2. Cryptography

When messages can be protected from disclosure by physical means, no interesting theoretical problems arise -- security rests on the effectiveness of physical barriers. An intriguing problem does arise, however, when a message must be securely communicated to a friendly recipient. The term 'communication' is to be taken in its most general sense. Figure 2.1 shows an idealized communications environment.

Figure 2.1
Generalized Communications Environment

The communications medium may be a slip of paper on which a message is written, an information-theoretic channel, or a physical file on a computer system. The security problem in such a setting is to 'write' the message in such a way that an enemy, even if he eavesdrops on the communications medium, cannot deduce the contents of the message.

The problem of secret writing is a very old one, and the many attempts to secure sensitive messages (particularly military and diplomatic messages) have endowed the field with a rich history and a devoted following of amateur and professional experts (see the bibliographic notes at the end of this section). The use of these techniques in computer systems has given rise to a new and increasingly sophisticated technology, and it is on these applications that we will concentrate.

2.1 Ciphers and Cryptosystems. To guarantee the security of messages placed in the communications medium, the system shown in Figure 2.1 is endowed with additional structure -- it becomes a cryptosystem. In a cryptosystem there is an enemy or attacker who has access to the communications medium. The enemy is also assumed to have computational resources and memory at his disposal. Like the noisy channel of information theory, the enemy may introduce errors into messages he sees in the medium. Unlike a noisy channel, however, the enemy does not always corrupt messages with random errors; he may introduce highly biased errors to confuse or deceive the receiver. The enemy may have three -- not always exclusive -- goals in affecting normal communications:

1. violating the secrecy of the communication,
2. confounding the receiver with a corrupted message,
3. deceiving either the transmitter or the receiver or both about the identity of the opposite party.

The first threat, violating the privacy of the transmitter, is the most familiar incarnation of the cryptographic problem. Protecting the communication in the presence of an enemy who pursues the latter goals are problems which have come into prominance only recently. These are problems of <u>integrity</u> and <u>authentication</u>, respectively. For example, the authentication problem arises in the login procedure for multiuser computer systems when the system software (the receiver) must insure that the login name it receives belongs to a legitimate user (a transmitter) and not to an intruder attempting to illegally use the computer. The integrity problem arises in electronic funds transfer (EFT) systems: a corruption of a funds transfer could result in funds being deposited in an unauthorized account belonging to the enemy.

These threats can be met by exploiting the most notable characteristic of a cryptosystem; the transmitter does not place the message itself in the communications medium. Rather, a second message is transmitted, a message <u>related</u> to the original message by a transformation known to the the transmitter and by a (possibly different) transformation known to the receiver. The difficulty of unraveling this relationship without exact knowledge of the transformations is what gives various cryptosystems their specialized properties. Figure 2.2 shows an idealized cryptosystem.

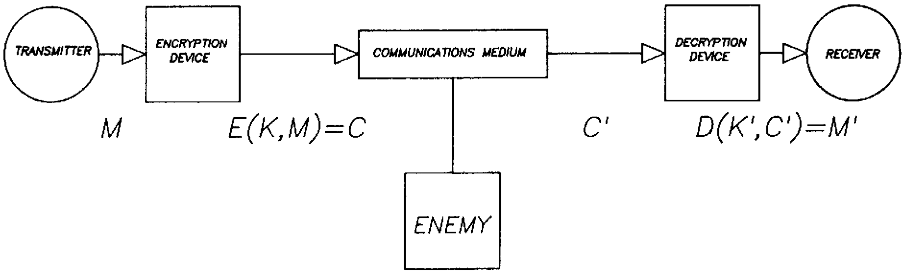

Figure 2.2
A Cryptosystem for Transmitting Message M

The transmitter composes a message M (the plaintext) and submits M to an encryption or enciphering device. The encryption device invokes an encrypting function E. The enciphering transformation is determined by two parameters. In addition to the plaintext, M, E also requires some additional information, K, called the encrypting key. In this sense, E determines a class of message transformations E_K such that $E_K(M) = E(K,M)$ whenever K is a valid key and M is a valid message. From the point of view of an observer who does not have access to an encrypting key, a good E will spread the plaintext messages evenly and apparently randomly over a set of messages called the ciphertext. The encrypting key may be held by the transmitter, by the device or by a third party. The ciphertext C is placed onto the communications medium where the enemy may examine it, store it, operate on it, and, finally, corrupt it to C'. The decryption device invokes a decrypting algorithm D. D takes two arguments: the alleged ciphertext C' and a decrypting key, K'. The nature of K' is fixed by the nature of the cryptosystem, but in general, K' depends on K (in a conventional cipher, $K=K'$) in such a way that $\{D_{K'}\}$ determines a class of algorithms for which

$$E_K(M) = C \text{ implies } D_{K'}(C) = M.$$

That is, K' represents the minimal amount of information that the decrypting device needs in order to recover plaintext from ciphertext. In Figure 2.2, the ciphertext may have been corrupted by the enemy so that $M' \neq M$. The threats that must be met are:

> <u>Threat to Secrecy</u>: $M = M'$ and the enemy is able to determine M from C and possibly other information.
>
> <u>Threat to Integrity</u>: $M \neq M'$ but neither the transmitter nor the receiver is able to detect the corruption of C.
>
> <u>Threat to Authorization</u>: C' is composed by the enemy, but the receiver believes that C' is ciphertext originating from the transmitter.

As we will discuss in a later section, it is possible to respond to authorization and integrity threats by responding to secrecy threats only and using <u>protocols</u> , i.e., algorithms known to all participants in a cryptographic exchange) to insure that the enemy cannot corrupt the message or impersonate the transmitter. Threats to secrecy cannot be met simply by insuring that $E_K = D_{K'}^{-1}$ -- that condition is trivially satisfied by the identity transformation which provides no secrecy whatsoever! It must be the case that the enemy cannot infer M from C. The process of deriving a general solution to such an inference problem (e.g., an algorithm to derive K' from samples of C) is known as <u>cryptanalysis</u>. What is the meaning of 'cannot infer'? Obviously, practical requirements vary from the strict security requirements placed on many military systems to the rather bland security requirements of some commercial applications. One view to which we will return later is that the communications medium together with the enemy constitutes a game-theoretic channel and the purpose of the cryptosystem is to make a winning strategy for the channel as unlikely as possible.

There are essentially only two possibilities for constructing the encrypting/decrypting devices: codes and ciphers. A code uses the entire plaintext language as a basis for constructing C;, that is, the code must have established the semantic content of every possible message to be sent through the system. The only way to encode messages through such a system is to provide a code-book, a list of possible messages and their encoded values, to the transmitter and to the receiver. Since there need be no relationship at all between the plaintext phrases and the ciphertext of the code-book (other than the completely arbitrary one established by the code-book itself) there is very little of theoretical significance that can be said about this approach. The problem with codes is that unless the phrase to be transmitted has already been conceived and placed in the code-book, communication is not possible. A cipher, on the other hand, exploits only the symbolic nature of the transmission; a cipher is a mapping that assigns new symbols of ciphertext to symbols or groups of symbols in the plaintext. Since every message is composed of plaintext symbols and the cipher completely specifies how plaintext symbols are to be replaced by ciphertext symbols, communication of arbitrary messages is possible.

There is a problem raised by ciphers: the relationship between ciphertext and plaintext is no longer the arbitrary one imposed by a code-book, rather, it is an extension of the simpler relationship between symbols in ciphertext and plaintext and so may be revealed by cryptanalysis.

For the purposes of this survey, we will distinguish between ciphers which carry out the encipherment by alphabetic substitutions, that is, which treat the plain text as a sequence of symbols to be encrypted (these ciphers are called <u>stream</u> ciphers) and ciphers which divide messages into blocks of characters of fixed length -- possibly after padding or compressing the message -- and derive ciphertext from plaintext by operating on blocks of characters rather than streams of characters (these ciphers are called <u>block</u> ciphers). Although stream ciphers retain their importance for many applications, it is block ciphers which have had the greatest recent impact on the field of cryptography. Modern computers naturally group alphabetic characters into strings of binary digits, so block encipherment can be viewed as an arithmetic problem and the security properties of the encrypting algorithms can be treated analytically. Moreover, by permitting substitutions on large blocks of plaintext at hardware speeds, very high speed encrypted data transmission can take place.

