 3106$. The entire message is enciphered as

$$C = 3106\ 0100\ 0931\ 2691\ 1984\ 2927\ . \quad ■$$

Because $\phi(n)$ cannot be determined without knowing the prime factors p and q, it is possible to keep d secret even if e and n are made public. This means that the RSA scheme can be used for public-key encryption, where the enciphering transformation is made public and the deciphering transformation is kept secret.

The security of the system depends on the difficulty of factoring n into p and q. The fastest known factoring algorithm, due to Schroeppel (unpublished), takes

$$T = \exp\left(\text{sqrt}\left(\ln(n)\ln(\ln(n))\right)\right)$$

FIGURE 2.18 Evaluate Jacobi symbol.

```
Algorithm $J(a, b)$: "Evaluate $\left(\frac{a}{b}\right)$"
if $a = 1$ then $J := 1$
else if $a \bmod 2 = 0$ then begin
  if $(b * b - 1)/8 \bmod 2 = 0$
    then $J := J(a/2, b)$ else $J := -J(a/2, b)$ end
else if $(a - 1) * (b - 1)/4 \bmod 2 = 0$
  then $J := J(b \bmod a, a)$ else $J := -J(b \bmod a, a)$
```

steps. This is the same number of steps required to compute the discrete logarithm in **GF**(n) when n is prime [see Eq. (2.6) and Figure 2.17]. Rivest, Shamir, and Adleman suggest using 100-digit numbers for p and q; then n is 200 digits, and factoring would take several billion years at the rate of one step per microsecond.

The security of the system also depends on using carefully selected primes p and q. If n is 200 digits, then p and q should be large primes of approximately 100 digits each. To find a 100-digit prime p, Rivest, Shamir, and Adleman recommend randomly generating numbers until a number b is found that is "probably" prime. To test b for primality, 100 random numbers $a \in [1, b - 1]$ are generated. Then b is almost certain to be prime if the following test (due to Solovay and Strassen [Solo77]) is satisfied for each a:

$$gcd(a, b) = 1 \text{ and } \left(\frac{a}{b}\right) \bmod b = a^{(b-1)/2} \bmod b \ , \qquad \textbf{(2.7)}$$

where $\left(\frac{a}{b}\right)$ is the Jacobi symbol (see [LeVe77,Nive72,Vino55] and discussion in following section). When $gcd(a, b) = 1$, the Jacobi symbol can be efficiently evaluated using the recursive function $J(a, b)$ shown in Figure 2.18. If b is prime, then Eq. (2.7) is true for all $a \in [1, b - 1]$. If b is not prime, Eq. (2.7) is true with probability at most 1/2 for each such a, and at most $1/2^{100}$ for 100 such a's.

For better protection against factoring, additional precautions should be taken in selecting p and q:

1. p and q should differ in length by a few digits.
2. Both $p - 1$ and $q - 1$ should contain large prime factors.
3. $gcd(p - 1, q - 1)$ should be small.

To find a prime p such that $p - 1$ has a large prime factor, first generate a large random prime p'. Then generate

$$p = i * p' + 1 \ , \quad \text{for } i = 2, 4, 6, \ldots \qquad \textbf{(2.8)}$$

until p is prime. Further protection may be obtained by picking p' such that $p' - 1$ has a large prime factor.

Simmons and Norris [Simm77] showed the scheme may be broken without factoring if p and q are not carefully chosen. They found that for certain keys, reenciphering a ciphertext message a small number of times restored the original plaintext message. Thus, given ciphertext $C_0 = M^e \bmod n$ and the public key (e, n), a cryptanalyst may be able to determine M by computing

$$C_i = C_{i-1}^e \bmod n\ , \quad \text{for } i = 1, 2, \ldots$$

until C_i is a meaningful message. Clearly, this type of attack is worthwhile only if the plaintext is restored within a reasonably small number of steps (e.g., a million). Rivest [Rive78b] showed that if each prime p is chosen so that $p - 1$ has a large prime factor p', where $p' - 1$ has a large prime factor p'', then the probability of this type of attack succeeding is extremely small. For primes larger than 10^{90}, this probability is at most 10^{-90}.

Blakley and Blakley [Blak78] and Blakley and Borosh [Blak79] show that for any choice of keys, at least nine plaintext messages will not be concealed by encipherment; that is, for any e and n, $M^e \bmod n = M$ for at least nine M. Although the probability of picking one out of nine such messages is small if messages are 200 digits long, a poor choice of keys will conceal less than 50% of all possible messages. They argue that the system will be more resistant to this type of attack and sophisticated factoring algorithms if safe primes are selected; a prime p is **safe** if

$$p = 2p' + 1, \quad \text{where } p' \text{ is an odd prime}$$

(every prime but 2 is odd). This represents a restriction on the form of Eq. (2.8) suggested by Rivest, Shamir, and Adleman. We observed earlier that Pohlig and Hellman also suggested using safe primes in their scheme.

Breaking the RSA scheme can be no more difficult than factoring, because a fast factoring algorithm automatically gives an efficient cryptanalytic procedure. This does not, however, rule out finding an efficient algorithm for cryptanalysis without finding a corresponding algorithm for factoring. Rabin [Rabi79] and Williams [Will80] have devised variants of the RSA scheme where the cryptanalytic effort is equivalent to factorization. There is, however, a drawback to these schemes arising from the constructive nature of the proofs. Rivest has observed that any cryptosystem for which there exists a constructive proof of equivalence of the cryptanalytic effort with factorization is vulnerable to a chosen-ciphertext attack (see [Will80]). Upon obtaining the deciphered message for a selected ciphertext message, a cryptanalyst can factor the modulus and break the cipher.

If n, d, and e are each 200 decimal digits (664 bits), the storage requirements per user are about 2,000 bits. Since both e and n must be made public, the public storage requirements are thus about 1.3 kilobits per user. By comparison, the DES requires only 56 bits (or 112 bits if longer keys or multiple encryption is used).

The time requirements are also considerably greater than for the DES. To encipher or decipher a 664-bit number requires 1–2 multiplications in modular arithmetic per bit, or about 1,000 multiplications total. Rivest has designed a special-purpose chip that will run at a few thousand bps [Rive80]. Although this is fast enough to support real-time communication over telephone lines, it is too slow for communication links capable of higher bit rates. Rabin's scheme [Rabi70] has a faster enciphering algorithm (requiring only one addition, one multiplication, and one division by the modulus—see exercises); the deciphering algorithm is comparable to the RSA algorithm.

Because the enciphering and deciphering functions are mutual inverses, the RSA scheme can be used for secrecy and authenticity. Each user A obtains a modulus n_A and enciphering and deciphering exponents e_A and d_A. A registers e_A and n_A with a public directory, thus making A's enciphering transformation E_A public. A keeps d_A and, therefore, the deciphering transformation D_A secret.

User B can send a secret message M to A by obtaining A's public transformation E_A and transmitting

$$E_A(M) = M^{e_A} \bmod n_A ,$$

which A deciphers using A's secret transformation D_A:

$$D_A(E_A(M)) = M^{e_A d_A} \bmod n_A = M .$$

Alternatively, A can send a signed message M to B by transmitting

$$D_A(M) = M^{d_A} \bmod n_A ,$$

which B authenticates using A's public transformation E_A:

$$E_A(D_A(M)) = M^{d_A e_A} \bmod n_A = M .$$

Because only A can apply D_A, it cannot be forged, and a judge can settle any dispute arising between A and B.

A slight difficulty arises when both secrecy and authenticity are desired, because it is necessary to apply successive transformations with different moduli. For example, in order for A to send a signed, secret message to B, A must transmit:

$$C = E_B(D_A(M)) .$$

If $n_A > n_B$, the blocks comprising $D_A(M)$ might not be in the range $[0, n_B - 1]$ of B's transformation. Reducing them modulo n_B does not solve the problem, because it would then be impossible to recover the original message. One solution is to reblock $D_A(M)$. Rivest, Shamir, and Adleman show that reblocking can be avoided using a **threshold** value h (e.g., $h = 10^{99}$). Each user has two sets of transformations: (E_{A1}, D_{A1}) for signatures and (E_{A2}, D_{A2}) for secrecy, where $n_{A1} < h < n_{A2}$. To send a signed, secret message to B, A, transmits

$$C = E_{B2}(D_{A1}(M)) ,$$

which is computable because $n_{A1} < h < n_{B2}$. User B recovers M and checks A's signature by computing

$$\begin{aligned} E_{A1}(D_{B2}(C)) &= E_{A1}(D_{B2}(E_{B2}(D_{A1}(M)))) \\ &= E_{A1}(D_{A1}(M)) \\ &= M . \end{aligned}$$

Previous Chapter Previous Page Home Next Page Next Chapter

Kohnfelder [Konf78] suggests another approach, pointing out that if $C = E_B(D_A(M))$ is not computable because $n_A > n_B$, then $C' = D_A(E_B(M))$ is computable. User B, knowing both n_A and n_B, can recover M by computing either of the following:

Case 1: $n_A < n_B$

$$\begin{aligned} E_A(D_B(C)) &= E_A(D_B(E_B(D_A(M)))) \\ &= E_A(D_A(M)) \\ &= M\,. \end{aligned}$$

Case 2: $n_A > n_B$

$$\begin{aligned} D_B(E_A(C')) &= D_B(E_A(D_A(E_B(M)))) \\ &= D_B(E_B(M)) \\ &= M\,. \end{aligned}$$

If a dispute arises between A and B on the authenticity of A's signature, a judge must be able to ascertain that M originated with A. If $n_A < n_B$, B applies B's private transformation to C and presents the judge with $X = D_B(C)$ and M. The judge computes

$$M' = E_A(X)$$

using A's public transformation, and verifies that $M' = M$. If $n_A > n_B$, another approach is needed because D_B must be applied after E_A, and B may not want to give D_B to the judge. The solution is for B to present the judge with C' and M. The judge computes

$$\begin{aligned} X &= E_B(M) \\ X' &= E_A(C') = E_A(D_A(E_B(M))) \end{aligned}$$

using both A's and B's public transformations, and verifies that $X = X'$. Table 2.11 summarizes.

