Privacy and Authentication: An Introduction to Cryptography

WHITFIELD DIFFIE MEMBER, IEEE, AND MARTIN E. HELLMAN, SENIOR MEMBER, IEEE

Invited Paper

Abstract—**This paper presents a tutorial introduction to contemporary cryptography. The basic information theoretic and computational properties of classical and modern cryptographic systems are presented, followed by cryptanalytic examination of several important systems and an examination of the application of cryptography to the security of timesharing systems and computer networks. The paper concludes with a guide to the cryptographic literature.**

I. INTRODUCTION

UNTIL RECENTLY, cryptography has been of interest primarily to the military and diplomatic communities. Private individuals and even commercial organizations have rarely considered it necessary to resort to encryption for the protection of their communications, and those that have, have seldom done so with particular care. Today, however, several factors have combined to stimulate great interest in commercial applications.

Electronic communications are replacing paper media in a rapidly increasing variety of applications. The effect of this is both to increase the amount and variety of information available to an eavesdropper, and to make the act of eavesdropping easier. Fortunately the same factors which promote the spread of electronic communications are producing a marked decrease in the cost of cryptography.

Where once the nation's business was conducted either by personal contact or written correspondence, it is now handled largely over the telephone, creating the environment for a substantial wiretapping industry. The replacement of wires by microwaves has even allowed the wireman to ply his trade without having to tap any actual wires. Finally, the introduction of **direct long distance dialing** has made it possible to identify calls of interest even when the tap is placed a long way from the person being spied upon, because each call is preceded by a digital sequence identifying the number being called.

It has come to light over the past two years that the Russians are monitoring telephones from their embassy in Washington and their consulates in other parts of the country. Microwave antennas intercept telephone traffic destined for such vital places as the Capitol building, and computer programs automatically select the conversations of interest [1], [2].

Manuscript received May 22, 1978; revised November 28, 1978. This work was partially supported by the National Science Foundation under NSF Grant ENG 10173.

W. Diffie was with the Department of Electrical Engineering, Stanford University and the Stanford Artificial Intelligence Laboratory, Stanford, CA. He is now with BNR, Inc., Palo Alto, CA 94304.

M. E. Hellman is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

Data communications are even more vulnerable than voice telephony because, with voice the eavesdropper lacks the ability to discern the content of a spoken message, unless a costly human monitor is employed, but if the intercepted material is in computer readable form (e.g., Telex), no such limitation is imposed. It has been reported [3, p. 765] that for some years the National Security Agency (NSA), in its operation Shamrock, scanned all telegraph and Telex messages passing in and out of the United States for keywords.

The cost of eavesdropping should continue to decline with time. There is at present extensive research into speech recognition [4], [5], and many communications will soon be carried on digitized packet-switched networks [6]. Electronic mail and Electronic Funds Transfer (EFT) are being developed to replace their present day counterparts, and telephone verification of credit data is already a reality.

As a result, private and commercial interest in cryptography has been rising, new papers have begun to appear, and the National Bureau of Standards (NBS) has adopted a cryptographic system [7] to be used as the federal Data Encryption Standard (DES) on sensitive but unclassified data.

This paper is intended as an introduction to the fascinating but forbidding subject of cryptography. It provides both a grounding in the fundamentals and a feel for the subject to anyone interested either in doing cryptographic research or employing cryptographic security. Cryptography is currently an engineering subject in which there are more facts and rules of thumb than theorems or systematic developments. The text naturally reflects this quality and, of necessity, combines a wide variety of material.

The next section surveys cryptographic fundamentals, defining concepts which are used and examined throughout the paper. This is followed in Section III by a survey of cryptographic systems ranging from puzzle cryptography to the DES and the new public key systems. Section IV explores the taxonomy of cryptographic systems, providing a framework for classifying many of the systems given as examples in the previous section. Section V covers some practical problems of applying cryptography, while Section VI reviews various applications. The last section contains a survey of the relevant literature.

II. CRYPTOGRAPHIC FUNDAMENTALS

A. Privacy and Authentication

When valuable or secret data must be stored or transmitted, they are frequently protected physically through the use of safes, armed couriers, shielded cables, and the like. As elec-

0018-9219/79/0300-0397$00.75

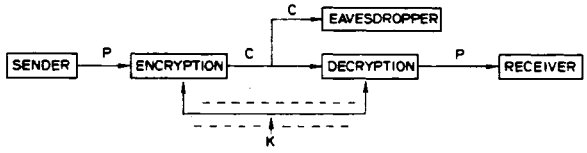


Fig. 1. The flow of information in a cryptographic privacy system.

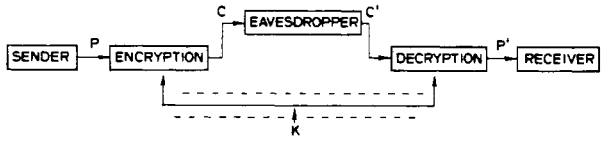


Fig. 2. The flow of information in a cryptographic authentication system.

tronic forms of communication and storage take over from their predecessors, however, such measures often become inapplicable, insufficient or uneconomical, and other techniques must be employed.

Central among these techniques is **cryptography**: the use of transformations of data intended to make the data useless to one's opponents. Such transformations provide solutions to two major problems of data security: the **privacy problem**, preventing an opponent from extracting information from a communication channel, and the **authentication problem**, preventing an opponent from injecting false data into the channel or altering messages so that their meaning is changed.

In telephone communication the problem of authentication predominates, since the called party cannot determine who is calling. Eavesdropping, which requires the use of a wiretap, is technically more difficult and legally more hazardous than calling and pretending to be someone else. In radio communication, the situation is reversed. Eavesdropping is passive and involves scant legal hazard, while injection exposes the illegitimate transmitter to discovery and prosecution.

Sometimes it is sufficient to authenticate that a message has not been modified by a third party (someone other than the sender or receiver). At other times, it is important for the receiver to be able to prove that he actually received the message from the sender, and that he has not modified it or originated it himself. The **problem of dispute** is to provide the recipient of a message with legal proof of the identity of the sender. On channels used for electronic funds transfer or contract negotiation, it is important to provide the electronic equivalent of a written signature, in order to settle any dispute between the sender and receiver as to what message, if any, was sent.

The problems of privacy and authentication are closely related and techniques for solving one can frequently be applied to the other. This paper examines the problem of privacy first because it is the older, the more widespread, and the more familiar of the two.

B. Basic Concepts

Fig. 1 illustrates the flow of information in a cryptographic privacy system. The transmitter generates a **plaintext** or unenciphered message P, which is to be communicated to a legitimate receiver over an insecure channel monitored by an eavesdropper. To prevent the eavesdropper from learning the contents of P, the transmitter **enciphers** or **encrypts** P with an invertible transformation S_K to produce the **cryptogram** or **ciphertext** $C = S_K(P)$. When the legitimate receiver obtains C he **deciphers** or **decrypts** it with the inverse transformation S_K^{-1} to obtain

$$S_K^{-1}(C) = S_K^{-1}(S_K(P)) = P \tag{1}$$

the original plaintext message.

The transformation S_K is chosen from a family of transformations known as a **cryptographic system**, **general system**, or merely **system**. The parameter that selects the individual transformation to be employed is called the **specific key** or **key**. The general system is a set of instructions, a piece of hardware, or a computer program which is capable of encrypting the plaintext (and decrypting the ciphertext) in a variety of ways, one of which is selected by the specific key. There is a close analogy here with a general purpose digital computer and a program. The computer, like the general system, is capable of a wide variety of behaviors, from which the program, like the specific key, selects one.

More formally, a **cryptographic system** is a single parameter family $\{S_K\}_{K \in \mathcal{K}}$ of invertible transformations

$$S_K : \mathcal{P} \longrightarrow \mathcal{C} \tag{2}$$

from a space $\mathcal{P}$ of plaintext messages to a space $\mathcal{C}$ of ciphertext messages. The parameter or key, K, is selected from a finite set $\mathcal{K}$ called the **keyspace**.

It is customary to regard the general system, that is the family of transformations, as public information. This is partly a matter of convention—that the portion that is publicly revealed is called the general system—and partly a reflection of a very important rule of security engineering: The security of a system should not depend on the secrecy of something which cannot be easily changed if it is compromised. The general system is usually a piece of equipment, which can be replaced only at considerable delay and expense, while the key is an easily changed datum such as an IBM card.

A cryptographic system is analogous to a resettable combination lock used to secure a safe. The structure of the lock is available to anyone who cares to purchase one. The combination, however, is kept secret and can be changed whenever it is suspected of having fallen into unauthorized hands. Even though an opponent knows the set of all possible keys or combinations, he may still be unable to discover which one is correct.

Despite the principle of publicity of the general system, it is still the practice of many users of cryptography to keep their systems secret. This is both because it is more difficult for an opponent to break a system with which he is unfamiliar, and because many suppliers of cryptographic equipment are themselves espionage organizations which want to prevent good systems from falling into the hands of their opponents. Unfortunately, this practice has carried over into the commercial area, making it difficult for a customer to obtain the information needed to make an intelligent choice among cryptographic products.

Since all security resides in the secrecy of the key, it must be conveyed to the sender or the receiver over a secure key distribution channel, such as courier service or registered mail, indicated by the shielded path in Fig. 1.

Fig. 2 illustrates why a cryptographic system can also be used to solve the authentication problem. In this case, the opponent not only sees all cryptograms flowing on the channel, but can alter them at will. The legitimate receiver protects himself from being deceived by an altered or injected message by decrypting all the messages he receives, and accepting only messages encrypted with the correct key.

Any attempt by the eavesdropper either to decrypt a cryptogram C to get the plaintext P, or to encrypt an inauthentic plaintext P' to get an acceptable cryptogram C', without obtaining the key K from the key channel is called **cryptanalysis.**

If cryptanalysis is impossible so that a cryptanalyst cannot deduce P from C, or C' from P' without prior knowledge of the key, the cryptographic system is said to be **secure.**

C. Cryptanalytic Attacks

The first step in assessing the adequacy of a cryptographic system is to classify the types of attack to which it may be subjected. For this purpose we must make a more thorough examination of the information which may be available to the cryptanalyst.

Usually the worst circumstance from the point of view of the cryptanalyst is to have nothing available to him but the material he has intercepted, knowledge of the general system, and some general knowledge of his opponent's messages. This may be limited to a knowledge of the statistical properties of the language in use (e.g., in English, the letter E occurs 13 percent of the time) and a knowledge of certain **probable words** (e.g., a letter probably ends "Sincerely yours,"). Although occasionally a cryptanalyst may be ignorant even of the language or system in use, this is the weakest threat to which a system is normally subjected, and any system which succumbs to it must be considered completely insecure. It is called a **ciphertext only attack.**

When cryptography is used to protect computer or business data, the cryptanalyst often knows substantial amounts of corresponding plaintext and ciphertext, making possible a **known plaintext attack.** The rigid structure of the formal languages used in programming, or of data such as business forms, guarantees that the opponent will know much detail of the plaintext *a priori*.