Categorizing ciphers into stream and block ciphers has another advantage. In a stream cipher, the basic allowable operation on a message is the <u>substitution</u> of one symbol for another. That is, if the message M to be transmitted is composed of symbols from an alphabet A,

$$M = a_1 \ldots a_n, \text{ each } a_i \in A,$$

then C must be defined with respect to n substituions $f_1, \ldots, f_n$ on A. In other words,

$$C = f_1(a_1) f_2(a_2) \ldots f_n(a_n),$$

each f_i:A-->A, $a_i \in A$. It is possible that each f_i also depends on one or more predecessors. With a block cipher, the concept of substitution still makes sense, but now there is an additional operation that can be defined: <u>permutation</u>. Suppose that the message M is composed of m blocks $B_1 \ldots B_m$ of n characters each. Then

$$C = C_1 \ldots C_m,$$

but now for each i, $1 \leq i \leq m$, if

$$B_i = b_1 \ldots b_n, \text{ each } b_j \in A,$$

then

$$C_i = b_{\pi(1)} \ldots b_{\pi(n)},$$

where π is a permutation of the integers 1,...,n. Thus a block cipher can be viewed as composed from a <u>network</u> of components which successively substitute for elements of A and permute positions within a block of characters. Figure 2.3 shows such a network (called an S/P network):

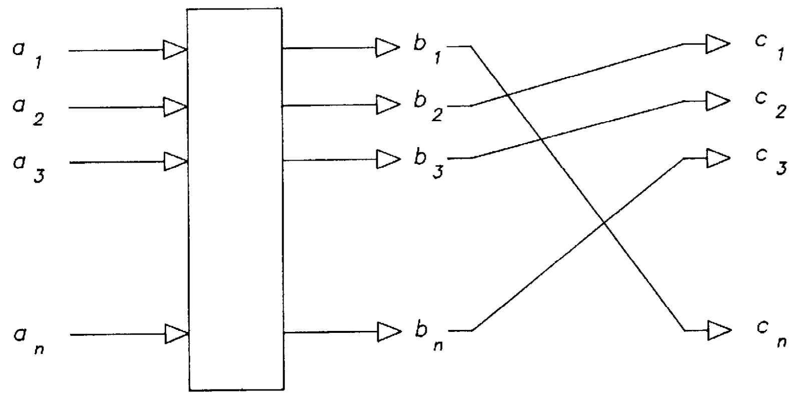

Figure 2.3
One Stage of a Substitution-Permutation (S/P) Network

2.2 Stream Ciphers. From now on, we will systematically confuse letters of the alphabet a,b,c, ... with their position in the usual ordering of the alphabet. Marks of punctuation and the digits are not directly represented in this way, but all of the following extends to a more realistic alphabet in an entirely straightforward way.

Legend assigns Julius Caesar credit for suggesting the following stream cipher:

$$f(x) = x+3 \pmod{26}.$$

Any cipher which is expressible by

$$f(x) = x+K \pmod{26}$$

is called a <u>Vigenere</u> cipher. Figure 2.4 shows the so-called Vigenere square -- a complete description of all possible Vigenere ciphers.

Plain--->	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
Cipher	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	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	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
V	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	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	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	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	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	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	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	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	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
	m	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	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	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	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	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	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	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	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	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	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	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	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	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	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

Figure 2.4
The Vigenere Alphabets

Each substitution alphabet is uniquely identified by both the amount of shift K and by its starting letter. Thus, there are only 26 keys for a cryptosystem built on such a cipher. It is easy to see how successful cryptanalysis of such a system would work, even if the enemy knows nothing about the message. Upon receipt of cipher text C, the enemy simply chooses a string of characters long enough to contain a meaningful message fragment. The enemy then carries out all possible substitutions on the reduced piece of ciphertext until he finds a meaningful translation. Solving the cipher is then equivalent to solving a linear congruence.

Thus, the major difficulty with Vigenere ciphers is that the cryptanalyst can count on the ordering of the ciphertext characters to be the same as in the alphabet A. The process of destroying this ordering information is known as <u>decimation</u> of an alphabetic sequence. If a and n are integers with (a,n)=1, then the residues a, 2a,...,na modulo n are all distinct. Thus, a cipher which destroys the normal alphabetic order is

$$f(x) = ax \pmod{26},$$

where a is relatively prime to 26. The constant a is known as the decimation interval. Once the sequence has been decimated it is, of course, possible to shift by an amount b (mod 26), so that a general decimating cipher is given by the linear transformation

$$g(x) = ax + b \pmod{26}.$$

To use g in a cryptosystem, the transmitter and receiver share the key (a,b). To encrypt, the transmitter computes for every symbol y, g(y)=c. To decrypt, the receiver solves the congruence

$$c = ax + b \pmod{26}.$$

If a and b are relatively prime, the congruence is uniquely solvable for x. This technique would appear to provide increased security. For example, comparing decimated alphabetic symbol frequencies for ciphertext and standard messages is not such an easy job, because the decimation has changed the shape of the frequency distribution. Still, if the enemy can determine two cipher-plaintext pairs (c_1,x_1) and (c_2,x_2), he can solve the congruences

$$c_1 = ax_1 + b \pmod{26}$$

$$c_2 = ax_2 + b \pmod{26}$$

for the decimation interval a and the shift amount b. This is the general line of attack for decimations based on linear transformations. One way of obtaining such ciphertext-plaintext pairs is to to examine the ciphertext for low frequency digraphs (symbol pairs) and high frequency <u>digraphs</u>. Since the plaintext alphabet separation of digraphs is known, the cryptanalyst can guess decimation intervals and plaintext-ciphertext pairs.

Thus, linear transformations have an inherent weakness: knowledge of cipher-plaintext pairs reveals too much about the substitution. This difficulty is apparently avoided if there is a <u>random</u> assignment of ciphertext letters to plaintext letters. However, the very randomness of the assignment means that the size of the key must be the size of the complete alphabetic sequence. Therefore, most cryptographic problems based on random assignment try to systematically induce a random-looking assignment. While these ciphers make entertaining puzzles, they also fall to frequency approaches -- usually by tables of suffixes and prefixes, distinguishing vowels from consonants, looking for 'pattern words', counting digraphs and trigraphs, and working on short (five letter) cipher fragments.

The encryption schemes above always encipher into a single fixed cipher alphabet. In such schemes, encipherment is said to be <u>monoalphabetic</u>. The usual method of attack on a monoalphabetic system is to use the fact that certain statistical properties of the plaintext alphabet are carried over into the cipher alphabet -- solution of the cipher is possible because a plaintext letter is always represented by the same ciphertext letter. An additional factor of confusion might, therefore, be added by using several cipher alphabets. For example, a key may be a sequence of designations of Vigenere alphabets. Even occurrences of digraphs can apparently be obliterated by these substitutions. Such ciphers are known as polyalphabetic ciphers, and most of conventional cryptanalytic theory is based on their solution.

The apparent security of complex polyalphabetic ciphers forms the basis for rotor machines. Very simply, the rotor machines determine, by mechanical gears and electrical interconnections of typewriter keys to indicator lights, a sequence of substitution alphabets. For example, the military version of the German Enigma determines a substitution alphabet by interactions of several substitutions. First, a plugboard substitution P results in a systematic interchange of letters. This substitution is self-reciprocal in the sense that that $P^{-1} = P$. A series of rewirings of keyboard to input lines results in a substitution E. Next, a series of enciphering rotors each determines a substitution. The rotors also determine the subsequent substitutions. If Q represents a rotor substitution at some position, then the other rotated positions are generated by the group of substitutions

$$C^{-i}QC^{i},$$

where C^{i} is the cipher x+i(mod 26). A final rotor, the reflector, yields a substitution based on 13 interchanges of alphabetic characters. A plaintext character x may then be enciphered by:

$$P^{-1}E^{-1}R_1{}^{-1}R_2{}^{-1}R_3{}^{-1}RR_3R_2\ R_1EP(x) = c.$$

The procedure for encipherment requires a daily setting of plugs and rotors. A message begins with a 'telegram key' of three letters, whose encipherment by the receiving machine determines the final setting of the three enciphering rotors to be used for the transmission. If E=I (as, in fact, was the case in the military version of Enigma) and the total rotor state is represented by Z, then a solution is represented by rewriting the equation above as

$$P^{-1}Z_iP(x_i) = c_i$$

for each of the rotor positions Z_i for the known plaintext $x_1,\ldots,x_n$. Since P is self-reciprocal, this reduces to

$$Z_iP(x_i) = P(c_i).$$

By careful analysis of the consistency of given data with the assumption of the existence of the required P, it is possible in principle to test a rotor position and its subsequent positions by mechanical means, but the number of possibilities to be tried grows combinatorially. However, it is possible to mechanically solve Enigma by careful pruning of the branching tree of possible P's to be tried. It is an interesting historical note that in the World War II cryptanalysis of Enigma, the transmitters made it easy to obtain plaintext to use in the cryptanalysis. Common telegram keys were AAA, BBB, CCC, and so on. Even when it was required that such keys not be used, this information was known to the cryptanalysts. Futhermore, several transmitters were prone to sending such stylized messages that the analysts could guess plaintext.

Although the amount of computation needed to cryptanalyze polyalphabetic ciphers can be quite forbidding, they are not secure. As the Enigma example shows, the application of knowledge concerning plaintext messages, the design of E, and general properties of sets of substitutions can be combined with massive calculations to exhaustively search out solutions to such systems. There is a notable exception, however.

When the key length is permitted to be as long as the message itself, a perfectly secure stream cipher can be constructed. A variation on the Vigenere cipher, called the Vernam cipher, or the one-time pad, is constructed as follows. Let M be the message $a_1 \ldots a_n$, where each a_i is expressed as a binary digit 0,1. A key for M is a sequence of binary digits $K = k_1 \ldots k_n$. Encipherment is obtained by the bitwise addition of messages and key modulo 2 (denoted $\oplus$). Thus, to encipher M, the transmitter forms

$$M \oplus K = C,$$

while the receiver decrypts by calculating

$$C \oplus K = M.$$

If the key is random, the cipher is theoretically unbreakable (see Section 2.3). To use such a cryptosystem, the correspondents must each have in their possession identical random 'key tapes' from which the 'next' n bits can be extracted to form the key for the current message. This may result in unacceptable amounts of key information to be exchanged, so it is natural to ask whether the key distribution requirements can be reduced in some fashion. For example, can a short random key be used over and over? The answer is no, not with absolute security, since the resulting periodicity can be exploited by a cryptanalyst. A variation on this idea is to use the $\oplus$ of two small keys of length n_1 and n_2, where $(n_1,n_2)=1$. The period of the resulting key is $n_1 n_2$. Even this scheme has been successfully cryptanalyzed.