In the above approach, the storage requirements for a signed message are the same as for the unsigned message. Thus, in applications where the unsigned message is stored in the clear or as ciphertext encrypted by some other method, the total storage requirements are twice that of the unsigned message alone. An alternative, described by Davies and Price [Davs80], is to compress a message into a "digest" by hashing, and then sign the digest. This reduces the storage requirements for a signed message to a fixed size block.

TABLE 2.11 Secrecy and authenticity in RSA scheme.

	$n_A < n_B$	$n_A > n_B$
A transmits	$C = E_B(D_A(M))$	$C' = D_A(E_B(M))$
B computes	$M = E_A(D_B(C))$	$M = D_B(E_A(C'))$
B gives judge	$M, X = D_B(C)$	M, C'
Judge computes	$M' = E_A(X)$	$X = E_B(M)$, $X' = E_A(C')$,
Judge tests	$M' = M$	$X = X'$

2.7.3 Mental Poker

Shamir, Rivest, and Adleman [Sham80a] show how any commutative cipher can be used to play a fair game of "mental poker". The scheme is easily implemented with an exponentiation cipher, where the players share a common modulus n.

Mental poker is played like ordinary poker but without cards and without verbal communication; all exchanges between the players must be accomplished using messages. Any player may try to cheat. The requirements for a fair game are as follows:

1. The game must begin with a "fair deal". Assuming the players have exchanged a sequence of messages to accomplish this, then
   a. The players should know the cards in their own hand but not the others.
   b. All hands must be disjoint.
   c. All possible hands must be equally likely for each player.
2. During the game, players may want to draw additional cards from the remaining deck; these must also be dealt fairly as described in (1). Players must also be able to reveal cards to their opponents without compromising the security of their remaining cards.
3. At the end of the game, the players must be able to check that the game was played fairly, and that their opponents did not cheat.

For simplicity, assume there are two players Alice and Bob, and each has a secret key. The keys are not revealed until the end of the game.

Let E_A and D_A denote Alice's secret transformations, and let E_B and D_B denote Bob's. The enciphering transformations must commute; that is, for any message M:

$$E_A(E_B(M)) = E_B(E_A(M)) .$$

The 52 cards are represented by messages:

M_1: "two of clubs"
M_2: "three of clubs"
.
.
.
M_{52}: "ace of spades" .

Bob is the dealer. The protocol for a fair deal is as follows:

1. Bob enciphers the 52 messages, getting 52 encrypted messages:

$$E_B(M_i) , \qquad i = 1, \ldots , 52 .$$

He then randomly shuffles the encrypted deck and sends it to Alice.

2. Alice randomly selects 5 encrypted messages and returns them to Bob. Bob deciphers them to determine his hand.
3. Alice randomly selects 5 more encrypted messages, $C_1, \ldots, C_5$. She enciphers them with her key, getting

$$C_i' = E_A(C_i) \;, \qquad i = 1, \ldots, 5 \;.$$

She then sends the doubly enciphered messages $C_1', \ldots, C_5'$ to Bob.

5. Bob deciphers each message C_i' with his key, getting

$$\begin{aligned} D_B(C_i') &= D_B\big(E_A(E_B(M_j))\big) \\ &= D_B\big(E_B(E_A(M_j))\big) \\ &= E_A(M_j) \end{aligned}$$

for some message M_j. He returns these to Alice. She then deciphers them with her key to determine her hand.

During the game, additional cards may be dealt by repeating the preceding procedure. At the end of the game, both players reveal their keys to prove they did not cheat.

Because exponentiation in modular arithmetic is commutative, mental poker may be implemented with an exponentiation cipher. Bob and Alice agree on a large modulus n with corresponding $\phi(n)$. Alice picks a secret key (e_A, d_A) such that $e_A d_A \bmod \phi(n) = 1$ and uses the transformations:

$$\begin{aligned} E_A(M) &= M^{e_A} \bmod n \\ D_A(C) &= C^{d_A} \bmod n \;. \end{aligned}$$

Similarly, Bob picks a secret key (e_B, d_B) and uses the transformations:

$$\begin{aligned} E_B(M) &= M^{e_B} \bmod n \\ D_B(C) &= C^{d_B} \bmod n \;. \end{aligned}$$

Lipton [Lipt79a] shows that it may be possible to cheat using this encryption method. One way uses quadratic residues. A number a is a **quadratic residue** modulo n if $gcd(a, n) = 1$ and there exists an x such that

$$x^2 \equiv_n a,$$

or, equivalently,

$$x^2 \bmod n = a \bmod n \;;$$

otherwise, it is a **quadratic nonresidue** modulo n. Any x satisfying the preceding equations is a **square root** of a modulo n. Let R_2 denote the set of quadratic residues modulo n. For any message $M = a$, enciphering (or deciphering) M preserves its membership in R_2 as shown by the following theorem:

Theorem 2.2:
Given a, $0 < a < n$, $a \in R_2$ if and only if $E_K(a) = a^e \bmod n \in R_2$, where $K = (e, n)$.

Proof:
First, suppose $a \epsilon R_2$. Then $x^2 \bmod n = a$ for some x. Because

$$E_K(a) = a^e \bmod n = (x^2)^e \bmod n = (x^e)^2 \bmod n,$$

$E_K(a)$ is in R_2.

Now, suppose $E_K(a) \epsilon R_2$. Because deciphering is the same operation as enciphering but with exponent d, $(E_K(a))^d \bmod n = a$ must also be in R_2. ■

Example:
Let $n = 11$. Then 3 is a quadratic residue modulo 11 because $5^2 \bmod 11 = 3$. Suppose $e = 4$. Then $3^4 \bmod 11 = 4$ is also a quadratic residue because

$$2^2 \bmod 11 = 4. \quad ■$$

Alice can exploit this result by noting which cards have messages in R_2. After Bob has encrypted and shuffled these messages, she still cannot decipher them. She can tell, however, which correspond to those in R_2. If the modulus n is a prime p (as in the Pohlig-Hellman scheme), the probability of a message being in R_2 is 1/2; this gives her one bit of information per card which could help her to win. For example, if she observes that the plaintext messages for all four aces are quadratic residues, she could select quadratic residues for her hand and quadratic nonresidues for Bob's.

For prime p, half the numbers in the range $[1, p - 1]$ are in R_2 and half are not. To prove this result, we first prove that the equation $x^2 \bmod p = a$ has either two solutions or no solutions. (See also [LeVe77,Nive72,Vino55].)

Theorem 2.3:
For prime $p > 2$ and $0 < a < p$,

$$x^2 \bmod p = a$$

has two solutions if $a \epsilon R_2$ and no solutions otherwise.

Proof:
If $a \epsilon R_2$, there is at least one solution x_1. But then $p - x_1$ must also be a solution because

$$\begin{aligned}(p - x_1)^2 \bmod p &= (p^2 - 2px_1 + x_1^2) \bmod p \\ &= x_1^2 \bmod p = a\ .\end{aligned}$$

Furthermore, the solutions are distinct because $p - x_1 = x_1$ is possible only if 2 divides p. ■

Theorem 2.4:
For prime $p > 2$, there are $(p - 1)/2$ quadratic residues modulo p and $(p - 1)/2$ quadratic nonresidues.

Proof:
Clearly the $(p - 1)/2$ residues

$$1^2, 2^2, \ldots, \left(\frac{p-1}{2}\right)^2 \bmod p$$

are quadratic residues. There can be no additional quadratic residues because for every $a \in R_2$, at least one of its roots x_1 or $p - x_1$ must fall in the range $[1, (p - 1)/2]$. ■

Example:
For $p = 7$, the quadratic residues are

$$1^2 \bmod 7 = 1$$
$$2^2 \bmod 7 = 4$$
$$3^2 \bmod 7 = 2\ . \quad ■$$

Now, if $gcd(a, p) = 1$, it is simple to determine whether $a \in R_2$ by computing $a^{(p-1)/2} \bmod p$. Theorem 2.5 shows the result will be congruent to 1 if $a \in R_2$ and to -1 otherwise.