The known plaintext attack is a natural extension of the use of probable words in a ciphertext only attack. In formal languages such as Algol, the almost certain and often repeated occurrence of words like PROCEDURE and INTEGER provide a close approximation to known plaintext in most cases. Diffie and Hellman [8] describe how a known plaintext attack on the DES can be adapted to be a ciphertext only attack when the plaintext is represented in ASCII. This is an illustration of how rigid structure in the underlying language may provide the cryptanalyst with the equivalent of known plaintext, and emphasizes the prudence of assuming that any system will be subject to a known plaintext attack.

Many secret messages sent for business purposes, press releases and product announcements, for example, are intended for subsequent public release. If such a message is sent in a system which is not secure against a known plaintext attack, then all messages encrypted in the same key will be compromised. In the past, users have tried to counter this threat by **paraphrasing** plaintexts which had been encrypted prior to their release. In an English message, such as a press release, paraphrasing introduces the danger that the meaning will be altered, while in a formal message, such as a program, it is next to impossible. It is far better to design the cryptosystem to be secure against a known plaintext attack.

While a known plaintext attack is not always possible, its occurrence is frequent enough that a system that succumbs to it is not considered secure. The NBS has accepted the known plaintext attack as appropriate in judging the security of its DES.

The cryptanalyst is sometimes in the even stronger position of being able to see the ciphertext corresponding to any plaintext he chooses. His problem is to determine the key for later use in enciphering or deciphering other messages. This is a **chosen plaintext attack.** It is sometimes appropriate to consider the cryptanalyst as being able to select either the ciphertext or the plaintext at will. This is the even more powerful **chosen text attack.**

For the purpose of certifying systems as secure, it is appropriate to consider the more formidable cryptanalytic threats, as these not only give more realistic models of the working environment of a cryptographic system, but also make the assessment of the system's strength easier. IBM used a chosen plaintext attack in certifying the national DES [9]. And, as will be shown in Section III, many systems which were difficult to break using a ciphertext only attack could have been ruled out immediately under known plaintext attack or chosen text attack.

Systems vulnerable to the less powerful attacks are no longer of interest in view of the ease with which systems resistant to chosen text attacks can now be constructed. Systems of World War II or earlier vintage were frequently insecure against even a known plaintext attack, and were therefore subject to a cumbersome set of **signaling rules** designed to prevent an opponent from gaining possession of corresponding plain and cipher texts. Examples were, "Never encipher the same plaintext in two different cryptosystems." and "Never declassify a plaintext without first paraphrasing it." Human nature being what it is, such rules were often broken, sometimes with disastrous results.

D. Unconditional and Computational Security

There are two fundamentally different ways in which cryptographic systems may be secure.

In some systems, the amount of information available to the cryptanalyst is actually insufficient to determine the enciphering and deciphering transformations, no matter how much computing power the cryptanalyst has available. A system of this kind if called **unconditionally secure.**

Even when the intercepted material contains sufficient information to allow a unique solution to the cryptanalytic problem, there is no guarantee that this solution can be found by a cryptanalyst with limited computational resources. It then becomes the goal of the designer of a cryptographic system to make the enciphering and deciphering operations inexpensive, while ensuring that any successful cryptanalytic operation would be too complex to be economical. What is required is that the task of the cryptanalyst, though known to be achievable with a finite amount of computation, is so overwhelming as to exhaust the physical computing resources of the universe. We will call a task of this magnitude **computationally unfeasible**, and the associated cryptographic system **computationally secure.**

While some unconditionally secure systems can be proven secure, the theory of computational complexity is at present inadequate to demonstrate the computational infeasibility of *any* cryptanalytic problem. Cryptography is therefore forced to rely on the less formal **certification** process of subjecting a prospective cryptosystem to cryptanalytic assault under circumstances considered favorable to the cryptanalyst.

The only unconditionally secure system in common use is the **one time tape**, in which the plaintext is combined with a totally random key of the same length. Usually the plaintext is represented as an n-bit binary string which is XORed with a

totally random key of the same length. As the name suggests, this key is never reused.

Even if the cryptanalyst could try deciphering under all 2^n possible keys, he would merely see all 2^n possible plaintexts, including not only the correct one, but also all other meaningful plaintexts of the same length. Because intercepting the cryptogram does not allow the cryptanalyst to rule out any plaintext messages, he learns absolutely nothing except the length of the message.

Shannon [10] analyzed unconditional security in more detail. If the cryptanalyst has unlimited computing time, he has no need for computational efficiency, and can do a complete cryptanalysis by trying all possible keys and keeping all meaningful plaintexts which result. In a one time tape, all meaningful messages of the same length as the cryptogram must be kept, but in other unconditionally secure systems there may be a smaller number of meaningful solutions. For example, the cryptogram XMDA resulting from simple substitution on English can stand for any four letter word with no letter repeated (e.g., TIME or FOUR, but not LOOK or HASH).

As the amount of intercepted text increases a point may be reached at which a unique solution is possible. Shannon calls this the **unicity distance** N_0. In a one time tape this never happens and $N_0 = \infty$ while in a simple substitution cipher N_0 is clearly finite. According to Friedman [11] almost any simple substitution cryptogram of 25 characters or more can be broken by a skilled cryptanalyst. Since the cryptanalyst has limited computational abilities, he cannot try all 26! $\approx 4 \times 10^{26}$ keys and must rely on suboptimal methods such as frequency analysis. Thus we can only say that $N_0 \leqslant 25$ characters.

Shannon provides a model for predicting the unicity distance of a cipher, and the result of this model often agrees with practice. According to this "random cipher" model

$$N_0 = H(K)/D \tag{3}$$

where $H(K)$ is the entropy of the key (usually this is just the length of the key measured in bits or $\log_2$ of the number of keys) and D is the redundancy of the language measured in bits per character. (e.g., in English Q is always followed by U so the U is redundant).

The reader is referred to Shannon's paper [10] or a more recent presentation by Hellman [12] for a complete derivation, but an intuitive feel can be obtained by rewriting (3) as the requirement for a unique solution

$$H(K) \leqslant ND. \tag{4}$$

$H(K)$ represents the number of unknowns in a binary representation of the key, and in a loose sense ND represents the number of equations available for solving for the key. When the number of equations is smaller than the number of unknowns, a unique solution is not possible and the system is unconditionally secure. When the number of equations is larger than the number of unknowns, as in (4), a unique solution is possible and the system is no longer unconditionally secure (although it may still be computationally secure).

In a one time pad, $H(K) = \infty$, so by (3) $N_0 = \infty$. In a simple substitution, $H(K) = \log_2 (26!) = 88.4$ bits, so to calculate N_0 we must find D. Each character could convey as much as $\log_2 26 = 4.7$ bits if all strings of characters were possible. Since spelling and gramatical rules rule out most strings, in an average sense [13], [14] only about 1.5 bits of information are conveyed by each character. The remaining 3.2 bits are redundant, so $D = 3.2$ bits/character. Equation (3) thus predicts $N_0 = 28$ characters, in excellent agreement with practice.

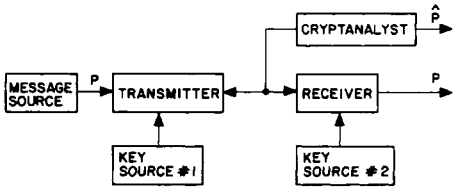


Fig. 3. The flow of information in a public key system.

While the Shannon theory approach to cryptography (i.e., assuming that the cryptanlyst has unlimited computational resources) is usually associated with a ciphertext only attack, known plaintext can be included as additional redundancy. If a 1000 character message is intercepted and some sequence of 100 plaintext characters is known, then the total redundancy is not just 3200 bits, but

$$(900 \text{ characters}) \times (3.2 \text{ bits/character}) + (100 \text{ characters}) \times (4.7 \text{ bits/character}) = 3350 \text{ bits} \tag{5}$$

because $\log_2(26) = 4.7$ bits of information are known about each of the 100 known characters.

Equation (3) indicates the value of data compression prior to encryption. Data compression removes redundancy thereby increasing the unicity distance. The redundant information can be added after deciphering so the legitimate receiver sees no difference. Perfect data compression would remove all redundancy and result in $N_0 = \infty$ with any size key, but is prohibitively expensive. Limited data compression tends to increase security while also reducing transmission costs.

E. Public Key Systems

The difficulty of distributing keys has been one of the major limitations on the use of conventional cryptographic technology. In order for the sender and receiver to make use of a physically secure channel such as registered mail for key distribution, they must be prepared to wait while the keys are sent, or have made prior preparation for cryptographic communication.

In the military, the chain of command helps to limit the number of user-pair connections, but even there, the key distribution problem has been a major impediment to the use of cryptography. This problem will be accentuated in large commercial communication networks where the number of possible connections grows as $(n^2 - n)/2$ if n is the number of users. A system with one million users has almost 500 billion possible connections, and the cost of distributing this many keys is prohibitive.

Section V-A discusses key management for conventional cryptographic systems in more detail, but at this point we introduce a new kind of cryptographic system which simplifies the problem of key distribution. Diffie and Hellman [15] and, independently, Merkle [16] have suggested that it is possible to dispense with the secure key distribution channel of Figs. 1 and 2, and communicate securely over the insecure channel without any prearrangement. As indicated in Fig. 3, two way communication is allowed between the transmitter and receiver, but the eavesdropper is passive and only listens. Systems which allow this are called **public key systems**, in contrast to conventional systems.

Two basic approaches have been suggested to this problem. In **public key distribution** the sender and receiver are able to agree on a key for use in a conventional cryptographic system. Even though the eavesdropper hears all exchanges, he is unable to compute the key, and thus cannot understand any of the conversation that follows. A public key distribution system is described in Section III-K. The second approach is through **public key cryptosystems**, which separate the keys used for enciphering and deciphering. Examples of such systems are examined in Sections III-L and III-M.

The reason that keys must be so carefully protected in conventional cryptographic systems is that the enciphering and deciphering functions are inseparable. Anyone who has access to the key in order to encipher messages can also decipher messages. If the enciphering and deciphering capabilities are separated, privacy can be achieved without keeping the enciphering key secret, because it can no longer be used for deciphering.

The new systems must be designed so that it is easy to generate a random pair of inverse keys E, for enciphering, and D, for deciphering, and easy to operate with E and D, but computationally infeasible to compute D from E.