2.3 Information-Theoretic Cryptanalysis. A cryptographic key represents information about which the enemy is ignorant. Due to the nature of a cryptosystem it is not possible to assume that the enemy does not know the nature of the cryptographic algorithm in use. In addition to the algorithm, the enemy may also have access to other information which can be of use in analyzing a cipher.

<u>Ciphertext Only Analysis</u>: In this attack on the system, the enemy only has access to the ciphertext he sees on the medium. Although it is a very weak sort of attack, there are many ciphertext only analyses available of apparently secure ciphers. Most ciphertext only attacks work through frequency counts and statistical properties of language.

<u>Known Plaintext Analysis</u>: The enemy in a known plaintext attack has several plaintext-ciphertext pairs from which to work. The example of the Enigma analysis shows how this information can be used to analyze the system. The Enigma example also shows that it is easy enough to gain probable plaintext -- a cryptographic system which cannot withstand a known plaintext assault cannot be considered secure.

<u>Chosen Plaintext Analysis</u>: In a chosen plaintext analysis, the enemy can submit unlimited portions of plaintext to the system and observe the corresponding ciphertext. It is obviously the most severe attack that can be mounted on a cryptosystem.

In assessing the security of any system, we usually assume the worst case. For cryptosystems, the worst case is that the enemy is attempting a chosen plaintext analysis of the system and has access to all system information <u>except</u> the cryptographic key. There are two ways to assess the security of a system. First, a system can be called secure if it is <u>unconditionally</u> secure, that is, if the enemy, regardless of the computational power he brings to bear on the problem, cannot analyze the system. That sounds very much like a worst case analysis, but it may, in fact, be too pessimistic. If the result of an unconditional security requirement is that no usable system is secure, then either the idea of security has to be abandoned or the worst case capabilities of the enemy must be more carefully assessed. If, for example, the enemy can only successfully cryptanalyze the system with an impossibly powerful computer, then an unconditionally insecure system may, in practice, be perfectly secure.

In this section, we will sketch the information-theoretic model of security. In the next section, we present a model based on <u>feasible</u> attacks by the enemy. The distinction between the concepts is crucial and has lead to the current advances in cryptographic theory.

The theory of unconditional security parallels the information theory of noisy channels. Let the encryption algorithm be $E(.,.)$ as described above. If message M is enciphered as $E(K,M)=C$, then the enemy observing the communications medium will attempt to obtain M from C by applying a cryptanalytic function h; in general, $h(C) = M' \neq M$ -- notice that h is not a function of the key K' since knowledge of K' is concealed from the enemy. E is <u>secure</u> if it is not possible that $h(C) = M$, regardless of how h is chosen. In other words, it seems reasonable to regard the cryptosystem as secure if

$$\mathrm{Prob}\{h(C)=M\} < \varepsilon.$$

It is usual to assume that each message occurs with probability $P(M)$ and that each potential ciphertext C will fall into the enemy's hands with probability $P(C)$. Let $\mathbf{K}$ be the set of available keys. Keys are chosen randomly and uniformly from $\mathbf{K}$. If $E_K(M) = C$ for all $K \in \mathbf{K}$, then the conditional probability of C given that a key in $\mathbf{K}$ is used for encipherment is denoted $P_M(C)$ (cf. the known-plaintext attack described above). Then E is unconditionally secure if $P_M(C)$ is independent of M:

$$P_M(C) = P(C).$$

In information theoretic terms, the uncertainty of the keys should be at least as great as the uncertainty of the messages.

Suppose that X is the transmitter of a stream of bits to Y in the presence of an enemy Z. If the communications medium is a <u>binary symmetric channel</u>, then Z will see $u \oplus 1$ with probability p, if u is the bit actually sent by X. If the channel is simply noisy, then Z <u>is</u> the receiver, and the corruption of u with probability p is due to the channel. If Z is the enemy, then the 'corruption' is really the result of encipherment. In either case, I -- the uncertainty <u>removed</u> -- is defined by

$$I = \sum \sum p(x,z) \; [\log(p(z|x)/p(z))],$$

where the summations are over the messages x,z sent and received by X and Z.

In applying information theory to noisy channel coding, the point is to maximize I. In cryptology, we would like to have I = 0. In this simple cryptosystem this condition obtains when p=0.5. Thus, in this model, the perfect concealer of messages is the Vernam cipher:

$$C = M \oplus K.$$

When Prob{K=1} = Prob{K=0} = 0.5,

$$\mathrm{Prob}\{h(c_1 \ldots c_k) = m_1 \ldots m_k\} \le 2^{-k}.$$

It is usually more convenient to work with the related measures of <u>entropy</u> and <u>equivocation</u>. If I is the uncertainty removed, then -I is the equivocation, the average ambiguity in the transmission.

More generally, assume that all messages of exactly N symbols chosen from a fixed alphabet A, |A| = L, so that there are L^N possible messages. The quantity $R_0 = \log(L)$ is sometimes called the <u>absolute rate</u> of the language, and so the number of possible messages is also given by

$$L^N = 2^{R_0 N},$$

and, assuming that the ciphertext is also written in A, this also expresses the number of possible ciphertexts. From the point of view of the transmitter, some messages are meaningful and so occur with nonzero probability, while some messages are meaningless, and, assuming a rational transmitter, occur with probability zero. The number of meaningful messages is assumed to be 2^{RN}, where the quantity R is called the <u>rate</u> of the language. Each meaningful message occurs with probability 2^{-RN}. The <u>redundancy</u> of a language is a measure of the number of different expressions for a given meaningful message: formally, the redundancy D is

$$D = R_0 - R.$$

The number of keys is defined to be 2^H, where each key is equally likely. The constant H is called the key <u>entropy</u>. The point of a cryptosystem is to create doubt on the part of the enemy as to the exact nature of the message M. As in the case of the binary symmetric channel, the only information to which the enemy does not have access is the key, so the entropy of a random key should represent the amount of uncertainty of the enemy given that he has received C. The <u>equivocation</u> introduced is the uncertainty introduced in the system by hiding the key. There are two sorts of equivocation which are important in determining the security of the system. Suppose that the message M_1 is enciphered to $E(K_1,M_1) = C_1$ by the key K_1. If there is no other meaningful message M so that $D(K,C_1) = M$, then -- remember that the enemy has arbitrary resources at his disposal -- it must be assumed that the enemy will eventually discover this fact and so compromise C_1. If, on the other hand. $E(K_2,M_2) = C_1$, where M_2 is also meaningful, then the enemy cannot compromise C_1 since it legally and meaningfully decrypts ambiguously (e.g. in the binary symmetric channel I = 0). Let N(m) be a random variable which represents the number of meaningful messages which encrypt to C. If, for a suitably large constant Q,

$$\text{Prob}\{N(m) > Q\} \dashrightarrow 1,$$

then the system must be secure in the sense outlined above. That is, the best that h can do is allow the enemy to make a list of Q possible meanings of C and to choose M′ randomly from this list; clearly for large Q the probability that M′ = M is small.

Another related measure of security arises when, for $E(K_1,M_1) = C_1$ as above, there is a key $K_2 \neq K_1$ such that $E(K_2,M_1) = C_1$. Thus, even though h may reveal the pair (M_1,C_1) it does not reveal the key used to encipher M_1.

Formally, if $P_C(K)$ and $P_C(M)$ represent the conditional probabilities of key K and message M given ciphertext C has been received, then the message equivocation $H_C(M)$ and key equivocation $H_C(K)$ are defined by:

$$H_C(X) = -\sum P(E,X)\log P_E(X),$$

where the summation is over E,K if X=K and E,M if X=M.

From the discussion above, an unconditionally secure system must have <u>incoherent</u> key streams. That is, the entropy of each key symbol must be at least as great as the average information content per symbol of the message. This condition is met by the Vernam cipher with a random nonrepeating key tape. Consider messages of length N in a language with rate R and let the rate of the key alphabet be R_K; then unconditional security requires

$$RN \leq R_K\sigma(N),$$

where $\sigma(N)$ is the minimal length key needed to encipher messages of size N. This condition is clearly satisfied by Vernam ciphers.

Finally, it is evident that the minimal amount of plaintext required for unique solution of a cipher can be made large by decreasing the redundancy of the language. At D=0, the cryptosystem admits no intersymbol dependencies (e.g., the output of a perfect source coder); unfortunately for such a system all errors go undetected, so the authorization and integrity problems are unsolvable for the cipher.

2.4 Feasibility of Cryptanalysis. So much for omnipotent enemies -- although we can get perfectly usable ciphers, the only provably secure system have severe drawbacks. What does it mean to <u>limit</u> the power of the enemy to feasible computation? To be precise, we must introduce the basic language of computational complexity theory. It seems fair to restrict the computational abilities of the enemy to effective procedures, that is, procedures that can be executed on computers. In solving a problem S on an idealized machine, the enemy will either fail to solve S or will solve it by using a finite amount of computational resources, such as time or memory. We will limit our attention to time as the resource, and denote by T(I) the amount of time taken on the idealized machine to solve instance I of S (forgetting for the moment about the units of time and the particular technology assumed in the idealized machine). We also assume that it is possible to consistently assign a measure of <u>size</u> to instances of problems. For example size(I) might be the number of bits required to represent I using a standard encoding of numbers, punctuation, etc. In this case, we usually simply write |I| for size(I). The number of instances of a given problem of some fixed size n is finite, so there is a number T(n) which is a lub on the set

$$\{T(I) \mid size(I) = n\}.$$

The <u>worst case complexity</u> of the indicated solution to the problem is completely described by the function T(n). Notice that the units of measured time can only determine T(n) up to a constant; that is, a rescaling of time units is a constant multiple of the given units. By the same token, a change in computing technology can be expected to change the running time of a given algorithm by only a constant factor (e.g., the popular representation of the advance of technology is the factor increase in computing speed). In summary, it seems enough to know about the growth rate of T(n) up to a constant factor. Therefore, we will characterize T(n) by its order: $T(n) = O(g(n))$ if there is a constant c such that $T(n) \leq c \cdot g(n)$ for all $n > n_0$. On the other hand, if every solution to S has worst case complexity $T(n) \geq g(n)$ for infinitely many values of n, then the complexity of S cannot be O(g(n)), so g(n) is a <u>lower bound</u> on the complexity of S.