Theorem 2.5:
For prime $p > 2$ and $0 < a < p$,

$$a^{(p-1)/2} \bmod p = \begin{cases} 1 & \text{if } a \in R_2 \\ p - 1 & \text{otherwise} \end{cases} \qquad \textbf{(2.9)}$$

Proof:
By Fermat's Theorem,

$$(a^{p-1} - 1) \bmod p = 0\ .$$

Because p is odd, we can factor $a^{p-1} - 1$, getting

$$(a^{(p-1)/2} + 1)\ (a^{(p-1)/2} - 1) \bmod p = 0\ .$$

This implies that p must divide either $a^{(p-1)/2} + 1$ or $a^{(p-1)/2} - 1$. (It cannot divide both because this would imply that p divides their difference, which is 2.) We thus have

$$a^{(p-1)/2} \equiv_p \pm 1\ .$$

Now, if $a \in R_2$, then there exists an x such that $a = x^2 \bmod p$, which implies

$$\begin{aligned} a^{(p-1)/2} \bmod p &= (x^2)^{(p-1)/2} \bmod p \\ &= x^{p-1} \bmod p \\ &= 1\ . \end{aligned}$$

Thus, the $(p-1)/2$ quadratic residues are solutions of

$$a^{(p-1)/2} \bmod p = 1\ .$$

There can be no additional solutions because the equation, being of degree $(p-1)/2$, can have no more than $(p-1)/2$ solutions. Thus, the $(p-1)/2$ quadratic nonresidues must be solutions of

$$a^{(p-1)/2} \bmod p = p - 1 \ . \ \blacksquare$$

Example:
We saw earlier that 1, 2, and 4 are quadratic residues modulo $p = 7$. We can verify this by computing $a^{(7-1)/2} \bmod 7 = a^3 \bmod 7$ for $a = 1, 2, 4$:

$$\begin{aligned} 1^3 \bmod 7 &= 1 \\ 2^3 \bmod 7 &= 1 \\ 4^3 \bmod 7 &= 1 \ . \end{aligned}$$

Similarly, we can verify that $a = 2, 3$, and 5 are quadratic nonresidues by computing

$$\begin{aligned} 3^3 \bmod 7 &= 6 \\ 5^3 \bmod 7 &= 6 \\ 6^3 \bmod 7 &= 6 \ . \ \blacksquare \end{aligned}$$

Note the relationship between Eq. (2.9) and Eq. (2.7), which is used to test a number b for primality by checking whether

$$\left(\frac{a}{b}\right) \bmod b = a^{(b-1)/2} \bmod b$$

for some random a relatively prime to b. If b is a prime p, then the Jacobi symbol $\left(\frac{a}{p}\right)$ is equivalent by the **Legendre symbol**, also denoted $\left(\frac{a}{p}\right)$, which is defined by

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a \in R_2 \\ -1 & \text{otherwise,} \end{cases}$$

when $gcd(a, p) = 1$. By Theorem 2.5, $\left(\frac{a}{p}\right) \bmod p = a^{(p-1)/2} \bmod p$. If b is not prime, let $b = p_1 p_2 \ldots p_t$ be the prime factorization of b (factors may repeat). Then the Jacobi symbol $\left(\frac{a}{b}\right)$ is defined in terms of the Legendre symbol as follows:

$$\left(\frac{a}{b}\right) = \left(\frac{a}{p_1}\right) \left(\frac{a}{p_2}\right) \cdots \left(\frac{a}{p_t}\right) .$$

Note that whereas the Jacobi symbol is always congruent to $\pm$ 1 (mod b), the expression $a^{(b-1)/2} \bmod b$ may not be congruent to $\pm$ 1 when b is not prime.

Example:
For $n = 9$ and $a = 2$, $2^4 \bmod 9 = 7$. $\blacksquare$

Note also that whereas $\left(\frac{a}{b}\right)$ is defined by the prime factorization of b, it can be evaluated without knowing the factors. The recursive function $J(a, b)$ given in the previous section for evaluating $\left(\frac{a}{b}\right)$ does so by exploiting several properties of the Jacobi symbol, namely:

1. $\left(\frac{1}{b}\right) = 1$.
2. $\left(\frac{a_1 a_2}{b}\right) = \left(\frac{a_1}{b}\right) \left(\frac{a_2}{b}\right)$.
3. $\left(\frac{2}{b}\right) = (-1)^{(b^2-1)/8}$.
4. $\left(\frac{b}{a}\right) = \left(\frac{b \bmod a}{a}\right)$.
5. $\left(\frac{a}{b}\right) \left(\frac{b}{a}\right) = (-1)^{(a-1)(b-1)/4}$ if $gcd(a, b) = 1$.

Returning to mental poker, we see that enciphering function may preserve other properties about a message $M = a$ as well. For example, let R_t be the set of elements congruent to x^t mod n for some x. Then $a \in R_t$ if and only if a^e mod $n \in R_t$.

Lipton proposes two modifications to the exponential method, each of which forces all messages (plaintext and ciphertext) to be quadratic residues [Lipt79b]. The first method appends extra low-order bits to each message M, set to make the extended message a quadratic residue. The original message is recovered by discarding the extra bits. The second method multiplies nonresidue messages by a fixed nonresidue w (the product of two nonresidues is a quadratic residue). The original message is recovered by multiplying by w^{-1}, where w^{-1} is the inverse of w (mod n).

These results show that for some applications, an encryption algorithm must be more than just computationally strong. This particular application requires an algorithm that conceals not only the messages but their mathematical properties.

2.7.4 Oblivious Transfer

Rabin has devised a protocol whereby Alice can transfer a secret to Bob with probability 1/2. Thus, Bob has a 50% chance of receiving the secret and a 50% chance of receiving nothing. On the other hand, Bob will know whether he has received the secret; Alice will not. Clearly, the uncertainty must be agreeable to both Alice and Bob, or one of them would refuse to cooperate. Called the "oblivious transfer", the protocol is described by Blum [Blum81a] as follows:

Oblivious transfer protocol

1. Alice sends to Bob the product n of two distinct odd primes p and q. The primes p and q represent her secret. They may, for example, be the secret parameters to an RSA deciphering transformation.

2. Bob picks a number x at random, where $0 < x < n$ and $gcd(x, n) = 1$, and sends to Alice

$$a = x^2 \bmod n \ . \tag{2.10}$$

3. Alice, knowing p and q, computes the four roots of a: $x, n - x, y, n - y$ (see discussion following). She picks one of these roots at random and sends it to Bob.
4. If Bob receives y or $n - y$, he can determine p and q from x and y by computing

$$gcd(x + y, n) = p \text{ or } q$$

(see exercises at end of chapter). If he receives x or $n - x$, he learns nothing.

Equation (2.10) has four roots because n has two distinct prime factors. By Theorem 1.7, we know that any solution x of Eq. (2.10) must be a common solution of

$$x^2 \bmod p = a \bmod p \tag{2.11}$$
$$x^2 \bmod q = a \bmod q \ . \tag{2.12}$$

By Theorem 2.3, Eq. (2.11) has two solutions: x_1 and $p - x_1$, and Eq. (2.12) has two solutions: x_2 and $q - x_2$. The four solutions of Eq. (2.10) are thus obtained using the Chinese Remainder Theorem (Theorem 1.8).

Now, finding the solutions of Eq. (2.11) and (2.12) is particularly easy if $p + 1$ and $q + 1$ are divisible by 4. Observe that

$$(a^{(p+1)/4})^2 = a^{(p+1)/2} \bmod p = a(a^{(p-1)/2}) \bmod p = a \ .$$

The last equality holds because a is a quadratic residue modulo p; thus, by Theorem 2.5, $a^{(p-1)/2} \bmod p = 1$. This gives us the two solutions:

$$x_1 = a^{(p+1)/4} \bmod p$$
$$x_2 = a^{(q+1)/4} \bmod q \ .$$

Note that Bob cannot find the root y of Eq. (2.10) without knowing p and q. If he accidentally picks an x that is a multiple of p or q, he can compute $gcd(x, n) = p$ or q, but the probability of this happening is small for large p and q.

Example:

Let $p = 3$ and $q = 7$. Then $n = pq = 21$. Suppose Bob picks $x = 5$ and sends to Alice

$$a = 5^2 \bmod 21 = 4 \ .$$

Alice computes the roots

$$x_1 = 4^{(3+1)/4} \bmod 3 = 1$$
$$x_2 = 4^{(7+1)/4} \bmod 7 = 2 \ .$$

Applying algorithm *crt* of Figure 1.24, Alice then computes

$$\begin{aligned}
z_1 &= crt(n, p, q, x_1, x_2) = 16\\
z_2 &= crt(n, p, q, x_1, q - x_2) = 19\\
z_3 &= crt(n, p, q, p - x_1, x_2) = 2\\
z_4 &= crt(n, p, q, p - x_1, q - x_2) = 5\ .
\end{aligned}$$

Note that $z_4 = x$, the number Bob picked. Let $y = z_2 = 19$. Then $z_3 = n - y$ and $z_1 = n - x$ (whence they can be determined from x and y directly rather than by the Chinese Remainder Theorem—see exercises at end of chapter).