A **public key cryptosystem** is a pair of families $\{E_K\}_{K\in\{K\}}$ and $\{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations

$$E_K\colon \{M\} \longrightarrow \{M\} \tag{6}$$

$$D_K\colon \{M\} \longrightarrow \{M\} \tag{7}$$

on a finite message space $\{M\}$, such that:

1) For every $K \in \{K\}$, D_K is the inverse of E_K. That is for any K and any M, $D_K E_K(M) = M$.

2) For every $K \in \{K\}$ and $M \in \{M\}$, the values $E_K(M)$ and $D_K(M)$ are easy to compute.

3) For almost every $K \in \{K\}$, any easily computed algorithm equivalent to D_K is computationally infeasible to derive from E_K.

4) For every $K \in \{K\}$, it is feasible to generate the inverse pair E_K and D_K from K.

The third property allows a user's enciphering key E_K to be made public without compromising the security of his secret deciphering key D_K. The cryptographic system is therefore split into two parts, a family of enciphering transformations, and a family of deciphering transformations in such a way that given a member of one family it is infeasible to find the corresponding member of the other.

The fourth property guarantees that there is a feasible way of computing corresponding pairs of inverse transformations when no constraint is placed on what either the enciphering or deciphering transformation is to be. In practice, the crypto-equipment must contain a true random number generator (e.g., a noisy diode) for generating K, together with an algorithm for generating the E_K–D_K pair from K.

A system of this kind greatly simplifies the problem of key distribution. Each user generates a pair of inverse transformations, E and D. He keeps the deciphering transformation D secret, and makes the enciphering transformation E public by placing it in a public directory, similar to directory assistance. Anyone can now encrypt messages and send them to the user, but no one else can decipher messages intended for him.

If instead of to conditions 1)–4) above, the set of transformations satisfy

1′) For every $K \in \{K\}$, E_K is the inverse of D_K. That is for any K and any M, $E_K D_K(M) = M$.

It is possible, and often desirable, to encipher with D and decipher with E. For this reason, we will usually refer to E_K as the public key and D_K as the *private key*, instead.

In the short time since public key cryptosystems were first proposed [15] several approaches have been discovered [17]–[19].

F. Digital Signatures

A second difficulty which has limited the application of conventional cryptography is its inability to deal with the problem of dispute. Conventional authentication systems, as shown in Fig. 2, can prevent third party forgeries, but cannot settle disputes between the sender and receiver as to what message, if any, was sent.

In current commercial practice, the validity of contracts and agreements is guaranteed by handwritten signatures. A signed contract serves as proof of an agreement which the holder can present in court if necessary, but the use of signatures requires the transmission and storage of written documents which is a major barrier to more widespread use of electronic communications in business.

The essence of a signature is that although only one person can produce it, anybody can recognize it. If there is to be a purely digital replacement for this paper instrument, each user must be able to produce messages whose authenticity can be checked by anyone, but which could not have been produced by anyone else, especially the intended recipient. In a conventional system the receiver authenticates any message he receives from the sender by deciphering it in a key which the two hold in common. Because this key is held in common, however, the receiver has the ability to produce any cryptogram that could have been produced by the sender and so cannot prove that the sender actually sent a disputed message.

The public key cryptosystems discussed in the previous subsection provide a direct solution to the signature problem, if they satisfy condition 1′). Systems which almost satisfy condition 1′) are also usable [18].

If user A wishes to send a signed message M to user B, he operates on it with his private key D_A to produce the signed message $S = D_A(M)$. D_A was used as A's deciphering key when privacy was desired, but is now used as his "enciphering" or "signing" key. When user B receives S he can recover M by operating on S with A's public key E_A.

B saves S as proof that user A sent him the particular message M. If A later disclaims having sent this message, B can take S to a judge who obtains E_A from the public file and checks that $E_A(S) = M$ is a meaningful message with A's name at the end, the proper date and time, etc. Only user A could have generated S because only he knows D_A, so A will be held responsible for having sent M.

This technique provides unforgeable, message dependent, digital signatures, but allows any eavesdropper to determine M because only the public information E_A is needed to recover M from S. To obtain privacy of communication as well, A can encrypt S with B's public key and send $E_B(S)$ instead of S. Only B knows D_B, so only he can recover S and thence M. B still saves S as proof that user A sent him M.

III. Examples of Systems and Solutions

So far we have discussed cryptography abstractly with few examples of specific cryptographic systems. In this section we restore the balance by examining in detail a number of exam-

ples representative of the variety of cryptographic systems presently or formerly in use, or of those planned for use in the future. In each case we will describe the operation of the system, and where possible we will outline its cryptanalysis. More thorough treatments of classical cryptanalysis can be found elsewhere [20]-[22].

A. Substitution

The simplest encryption technique: simple substitution is widely used as a puzzle. The key is a **permuted alphabet** (e.g., BWEKQFMVYALUCONPHSIDXTRGZJ) the letters of which are substituted for those of the normal alphabet. Using this alphabet: A is replaced by B, B is replaced by W, C is replaced by E, etc. If word divisions are preserved, the plaintext message

THE QUEEN HAS GIVEN BIRTH TO A HEALTHY SIX POUND BOY.

is transformed into

DVQ HXQQO VBI MYTQO WYSDV DN B VQBUDVZ IYG PNXOK WNZ.

As is well known to puzzle solvers, simple substitution ciphers on short alphabets such as Roman or ASCII offer little protection to the underlying plaintext. Substitution cryptograms can be solved by making frequency tables of letters, letter pairs (digrams), and letter triples (trigrams). In these tables the high frequencies of such letters as E and T, the low frequencies of Q and Z, and the frequent association of vowels with consonants are conspicuous. These allow the identification of the plaintext letters corresponding to the letters in the cryptogram. Similar results may be obtained by scanning the ciphertext for evidence of **pattern words** considered likely to be in the plaintext. Frequent words such as "EVERY," "THAT," and "LOOK" are conspicuous because of their repeated letters.

As the alphabet size is increased, these techniques become more and more expensive to apply. If instead of ASCII with its 128 7-bit characters, we had chosen the set of all 32-bit words, the compilation of a frequency table would have required an unreasonable amount of effort. Unfortunately, the compilation of such a cipher is also prohibitively expensive. The permuted alphabet, like the frequency table, is 2^{32} or about four billion words in length. Substitution ciphers on large "alphabets" do not, therefore, allow all possible permutations, in order to limit the key to a more reasonable size. The DES of Section III-I is a good example.

B. Transposition

In **transposition**, the positions of the plaintext letters in the message rather than the letters of the alphabet are permuted. As an example, if the above message is broken into five character groups (including spaces) and the letters in each group rearranged according to the permutation $\left(\begin{smallmatrix}1&2&3&4&5\\2&5&1&4&3\end{smallmatrix}\right)$ (i.e., the third character of each group is written first, the first character is written second, etc.) the cryptogram becomes

ETQ HEU NESHG AEI NVRBHTIO A TE LAHYTS H IOPXDUB NXOXXY.

In the case of transpositions, frequency tables of letter pairs and triples reveal the breakup of common letter pairs, such as the T and H of THE, allowing the plaintext to be reconstructed by seeking permutations which rejoin them. In this way the key used in transforming the plaintext, and thus the plaintext itself, can be recovered from the cryptogram alone.

Substitution and transposition, although of little use by themselves, are important components in more complex cipher systems, some of which will be treated in detail later in this section.

C. Polyalphabetic Ciphers

In an effort to defeat frequency analysis, cryptographers developed substitution ciphers in which several different substitution alphabets were used periodically to encipher the message. For example, if 5 alphabets are used, letters numbered $5n+i$ are enciphered in the ith alphabet (e.g., letters numbered 1, 6, 11, 16, . . . are enciphered in the first alphabet).

A frequency count, either on individual letters or letter pairs, is now much flatter and provides few clues. This barrier blocked a general solution of polyalphabetic ciphers for over 300 years [23, p. 207], until the appearance in 1863 of a book by a Prussian officer, Friedrich Kasiski.

Kasiski's approach is to determine the period by first looking for repeated groups of three or more ciphertext letters, which usually occur due to a frequent plaintext trigram (which may or may not be a word by itself) such as THE (4.5-percent probability) or ING (1.2-percent probability), occurring twice in the same phase. For example, if the plaintext message "THEY ARE NOT THERE." is enciphered using five alphabets, the two occurrences of THE will result in identical ciphertext since they are $10 = 2 \times 5$ characters apart. The cryptanalyst then counts the number of characters between repeated groups. The period will divide all these numbers, except for a few that were not caused by repeated plaintext groups. Factoring the numbers of characters between repeated groups, and looking for a frequently repeated factor usually identifies the period rather rapidly.

The assumed period n is checked by making a frequency count on every nth ciphertext letter. This is done for each of the n possible phases. If each of the n frequency counts thus obtained has the highly nonuniform distribution characteristic of a monoalphabetic substitution the assumed period is correct. This is because the periodic use of alphabets consists precisely in enciphering every nth letter in the same alphabet.

Once the period has been determined, the problem can be solved as a set of n different simple substitutions. Examples are worked out in full detail in [23, pp. 209-213] and [20, ch. 3].

If a plaintext is enciphered with a periodic polyalphabetic cipher, and the resulting cryptogram is superenciphered with a second periodic polyalphabetic cipher with a different period, the period of the resulting cipher will usually be much longer than the period of either component. If the two periods, ℓ_1 and ℓ_2, are relatively prime (have no common factor) then the resulting period will be $\ell_1 \times \ell_2$. Cryptanalysis of such multiple loop Vigenere systems is more difficult, but can still be accomplished [24].

D. Running Key Cipher

In an effort to remove the weakness of periodic polyalphabetic ciphers, cryptographers turned to running key ciphers which are aperiodic polyalphabetics. The key is typically the name of a readily available book, together with a page, line, and column number (e.g., *Proceedings of the IEEE*, October 1976, p. 1488, line 4, column 19). To encipher the message

"THIS MATERIAL IS ENCIPHERED IN A RUNNING KEY." we write the message and the text of the book ("on a non-interfering basis over the Defense Satellite . . .") one above the other as shown below and add them mod-26, by regarding A as 0, B as 1, · · · , Z as 25.

Plaintext:

THIS MATERIAL IS ENCIPHERED IN A RUNNING KEY.

Key:

ONAN ONINTERF ER INGBASISOV ER T HEDEFEN SES.

Ciphertext:

HUIF ANBRKMRQ MJ MAIJPZMJSY ME T YYQRNRT CIQ.

Spaces are usually deleted from the ciphertext to hinder cryptanalysis. Decryption is carried out by subtracting the key from the ciphertext mod-26.