Assessing feasibility in computation is based on empirical observations of time bounds of algorithms. A problem is said to be <u>feasibly solvable</u> if the running time of a solution is upper bounded by a <u>polynomial</u> function of problem size. Examples of feasible problems are sorting lists of n ordered objects (complexity $O(n(\log_2 n))$) and determining planarity of graphs (complexity O(n)). This does not imply that all polynomial running times are practically feasible, since n^{1000} is an exceedingly long running time for even modestly large n. But we do mean that polynomial growth in problems that we encounter <u>on a daily basis</u> is sufficient for computational tractability. On the other hand, nonpolynomial running times are very restrictive. Algorithms with nonpolynomial running times prohibit (in general) all but the smallest instances, so it is usual to call problems with nonpolynomial lower bound on their running times <u>intractable</u>.

Among the intractable problems which are frequently encountered in mathematics, operations research and computer science are those that are solved by backtrack search procedures. Among these problems are the ones for which the backtrack search tree can always be bounded in depth by a polynomial function of the size of the input problem. The class of (suitably encoded) problems which are solvable in polynomial time is denoted by the letter P while problems that can be solved by polynomial depth backtrack search are denoted by NP. Clearly every problem in P is also in NP. At the present time it is not known if there are polynomially bounded algorithms for all problems in NP, but since NP contains some very difficult problems, it seems likely that many problems in NP have nonpolynomial lower bounds and are truly intractable. In NP are problems which are, in a formal sense, the most difficult problems in NP, the <u>NP-complete problems</u>. An NP-complete problem is remarkable in the following respect: if an NP-complete problem admits no polynomial solution, then obviously P ≠ NP, but if it is solvable in polynomial time then that polynomial time algorithm automatically yields a polynomial time algorithm for all problems in NP and so P = NP. At present, it is not known whether or not P = NP; however, it is widely believed that P ≠ NP. In that case, no NP-complete problem is in P and hence, no such problem has an efficient solution.

An example of an NP-complete problem is the <u>knapsack problem</u>. An instance of the knapsack problem is given by an integer S (the 'knapsack') and n integers $w_1,\ldots,w_n$ (the 'objects') The problem is to determine whether or not the knapsack can be filled by some subset of the given objects. In more convenient terms: given S and n-vector w, does there exist a binary vector x such that $w \cdot x = S$? The best known algorithms for the 0/1 knapsack problem have running times of order $2^{(n/2)}$ or worse, although there are efficient solutions for special cases of w.

There are also difficult problems in NP that are not NP complete. For example, factorization of large integers into prime factors is at the current time thought to be intractable: the best known bounds are on the order of

$$e^{\sqrt{\ln(n)\ln(\ln(n))}},$$

where n is the number being factored. Another important, apparently complex, problem is taking logarithms in finite fields. In a Galois Field GF(p) for a prime p, exponentiation can be carried out by 2log(p) multiplications, but computing logarithms requires an exponential number of multiplications.

This last example points to another property of NP problems that make them important to cryptographers -- as opposed to, say, a problem which has provably exponential complexity: it is easy to verify that a solution has been found. To verify that one has obtained a logarithm in GF(p), it is necessary only to exponentiate the reputed logarithm, and we have already remarked that this operation has an efficient algorithm. Although it may be hard to find the factors of a composite number, the factors can be verified to yield the composite by a single multiplication of n bit numbers which can be carried out in time proportional to n(logn)(loglog n). Finally, a proposed solution vector x to a knapsack problem can be verified by carrying out the dot product in time proportional to n.

Most of the current activity in cryptographic theory is based on the following principle: the cryptanalytic function h should be computationally intractable (a hard NP problem, an NP-complete problem, etc.). We will examine this principle more critically in Section 2.6.

Secure cryptosystems based on the 'feasibility' model of cryptanalysis must satisfy certain minimal requirements. Let $E_K(.) = E(K,.)$ and $D_{K'}(.) = D(K',.)$, where K' is the key required for decryption. Then the cryptosystem must satisfy 1-4 (5 is a technical requirement which will play a role in certain protocols).

1. For each message M, $D_{K'}(E_K(M)) = M$,
2. E_K is computationally tractable,
3. $D_{K'}$ is computationally tractable,
4. D is concealed from an enemy who knows E (i.e., K but not K') by a computationally intractable problem,
5. for each message M, $E_K(D_{K'}(M)) = M$.

For such cryptosystems the information theoretic model of security does apply -- the enemy may indeed be able to break the cipher if the feasibility of the process of computing h is factored out of the model. But if h is intractable, growing in complexity say at an exponential rate, then by providing h with problems of large enough size, the resource bounds on any algorithm for h can be made impossibly large, thus separating the enemy from the messages (or, equivalently from the knowlege of the deciphering key K'). This is guaranteed by condition 4. Condition 1 simply insures that D and E behave correctly as ciphers, while condition 5, which says that E and D commute is a technical property not required to insure secrecy. This condition is required, however, to give the system security from certain threats to authorization or integrity. Conditions 2 and 3 represent the commonsense requirement that E and D both be efficiently computable, so that the encryption/decryption devices can run at rates which are limited by the transmission capacity of the communications medium and not by requirement of the cryptosystem design.

2.5 Modern Block Ciphers. From the point of view of the cryptanalyst, block ciphers increase the rate of the language being enciphered. This removes, for example, any reasonable possibility of ciphertext only attack by statistical means, since the basic unit of transmission has no direct relationship to the underlying language. On the other hand, dealing with blocks of tens of bits -- sometimes hundreds of bits -- in a single operation is beyond all but the most sophisticated mechanical transmitting devices and certainly beyond human capabilities. It is only through electronically implemented cryptosystems that block ciphers become feasible.

A simple device which at first sight might seem to give a very good block cipher is a maximal length linear feedback shift register (LFSR) as shown in Figure 2.5.

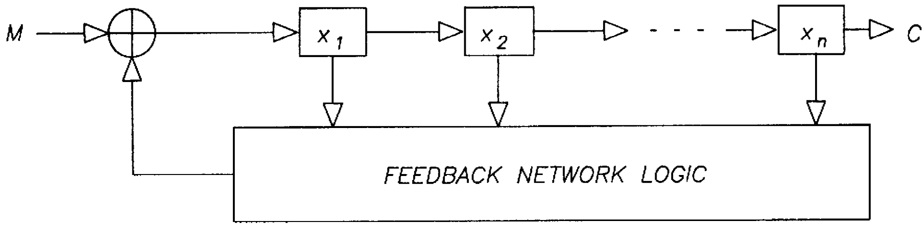

Figure 2.5
Logical Organization of LFSR

The LFSR implements a polynomial f(x) with coefficients from the field of integers modulo a prime p. Furthermore, $f(x) | (x^n-1)$, where $n=p^t-1$, if f is of degree t.

Since LFSR's produce output which meets several of the statistical tests for randomness, they are often suggested as block ciphers. To use a LFSR as an enciphering device, an n bit message is loaded into the shift register $x_1,...x_n$. The device is then stepped through some number of steps greater than n and the output C is read from the register. Decipherment is carried out by stepping the inverse logic through the same number of steps. Although the output is apparently random, the fact that LFSR's implement polynomials makes them quite susceptible to known plaintext attacks. An enemy who knows some plaintext-cipher text pairs can set up a system of 2n linear equations and solve the system to obtain the feedback logic and the polynomial coefficients which allow him to duplicate both the encryption and decryption devices.

Since LFSR's reveal their keys too easily, block ciphers must be sought which satisfy the basic computional requirements outlined above. Surprisingly, the place to begin looking for such an algorithm is the simple monographic substitution cipher which was discarded as insecure for stream ciphers. Random substitution alphabets for blocks of n bits can be realized by a device such as the one shown in Figure 2.6.

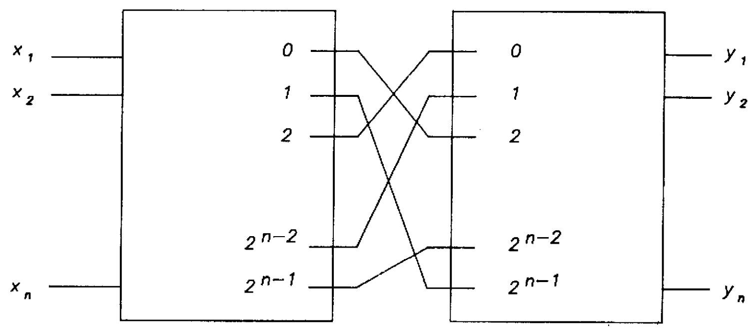

Figure 2.6
Substitution Box

The substitution algorithm consists of two devices: one to transform an n-bit block into a corresponding base n value by signaling on one of the 2^n output lines, and the other to carry out the reverse transformation. The base n output lines of the first device can be connected to the base n input lines of the second device in any of n! ways. There is a simple frequency analysis of such a scheme for values of n between 4 and 5 (the number of bits required to represent the standard alphabet) and the information theoretic cryptanalysis of such a scheme paints a very dismal picture. But size is an important component here. For large values of n (say, n = 128) the frequency attack is no longer possible. However, such a device is impossible to build since it requires 2^{128} internal switches. By cascading a number of more economical devices, it is possible to create a product polygraphic block substitution cipher that has many of the advantages of the n = 128 device.