Suppose now that Alice sends the root y to Bob. Bob, using x, computes p and q as follows:

$$\begin{aligned}
gcd(x + y, n) &= gcd(5 + 19, 21)\\
&= gcd(24, 21) = 3 = p\ .
\end{aligned}$$

$$q = \frac{n}{p} = \frac{21}{3} = 7\ . \quad ■$$

The oblivious transfer protocol can be used to flip coins by telephone, exchange secrets, and send certified mail [Blum81a,Rabi81,Blum81b]. We shall describe the protocol for coin flipping by telephone. The problem here is to devise a scheme whereby Bob can call HEADS or TAILS and Alice can flip in such a way that each has a 50% chance of winning. Flipping a real coin over the telephone is clearly unsatisfactory because if Bob calls HEADS, Alice can simply say "Sorry, TAILS." The solution given by Blum [Blum81a] is as follows:

Coin flipping protocol

1. Alice selects two large primes p and q and sends $n = pq$ to Bob.
2. Bob checks if n is prime, a prime power, or even; if so, Alice cheated and loses. Bob picks an x and sends $a = x^2 \bmod n$ to Alice.
3. Alice computes the four roots of a, picks one at random, and sends it to Bob.
4. Bob wins if he can factor n.

2.8 KNAPSACK CIPHERS

We shall describe three public-key encryption schemes based on the **NP**-complete knapsack problem. The first two can be used for secrecy, but not authentication. The reason is that the enciphering transformation does not map the entire message space back onto itself; thus, it is not possible to take an arbitrary message and sign it by applying the deciphering transformation. By contrast, the third scheme can be used for authentication but not secrecy. The problem here is just the opposite: although the deciphering algorithm can be applied to all messages, the enciphering algorithm cannot.

Shamir [Sham79] studied the feasibility of constructing a knapsack system for both secrecy and authentication. In order to use a secrecy knapsack system for authentication, the system must be sufficiently dense that most messages can be signed. The interesting result is that any knapsack system with this property is polynomial solvable; thus a single knapsack system cannot be used for both secrecy and signatures.

2.8.1 Merkle-Hellman Knapsacks

Merkle and Hellman [Merk78] proposed a scheme whose security depends on the difficulty of solving the following 0—1 knapsack problem:

> Given a positive integer C and a vector $A = (a_1, \ldots, a_n)$ of positive integers, find a subset of the elements of A that sum to C; that is, find a binary vector $M = (m_1, \ldots, m_n)$ such that $C = AM$, or
>
> $$C = \sum_{i=1}^{n} a_i m_i \; . \qquad \textbf{(2.13)}$$
>
> This knapsack problem is adapted from Karp's knapsack problem [Karp72], which is to determine simply whether a solution M exists.
>
> ***Example:***
> Let $n = 5$, $C = 14$, and $A = (1, 10, 5, 22, 3)$. Then $M = (1, 1, 0, 0, 1)$ is a solution. ■

The knapsack problem is an **NP**-complete problem. The best known algorithms for solving arbitrary instances of size n require $O(2^{n/2})$ time and $O(2^{n/4})$ space [Schr79]. There is, however, a special class of knapsack problems, referred to as **simple knapsacks**, that can be solved in linear time. In a simple knapsack, the elements a_i $(i = 1, \ldots, n)$ are **super increasing** so that

$$a_i > \sum_{j=1}^{i-1} a_j$$

for $i = 2, \ldots, n$. This implies that

$$m_n = 1 \quad \text{iff} \quad C \geq a_n$$

and, for $i = n - 1, n - 2, \ldots, 1$,

$$(C - \sum_{j=i+1}^{n} m_j a_j) \geq a_i \, .$$

An algorithm for solving simple knapsacks is shown in Figure 2.19.

FIGURE 2.19 Solution to simple knapsack.

```
Algorithm snap (C, A): "Simple Knapsack Solution"
for i := n downto 1 do
  begin
    if $C \geq a_i$ then $m_i$ := 1 else $m_i$ := 0;
    $C := C - a_i * m_i$
  end;
if C = 0 then snap := M else "no solution exists"
```

Example:
Rearranging the elements of the vector in the preceding example to give A' = (1, 3, 5, 10, 22) shows that A' is a simple knapsack vector, whence $snap(14, A')$ gives the solution (1, 1, 0, 1, 0). ■

Merkle and Hellman show how to convert a simple knapsack into a **trapdoor knapsack** that is hard to solve without additional information. First, a simple knapsack vector $A' = (a'_1, \ldots, a'_n)$ is selected. This allows an easy solution to a problem $C' = A'M$. Next, an integer u is chosen such that

$$u > 2a'_n > \sum_{i=1}^{n} a'_i \ .$$

Then an integer w is chosen such that $gcd(u, w) = 1$, and the inverse w^{-1} of w mod u is computed using $w^{-1} = inv(w, u)$ (see Section 1.6.2). Finally, the vector A' is transformed into a hard knapsack vector $A = wA' \bmod u$; that is,

$$a_i = w * a'_i \bmod u \ .$$

Now, solving $C = AM$ is difficult, but with knowledge of the trapdoor information w^{-1} and u, the problem can be transformed into the easy problem:

$$\begin{aligned} C' &= w^{-1}C \bmod u \\ &= w^{-1}AM \bmod u \\ &= w^{-1}(wA')M \bmod u \\ &= A'M \bmod u \\ &= A'M \ . \end{aligned}$$

To apply the trapdoor knapsack problem to public-key encryption, let the public key be the hard knapsack vector A, and let the secret key be the simple knapsack vector A' together with the trapdoor information u and w^{-1} (actually A' can be computed from A, u, and w^{-1} by $A' = w^{-1}A \bmod u$). Let E_A denote the enciphering transformation using the public key A, and let D_A denote the deciphering transformation using the secret key (A', u, w^{-1}).

To encipher, the plaintext is broken into blocks $M = (m_1, \ldots, m_n)$ of n bits each. Each block M is then enciphered as

$$C = E_A(M) = AM \ .$$

C is deciphered by computing

$$D_A(C) = snap(w^{-1}C \bmod u, A') = M \ .$$

Example:
Let $A' = (1, 3, 5, 10)$, $u = 20$, and $w = 7$. Then $w^{-1} = 3$. The simple vector A' is transformed into the "hard" vector

$$\begin{aligned} A &= (7 * 1 \bmod 20, 7 * 3 \bmod 20, 7 * 5 \bmod 20, 7 * 10 \bmod 20) \\ &= (7, 1, 15, 10) \ . \end{aligned}$$

Let $M = 13$, which is the binary vector (1, 1, 0, 1).

Then

$$C = E_A(M) = 7 + 1 + 10 = 18\ ,$$

and

$$\begin{aligned} D_A(C) &= D_A(18) \\ &= snap(3 * 18 \bmod 20, A') = snap(14, A') \\ &= (1, 1, 0, 1) \\ &= 13\ . \quad \blacksquare \end{aligned}$$

Merkle and Hellman originally suggested using $n = 100$ or more. Schroeppel and Shamir [Schr79], however, have developed an algorithm that can solve knapsacks of this size. By trading time for space, their method can solve the knapsack problem in time $T = O(2^{n/2})$ and space $S = O(2^{n/4})$. For $n = 100$, $2^{50} \cong 10^{15}$; thus, a single processor could find a solution in about 11,574 days, and 1000 processors could find a solution in about 12 days (ignoring constants and figuring 8.64×10^{10} instructions per day). But if $n = 200$, $2^{100} \cong 10^{30}$, whence the algorithm is computationally infeasible.

Merkle and Hellman suggest choosing several pairs (u, w) and iterating the transformation $A = wA' \bmod u$ to obscure the transformation. Indeed, Shamir and Zippel [Sham80a] show that if this extra precaution is not taken, the scheme is highly vulnerable to cryptanalysis when u is known.

Although the (hard) knapsack problem is **NP**-complete, this does not imply that the trapdoor knapsack problem is also **NP**-complete. It could be that the peculiar structure of the system provides a shortcut solution. No faster solution has yet been found.

Pohlig [Pohl78a] has shown that if a hard knapsack has large simple subsets, it may be feasible to find a solution in a much shorter period of time. The probability of an arbitrary knapsack having large simple subsets is extremely small, however, so this does not seem to be a serious threat.