Although a Kasiski solution is no longer possible due to the aperiodic selection of the 26 alphabets used (one for each of the 26 possible values of the running key), Bazeries [23, p. 246] solved running key ciphers in the late 1890's. Friedman published a solution in 1918 [25] and added additional techniques in his *Military Cryptanalysis* [22]. There are two basic approaches. The more powerful of the two depends on the cryptanalyst knowing a **probable word**, one which probably occurs in the plaintext. In military communications the words BATTALION, COMPANY, ATTACK, etc., are probable. In a message relating to bidding on oil leases, probable words are OFFSHORE, COMPETITION, LEASES, etc. Lacking even this, the cryptanalyst can use common words or groups of letters such as OFTHE, TION, WHICH, etc. If a number of probable words are tried, one will usually produce success.

To test for the existence of a probable word the cryptanalyst subtracts it from the ciphertext, mod-26, in all possible locations. If the word is present, this process produces the key when tried at the proper location. When tried at an incorrect location (and all locations are incorrect if the probable word is not present), the cryptanalyst finds a random looking result. Considering the ciphertext in our example, HUIFANB . . . TCIQ, the cryptanalyst might use CIPHER as a probable word or partial word. Since the word ENCIPHERED did occur, when the cryptanalyst subtracts CIPHER from IJPZMJ he will obtain GBASIS. The inclusion of the word BASIS is indicative of success, and the cryptanalyst then tries to extend either the plaintext or the key in either direction (e.g., by trying INGBASIS to get ENCIPHER). When CIPHER is subtracted off at the point just before the correct one (i.e., ciphertext AIJPZM) the cryptanalyst finds the key would have to have been YAUIVV, an impossibility.

Occasional false alarms will occur, especially with short probable words, but these will be detected as analysis proceeds since they will not allow reasonable extensions. It should be noted that our example is too short to allow an easy solution and is being used solely for illustrative purposes.

The second approach to solving a running key cipher is to note that each ciphertext letter tends to represent certain plaintext-key letter pairs. In our example, ciphertext *M* occurs 5 times, and each time results from the pair I-E or E-I. This is rather unusual, but indicative of the approach. Each ciphertext *M* can represent A-M, B-L, C-K, D-J, E-I, F-H, G-G, N-Z, O-Y, P-X, Q-W, R-V, S-U, T-T and the reversals thereof (i.e., M-A, L-B, etc.). It is seen that E-I, I-E, and T-T are the only pairs where both letters in the pair are highly probable. Calculation shows that these three pairings account for 64 percent of the occurrences of ciphertext M. At the other extreme only 0.1 percent of the time that ciphertext *M* occurs does it stand for the pair Q-W. The ten lowest probability pairs (Q-W, W-Q, Z-N, N-Z, K-C, C-K, J-D, D-J, X-P, and P-X) account for only 2 percent of the occurrences of ciphertext *M*.

Therefore, while there are 26 possible pair substitutes for each letter of the ciphertext, and 26^n possible pairs of message and key to try on an n letter cryptogram, most of these can be tentatively discarded without introducing too great an error into the solution.

The cryptanalyst substitutes the several highest probability pairs that correspond to each ciphertext letter and attempts to find adjacent pairs which have a high digram probability. He then tries extending this path to produce good trigrams, etc.

The fact that running key ciphers can be solved is proof that English is at least 50-percent redundant, since two n character texts (the plaintext and the running key) can be recovered from a single n character string (the cryptogram). This observation leads to an obvious, but little used method for strengthening a running key cipher: use two or more successive encipherments with different running keys. Since English is about 75-percent redundant [13], [14], four encipherments would be secure against all attacks. There cannot be a way for the cryptanalyst to recover the plaintext since, by symmetry, he could then also recover the four running keys, implying that English is at least 80-percent redundant.

While techniques at least as clumsy as this were often used (e.g., see Kahn's description of a Russian spy cipher [23, pp. 669–670]), none of them offer its ironclad guarantee of security. It is therefore somewhat surprising that multiple running key ciphers were not more widely used. Today, they are of little value because of the availability of inexpensive, more easily used electronic techniques.

In the limit as the number of running keys tends to infinity, the mod-26 sum approaches a totally random character sequence, which can be regarded as the key in a one time tape system. It is now clear why the cryptanalyst is unable to learn anything about plaintext which has been enciphered with a one time tape. If he could learn even 1 percent of the plaintext, he could learn (by symetry) 1 percent of each key, for an infinite information gain from a finite message.

E. Codes

Cryptographic techniques like substitution and transposition which operate on the plaintext without regard to its linguistic structure are called **ciphers** and the cryptotext they produce is referred to as **ciphertext**. A cryptographic system which operates on larger linguistic units of the plaintext, such as words or phrases, is called a **code**. A code usually consists of a list of words and phrases together with corresponding random groups of numbers or letters called **codegroups**. Since codegroups are typically shorter than the expressions they represent, codes offer the advantage of data compression as well as secrecy.

If used properly, codes are far more difficult to break than other classical paper and pencil systems. There are three basic reasons for their success. Probably the most important is the large amount of key involved. A typical cipher system employs at most a few hundred bits of key. For example, the key to a simple substitution cipher is a permuted alphabet, representing fewer than 90 bits, while a good-sized code book

may contain a hundred thousand or even a million bits. As pointed out by Shannon [10], the job of the cryptanalyst becomes more difficult as redundancy is removed from the message, and codes remove redundancy. Finally, the code operates on relatively large blocks of plaintext (words or phrases) and thereby conceals local information which might otherwise provide valuable cryptanalytic clues.

Against these strengths it must be said that the key is not well used in a code, since only a very small amount of the code book comes to bear in encoding an individual word or phrase. As a result, codes succumb to frequency analysis under heavy use, and are particularly vulnerable to attack by known plaintext, which becomes more available the longer a code is in service. For these reasons, codes must be changed frequently in order to be secure.

Despite their success in some circumstances, codes are not well adapted to modern communications, because they are not easily automated, and because the key (codebook) is not easily changed if compromised. This violates a basic security principle, and was often responsible for code failures.

F. Hagelin Machine

The Hagelin C-48 machine was widely used as an American field cipher during World War II, under its military designation M-209, and is still in occasional use today (Fig. 4). We treat it here because it is one of the few systems whose complete description is public knowledge, and because its solution brings out some techniques which are valuable in general. Our description of the solution draws on Kahn's outline of an attack [23, p. 431], on unpublished work of Jim Reeds of the Statistics Department of the University of California at Berkeley, and Robert Morris and Dennis Ritchie of Bell Labs [26], [27]. Wayne G. Barker [28] has recently published a book which has significant overlap with these approaches.

Our basic approach is due to Reeds and was first implemented by Morris and Ritchie. According to their results, one to two thousand characters of ciphertext suffice and to a large extent the language used in the plaintext need not be known *a priori!* For the sake of clarity, we have simplified their technique. As a result, the technique presented here requires substantially more text. At the end of this subsection, we indicate the nature of their improvements.

Kahn [23, pp. 427–431] gives a complete mechanical description of the machine. For ease of understanding we delete some of the mechanical interactions which are not involved in a mathematical description of the process.

The M-209 combines the plaintext, character by character, with the **keystream** (a long pseudorandom sequence, derived from the key) to produce the ciphertext in a manner similar to Vigenere or running key ciphers. The plaintext, the keystream, and the ciphertext are all written in a 26-character alphabet, and the plaintext P is subtracted from (rather than added to) the keystream KS mod-26, to make enciphering self-inverse

$$C = KS - P \quad \text{mod-26}$$

and

$$P = KS - C \quad \text{mod-26}.$$

Two major components are required in the keystream generation process: the **keywheels**, which generate a long pseudorandom sequence of six bit groups and the **cage**, which converts the 6-bit groups into characters. The machine is keyed by adjustments to both the keywheels and the cage.

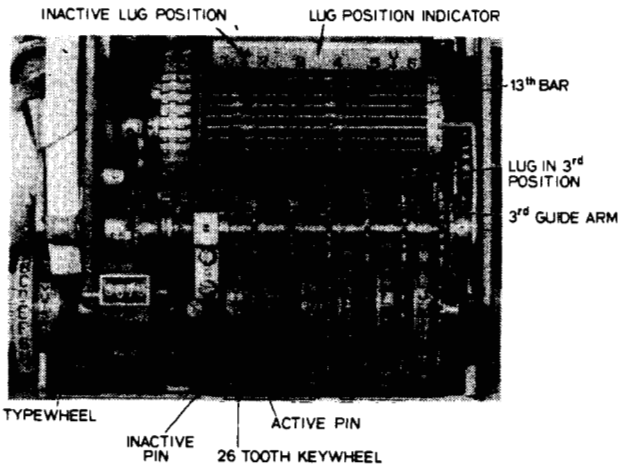


Fig. 4. Essential parts of Hagelin machine.

The six keywheels may be regarded as gears with 26, 25, 23, 21, 19, and 17 teeth, respectively. Next to each tooth of each wheel is a pin which can be either extended or retracted. One bit of key is used to set each pin, and we will use the convention that an extended pin corresponds to a key bit equal to 1.

Once this portion of the key has been set, the keywheels are set to their initial reference positions. This is facilitated by labeling the teeth of each gear with as many letters of the alphabet as there are teeth, and providing a window over each gear that shows which teeth are active (in position to interact with the cage).

At the begining of the encryption process the six windows will show AAAAAA. The 6 bits which correspond to the A teeth on each gear determine the first letter of the keystream. After the first character has been enciphered each keywheel is rotated one position so that the windows show BBBBBB. Six new bits of key (pins) come into play so the second character in the keystream is independent of the first. This holds true for the first 17 characters, at which point the windows show QQQQQQ. The 18th character of the keystream, however, is determined by the pins corresponding to teeth RRRRRA since the last wheel has finished one complete revolution. Thus the 18th character of the keystream is correlated with the first character of the keystream. Similarly the 19th is correlated with the second since the windows show SSSSSB. The 20th is correlated with both the 3rd and 1st since the next to last gear has now also come full cycle and the windows show TTTTAC.

Because 26, 25, 23, 21, 19 and 17 do not contain any common factors the keywheels return to the AAAAAA position only after $26 \times 25 \times 23 \times 21 \times 19 \times 17 \approx 101$-million characters have been enciphered.

In most of what follows, we will treat the cage as a programmable READ-ONLY memory (PROM) holding $2^6 = 64$ characters. These characters are drawn from the Roman alphabet, and are represented by the numbers 0 through 25. Each six bit group produced by the keywheels is transformed into a character by using the six bits to designate a memory location in the PROM and taking the contents of this location as output.