Such a scheme is shown in Figure 2.7.

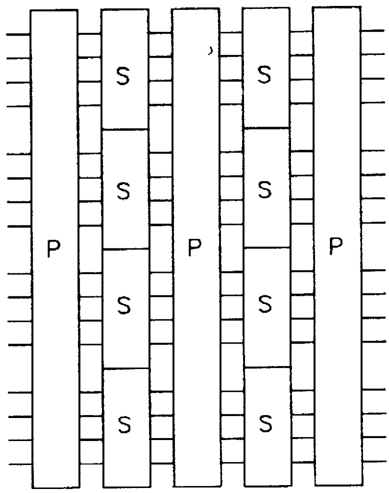

Figure 2.7.
A Product Cipher S/P Network

This is essentially the design of the IBM Lucifer system. The choice of n for each substitution box is quite small (n=4), although the block size for the entire device is large. A key for such a system is a specification for each permutation box, of which the 64! or 128! permutations is to be selected and for each substitution box which of the 16 substitutions is to be performed. It is possible to construct a S/P network that behaves quite badly by choosing keys which permute the plaintext bits in the cipher text. This permutation can be discovered by chosen plaintext analysis of blocks that contain a single 1. Tracing the path of the bit through the network is then simply a process of observing the output wires for the single on bit. Lucifer permanently keys each P and each S box with "good" keys; for example, each S box is specified by selecting a substitution from two permanently keyed substitutions -- call them S_0 and S_1. So to describe a key, one sets a binary string $b_1b_2...b_k$, where $b_i = 0$ if the ith S box is S_0 and $b_i = 1$ otherwise.

The Lucifer system is a high-quality block cipher. It is, however, not now considered to be a secure system -- Lucifer's operating environment assumes that keys (which are contained on passcards issued to bank customers) cannot be read and altered by enemies. Lucifer also implements a strong form of information theoretically secure (for authorization) encipherment, but cannot be widely used for other cryptographic tasks (cf. Chapter 4).

2.5.1 The Data Encryption Standard. In 1977 the U.S. National Bureau of Standards (NBS) issued a Data Encryption Standard (DES). The DES algorithm is a block cipher developed by IBM based on an S/P network as in the Lucifer system described above. The intention in issuing DES was to provide a cryptographic standard to be used for secure data transmission for all but national security-related data. Figure 2.8 is a schematic diagram of the DES algorithm as it was issued by NBS.

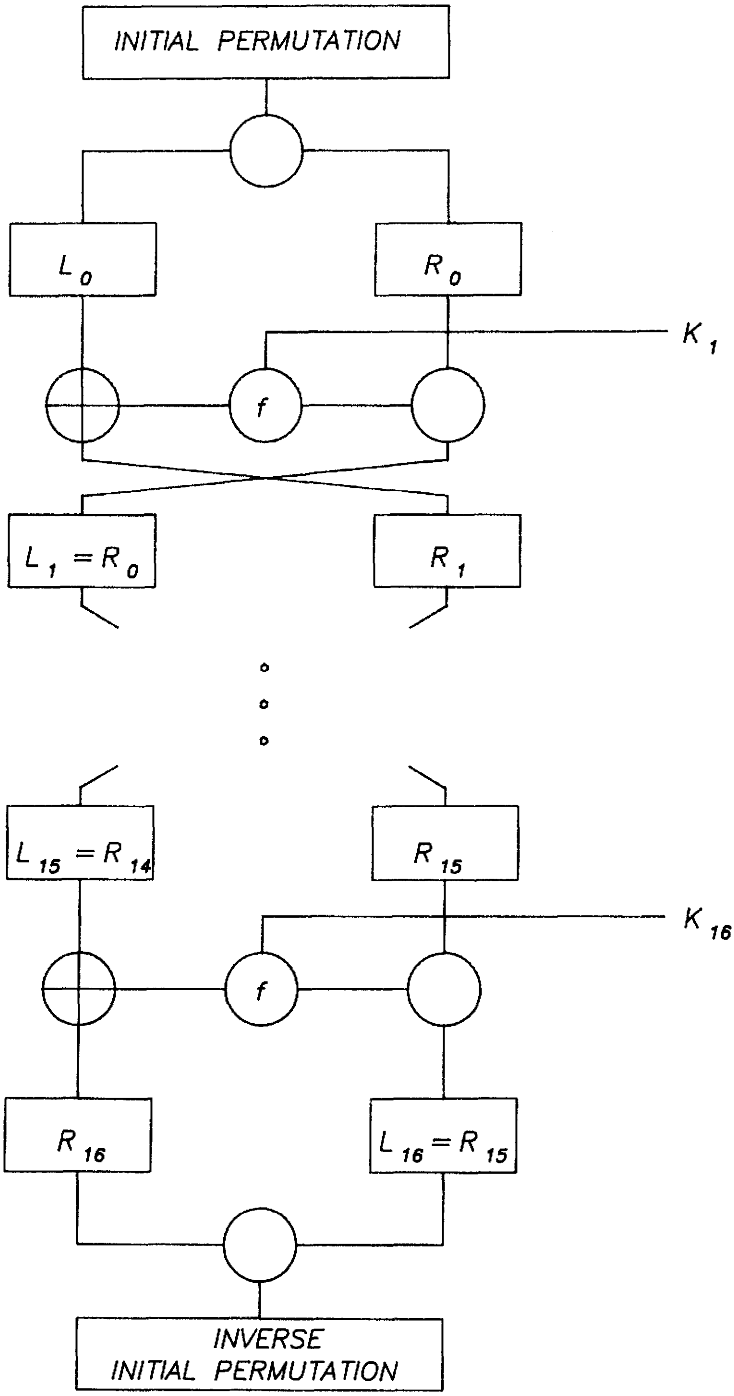

Figure 2.8
The DES Algorithm

Briefly, DES enciphers 64 bit blocks of plaintext using a 64 bit key (56 bits of key and 8 parity bits). Encryption is by 16 separate rounds of encipherment, each round a product cipher under the control of a 48 bit key. That is, each round uses a distinct 48 bit key $K_1,\ldots K_{16}$ obtained by a scheduling algorithm from the external key K. To decipher, the keys are scheduled in reverse order using the same external key K. The cryptographic function E(K,M) = C can be defined in terms of an equation

$$IP^{-1}(F)IP(M) = C,$$

where IP is the initial input permutation. F represents the portion of the cipher that depends on the key K and is described by the function f and a key scheduling algorithm KS. f is a product cipher (a S/P network). It is sufficient to describe only the first round since the remaining rounds behave identically. The 64 bits of the initial permutation are split into two 32 bit blocks, bits 1 through 32 being designated the L_0 block and bits 33 through 64 the R_0 block, so at the start of this round IP(M) = L_0R_0. KS produces the first 48 bit key K_1. The output of the round is a block L_1R_1 which is also the input to the next round. These 32 bit blocks are described by the equations:

$$L_1 = R_0$$

$$R_1 = L_0 \oplus f(R_0,K).$$

After the 16th round, the blocks are permuted to obtain C. The key scheduling algorithm is a complex sequential shifting process; since it has little cryptographic significance, we will not describe it. The same algorithm can be used for deciphering as long as the output permutation is an inverse of the input permutation, the same key is used and if L',R' are the output of a round:

$$R = L'$$

$$L = R' \oplus f(L', K_i).$$

The symmetry in the encryption/decryption process can be seen by writing the rounds in reverse order. Let $K_1, \ldots, K_{16}$ be the 48 bit keys produced by the key scheduling algorithm. Then

$$L_n = R_{n-1}$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n),$$

and

$$R_{n-1} = L_n$$

$$L_{n-1} = R_n \oplus f(L_n, K_n).$$

Clearly, the scheduled keys are used in reverse order and in the forward direction the permuted input block is L_0R_0, while the preoutput before final permutation is $R_{16}L_{16}$. During decipherment $R_{16}L_{16}$ is the permuted input and L_0R_0 is the preoutput.

The product cipher f is described in Figure 2.9. The function accepts a 32 bit input block and outputs 8 blocks of 6 bits each, where each 6 bit block has its bits selected from the input block according to a fixed selection table. After combination with the scheduled key, the blocks are passed to 8 substitution boxes which compress the blocks, substitute and produce 4 bit output blocks. This 32 bit output is then subjected to a permutation and the result is returned as the result of the f operation.