For $n = 200$, the a'_i are chosen from the range $[(2^{i-1} - 1)\ 2^{200} + 1, (2^{i-1})2^{200}]$. This gives 2^{200} choices for each a$'_i$, making it difficult for a cryptanalyst to determine any one of them. Because $u > 2a'_{200}$ is required, u is chosen from the range $[2^{401} + 1, 2^{402} - 1]$ and w from the range $[2, u - 2]$. Note that this puts each a_i in the range $[1, 2^{402} - 2]$. Thus, a 200-bit plaintext message M has a ciphertext message (knapsack problem) $C = AM$ with a 410-bit representation (summing 200 402-bit values can add up to $\lceil \log_2 200 \rceil = 8$ bits to the representation, where "$\lceil\ \rceil$" denotes the ceiling function). This is why the scheme cannot be used for authentication. There will be many 410-bit knapsack problems that do not correspond to 200-bit messages, whence the deciphering transformation cannot be applied to arbitrary messages of length 410.

For $n = 200$, the storage requirements for each public vector A are approximately $200 * 400 = 80$ kilobits. In contrast, the RSA scheme uses only about 1 kilobit per public key. Shamir [Sham80b] investigated the feasibility of reducing the storage requirements by either shortening the elements a_i or reducing their number. Let t be the length (in bits) of each a_i ($t = 400$ in the implementation suggested previously). The first strategy fails because the deciphering algorithm becomes ambiguous when $t < n$, and the scheme is insecure when t is sufficiently small. To implement the second strategy, an n-bit message M is broken into d multi-bit chunks $m_1, \dots, m_d$ such that each coefficient m_i in Eq. (2.13) is n/d bits, and only d elements a_i are needed. This strategy also fails because such "compact knapsacks" are easier to solve than 0–1 knapsacks.

Enciphering and deciphering are faster, however, than in the RSA scheme. For $n = 200$, enciphering requires at most 200 additions, while deciphering requires at most 200 subtractions plus one multiplication in modular arithmetic. In contrast, the RSA scheme requires about 1000 multiplications in modular arithmetic to encipher and decipher. Henry [Henr81] presents a fast knapsack decryption algorithm that optimizes the evaluation of

$$C' = w^{-1}C \bmod u \ .$$

Letting $b_{n-1}2^{n-1} + \cdots + b_0 2^0$ denote the binary expansion of C, evaluation of C' can be expressed as

$$[b_{n-1}(2^{n-1}w^{-1} \bmod u) + \cdots + b_0(2^0 w^{-1} \bmod u)] \bmod u \ .$$

Since the terms in parentheses are independent of C, they can be precomputed and stored in a table. Computation of C' thus reduces to a sequence of at most n table lookups and $n - 1$ additions, followed by a single reduction mod u. The reduction mod u is "easy" in that the sum can be no larger than nu.

2.8.2 Graham-Shamir Knapsacks

Graham and Shamir independently discovered a way of obscuring the superincreasing property of trapdoor knapsacks [Sham80c,Lemp79]. A Graham-Shamir trapdoor knapsack vector $A' = (a_1', \dots, a_n')$ has the property that each a_j' has the following binary representation:

$$a_j' = (R_j, I_j, S_j)$$

where R_j and S_j are long random bit strings, and I_j is a bit string of length n such that the jth high-order bit is 1 and the remaining $n - 1$ bits are 0. Each random bit string S_j has $\log_2 n$ 0's in its high-order bit positions so that summing does not cause them to overflow into the area of the I_j's. Thus a sum $C' = A'M$ has the binary representation:

$$C' = (R, M, S) \ ,$$

where $R = \sum_{j=1}^{n} R_j m_j$ and $S = \sum_{j=1}^{n} S_j m_j$. Notice that the vector of bit strings $((I_n, S_n), \dots, (I_1, S_1))$ (i.e., the elements a_j listed in reverse order and without the R_j's) is a simple knapsack vector. The R_j's are added to obscure this property. These knapsacks are even easier to solve than Merkle-Hellman trapdoor knapsacks, however, because M can be extracted directly from the binary representation of C'.

Example:
Let $n = 5$ where A' is given by

j	R_j	I_j	S_j	
1	011010	10000	000101	$= a_1'$
2	001001	01000	000011	$= a_2'$
3	010010	00100	000100	$= a_3'$
4	011000	00010	000111	$= a_4'$
5	000110	00001	000001	$= a_5'$

Let $M = (0, 1, 0, 0, 1)$. Then

$$\begin{aligned} C' &= A'M \\ &= a_2 + a_5 \\ &= (R_2 + R_5, I_2 + I_5, S_2 + S_5) \\ &= 001111 \quad 01001 \quad 000100 \ . \quad ■ \end{aligned}$$

A trapdoor knapsack vector A′ is converted to a hard knapsack vector A as in the Merkle-Hellman scheme; that is, by picking u and w and computing $A = wA' \bmod u$. Similarly, a message M is enciphered as in the Merkle-Hellman scheme, whence $C = E_A(M) = AM$. C is deciphered by computing $C' = w^{-1}C \bmod u$ and extracting from C' the bits representing M. Shamir and Zippel [Sham80c] believe this variant is safer, faster, and simpler to implement than the original scheme proposed by Merkle and Hellman.

2.8.3 Shamir Signature-Only Knapsacks

Unlike the RSA exponentiation scheme, the trapdoor knapsack schemes cannot be used for authentication. The reason is that the enciphering function is not "onto" the entire message space; thus, certain messages (indeed most!) cannot be deciphered before they are enciphered.

Shamir [Sham78a] shows how a trapdoor knapsack can be constructed to provide digital signatures. Shamir's knapsacks, however, cannot be used for secrecy. The scheme is based on the following **NP**-complete knapsack problem, which is also an extension of the one defined by Karp [Karp72].

> Given integers n, M, and $A = (a_1, \ldots, a_{2k})$, find $C = (c_1, \ldots, c_{2k})$ such that $M = CA \bmod n$; that is, such that
>
> $$M = \sum_{j=1}^{2k} c_j a_j \bmod n \ , \qquad \textbf{(2.14)}$$
>
> where each c_j is an integer in the range $[0, \log n]$.

In a signature-only knapsack system, n is a k-bit random prime number ($k = 100$ would be appropriate). The pair (A, n) is the public key, M is a message in the range $[0, n-1]$, and C is the signature of M. The recipient of a pair (M, C) can validate the signature by checking that

$$E_A(C) = CA \bmod n = M \ .$$

But the recipient cannot forge a signature for another message M' without solving the knapsack problem. The signer, however, has secret trapdoor information for generating a signature $C = D_A(M)$.

The secret trapdoor information is a $k \times 2k$ binary matrix H whose values are chosen at random. The vector A is constructed to satisfy the following system of modular linear equations:

$$\begin{pmatrix} h_{1,1} \dots h_{1,2k} \\ h_{2,1} \dots h_{2,2k} \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad \cdot \\ h_{k,1} \dots h_{k,2k} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_{2k} \end{pmatrix} = \begin{pmatrix} 2^0 \\ 2^1 \\ \cdot \\ \cdot \\ \cdot \\ 2^{k-1} \end{pmatrix} \bmod n \ ,$$

giving

$$\sum_{j=1}^{2k} h_{ij}a_j = 2^{i-1} \bmod n \quad i = 1, \dots, k \ .$$

Because there are only k equations in $2k$ unknowns, k values of A can be chosen at random, and the remaining values determined by solving the preceding system.

Let M be a message, and let $\overline{M} = (m_1, \dots, m_k)$ be the reversal of M in binary (i.e., m_i is the ith low-order bit in M, $1 \le i \le k$). M is signed by computing

$$C = D_A(M) = \overline{M}H \ ,$$

whence

$$c_j = \sum_{i=1}^{k} m_i h_{ij} \quad (1 \le j \le 2k) \ .$$

We see C is a valid signature because

$$\begin{aligned} E_A(C) &= CA \bmod n = \sum_{j=1}^{2k} c_j a_j \bmod n \\ &= \sum_{j=1}^{2k} \left(\sum_{i=1}^{k} m_i h_{ij} \right) a_j \bmod n \\ &= \sum_{i=1}^{k} m_i \left(\sum_{j=1}^{2k} h_{ij} a_j \right) \bmod n \end{aligned}$$

$$= \sum_{i=1}^{k} m_i 2^{i-1} \bmod n$$
$$= M \ .$$

Example:
Let $k = 3$ and $n = 7$. This will allow us to sign messages in the range [0, 6]. Let H be as follows:

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

and pick $a_1 = 1$, $a_2 = 2$, and $a_3 = 3$. Solving for the remaining values of A, we get $a_4 = 0$, $a_5 = 0$, and $a_6 = 4$, whence $A = (1, 2, 3, 0, 0, 4)$. Let $M = 3$; because this is 011 in binary, $\overline{M} = (1, 1, 0)$. The signature C is thus:

$$\begin{aligned} C &= \overline{M}H \\ &= (1 \quad 1 \quad 0) \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \\ &= (1, 1, 2, 1, 0, 2) \ . \end{aligned}$$

To validate C we compute

$$\begin{aligned} CA \bmod 7 &= [(1, 1, 2, 1, 0, 2)\ (1, 2, 3, 0, 0, 4)] \bmod 7 \\ &= [1 + 2 + 6 + 0 + 0 + 8] \bmod 7 \\ &= 17 \bmod 7 \\ &= 3 \ . \quad \blacksquare \end{aligned}$$