The 1940 vintage M-209 uses a mechanical technique instead of a PROM. This technique restricts the mapping in a way which makes the solution easier. There are 27 bars with 2 lugs on each bar. These 2 lugs can be set to any of 8 positions. Six

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

Fig. 5. Hagelin ciphertext matrix for 17 pin wheel.

of these 8 positions line up with the 6 pins on the keywheels, and 2 positions are "inactive." After a new 6-bit pattern has been established by rotating each keywheel, each of the 27 bars passes over these 6 pins. If either or both of the lugs on a bar hits an extended pin, that bar is pushed to the left (the lugs are beveled), and moves a gear connected to the printing wheel which prints the ciphertext. Thus the number of bars activated equals the shift or keystream value KS. (Of course, if 26 or 27 bars are activated $KS = 0$ or 1 since the printing wheel has only 26 positions.)

Although it does not affect our analysis, we note that the ciphertext alphabet is displaced one unit on the printing wheel so that the actual value of KS is one less than in our description.

Reeds approach makes use of the fact that the mod-26 sum of two independent, nonuniformly distributed character streams is typically not uniformly distributed. The plaintext, being in a natural language, has a nonuniform probability distribution on individual characters (e.g., in English Pr(E) = 0.13 while Pr(Z) = 0.001). And, since 26 does not evenly divide 64 (the number of PROM addresses) the keystream characters occur with unequal frequencies, even if all 6-bit patterns occur equally often. At best 12 letters occur with probability 3/64 and 14 occur with probability 2/64.

If one of the six bits that enter the PROM is held constant, the nonuniformity of the keystream tends to increase, and this will be reflected in an increase in the nonuniformity of the ciphertext. If the first 17 ciphertext characters are written as the first row of a matrix with 17 columns, the next 17 are written as the second row, etc. (Fig. 5), all characters in column i will have been enciphered with the ith bit on the last (17 tooth) keywheel in the active position. If the ith and jth pins on the last keywheel are either both 0 or both 1 then the ith and jth columns will be drawn from the same distribution, while if these bits differ the distributions will differ. Cluster analysis can be used to separate the columns into two groups in such a way that all of the columns in each group are drawn from the same distribution. Equivalently, we have decided which pins on the last wheel are in the same position. At this point it makes no difference which group we assign to 0 and which group we assign to 1 since our later estimate of the PROM's contents can correct for any mislabeling of the groups, although not for assigning a column to the wrong group.

By writing the ciphertext in a matrix with 19 columns, the cryptanalyst can perform a similar test and grouping to obtain two equivalence classes for the 19 pins on the fifth keywheel. He can then continue to obtain the equivalence classes for the remaining keywheels. While some errors may be made in these assignments they will be caught in the next phase.

Having tentatively assigned the value 0 or 1 to each of the pins on each wheel, the cryptanalyst calculates the sequence of 6-bit patterns which would have occured for the run of ciphertext, assuming all his estimated pin settings are correct. He also sets up 64 columns marked 000000, 000001, $\cdots$, 111111, and as an estimated 6 bit pattern occurs he writes the corresponding ciphertext character in the appropriate column. If most pin setting estimates are correct, the characters in a column will have a frequency distribution which is close to a reversed and shifted distribution on the plaintext (since $C = KS - P$) and the shift or keystream value KS corresponds to the PROM contents for the address specified by the column heading.

Errors in the pin settings can be corrected by deciphering the ciphertext with the assumed key and noting the locations of errors in the plaintext. By writing the time of occurrence next to each error the cryptanalyst can get valuable clues as to which pin settings are wrong. If the second pin on the last keywheel is in error it can cause errors only at times of the form $17n + 2$. Each of the pins has a similar "error trace" which can be identified. For example if the only errors occur at time 2, 12, 36, 70, 87, 90, and 116 we can identify 2, 36, 70, 87 as being of the form $17n + 2$ with $n = 0, 2, 4$, and 5; and 12, 90, 116 as being of the form $26n + 12$ with $n = 0, 3$, and 4. This indicates that the second pin on the 17 tooth wheel and the twelfth pin on the 26-tooth wheel are in error. This hypothesis can be checked by complementing these pins and seeing if the deciphered text is now correct.

In a known plaintext attack, the cryptanalyst can obtain the portion of the keystream corresponding to the known plaintext as

$$KS = C + P \text{ (mod-26)}.$$

The settings of pins on the keywheels are determined in a manner similar to the ciphertext only case with the matrix of ciphertext characters replaced by a matrix of keystream characters. Once the pin settings have been tentatively identified, the keystream is written out in a 64-column matrix as before. Now, if the pin settings are correct, each column contains a single letter (the PROM value for that address), rather than a distribution, and the process is vastly simplified.

In a known plaintext attack, the details of cage structure are more important than in a ciphertext only attack. Morris reports that using this information, solutions can be obtained with approximately 75 characters. Consideration of the actual mechanical structure of the cage aids cryptanalysis. For example, the bit pattern 000000 always produces $KS = 0$, since there are no extended pins. Similarly, when the bit pattern has a single 1 not many bars will be activated, and these 6 "PROM" addresses will usually produce small values of KS. As the Hamming weight w of the bit pattern (i.e., the number of 1's) increases, the value of KS tends to increase. The only exception is that when 111111 occurs ($w = 6$) the values $KS = 0$ and $KS = 1$ become more probable due to the possibility of 26 or 27 bars being activated.

An earlier version of the M-209, known as the C-36 [23, p. 428] has only 25 bars. The folding over present in the M-209's distribution was therefore absent and $KS = 0$ and 1 almost never occurred. A later Hagelin machine, the CD-57, had the equivalent of 40 bars causing even greater foldover and tending to smooth out the KS distribution.

The unevenness of the keystream distribution led the Army to issue guidelines for lug placement [28, pp. 198–210] to

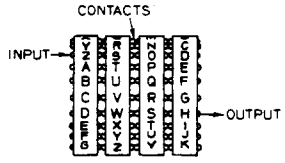


Fig. 6. The path of signal passing through a rotor.

ensure a more uniform distribution, but these guidelines introduce another weakness which can be exploited in cryptanalysis. One keywheel will have a large number of lugs opposite it and the other wheels will have successively smaller numbers of lugs.

In the extreme case, if a wheel had no lugs opposite it, it would be impossible (and unnecessary) to recover that keywheel's pin settings since they would have no effect. The cryptanalyst therefore chooses the keywheel with the largest number of lugs and thus the greatest effect, by looking for two well defined cluster distributions, and solves for its pin settings first. He then attacks successively harder keywheels, making use of the information gained about the previously considered keywheels' pin settings.

Because the keystream distribution depends on w, he can make estimates of w, and indirectly of the pin settings, based on his knowledge of the keystream. Whenever a larger (smaller) than average keystream value occurs, it indicates that each of the 6 active pins is more likely to be extended (retracted).

These improvements allow known plaintext attacks to succeed with 50 to 100 characters of text, and ciphertext only attacks to succeed with 1000 to 2000 characters of text.

G. Rotor Machines

The rotor machine was the most important cryptographic device of World War II, and remained dominant at least until the late nineteen fifties. The M-134 or SIGABA, the highest level American system of World War II, the British TYPEX, and the German Enigma were all rotor machines, and undoubtedly many are still in use today.

The central component of the rotor machine is the rotor or wired wheel, a disk about the size of a hockey puck which serves to implement a cipher alphabet. Around the perimeter of each circular face of the disk there are M evenly spaced electrical contacts where M is the number of characters in the alphabet. If only the Roman alphabet is represented, $M = 26$, whereas if full ASCII were in use M would equal 128. Each contact on the front face is wired to exactly one contact on the rear face as shown in Fig. 6. An electrical signal representing a character will thus be permuted as it passes through the rotor from the front face to the rear face.

If the rotor is rotated from one position to another, the permutation which it produces on an incoming signal will change. In general this permutation can be represented as

$$P = C^j R C^{-j}$$

where R is the permutation implemented by the rotor in its initial position, C is a cyclic shift through one position and C^j is a cyclic shift though j positions. For example if $R(A) = G$ and the rotor is moved 3 positions ($j = 3$) then plaintext D will be opposite the rotor contact which used to represent plaintext A and ciphertext J will be opposite the rotor contact which used to represent ciphertext G and $P(D) = J$ when $j = 3$. This is expressed algebraically as

$$P(D) = C^3 R C^{-3}(D) = C^3 R(A) = C^3(G) = J \qquad (8)$$

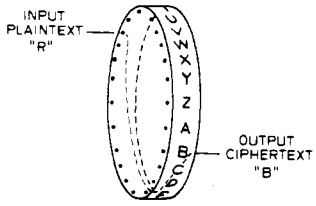


Fig. 7. A bank of rotors.

Rotors can be connected one after the other in a bank in such a way that a signal entering at one end will be permuted by each of the rotors before leaving at the other. The effect of this bank of rotors (Fig. 7) can be described mathematically as

$$\begin{aligned} P &= C^{j_1} R_1 C^{-j_1} C^{j_2} R_2 C^{-j_2} \cdots C^{j_{n-1}} R_{n-1} C^{-j_{n-1}} C^{j_n} R_1 C^{-j_n} \\ &= C^{j_1} R_1 C^{j_2 - j_1} R_2 \cdots R_{n-1} C^{j_n - j_{n-1}} R_n C^{-j_n}. \end{aligned} \qquad (9)$$

The bank of rotors is capable of implementing a large variety of different permutations as the rotors are rotated to different positions. In order to build a strong cryptographic system, the state of the rotors must be changed from character to character, much as the state of the Hagelin machine is changed from character to character. A rotor machine consists of a bank of rotors, together with a mechanism for changing the positions of the rotors each time a character is enciphered, along with input and output devices such as a paper tape reader and a printer.

The simplest possible rotor motion is that of an odometer, and was used by the German Enigma machine in World War II. The rightmost wheel rotates once each time a character is enciphered. When this (or any other) wheel has moved M positions and made one complete rotation, the wheel to the left of it is advanced one place and the process is repeated. This process will carry the bank of rotors through all of its possible states before it ever repeats.

The characteristics which are desirable in a pattern of rotor motion are straightforward but difficult to achieve with purely mechanical devices.

1) The period must be long.

2) Each change of state should be a large one, that is all or most of the rotors should rotate relative to each other, after each character so that few, if any, of the exponents, $j_i - j_{i-1}$, in (9) are zero.

Consideration of the odometer motion shows that it is optimal with respect to the first property, but fails dismally with respect to the second. In order to improve this aspect of the odometer's performance, each wheel can rotate more than once when it is moved. The period is still maximal if the displacement of each wheel has no factor in common with the alphabet size M.