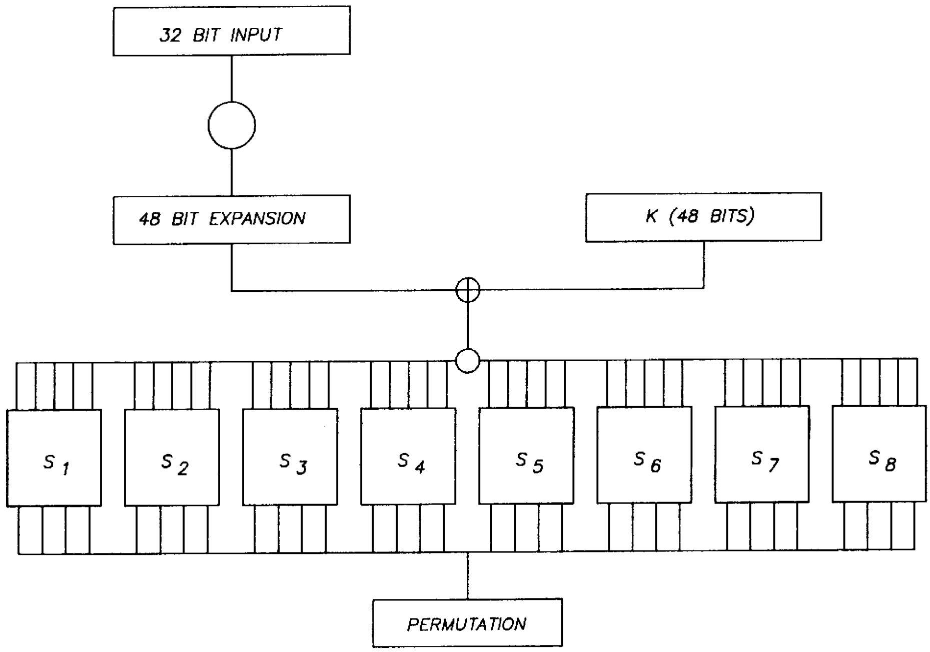

Figure 2.9
The Calculation of f

Many standard tests of cryptographic strength have been applied to the DES algorithm with positive results. DES exhibits, for example, strong intersymbol dependence: a small change in the input message causes massive changes in the output cipher.

DES can be used as a stream cipher simply by observing that it is a good generator of random numbers. The key K is used as a seed, and subsequent outputs are recycled through the system to produce a long pseudo random key which can be used as a Vernam cipher key. This process is sometimes called <u>feedback chain encipherment</u>.

2.5.2 Cryptanalysis. In the information-theoretic model, DES does not exhibit strong security. By computing equivocation for DES under various attacks, it can be shown that one or two blocks determine a key even if only ciphertext is available to the enemy. The key size of the standard has been criticized. The crux of the problem with DES is that the 56 bit key size seems to admit a known plaintext analysis by an enemy with massive -- but feasible -- computational resources. The analysis is simply a brute force search. Suppose that known message M is enciphered using DES with key K and that $E(K,M) = C$. To discover K, the cryptanalyst deciphers C with each of the 2^{56} possible keys. When he discovers M, he stops and announces the key. Since 2^{56} is approximately 10^{17}, this exhaustive search sounds prohibitive. But it has been argued that special purpose hardware can be constructed to enable a million parallel searches to take place in about twelve hours. With available technology such a machine can be constructed for about $20 million. With some reasonable assumptions about the operating environment this cost can be amortized to about $5,000 per solution -- well within the range of a determined enemy! NBS has argued vigorously against the feasibility of this analysis.

No one has yet questioned the practical security of DES-style algorithms. It seems, for example, that simply by increasing the key size from 56 to 128 an exhaustive cryptanalysis is ruled out.

2.5.3 Public Key Systems. The cryptosystems given above all have the property that the encryption key K and the decryption key K' are the same. They are conventional ciphers. At first blush, it is hard to imagine how requiring $K \neq K'$ can help the cryptosystem since differing encryption and decryption keys would require an extra exchange of keys and therefore an extra possibility of compromise. Consider the key exchanges in an n-person conventional cryptosystem. To provide for pairwise secure communications $O(n^2)$ keys must be exchanged. Our goal is to exploit the fact that $K \neq K'$ to reduce the "exhanges" to $O(n)$ keys.

In any case, the entire question may be irrelevant since it is not at all clear that there exists any ciphers that satisfy conditions 1 - 5: the troublesome condition seems to be the condition that the deciphering algorithm be hard to infer from the enciphering algorithm. More exactly, if K' is fixed and a function f(K') is used to derive an enciphering key K, then knowing E_K should not reveal $f^{-1}(K)$. Such a function is <u>one-way</u> in that it is easy to compute but hard to invert.

Given such functions, it is easy to see how a scheme based on 1 - 5 might operate. All participants in the cryptosystem agree to derive (according to special rules that depend on the cipher) a pair of keys satisfying 1 - 5 -- call them K and K'. The enciphering key K is made public; for example, K can be published in a directory which is distributed to all users. The deciphering key K' is kept secret, however. Suppose that A wants to communicate with B. A consults the public directory to find B's encipherment key K, P_B. A then computes E(K,M) = C and sends C to B. When B receives C, he uses his secret key $K' = S_B$ to recover M = D(K',C). Since K' is concealed by a computationally intractable problem from anyone who does not know K', the message is secure from all attacks. Such a cryptosystem is called a <u>public key</u> cryptosystem.

2.5.4 The RSA Algorithm. At this time it is not known whether or not provably secure public key encryption algorithms exist (cf. Section 2.6). There have been several proposals for algorithms whose security has the essentially the same status as the P=NP problem. The first such algorithm to be announced was the RSA algorithm.

The RSA algorithm depends on some simple number theory. We use the familiar notation $\phi(n)$ to denote the Euler function. Clearly, if p is a prime number, then $\phi(p) = p-1$. Also, if p and q are primes, then $\phi(pq) = (p-1)(q-1)$.

The private key is a triple (p,q,d), where p and q are large primes and the public key is a pair (n,e), where $n=p \cdot q$. Notice that the public key does not mention the factorization of n. Since p and q are prime $\phi(n) = (p-1)(q-1)$ can be computed quickly. The remaining portions of the keys are derived as follows. Let e be chosen so that $(e,\phi(n)) = 1$; e.g., e may be any prime between max(p,q) and n-1. To obtain d, the Euclidean Algorithm can be used to find integers d, b such that

$$de + b\phi(n) = 1.$$

Thus, a d for which

$$de = 1 \bmod \phi(n)$$

is determined.

Encipherment is carried out on a message M by

$$M^e \bmod n = c$$

and decipherment is carried out by

$$C^d \bmod n = M'.$$

We first show that encipherment and decipherment behave properly; i.e. $M = M'$.

Two preliminary facts are necessary. The first is Fermat's Little Theorem: if p is prime and x is an integer which does not contain p as a factor,

then

$$x^{p-1} \equiv 1 \bmod p.$$

The second fact is a technical lemma. Suppose $n = pq$ and that p and q are both relatively prime to x. Then

$$x^{\phi(n)} \equiv 1 \bmod p\, q.$$

To see why, consider the following instances of Fermat's Little Theorem:

$$x^{p-1} \equiv 1 \bmod p \text{ and } x^{q-1} \equiv 1 \bmod q.$$

Since

$$x^{\phi(n)} = x^{(p-1)(q-1)} = (x^{p-1})^{q-1} = (x^{q-1})^{p-1},$$

we have

$$x^{\phi(n)} = 1 \bmod p \text{ and } x^{\phi(n)} = 1 \bmod q.$$

Since $p | x^{\phi(n)}-1$ and $q | x^{\phi(n)} - 1$,

$$x^{\phi(n)}-1 \equiv 0 \bmod pq,$$

giving the result.

We now claim that

$$M^{ed} \equiv M \bmod n.$$

There are two cases according to whether or not M and n are relatively prime. If (M,n) = 1, then

$$M^{ed} = M^{1+c\cdot\phi(n)} \equiv M \bmod n.$$

Therefore, suppose that M = py, where (y,q) = 1. Then

$$M^{ed} - M = M^{1+c\cdot\phi(n)} - M = M(M^{c\cdot\phi(n)}-1).$$

By the technical lemma,

$$M^{ed} - M \equiv 0 \bmod p\ q.$$

How is condition 4 satisfied? If n can be factored then d can be determined from e as described above. It does <u>not</u> follow that determining d from e requires the factorization of n, although there does not appear to be any way around it.

Choosing large enough primes to satisfy the security requirements is fairly simple. There are already hardware devices available which will generate prime numbers of a hundred bits or more in several seconds.

There is very little cryptanalytical experience with the RSA algorithm. Of course, it is not known whether or not factorization is an intractable problem, but there are certainly no fast algorithms in existence today, and it seems safe to assume that this situation will not change soon.

The RSA algorithm may have unfortunate behavior in some cases. It is known, for example, that for every public key (n,e) at least nine messages will be encrypted as themselves. In addition, there are very bad choices of e which leave half of the messages unaffected by encryption. This can be avoided by choosing <u>safe</u> primes p and q: for example p = 2p'+1 is a safe prime. In addition, there may be iterative attacks. A cryptanalyst may successively encrypt M^e until he obtains plaintext. To avoid this attack, p must be chosen so that p-1 has a large prime factor p' and p'-1 has a large prime factor p".