The signing procedure thus far is insecure, because someone might be able to determine H by examining enough (M, C) pairs. To prevent this, messages are randomized before they are signed. This is done using a random binary vector $R = (r_1, \ldots, r_{2k})$. First,

$$M' = (M - RA) \bmod n$$

is computed; thus

$$M = (M' + RA) \bmod n \ .$$

Next, M' is signed as described previously, giving a signature C'. Finally, the signature of M is computed from C' by adding R, giving $C = C' + R$. C is a valid signature of M because

$$\begin{aligned} CA \bmod n &= (C' + R)A \bmod n \\ &= (C'A + RA) \bmod n \\ &= (M' + RA) \bmod n \\ &= M \ . \end{aligned}$$

Example:

Let k, n, H, A, and M be as before, and let $R = (1, 0, 0, 0, 1, 1)$. Then

$$\begin{aligned} M' &= (M - RA) \bmod n \\ &= (3 - [(1, 0, 0, 0, 1, 1)\ (1, 2, 3, 0, 0, 4)] \bmod 7 \\ &= (3 - [1 + 0 + 0 + 0 + 0 + 4]) \bmod 7 \\ &= (3 - 5) \bmod 7 = -2 \bmod 7 \\ &= 5 \ , \end{aligned}$$

and

$$C' = \overline{M}'H \bmod n = (2, 0, 2, 1, 1, 1) \ .$$

The signature of M is thus

$$\begin{aligned} C = C' + R &= (2, 0, 2, 1, 1, 1) + (1, 0, 0, 0, 1, 1) \\ &= (3, 0, 2, 1, 2, 2) \ , \end{aligned}$$

which the reader should verify is also valid. ■

Because a signature depends on the random vector R, a message M can have multiple signatures C satisfying Eq. (2.14), as illustrated in the preceding examples for $M = 3$. This explains why the scheme cannot be used for secrecy. Because D_A is one-to-many, computing $D_A(E_A(M))$ might not give back M.

The signature-only knapsack system has the advantage of being fast. But, like the other knapsack schemes, it is not known whether it is as difficult to solve as the **NP**-complete problem on which it is based.

2.8.4 A Breakable NP-Complete Knapsack

Lempel [Lemp79] describes a conventional (one-key) cipher derived jointly with Even and Yacobi with the peculiar property of being **NP**-complete under a chosen-plaintext attack, yet easily breakable given enough known plaintext. The cipher uses an n-bit secret key of the form $K = (k_1, \ldots, k_n)$, and a knapsack vector $A = (a_1, \ldots, a_n)$ of positive elements, assumed known to the cryptanalyst. Messages are enciphered by breaking them into t-bit blocks of the form $M = (m_1, \ldots, m_t)$, where $t = \lceil \log_2(1 + \sum_{i=1}^{n} a_i) \rceil$.

To encipher a message M, the sender generates a random n-bit vector $R = (r_1, \ldots, r_n)$ and forms the t-bit sum:

$$S = A(K \oplus R) = \sum_{i=1}^{n} a_i(k_i \oplus r_i) \ .$$

M is then enciphered as the $(t + n)$-bit vector

$$C = (L, R), \text{ where } L = M \oplus S \ .$$

Because the last n bits of C contain R, a receiver knowing K and A can compute S and exclusive-or it with L (the first t bits of C) to recover M.

A cryptanalyst, knowing A and a single (M, C) pair, can find S by computing

$$L \oplus M = (M \oplus S) \oplus M = S \ .$$

To determine K, the cryptanalyst must solve the knapsack problem $S = A(K \oplus R)$, which is **NP**-complete.

If, however, the cryptanalyst knows a set of n pairs $(M_i, C_i) = (M_i, L_i, R_i)$, for $i = 1, \ldots, n$, such that the n vectors $U_i = 1^n - 2R_i$ are linearly independent (1^n is a vector of n 1's), the cryptanalyst can easily solve for the key K. To see how this can be done, observe that

$$\begin{aligned} K \oplus R_i &= K + R_i - 2(K * R_i) \\ &= R_i + K * U_i \ , \end{aligned}$$

where multiplication ($*$) is componentwise. This leads to the system of equations

$$\begin{aligned} S_i &= A(K \oplus R_i) \\ &= A(R_i + K * U_i) \\ &= AR_i + (U_i * A)K \quad i = 1, \ldots, n \ . \end{aligned}$$

Letting $T_i = S_i - AR_i$, the n equations can be expressed in matrix form as

$$\begin{pmatrix} T_1 \\ T_2 \\ \cdot \\ \cdot \\ \cdot \\ T_n \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ \cdot \\ \cdot \\ \cdot \\ U_n \end{pmatrix} \begin{pmatrix} a_1 & 0 & \ldots & 0 \\ 0 & a_2 & \ldots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \ddots & \cdot \\ \cdot & \cdot & & \cdot \\ 0 & 0 & \ldots & a_n \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ \cdot \\ \cdot \\ \cdot \\ k_n \end{pmatrix} .$$

Thus the system is readily solved for K when the U_i are linearly independent and the a_i positive. The probability of $N \geq n$ pairs (M_i, C_i) containing a subset of n linearly independent U_i is bounded below by approximately 1/3 for $N = n$, and quickly approaches 1 as N increases.

This example shows that it is not enough to base a cipher on a computationally hard problem. It is necessary to show that the cipher cannot be broken under any form of attack. The weakness in the scheme is caused by the linear relationship between the plaintext and ciphertext, which does not hold in the other knapsack schemes.

EXERCISES

2.1 Decipher the Churchyard cipher shown in Figure 2.2.

2.2 Decipher the following ciphertext, which was enciphered using a Vigenère cipher with key ART:

YFN GFM IKK IXA T .

2.3 Decipher the following ciphertext, which was enciphered using a Beaufort cipher with key ART:

CDZ ORQ WRH SZA AHP .

2.4 Decipher the following ciphertext, which was enciphered using a Playfair cipher with the key shown in Figure 2.11.

AR HM CW CO KI PW .

2.5 Decipher the ciphertext LJ (11 9) using the decipher matrix

$$\begin{pmatrix} 15 & 20 \\ 17 & 9 \end{pmatrix}$$

with the Hill cipher. (The plaintext is a two-letter word of English).

2.6 Solve the cipher of Figures 2.8–2.10 by finding the remaining two key characters and deciphering the text.

2.7 Consider a linear substitution cipher that uses the transformation $f(a) = ak \bmod 26$. Suppose it is suspected that the plaintext letter J (9) corresponds to the ciphertext letter P (15); i.e., $9k \bmod 26 = 15$. Break the cipher by solving for k.

2.8 Consider an affine substitution cipher using the transformation $f(a) = (ak_1 + k_0) \bmod 26$. Suppose it is suspected that the plaintext letter E (4) corresponds to the ciphertext letter F (5) and that the plaintext letter H (7) corresponds to the ciphertext letter W (22). Break the cipher by solving for k_1 and k_0.

2.9 Determine the unicity distance of ciphers based on affine transformations of the form $f(a) = (ak_1 + k_0) \bmod 26$. Assume the keys k_0 and k_1 generate a complete set of residues, and that all such keys are equally likely.

2.10 Consider a homophonic cipher that uses $26h$ ciphertext symbols, assigning h homophones to each letter of the English alphabet. Determine the number of possible keys (i.e., assignments of homophones), and use your result to calculate the unicity distance of the cipher.

2.11 Suppose that the keys used with DES consist only of the letters A–Z and are 8 letters long. Give an approximation of the length of time it would take to try all such keys using exhaustive search, assuming each key can be tested in one μsec. Do the same for keys 8 letters or digits long.

2.12 Let X' denote the bit-by-bit complement of a block X. Show that if $C = \mathrm{DES}_K(M)$, then $C' = \mathrm{DES}_{K'}(M')$. Explain how this property can be exploited in a chosen-plaintext attack to reduce the search effort by roughly 50%. (*Hint*: Obtain the ciphertext for a plaintext M and its complement M'.) (This symmetry in the DES was reported in [Hell76].)

2.13 Consider the RSA encryption scheme with public keys $n = 55$ and $e = 7$. Encipher the plaintext $M = 10$. Break the cipher by finding p, q, and d. Decipher the ciphertext $C = 35$.

2.14 Consider the Pohlig-Hellman exponentiation cipher, where

$$\begin{aligned} E_K(M) &= M^e \bmod p(x) \\ D_K(C) &= C^d \bmod p(x) \ , \end{aligned}$$

$e = 5$, $d = 3$, $p(x) = (x^3 + x + 1)$, and exponentiation is performed in the field GF(2^3). Because $\phi(p) = 7$ (see Section 1.6.3), $ed \bmod \phi(p) = 1$. Let $M = x^2 + 1 = 101$ in binary. Compute $C = E_K(M)$. Compute $D_K(C)$, showing that deciphering restores M.

2.15 Consider the RSA encryption scheme with $n = pq$, where $p = 5$ and $q = 7$. Prove that all keys d and e in the range $[0, \phi(n) - 1]$ must satisfy the equality $d = e$.