Another approach is to move each rotor one position as in the motion of the Hagelin machine. Since each of the rotors has the same number of contacts, however, this would result in an unacceptably short period of at most M for the entire machine. The usual solution is to limit the number of permitted stopping places of each rotor by adding a plastic outer ring on which the stopping places are marked in some way. A rotor machine using the Roman alphabet can thus be made to move in much the same way as the wheels of a Hagelin machine as follows. The first wheel is allowed to stop in each of its 26 positions. The second wheel, however, is allowed to stop in only 25. The third can stop in only 23, and so on until the

sixth which can only stop in 17. The period of this rotor machine is now 101 million, the same as the period of the Hagelin machine, instead of $26^6 \approx 309$ million with odometer motion. The loss in length of period is well justified by the gain in complexity of motion. The second property is moderately well satisfied, since each of the wheels moves after the encipherment of each character, and many can move with respect to each other.

Further refinements in the motion of rotor machines are possible, but this is primarily a question of mechanical engineering which is not of central interest in modern cryptography.

Rotor machines may be keyed by changing any of their variables: the rotors, the order of the rotors, the number of stopping places per wheel, the pattern of motion etc. Because rotors are difficult to rewire, the usual practice is to supply the machine with a **basket** of rotors containing more rotors than it can hold at any one time. **Primary keying** is done by selecting the rotors which make up the basket. **Secondary keying** is done by selecting rotors from the basket, and setting parameters which govern the machine's motion.

The Enigma machine, used heavily by the Germans in World War II required three rotors and, at first, a basket of only three rotors was issued. The order of these in the rotor bank and their initial positions constituted part of the key. Later in the war, the number of rotors in the basket was increased to five, but even then there were only $5 \times 4 \times 3 = 60$ possible ways to load the rotors. Most of the key was in the initial rotor positions ($26^3 = 17\,576$ possible settings) and in a plugboard which effected a simple substitution on the plaintext prior to its being enciphered and on the ciphertext after encipherment ($26! = 4 \times 10^{26}$ possibilities).

Cryptanalysis of rotor machines when the basket of standard rotors is known merits further study, but is obviously simpler than the problem of determining the rotors directly. The problem of determining the rotors (assuming known rotor motion) is treated by Pohlig [29], for a single rotor machine under a ciphertext only attack and for a two-rotor machine under a known plaintext attack. His approach can be generalized to larger machines, the real question being the rate of growth of the required computation as a function of the number of rotors. It appears that modern computational aids will allow this approach to recover between three and five rotors.

Dov Andelman of Stanford University and Jim Reeds of the University of California at Berkeley are experimenting with a maximum likelihood estimation approach in which the rotors are modeled as noisy "channels" whose transition probabilities are to be estimated. Their results, which appear very promising, will be treated in a forthcoming paper.

The rotors cannot be determined completely because R_i, R_{i+1} cannot be distinguished from R_iC, $C^{-1}R_{i+1}$ as shown in (g) above. The cryptanalyst can therefore choose one wire in each rotor, save one, arbitrarily. Whatever error he makes in R_i will be compensated for by shifting all other wires on R_i a like amount, and all wires on R_{i+1} by the negative amount.

The encipherment of a single character depends only on those wiring choices over which its signal travels. If two wires are interchanged which are not in this signal path, the encipherment is not changed. The cryptanalyst can therefore test hypotheses concerning rotor wirings more easily. Testing one contact's wiring rules out many keys (rotors), not just one, and is therefore much more efficient than exhaustive search of all possible rotor wirings. This is a basic weakness in a cryptographic system (the ciphertext should depend on the entire key), and has been present in all of the systems discussed thus far in this section.

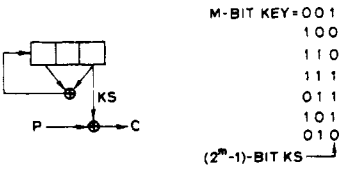


Fig. 8. A three-stage shift register.

H. Shift Registers

In the provably secure one time tape cipher, the key is a totally random bit string which is XORed with the plaintext, also represented in binary form (e.g., $A = 00000$, $B = 00001$, $C = 00010$, etc.) to produce the ciphertext. The key must be as long as the message and cannot be reused. In an effort to use shorter keys, people often think of using pseudorandom bit generators, notably feedback shift registers to produce long, nonrepeating, random like binary sequences from a short key.

Fig. 8 illustrates a three stage shift register. The first and third stages are added mod-2 (XORed) to produce the next input to the register. If the key is 001, the register is started in this state (with the 1 in the rightmost stage) and the sequence of states is shown in Fig. 8. The register goes through all seven nonzero states before returning to its initial state, 001. Because the eighth possible state, 000, loops on itself, this is the longest period obtainable with a linear (in mod-2 arithmetic) feedback shift register, and the sequence is called a Maximal Length Shift Register Sequence (MLSRS).

As depicted in Fig. 8, the plaintext bits are XORed with the MLSRS bits to produce the ciphertext. Because the XOR operation is self-inverse, deciphering is carried out in the same fashion.

The period of the sequence depends on the choice of taps. If, for example, all three stages had been included in the mod-2 sum, then the period would have been dependent on the choice of the initial state and would be at most four (if started in state 001), and could be as small as one (if started in state 111).

As shown in Golomb [30] or Peterson and Weldon [31], for any integer value of m there is an m-stage MLSRS with period $2^m - 1$. If $m = 100$ then the sequence will not repeat in 10^{16} years on a 1-Mbit/s data link. Typical tap sequences can be found in Peterson and Weldon's Appendix C for all values of m up to 34. Taps are easily found for much larger values of m [32], [33].

In our small example, the output sequence (keystream) of the register is 1001110, after which it repeats. There are four 1's and three 0's, as close to an even distribution as could be asked of a sequence of length 7. If successive pairs of bits are considered, there are two occurrences each of 01, 10, and 11 and one occurrence of 00, again as close as possible to an even distribution. With an m-stage MLSRS, this equidistribution property extends to triples, quadruples, etc. up to m-tuples. Because they are so evenly distributed, MLSRS's are often used as pseudorandom bit sequences in cryptosystems which imitate the much more secure one time tape.

While this cryptosystem is meant to imitate the provably secure one time tape, it is not only not provably secure, it is demonstrably insecure and can be broken in a few seconds on a minicomputer. If the taps are fixed, only m bits of known

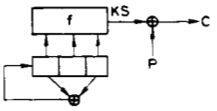


Fig. 9. Linear shift register with nonlinear output logic.

plaintext are needed to recover the initial contents of the register. The m bits of known plaintext are XORed with the corresponding m bits of the ciphertext to recover m bits of the keystream. These m bits give the state of the shift register at some time and, by running the register backward, its initial contents can be found.

If the taps are part of the key, only $2m$ bits of known plaintext are needed to allow rapid solution for both the taps and the initial contents of the register. Let $s(i)$ be a column vector of m 0's and 1's which gives the state of the register at time i. Then

$$s(i+1) = As(i) \mod 2 \tag{10}$$

where A is an $m \times m$ matrix which specifies the taps. The A matrix for the three stage register of Fig. 8 is

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

A is always of this form, with 1's just below the main diagonal, the tap sequence in the first row, and 0's elsewhere.

The $2m$ bits of known plaintext allow calculation of $2m$ successive bits of the keystream. To simplify notation, we take these to be the first $2m$ bits of the keystream. We then have $s(1)$, which is the first m bits of the keystream; $s(2)$, which is the second through $(m+1)$th; $\cdots$; and $s(m+1)$ which is the last m.

We can then form the two $m \times m$ matrices

$$X(1) = [s(1), s(2), \cdots, s(m)]$$

$$X(2) = [s(2), s(3), \cdots, s(m+1)]$$

which are related by

$$X(2) = A\,X(1) \mod 2. \tag{11}$$

It can be shown that, for any MLSRS, $X(1)$ is always nonsingular so A can be computed as

$$A = X(2)\,[X(1)]^{-1} \mod 2.$$

The matrix inversion requires at most on the order of m^3 operations, and is thus easily accomplished for any reasonable value of m. For example, if $m = 100$, the $m^3 = 1\,000\,000$ and a minicomputer with a 2-μs instruction time would require 2 seconds for the inversion. Even if $m = 1000$ such a minicomputer would need only 2000 s (less than an hour) for the inversion. (The structure of $X(1)$ actually allows the inversion to be accomplished even more rapidly.)

MLSRS's can be strengthened for cryptographic applications through the use of nonlinear logic. Groth [34] and Key [35] have suggested using linear logic to generate a MLSRS, but then using a nonlinearly "filtered" version of the shift register contents as the keystream, as indicated in Fig. 9. The function f should be chosen to have a good balance between 0's and 1's, so that the filtered sequence is close to evenly distributed.

It also should be chosen so that the filtered sequence has a long period. If $2^m - 1$ is a prime number (as $7 = 2^3 - 1$ is prime), then the filtered sequence must have a period of either 1 (which can be easily checked) or $2^m - 1$. Many such values of m are known including $m = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203$, and 2281 [36, p. 50], and the associated primes are called Mersenne primes.

While the filtered output sequence cannot usually be generated by an m-stage linear feedback shift register, it can always be generated by a larger linear feedback shift register [37]. In the extreme, a $2^m - 1$ stage register will always do, and often a much shorter one will suffice. The Berlekamp–Massey algorithm [37], [38] is a computationally efficient method for finding the minimal length linear feedback shift register which generates a sequence, and this test should be applied to ensure that the nonlinear output logic is adequate in this respect [35]. This test rules out some cryptographically weak sequences, but does not guarantee that there is not some other weakness. The sequence $000000\cdots0001$, for example, requires a shift register as long as the sequence, but is clearly very weak.

Some thought shows that the linear feedback shift register is acting primarily as a form of counter, and it is not clear that anything is lost by driving the nonlinear logic f with a normal counter instead. This point is treated further in Section IV.

Using nonlinearities in the feedback logic appears even more attractive, but the theory of such circuits is not well understood (at least in the open literature) and it is difficult to guarantee long periods, good distribution properties, etc.

I. IBM's Systems and DES

A block cipher is a cryptographic system which divides the plaintext into separate blocks, usually all of the same size, and operates on each independently to produce a sequence of ciphertext blocks.

The most general possible block cipher is one which can transform any possible plaintext block into any possible ciphertext block as long as the overall transformation is invertible. If the blocks are n bits long, there are 2^n distinct blocks and $2^n!$ ways in which this can be done. The number of bits of key necessary to select one of these ways is thus $\log_2 (2^n!)$ or approximately $n \times 2^n$. This number is prohibitively large even for fairly small values of n. For n equal to 8, it requires about 2000 bits of key and for n equal to 16 it requires roughly a million bits. A cipher of this type is a simple substitution cipher, and has been described earlier in this section. As shown there, simple substitutions on small alphabets can be cryptanalyzed efficiently by compiling frequency tables for the characters of the ciphertext, so the blocks must be of substantial size if the cipher is to be secure.