2.5.5 Related Algorithms. Another public key system can be based on the computation of logarithms in finite fields. A user selects a secret integer less than the number p-1 for the prime p (call it X) and a public key

$$Y = a^X \pmod p,$$

where $a \varepsilon GF(p)$. To transmit from user A with public key X to user B with public key x, A computes

$$a^{Xx} \pmod p = y^X \pmod p$$

$$= Y^x \pmod p,$$

where Y and y are the secret keys of A and B respectively. To compromise this system, the enemy is required to compute logarithms base a in GF(p):

$$Y^{\log_a y} \pmod p.$$

A more subtle example is based on the knapsack problem. This algorithm is based on the following special case of the knapsack problem (find boolean vector x such that S = a · x). If, for all values of i,

$$a_i > a_1 + a_2 + \ldots + a_{i-1},$$

then $x_i = 1$ iff either of the following two conditions are met:

1. $i = n$ and $S > a_n$, or

2. $i < n$ and

$$S - \sum_{j=1}^{i-1} x_j a_j \geq a_i$$

The public key in this case is a vector a. To send x to a user with public key a, a user sends $S = x \cdot a$. Thus recovering x is equivalent to solving a knapsack problem. The owner of a, however, has chosen a to correspond to another vector a' which he keeps secret and which satisfies (1) and (2) above so that $S' = a' \cdot x$ can be easily solved. He chooses values of m and w so that the multiplicative inverse of w (mod m) exists and derives a from a' by the rule:

$$a_i = wa_i' \pmod m.$$

Therefore, to decrypt S, the designer need only compute

$$S' = vS \bmod m,$$

where v is the inverse of w. This, in turn, is equal to

$$x_1a_1' + x_2a_2' + \ldots \pmod m,$$

and if m is larger than the sum of the components of a', then

$$S' = x \cdot a'$$

which is an easy case of the knapsack problem.

Shamir has recently provided a successful cryptanalysis of the knapsack-based public key cryptosystem. Specifically, Shamir has shown that whenever the public key $(a_1,...,a_n)$ has the property that each a_i is a modular multiple of a sequence $(b_1,...,b_n)$ such that

$$b_i > b_1+b_2+...+b_{i-1},$$

Then almost all equations

$$\sum_{i=1}^{n} x_i a_i = c$$

can be found in polynomial time, giving an efficient method of recovering plaintext $(x_1,...,x_n)$ from ciphertext c. A fast integer programming algorithm is used to find a real number d ε (0,1) such that a necessary condition on the values m and w used in the knapsack algorithm is that $m/w \in [\alpha,\alpha+\varepsilon]$ for small ε. Next n^2 subintervals (β_i,γ_i) of $[\alpha,\alpha,+\varepsilon]$ are constructed so that if $m/w \in (\beta_i,\gamma_i)$, then m, w can be used in the knapsack algorithm. Since at least one pair of integers m, w is known to exist, at least one (β_i,γ_i) must be nonempty. An efficient diophantine approximation algorithm locates the smallest m and w such that m/w is in some (β_i,γ_i). Notice that Shamir's algorithm does not necessarily construct the <u>same</u> m, w used in the original knapsack. This argument has recently be extended to cover various generalizations of the knapsack cipher, which must now be regarded as insecure.

There have been efforts to refine the RSA algorithm. The basic rationale of the RSA approach is to force the cryptanalyst to solve an equation of the form

$$x^m = y \pmod{n}.$$

If it is hard to solve such a congruence, then it should be harder still to solve a system of the form

$$f_i(x) = y_i \pmod{n},$$

where the function f_i are 'difficult' functions in the ring of algebraic numbers $Q(\sqrt{s})$ for s a square-free rational integer. Let n=pq for large primes p,q and

$$(s/p) = (s/q) = -1,$$

where (α/β) denotes the Legendre symbol, so that s is a quadratic non-residue mod p and q, and $\phi(n) = (p^2-1)(q^2-1)$ so that $m_1m_2=1 (\bmod\ \phi(n))$ is as hard as factorization. Identify the point (a,b) for rational a,b with $a+b\sqrt{s}$. A public key is a triple (s,n,e). The corresponding private key is (p,q,d), where,

$$ed \equiv 1 \bmod (p^2-1)(q^2-1).$$

Letting * represent pointwise defined multiplication in $Q(\sqrt{s})$ (i.e., $(x,y)*(u,v) = (a,b)$ when $a = xu+syv \pmod{n}$ and $b = xv+yu \pmod{n}$) encryption of a message M is defined by

$$M^e \bmod n$$

and decryption of ciphertext c is defined by

$$C^d \bmod n,$$

where exponentiation is the iteration of *.

Factorization in Z (as in RSA) or in $Q(\sqrt{s})$ is sufficient to allow successful cryptanalysis of the public key systems, but it may not be necessary. At present, no functions that are suitable for communication are known which are provably equivalent to factorization.

There is, however, a public key function that is suitable for certain protocols that has the property that cryptanalyzing it implies factoring. The public key K is a pair (n,b), where n is the product of two large primes p and q and $0 \leq b < n$.

The encryption of a message $M \leq n/2^k$, for some fixed k is

$$C = E_{(n,b)}(M) = M(M+b) \bmod n.$$

The receiver of the cipher text C knows p, q and also that

$$C = x(x+b) \bmod n.$$

Hence, Berlekamp's Algorithm can be used to solve

$$x(x+b) = C \bmod p$$

and

$$x(x+b) = C \bmod q,$$

and, by Chinese remaindering, obtain x_1, x_2, x_3, x_4 so that

$$x_i^2 + bx_i = E_{(n,b)} \bmod n.$$

There are two possibilities. If only one of the solutions x_i (in binary) ends in k zeroes, then $x=x_i/2^k$ is the proper decryption of C. If, on the other hand, more than one of the solutions ends in k zeroes, then C decrypts ambiguously. This 4-1 nature of decryption may make this function unsuitable for normal communications. However if k is large enough, the second case rarely arises.

The cryptanalytic function is h(y), where:

$$y = E_{(n,b)}(x),\ x \leq n/2^k \text{ implies } h(y) = x.$$

Notice that the behavior of h is unspecified when y is not in the proper form. Computing such an h is equivalent to factorization by the following argument. Suppose that the system can be cryptanalyzed. Pick a random x and let $y = x^2 \bmod n$. Suppose that the cryptanalyst has obtained in a z such that $y = z^2 \bmod n$, so that $x^2 = z^2 \bmod n$. Then a factorization of n is obtained as follows. If $x \neq \pm z \bmod n$, then by computing $(x-z,n)$ or $(x+z,n)$, we get a factor of n. If, on the other hand, $x = \pm z$ modulo n, then since x was chosen randomly, we have equality with probability at most 1/2. If we repeatedly select random values of x, we will quickly get $x \neq \pm z$ and thus factor n.

Public key cryptosystems are also susceptible to various styles of attack that do not address the strength of the complex problem used to define the public and private keys. These are attacks on the <u>protocols</u> used to implement the cryptosystem and are discussed in Chapter 4.

2.6 Intractability and Cryptanalysis. Such problems as factoring integers, the travelling salesman problem, graph coloring, and integer programming are all in the class NP. One way of looking at a backtrack procedure for solving these problems is to look at the branching backtrack tree describing the backtrack search. At each step of the backtrack procedure, the search algorithm must 'guess' which elementary item is to be considered next in the search. The algorithm proceeds with the search with this item under consideration until one of two things happens. Either the search is successful (in which case it terminates) or the search is unsuccessful. In the latter case, the procedure 'backtracks'; that is, it proceeds to the most recent point at which it is possible to undo the effects of a wrong guess and makes a new guess. If it is not possible to make a new guess, then the algorithm terminates unsuccessfully.

Thus, a backtrack search algorithm, B, running on an input I, of size n, defines a set of possible computations C(B,I). If the search is supposed to determine whether or not $I \in S$, then each computation in C(B,I) should terminate and announce either $I \in S$ or $I \notin S$. The algorithm is said to <u>accept</u> I in time T(n) if there is a sequence of guesses in C(B,I) which leads to the result "$I \in S$" in at most T(n) elementary operations. B accepts S in time T if for every $I \in S$, B accepts I in time T(n). Formally, the set **NP** is the set of problems S which are accepted by algorithms in time $T(n) = O(n^k)$ for $k \geq 0$.

The theoretical superstructure of computational intractability is essentially the conjecture that **P** does not equal **NP**. There are, in addition, intermediate notions that often prove to be useful in the theory. One such notion is that of a co-NP problem. Suppose that S is simply a subset of a fixed reference set U (for example, if S is the set of composites, then U may be the set of all integers in some suitable notation). If $S \in$ **NP**, then it may be useful to consider the set $U-S = \bar{S}$. $\bar{S}$ is said to be in the class co-NP. There is a characterization of these classes that may be helpful in visualizing the relationship between them. One may think of S as defined by a predicate:

$$I \varepsilon S \text{ iff } (\exists y)(|y| \leq n^k \wedge B(I,y)),$$

where y is intended to denote the sequence of backtrack guesses and B(I,y) is true exactly when the polynomial time algorithm B on input I will evaluate "I ε S" when the guesses y are made. Then clearly,

$$I \varepsilon \bar{S} \text{ iff } (\forall y)(|y| \leq n^k \Rightarrow \neg B(n,y)).$$

How essential is this change in logical quantification? If S is in P, then clearly $\bar{S}$ is in P, since the backtrack procedure need not guess at all. On the other hand, there are problems in NP and co-NP which appear to be difficult indeed (although not necessarily NP-complete). Factorization is one such problem. Although there are NP-style algorithms to determine both the set of primes and the set of composites, the problem of factoring in polynomial time remains unsolved.

Thus, theoreticians like to add a corollary conjecture to the P = NP conjecture; that is, NP = co-NP. If P = NP, then NP = co-NP.

Let us now return to our axioms for D and E given in Section 2.4. We would like to know exactly how complex the problem is which separates D and E in a public key system. If the problem of determining D from E is, for example, NP-complete, then we have proved (modulo the P vs. NP conjecture) that the the cryptanalysis must solve an intractable problem.