2.16 Consider the equations

$$\begin{aligned} x \bmod p &= x_1 \text{ or } p - x_1 \\ x \bmod q &= x_2 \text{ or } q - x_2 \ , \end{aligned}$$

where $n = pq$ for primes p and q. There are four common solutions, given by

$$\begin{aligned} z_1 &= crt(n, p, q, x_1, x_2) \\ z_2 &= crt(n, p, q, x_1, q - x_2) \\ z_3 &= crt(n, p, q, p - x_1, x_2) \\ z_4 &= crt(n, p, q, p - x_1, q - x_2) \ . \end{aligned}$$

Show that $z_4 = n - z_1$ and $z_3 = n - z_2$.

2.17 Using the result of the preceding exercise, find the 4 solutions to the equation $x^2 \bmod 77 = 4$ by first finding solutions to $x^2 \bmod 7 = 4$ and $x^2 \bmod 11 = 4$.

2.18 Let $n = pq$ for primes p and q. Given a, $0 < a < n$, let x and y be square roots of a modulo n such that $y \neq x$ and $y \neq n - x$. Show that $gcd(x + y, n) = p$ or q.

2.19 Show how coin flipping by telephone can be implemented using a scheme based on the mental poker protocol. Could either Bob or Alice have an advantage?

2.20 Rabin's public-key encryption scheme enciphers a message M as

$$C = M(M + b) \bmod n \ ,$$

where b and n are public and $n = pq$ for secret primes p and q. Give a deciphering algorithm for the case where $p + 1$ and $q + 1$ are divisible by 4. (*Hint:* compute d such that $2d \bmod n = b$. Then

$$(M + d)^2 \bmod n = (C + d^2) \bmod n \ .)$$

2.21 Suppose that the public key for a Merkle-Hellman knapsack system is given by the vector $A = (17, 34, 2, 21, 41)$, and that the ciphertext $C = 72$ resulted from enciphering some number M with A. The secret key is given by the values $u = 50$ and $w = 17$. Find M by deciphering C.

2.22 Consider Shamir's signature-only knapsack scheme, and let $n = 7$ and

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Given $a_1 = 1$, $a_2 = 2$, and $a_3 = 3$, compute a_4, a_5, and a_6. Sign the message $M = 3$, first without randomizing, and then using the random vector $R = (0, 1, 0, 1, 0, 1)$. Check the validity of both signatures.

2.23 *Class Computer Project:*

Teacher: Write a program to encipher a reasonably long message using a Vigenère or Beaufort cipher. (*Optional:* provide programs to compute the index of coincidence *IC* and print histograms of letter frequencies.)

Students: Break the cipher using the computer to analyze the ciphertext and to decipher the message.

2.24 *Class Computer Project:* Implement the DES.

Teacher: Write programs to convert ASCII character sequences into 64-bit blocks and vice versa. Each 64-bit block can be represented internally as a bit or character array of '1's and '0's so the bits are easily addressed; it can be represented externally as a record containing 64 characters ('1's and '0's). Create a file containing the data for the DES tables. Implement the DES, and encipher a message for the students. Give the students a skeleton of your program containing the declarations and statements used to input the DES tables.

Students: Complete the program and decipher the message.

2.25 *Class Computer Project:* Implement the RSA scheme using a 7-digit number n (this can be guaranteed by picking p and q in the range [1000, 3162]) and 6-digit data blocks.

Teacher: Write programs to convert character streams into 6-digit blocks and vice versa (assign a 2-digit number to each character).

Students: Generate keys. Exchange public keys and messages.

2.26 *Class Computer Project:* Implement one of the trapdoor knapsack encryption schemes.

REFERENCES

Adle79. Adleman, L., "A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography," *Proc. IEEE 20th Annual Symp. on Found. of Comp. Sci.*, pp. 55–60 (Oct. 1979).

Bark77. Barker, W. G., *Cryptanalysis of the Hagelin Cryptograph*, Aegean Park Press, Laguna Hill, Calif. (1977).

Beal78. "The Beale Ciphers," The Beale Cypher Assoc., Medfield, Mass. (1978).

Blak78. Blakley, B. and Blakley, G. R., "Security of Number Theoretic Public Key Cryptosystems Against Random Attack," *Cryptologia*. In three parts: Part I: Vol. 2, No. 4 (Oct. 1978), pp. 305–321; Part II: Vol. 3, No. 1 (Jan. 1979), pp. 29–42; Part III: Vol. 3, No. 2 (Apr. 1979), pp. 105–118.

Blak79. Blakley, G. R. and Borosh, I., "Rivest-Shamir-Adleman Public Key Cryptosystems Do Not Always Conceal Messages," *Comp. & Math. with Applic.* Vol. 5 pp. 169–178 (1979).

Blum81a. Blum, M., "Three Applications of the Oblivious Transfer: 1. Coin Flipping by Telephone, 2. How to Exchange Secrets, 3. How to Send Certified Electronic Mail," Dept. EECS, Univ. of California, Berkeley, Calif. (1981).

Blum81b. Blum, M. and Rabin, M. O., "How to Send Certified Electronic Mail," Dept. EECS, Univ. of California, Berkeley, Calif. (1981).

Bran79. Branstad, D., "Hellman's Data Does Not Support His Conclusion," *IEEE Spectrum* Vol. 16(7) p. 41 (July 1979).

Davi79. Davida, G. I., "Hellman's Scheme Breaks DES in its Basic Form," *IEEE Spectrum* Vol. 16(7) p. 39 (July 1979).

Davs80. Davies, D. W. and Price, W. L., "The Application of Digital Signatures Based on Public Key Cryptosystems," NPL Report DNACS 39/80, National Physical Lab., Teddington, Middlesex, England (Dec. 1980).

Deav80a. Deavours, C. A., "The Black Chamber: A Column; How the British Broke Enigma," *Cryptologia* Vol. 4(3) pp. 129–132 (July 1980).

Deav80b. Deavours, C. A., "The Black Chamber: A Column; La Methode des Batons," *Cryptologia* Vol. 4(4) pp. 240–247 (Oct. 1980).

Diff76. Diffie, W. and Hellman, M., "New Directions in Cryptography," *IEEE Trans. on Info. Theory* Vol. IT-22(6) pp. 644–654 (Nov. 1976).

Diff77. Diffie, W. and Hellman, M., "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," *Computer* Vol. 10(6) pp. 74–84 (June 1977).

Diff79. Diffie, W. and Hellman, M., "Privacy and Authentication: An Introduction to Cryptography," *Proc. IEEE* Vol. 67(3) pp. 397–427 (Mar. 1979).

Diff81. Diffie, W., "Cryptographic Technology: Fifteen Year Forecast," BNR Inc., Mountain View, Calif. (Jan. 1981).

Feis70. Feistel, H., "Cryptographic Coding for Data-Bank Privacy," RC-2827, T. J. Watson Research Center, Yorktown Heights, N.Y. (Mar. 1970).

Feis73. Feistel, H., "Cryptography and Computer Privacy," *Sci. Am.* Vol. 228(5) pp. 15–23 (May 1973).

Feis75. Feistel, H., Notz, W. A., and Smith, J., "Some Cryptographic Techniques for Machine to Machine Data Communications," *Proc. IEEE* Vol. 63(11) pp. 1545–1554 (Nov. 1975).

Frie18. Friedman, W. F., "Methods for the Solution of Running-Key Ciphers," Riverbank Publication No. 16, Riverbank Labs, Geneva, Ill. (1918).

Frie20. Friedman, W. F., "The Index of Coincidence and Its Applications in Cryptography," Riverbank Publication No. 22, Riverbank Labs., Geneva, Ill. (1920).

Frie67. Friedman, W. F., "Cryptology," *Encyclopedia Britannica* Vol. 6 pp. 844–851 (1967).

Gain56. Gaines, H. F., *Cryptanalysis*, Dover, New York (1956).

Gard77. Gardner, M., "Mathematical Games," *Sci. Am.* Vol. 237(2) pp. 120–124 (Aug. 1977).

Gill80. Gillogly, J. J., "The Beale Ciphers: A Dissenting Opinion," *Cryptologia* Vol. 4(2) pp. 116–119 (Apr. 1980).

Hamm71. Hammer, C., "Signature Simulation and Certain Cryptographic Codes," *Comm. ACM* Vol. 14(1) pp. 3–14 (Jan. 1971).

Hamm79. Hammer, C., "How Did TJB Encode B2?" *Cryptologia* Vol. 3(1) pp. 9–15 (Jan. 1979).

Hamm81. Hammer, C., "High Order Homophonic Ciphers," *Cryptologia* Vol. 5(4) pp. 231–242, (Oct. 1981).

Hell76. Hellman, M., Merkle, R., Schroeppel, R., Washington, L., Diffie, W., Pohlig, S., and Schweitzer, P., "Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard," Information Systems Lab., Dept. of Electrical Eng., Stanford Univ. (1976).