Because of the infeasibility of doing arbitrary substitutions on large blocks of data, cryptographers have sought restricted families of transformations. In 1929, Hill [39], proposed the use of linear transformations on blocks of text, but this system is too vulnerable to a known plaintext attack for serious use [20, ch. 4], and so more complex families of transformations have been developed.

Motivated by the growing need for data security in its products, IBM initiated a cryptographic research effort, concentrating on nonlinear block ciphers, in the late 1960's. This work has added greatly to the quality of the unclassified literature on cryptography [40]–[44] and has produced several important cryptosystems. In January 1977, one of these was adopted by the NBS as the national DES [7].

The IBM systems have their roots in Shannon's brilliant 1949 paper [10] connecting cryptography with information theory. Shannon suggested using product ciphers to build a

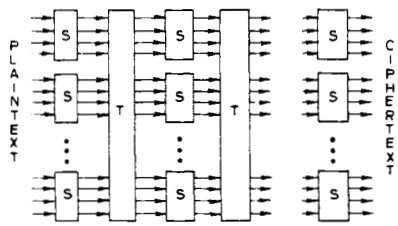


Fig. 10. A primitive system based on substitution and permutation.

strong system out of simple, individually weak, components. He suggested using products of the form $B_1MB_2M\cdots B_n$ where M is an unkeyed "mixing transformation" and the B_i are simple cryptographic transformations.

High speed electronic circuitry allows the product system to be implemented almost as economically as a single BM pair. The data are encrypted in a number of "rounds" (iterations) each consisting of a single pair B_iM and each using the same hardware. The output of the ith round becomes the input to the $(i+1)$th round. A high-speed terminal operating at 120 characters/s allows 67 ms for encryption of an eight character block. If each round involves 20 gate delays of 20 ns each, then over 150 000 rounds are possible. Tape transports, disk drives and similar high data rate devices place more of a demand on the circuitry and pipelined systems which replicate the circuitry n times may be needed. For this reason IBM has limited its systems to 16 rounds.

Although there are minor variations, all of the IBM systems basically use a transposition of the bits within a block for the fixed mixing transformation M, and substitution on four bit groups of the block for the simple cryptographic transformations B_i. Fig. 10 shows a primitive version of such a system described by Feistel [41]. The 128 bit plaintext block is broken into thirty-two 4-bit bytes, each of which is acted on by a different, invertible, 4-bit to 4-bit mapping or S (for substitution) box determined by a portion of the key. The resulting 128 bits are then scrambled by the fixed transposition T and input to the next round.[1] The transposition mixes bits from different S-boxes and is necessary to prevent the overall transformation from degenerating into a substitution on 4-bit blocks.

Deciphering is accomplished by running the data backward, using the inverse of each S-box.

The system as described above is difficult to implement because of the large key required (2^4 words/S-box $\times$ 4 bits/word $\times$ 32 S-boxes/round $\times$ 16 rounds = 32 768 bits of key), because of the restriction on the key (each of the words in a given S-box must be different, so a randomly generated key is not directly usable), and because it does not lend itself to repeated use of the same circuitry in low speed applications (the key for each round may as well be stored as a set of S-boxes).

To avoid these difficulties, Feistel [41] used only two different S-boxes, S_0 and S_1, which are publicly known. Each of the S-boxes shown in Fig. 10 is set to either S_0 or S_1 depending on one bit of the key. This reduces S-box memory requirements to 128 bits and the key to 512 bits. An even further reduction in key size, to 128 bits, was accomplished by use of a key expansion algorithm which repeats each bit of the key 4 times.

Any k-bit S-box can be implemented as a 2^k word memory with k-bit words. If the S-box is to be invertible then each of the words must be different. The simplest implementation of a 4-bit S-box, therefore, requires a $2^4 \times 4 = 64$-bit memory. An 8-bit S-box would require a $2^8 \times 8 = 2048$-bit memory, and a 16-bit S-box would require a 1 048 576 bit memory. IBM selected $k = 4$ as a compromise between cost and security.

If $k = 2$ had been chosen, the entire system would be solvable on a minicomputer in a few seconds' time. This is because all $4! = 24$ 2-bit to 2-bit invertible mappings from $x = (x_1, x_2)$ to $y = (y_1, y_2)$ are affine (linear plus a constant) and can be represented in the form

$$y = Ax \oplus b \tag{12}$$

where all arithmetic is done mod-2. Because transposition is a linear operation, and the composition of affine mappings is affine, the use of affine S-boxes would make the overall relation between plaintext p and the ciphertext c affine,

$$c = A_K p \oplus b_K. \tag{13}$$

While the key itself might be difficult to determine, cryptanalysis under a known plaintext attack easily produces the equivalent information A_K, A_K^{-1}, and b_K, by solving a set of linear equations. Enciphering via (13) or deciphering via

$$p = A_K^{-1}(c - b_K) \tag{14}$$

can then be accomplished. The linear equations are obtained from a set of $n+1$ $p-c$ pairs, $(p_0, c_0), (p_1, c_1), \cdots, (c_n, p_n)$ by letting

$$p_i' = p_i - p_0 \tag{15}$$

$$c_i' = c_i - c_0 \tag{16}$$

and noting that the (p_i', c_i') are related by the linear mapping

$$c_i' = A_K p_i'. \tag{17}$$

Letting

$$C = [c_1', c_2', \cdots, c_n'] \tag{18}$$

$$P = [p_1', p_2', \cdots, p_n'] \tag{19}$$

then

$$C = A_K P$$

and

$$A_K = CP^{-1}. \tag{20}$$

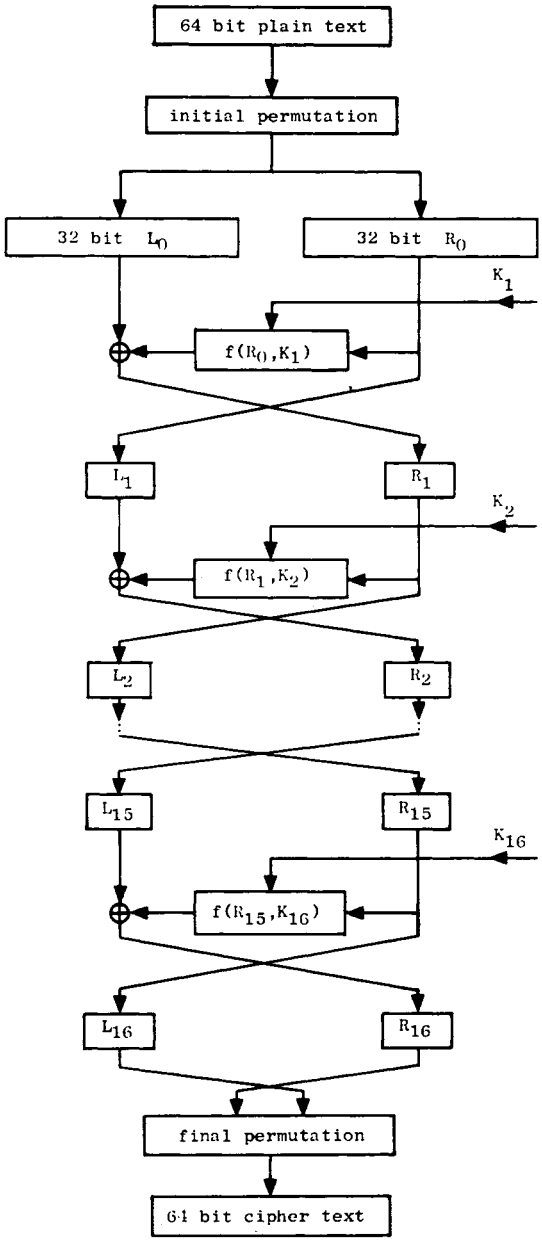
Then b_K can be obtained from

$$b_K = c_0 - A_K p_0. \tag{21}$$

With a 128-bit block size, approximately $(128)^3 \approx$2-million operations are required to compute A_K and b_K. A similar number of operations yields A_K^{-1}. Numerical stability is not a problem in the matrix inversion because there is no roundoff error in mod-2 arithmetic. Assuming a 1-μs instruction time results in 2 s for cryptanalysis when the S-boxes are affine.

The moral is clear: affine mappings are to be avoided. Mappings which are "close" to affine should also be avoided, and probably resulted in IBM's not using $k = 3$ even though only 3 percent of the invertible 3-bit-to-3-bit mappings are affine. Further study is needed to determine the minimum usable value of k. IBM apparently feels $k = 4$ is adequate. A larger

[1] The terms transposition and T-box have been used here in preference to the terms permutation and P-box used in the IBM literature. This is done to conform to classical cryptographic terminology and emphasize the fact that both S and T boxes represent permutations, on the alphabet and on the bits of the block, respectively. In describing DES, however, we conform to the terminology of the standard, referring to the transposition within the f function as P.



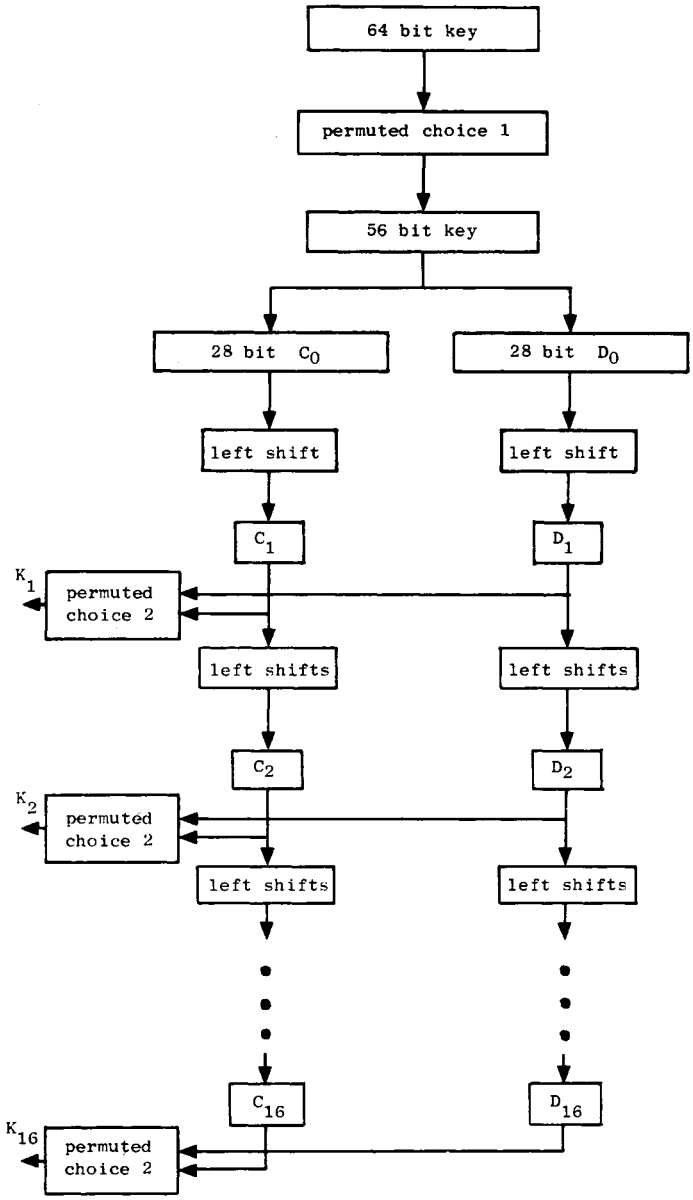


Fig. 11. Data encryption standard.

value of k would provide a greater margin of safety against unforeseen weaknesses, but may not be necessary.