But deriving D from E cannot be equivalent to an NP-complete problem because of the following theorem: the computational task to deriving D from E is in NP and co-NP. Thus if the cryptanalytical problem is NP-complete, it follows that NP=co-NP, which is considered to be just as unlikely as P=NP. To see why the theorem is true, let h be the cryptanalytic function. Since h^{-1} is simply the process of computing enciphering keys from deciphering keys, practice dicatates the following conditions.

1. $|h^{-1}(i)| = |i|$,
2. $h^{-1}: N \dashrightarrow N$ is a bijection
3. h^{-1} is in P.

Define the set

$$S = \{(n,m) | h(n) > m\}.$$

First, S must be in NP: a backtrack algorithm B can guess an $|n|$ bit integer i, verify that $h^{-1}(i)=n$ and accept only if $i>m$. Second S must be in co-NP since there is a similar algorithm B* that accepts only if $i \geq m$. Now h(n) has $2^{|n|}$ possible values. But in order to avoid exhaustively searching all values, we need only use B and B* to perform a binary search. That is, search for x in an interval of s numbers, by simply successively bisecting into subintervals at the midpoints t_0, t_1, ..., $t_{\log s}$, where the ith subinterval S_i is determined by the following rule: if $x<t_{i-1}$, then

$$S_i = \{j \in S_{i-1} \mid j < t_{i-1}\},$$

otherwise,

$$S_i = \{j \in S_{i-1} \mid j > t_{i-1}\}.$$

This uses at most |n| operations. Thus h is in both NP and co-NP.

While it is desirable to have E and D be separated by a complex problem, there are as yet no examples of such a function. Failing this, we have been reduced to a kind of argument by analogy. We have argued that breaking a particular system is "similar" to solving other hard problems (including some that are NP-complete). Such arguments while useful in practice fall far short of <u>proving</u> the intractability of the cryptanalytic problem. As in the case of the classical ciphering problems, it is tempting to try to read too much into the <u>apparent</u> difficulty of decipherment.

In the modern case, the safety of the cipher depends on a conjecture: perhaps it is false!

Moreover, even if P $\neq$ NP, none of the following are ruled out:

1. There is an algorithm for each NP problem that runs in $n^{t(n)}$ steps, where $t(n) \to \infty$ very slowly; Such an algorithm is not polynomially bounded, but might well be able to derive D from E;
2. there is a random algorithm that works in random polynomial time;
3. there is an algorithm that solves all instances of the travelling salesman problem of length $n \leq 10^{10}$ in n^3 steps; since 10^{10} is a generous bound on key size, such an algorithm could feasibly compute h;

4. there is an algorithm for each NP problem that solves 1% of the problem instances in polynomial time. For most applications this compromises the security of the deciphering key.

Thus, the question of whether or not P = NP seems to be a rather narrow one on which to rest cryptographic security, and it is probably best not to gain too much confidence from it alone since none of the above are ruled out by a proof that P ≠ NP.

We can see an example of this in the 4-1 encryption function which we showed to be as hard as factorization in Section 2.5. Let $E_1 = E_{(n_1,0)}$ and let $E_2 = E_{(n_2,0)}$, where n_1 and n_2 have distinct prime factors and $|n_1| = |n_2|$. Since frequent key change is commonly thought to be a good additional security measure, it is perhaps reasonable to ask whether or not putting $E_1(x)$ <u>and</u> $E_2(x)$ in the hands of the cryptanalyst will ever compromise the message. Notice carefully that the proof of equivalence to factorization says nothing at all about multiple systems, and so no contradiction arises. To see how the message can be compromised let

$$x^2 = a + yN$$

$$x^2 = b + zM$$

where (N,M)=1 and a,b are residues with $0 \leq M < N$. But, given a,b,N,M, x can be quickly determined. Notice

$$a - b = zM - yN.$$

Thus, by the Euclidean Algorithm, we can obtain

$$a - b = z_0M - y_0N,$$

for known z_0, y_0 with $|z_0| \leq N$ and $|y_0| \leq M$. Then

$$z = z_0 + tN$$

$$y = y_0 + tM$$

are the general solutions to the equations . It follows that

$$x^2 - a - y_0N = tNM,$$

so,

$$|x^2 - a - y_0N| \leq 3N^2.$$

Therefore t is at most 3N/M, but N and M are the same order of magnitude so only a constant number of t's need be tried to find x.

There is another threat to this function: a message may be <u>partially</u> decrypted without factoring the key. Let $E(x) = E_{(n,b)}(x)$. We will let the message be k bits in size -- say k=300 -- so that at 6 bits per character, 50 characters are encoded in a message. In the example of the cryptanalysis of Enigma we noted that the cryptanalysts were aided greatly by the stylized form of the plaintext. In modern electronic applications, the situation frequently occurs that series of highly similar messages are sent. For example EFT orders frequently differ only in the amount to be transferred, and many other messages are simply updated versions of previously sent messages. Thus, we may find applications for which cryptanalysis need only analyze messages that are close to each other in that they differ only in a fixed number of characters. We will consider the simplest case of this situation: x and y differ in at most one character. Notice that this case is equivalent to $x = y+d$, for some $d \in \Delta$, where $|\Delta| = 64 \cdot 50$. This is simply due to the fact that each character position can be changed by adding a suitable d and there are at most 64 · 50 such changes.

We now show that if x and y differ in at most one character, x and y can be recovered from E(x) and E(y) in polynomial time. Since $x = y+d$,

$$E(y) - E(x) = -x^2bx + [(x+d)^2+b(x+d)] \pmod n.$$

Thus,

$$E(y) - E(x) = 2dx + d^2 + bd \pmod n.$$

This is a linear equation and so can be solved for x. Notice that this result also does not lead to contradiction to the equivalence to factorization since if x and y are random messages they will only rarely differ in one character.

Chapter 4 describes additional attacks on cryptosystems which do not actually result in key compromise.

For conventional ciphers (those for which K=K'), computational intractability is no theoretical impediment at all. There are, for example, ciphers which are provably intractable in the sense that they are NP-complete, but which are easy to compromise. That is, there is a cipher with the following properties:

1. computing h under chosen plaintext attack is NP-complete,

2. given enough (M,C) pairs h can be efficiently com-

puted with probability approaching unity.

The cipher uses a key $k=(x_1,...x_n)$ of

$$m = \lceil \log_2(1+a_1+...+a_n]) \rceil,$$

and $A=(a_1,...,a_n)$ is known to the cryptanalyst.

To generate C from M:

1. generate a random vector R,
2. compute $S=A(K \oplus R)^T$
3. set $C=(M \oplus S_2, R)$, where S_2 is S encoded in binary.

Notice that the ciphertext is of length m+n. Knowledge of K insures easy decryption.

The cryptanalyst has available to him many pairs (M_i,C_i), where

$$C_i = (M_i \oplus S_i, R_i).$$

It is very likely that n of these will be such that

$$U_i = 1^n - 2R_i,$$

(where 1^n is a vector of n 1's) are linearly independent over the reals. This follows since the R_i' are randomly chosen. Now,

$$K \oplus R_i = K + R_i - 2K*R_i,$$

where * denotes component-wise multiplication. Thus,

$$K \oplus R_i = R_i^T K*U_i,$$

and, therefore,

$$S_i = A(K \oplus R_i)^T$$

$$= A(R_i + K*U_i)^T$$

$$= AR_i^T + A(K*U_i)^T$$

Now, letting $t_i = S_i - AR_i^T$, we obtain the following equations in matrix form:

$$\begin{bmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ \cdot \\ t_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{bmatrix} \begin{bmatrix} a_1 & & & & \\ & a_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & a_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

Since the U_i's are independent and all $a_i > 0$, it follows that we can easily solve for the key k.

2.7 Bibliographic Notes. The classic survey of conventional cryptography -- particularly its applications is [45]. For a more technical treatment of mathematical cryptography and cryptanalysis, a good treatment is [47], while an elementary survey of substitution ciphers is presented in [83]. The history of American involvement in diplomatic and military codes is available in [85], which has recently been reprinted. The account of how the Enigma code was broken is taken from [23].

The information theoretic approach to cryptography is due to Shannon [80]. Important recent contributions have also been made by Hellman [38]. The complexity theoretic approach to cryptanalysis was proposed in [27] and [56]; for a less technical survey, see also [39] and [40]. Computational complexity theory is presented in many modern computer science texts: [1] is representative. The concept of NP-complete problems originated with Karp in [46]. Public Key encryption was proposed by Diffie and Hellman in [27] and by Merkle in [56]. The RSA algorithm is due to Rivest, Shamir and Adelman [70]. Since then, there has been extensive literature on the mathematics of public key systems; see for example [51,58,74,78,79]. The algebraic number theory algorithm is due to Lipton [53], while the quadratic algorithm was proposed by Rabin as a signature scheme [67]. Critical analyses of public key systems have been presented in [41,65]. The results about weaknesses in certain public key algorithms are due to Blakely and Blakely [5]. For a survey article which relates conventional ciphers and public key systems, see [81].

The outlines of S/P network encryption is sketched in [32], where a description of the Lucifer system also appears. The DES algorithm can be obtained from the US Printing Office [61]. The cryptanalysis of DES was carried out by Hellman and is described in [28]. An extensive controversy has raged over this and other approaches to secure cryptosystems; accounts of these appear regularly in Science and other technical publications. The public-key encryption algorithms have also been popularized [34].

The relationship between NP and public key systems was developed in [7] and [31]. The example of an insecure system based on an NP-complete problem appears in [31]. Shamir's analysis of the knapsack algorithm was announced in [77].