Hell79. Hellman, M. E., "DES Will Be Totally Insecure Within Ten Years," *IEEE Spectrum* Vol. 16(7) pp. 32–39 (July 1979).

Hell80. Hellman, M. E., "A Cryptanalytic Time-Memory Tradeoff," *IEEE Trans. on Info. Theory* Vol. IT-26(4) pp. 401–406 (July 1980).

Henr81. Henry, P. S., "Fast Decryption Algorithm for the Knapsack Cryptographic System," *Bell System Tech. J.*, Vol. 60 (5) pp. 767–773 (May–June 1981).

Hill 29. Hill, L. S., "Cryptography in an Algebraic Alphabet," *Am. Math. Monthly* Vol. 36 pp. 306–312 (June–July 1929).

Kahn67. Kahn, D., *The Codebreakers*, Macmillan Co., New York (1967).

Kam78. Kam, J. B. and Davida, G. I., "A Structured Design of Substitution-Permutation Encryption Networks," pp. 95–113 in *Foundations of Secure Computation*, ed. R. A. DeMillo et al., Academic Press, New York (1978).

Karp72. Karp, R. M., "Reducibility Among Combinatorial Problems," pp. 85–104 in *Complexity of Computer Computations*, ed. R. E. Miller and J. W. Thatcher, Plenum Press, New York (1972).

Kasi63. Kasiski, F. W., *Die Geheimschriften und die Dechiffrir-kunst*, Mittler & Son (1863).

Knut69. Knuth, D., *The Art of Computer Programming; Vol. 2, Seminumerical Algorithms*, Addison-Wesley, Reading, Mass. (1969).

Konf78. Kohnfelder, L. M., "On the Signature Reblocking Problem in Public-Key Cryptosystems," *Comm. ACM* Vol. 21(2) p. 179 (Feb. 1978).

Konh81. Konheim, A. G., *Cryptography: A Primer*, John Wiley & Sons, New York (1981).

Kowa80. Kowalchuk, J., Shanning, B. P., and Powers, S. A., "Communications Privacy: Integration of Public and Secret Key Cryptography," *Proc. Nat'l. Telecommunications Conf.*, pp. 49.1.1–49.1.5 (Dec. 1980).

Kruh77. Kruh, L., "The Churchyard Ciphers," *Cryptologia* Vol. 1(4) pp. 372–375 (Oct. 1977).

Lemp79. Lempel, A., "Cryptology in Transition," *Computing Surveys* Vol. 11(4) pp. 285–303 (Dec. 1979).

LeVe77. LeVeque, W. J., *Fundamentals of Number Theory*, Addison-Wesley, Reading, Mass. (1977).

Lipt79a. Lipton, R. J., "How to Cheat at Mental Poker," Comp. Sci., Dept. Univ. of Calif., Berkeley, Calif. (Aug. 1979).

Lipt79b. Lipton, R. J., "An Improved Power Encryption Method," Comp. Sci., Dept. Univ. of Calif., Berkeley, Calif. (Aug. 1979).

Mell73. Mellen, G. E., "Cryptology, Computers, and Common Sense," pp. 569–579 in *Proc. NCC, Vol. 42*, AFIPS Press, Montvale, N.J. (1973).

Merk78. Merkle, R. C. and Hellman, M. E., "Hiding Information and Signatures in Trapdoor Knapsacks," *IEEE Trans. on Info. Theory* Vol. IT-24(5) pp. 525–530 (Sept. 1978).

Merk81. Merkle, R. C. and Hellman, M. E., "On the Security of Multiple Encryption," *Comm. ACM* Vol. 27(7) pp. 465–467 (July 1981).

Morr77. Morris, R., Sloane, N. J. A., and Wyner, A. D., "Assessment of the National Bureau of Standards Proposed Federal Data Encryption Standard," *Cryptologia* Vol. 1(3) pp. 281–291 (July 1977).

NBS77. "Data Encryption Standard," FIPS PUB 46, National Bureau of Standards, Washington, D.C. (Jan. 1977).

Nive72. Niven, I. and Zuckerman, H. S., *An Introduction to the Theory of Numbers*, John Wiley & Sons, New York (1972).

Pele79. Peleg, S. and Rosenfeld, A., "Breaking Substitution Ciphers Using a Relaxation Algorithm," *Comm. ACM* Vol. 22(11) pp. 598–605 (Nov. 1979).

Pohl78a. Pohlig, S. and Hellman, M., "An Improved Algorithm for Computing Logarithms over **GF**(p) and its Cryptographic Significance," *IEEE Trans. on Info. Theory* Vol. IT-24(1) pp. 106–110 (Jan. 1978).

Pohl78b. Pohlig, S., "Bounds on a Class of Easily Solved Knapsacks," MIT Lincoln Lab., Lexington, Mass. (1978).

Prat42. Pratt, F., *Secret and Urgent*, Blue Ribbon Books, Garden City, N.Y. (1942).

Rabi79. Rabin, M. O., "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," MIT/LCS/TR-212, MIT Lab. for Computer Science, Cambridge, Mass. (Jan. 1979).

Rabi81. Rabin, M. O., "Exchange of Secrets," Dept. of Applied Physics, Harvard Univ., Cambridge, Mass. (1981).

Rive78a. Rivest, R. L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM* Vol. 21(2) pp. 120–126 (Feb. 1978).

Rive78b. Rivest, R. L., "Remarks on a Proposed Cryptanalytic Attack of the M.I.T. Public Key Cryptosystem," *Cryptologia* Vol. 2(1) pp. 62–65 (Jan. 1978).

Rive80. Rivest, R. L., "A Description of a Single-Chip Implementation of the RSA Cipher," *Lambda* Vol. 1(3) pp. 14–18 (1980).

Rive81. Rivest, R. L., "Statistical Analysis of the Hagelin Cryptograph," *Cryptologia* Vol. 5(1) pp. 27–32 (Jan. 1981).

Sam79. Sam, E., "Musical Cryptography," *Cryptologia* Vol. 3(4) pp. 193–201 (Oct. 1979).

Schr79. Schroeppel, R. and Shamir, A., "A $TS^2 = O(2^n)$ Time/Space Tradeoff for Certain **NP**-Complete Problems," *Proc. IEEE 20th Annual Symp. on Found. of Comp. Sci.*, (Oct. 1979).

Sham78. Shamir, A., "A Fast Signature Scheme," MIT/LCS/TM-107, MIT Lab. for Computer Science, Cambridge, Mass. (July 1978).

Sham79. Shamir, A., "On the Cryptocomplexity of Knapsack Systems," *Proc. 11th Annual ACM Symp. on the Theory of Computing*, pp. 118–129 (1979).

Sham80a. Shamir, A., Rivest, R. L., and Adleman, L. M., "Mental Poker," in *The Mathematical Gardner*, ed. D. Klarner, Prindle, Weber & Schmidt, Boston, Mass. (1980).

Sham80b. Shamir, A., "The Cryptographic Security of Compact Knapsacks (Preliminary Report)," pp. 94–99 in *Proc. 1980 Symp. on Security and Privacy*, IEEE Computer Society (Apr. 1980).

Sham80c. Shamir, A. and Zippel, R. E., "On the Security of the Merkle-Hellman Cryptographic Scheme," *IEEE Trans. on Info. Theory* Vol. IT-26(3) pp. 339–40 (May 1980).

Shan49. Shannon, C. E., "Communication Theory of Secrecy Systems," *Bell Syst. Tech. J.* Vol. 28 pp. 656–715 (Oct. 1949).

Simm77. Simmons, G. J. and Norris, J. N., "Preliminary Comments on the M.I.T. Public Key Cryptosystem," *Cryptologia* Vol. 1(4) pp. 406–414 (Oct. 1977).

Sink66. Sinkov, A., *Elementary Cryptanalysis*, Math. Assoc. Am. (1966).

Solo77. Solovay, R. and Strassen, V., "A Fast Monte-Carlo Test for Primality," *SIAM J. Computing* Vol. 6 pp. 84–85 (Mar. 1977).

Stah73. Stahl, F. A., "A Homophonic Cipher for Computational Cryptography," pp. 565–568 in *Proc. NCC*, Vol. 42, AFIPS Press, Montvale, N.J. (1973).

Suga79. Sugarman, R., "On Foiling Computer Crime," *IEEE Spectrum* Vol. 16(7) pp. 31–32 (July 1979).

Tuch79. Tuchman, W., "Hellman Presents No Shortcut Solutions to the DES," *IEEE Spectrum* Vol. 16(7) pp. 40–41 (July 1979).

Vino55. Vinogradov, I. M., *An Introduction to the Theory of Numbers*, Pergamon Press, Elmsford, N.Y. (1955).

Ward85. Ward, J. B., *The Beale Papers*, Pamphlet printed by Virginian Book and Job Print; reprinted by The Beale Cypher Assoc., Medfield, Mass. (1885).

Will80. Williams, H. C., "A Modification of the RSA Public-Key Encryption Algorithm," *IEEE Trans. on Info. Theory* Vol. IT-26(6) pp. 726–729 (Nov. 1980).