The algorithm adopted as a national DES uses a different structure, but is still based on S and T boxes. The main difference is apparent in Fig. 11 which shows that after a fixed initial permutation (IP) the 64-bit plaintext block is broken into left and right-halves, L_0 and R_0, each 32 bits long. The algorithm then performs 16 rounds, each of the form

$$L_i = R_{i-1} \tag{22}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{23}$$

where $\oplus$ represents componentwise addition mod-2 (XOR), K_i is a 48 bit portion of the key used in round i, and f is a function with a 32-bit output.

Note that the function f need not be inverted during decryption, because $L_{i-1}R_{i-1}$ can be computed from L_iR_i as

$$R_{i-1} = L_i \tag{24}$$

$$L_{i-1} = R_i \oplus f(L_i, K_i) \tag{25}$$

even if the function $f(\cdot, K)$ is many-to-one.

Because the other operations are all linear in binary arithmetic, the function $f(R, K)$, diagrammed in Fig. 12, is the source of all security. The 32-bit R is expanded into a 48-bit R' by repeating bits numbered 1, 4, 5, 8, 9, 12, $\cdots$, 24, 27, 28, 32 (the edge bits of each successive 4-bit byte). The repeated bits are appended to the cyclically adjoining bytes to make eight 6-bit bytes

$$r_{32}r_1r_2r_3r_4r_5, r_4r_5r_6r_7r_8r_9, \cdots, r_{28}r_{29}r_{30}r_{31}r_{32}r_1. \tag{26}$$

The two 48-bit quantities R' and K are then XORed and broken into eight 6-bit bytes which are the inputs to eight different 6-bit-to-4-bit S-boxes ($S_1, S_2, \cdots, S_8$). The 32 output bits of the S-boxes are permuted by P and then taken as $f(R, K)$.

In order to obtain the $16 \times 48 = 768$ bits of K_1 through K_{16} from the 56 key bits, the key is loaded into two 28-bit shift registers, each with 24 taps. These 48 taps provide the current 48 key bits for use in $f(R, K)$. After each round the shift registers are cyclically left shifted to produce the next 48-bits of key. The locations of these taps, the S-boxes, and other details of DES are described in [7] and are not needed for the cursory analysis of this paper.

All of the operations involved in the algorithm, except the S-box mappings, are linear in binary arithmetic. It is therefore crucial that the S-boxes not be affine, or the overall algorithm would be affine of the form

$$c = Ap \oplus Bk \oplus b$$

where A, B, and b would be fixed and k is the 56-bit key. Knowledge of even one $p - c$ pair would allow k to be computed as

$$k = B^{-1}(c - Ap - b)$$

where B^{-1} is the pseudoinverse of B.

The authors, together with R. Merkle, R. Schroeppel, L. Washington, S. Pohlig, and P. Schweitzer performed a preliminary analysis of DES [45] during August 1976. This study found that none of DES's S-boxes are affine, but also found other structure, some of which tends to strengthen and some of which tends to weaken the algorithm. Here we describe only the most important points.

Each S-box has the property that changing any single input changes at least two outputs. This produces a large, rapid avalanche of changes if even one bit of the plaintext or key is changed, a property called **error propagation**, and helps foil a key cluster attack. In such an attack, the cryptanalyst first finds a key which is close to the correct key and then searches all keys in that neighborhood (a cluster of keys) until he finds the correct one. This is much more rapid than an exhaustive search. For example, if a test could be devised to tell when a key was correct in all but one or two positions, the number of keys tried in the first and second phases of the search would be about 10^{14} and 10^3, respectively, well below the $2^{56} \approx 10^{17}$ required for exhaustive search. If the cluster radius could be extended to 7 then each phase of the search would involve about 3×10^8 keys, making the algorithm useless. Error propagation is necessary, but not sufficient, to foil key cluster attacks.

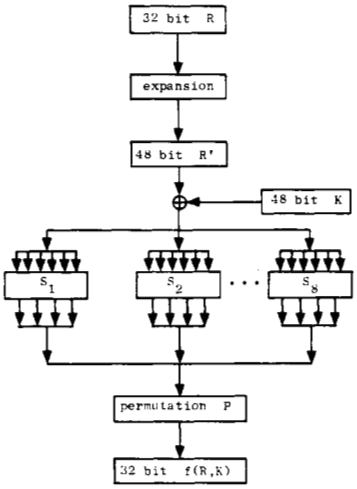


Fig. 12. $f(R, K)$ flowchart.

The expansion E and permutation P of Fig. 12 are coupled so that the four outputs from any S-box, after going through P and E, appear as inputs to 6 different S-boxes. This also encourages rapid error propagation and foils a clustering attack.

Turning to weaknesses, we found that DES possesses a symmetry under complementation which allows a 50-percent reduction in search effort under a chosen text attack. If S_K denotes enciphering with DES using key K then

$$C = S_K(P) \tag{27}$$

implies

$$\overline{C} = S_{\overline{K}}(\overline{P}). \tag{28}$$

where $\bar{x}$ denotes the bit-by-bit complement of x. This symmetry is a result of (22) and (23), and of the fact that

$$f(R, K) = f(ER \oplus K) \tag{29}$$

so f is invariant under complementation of R and K,

$$f(R, K) = f(\bar{R}, \bar{K}). \tag{30}$$

Cryptanalysis can exploit this symmetry if two plaintext-ciphertext pairs (P_1, C_1) and (P_2, C_2) are available with $P_1 = \overline{P_2}$ ($C_1 = \overline{C_2}$ would also suffice). Under a chosen text attack such pairs can always be obtained. Under a more normal attack, it may be that the plaintext sequence $010101 \cdots$ is sent during idle periods. Obtaining the two phases $0101 \cdots$ and $1010 \cdots$ would then be sufficient.

The search proceeds by enciphering P_1 under each of the 2^{55} keys which has a 0 as its least significant bit. If the resultant ciphertext $C \neq C_1$ then K, the key tried, is not correct. If $C \neq \overline{C_2}$, then $\bar{K}$ is not correct either. Two keys, K and $\bar{K}$, are effectively tried with only one encipherment. Since testing whether $C = C_2$ is much faster than an encipherment, the computational savings is very close to 50 percent.

We found no other weaknesses in DES which allowed us to further reduce the search effort. However, we did find a number of questionable quasi-linear structures which lead us

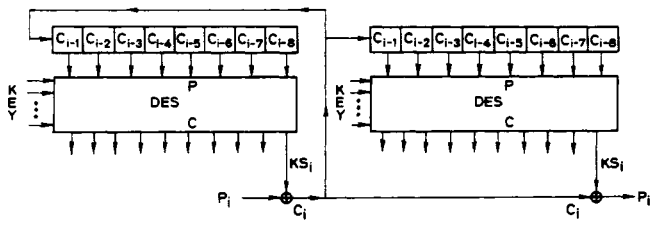


Fig. 13. DES in cipher feedback mode.

to believe that a more detailed analysis might be able to reduce the search effort substantially. The reader is referred to [45] for details of these other structures. NBS and NSA have since stated [46] that they are not aware of methods for exploiting the quasi-linearities.

Because there are 2^{56} or almost 10^{17} keys it might appear that even a substantial reduction in search effort would leave DES secure. For example, recovery of 10 bits of key would cut the search effort by a factor of 1024, to a search over 10^{14} keys, which is still a very large number. Unfortunately while 10^{14} and 10^{17} are large numbers, technology has progressed to the point where searches of this size are economically feasible. To facilitate exhaustive search, an LSI chip can be built [8] which can search one key every microsecond. By building a search machine with a million such chips, all searching in parallel, 10^{12} keys can be searched per second. The entire keyspace can then be searched in 10^5 seconds, which is about one day. The authors [8] have given a detailed justification for the feasibility of this machine, and estimate its cost to be $20-million and the average cost per solution to be $5000.

This estimate has not met with universal agreement, and there has been much controversy about the security of DES, particularly about the key size [47]-[50]. Although twenty million dollars is a large enough sum that, today, the machine could only be built covertly by a large intelligence organization, the authors feel that the steadily decreasing cost of computer hardware will reduce this figure to an unacceptably small $200 000 within the decade [51]-[54]. Second, there is no safety margin. If 10 bits of key can be recovered the prices drop to $20 000 for the machine and $5 for each solution.

A way to improve the security of DES through multiple encipherment is described in [8] and has also been suggested by IBM [55], but it would be preferable for the standard itself to be improved. If multiple encipherment, imperfect though it is, is to be used at all it must be written into the standard. No such action seems imminent. We predict that the standard will be improved in five to ten years and that part of that improvement will be to use a larger key. Equipment which uses DES should be designed with this possibility in mind.

In order to adapt the DES to applications which require the data to be enciphered character by character, or applications in which the data vary in length, NBS has suggested that DES can be operated in a mode called **cipher feedback**, as shown in Fig. 13. The ith plaintext character is XORed with a pseudorandom character which depends only on the key and the previous eight ciphertext characters, $C_{i-1} \cdots C_{i-8}$. This system is **self-synchronizing** and removes the need for special synchronization protocols since a transmission error or lost character causes only seven additional characters to be deciphered incorrectly, after which the error propagation ceases.

The number of bits fed back can be anywhere from 1 to 64. If cipher feedback is done on eight bit characters, the maximum encryption speed will be one eighth of the block mode speed. Similarly, if cipher feedback is done on a bit basis, the maximal speed will be one 64th of the block mode speed. On the other hand, if cipher feedback is done on whole words, there will be very little reduction in speed over block mode.

J. Analog Systems

When voice, facsimile, or other analog communications must be encrypted, this is frequently done using analog rather than digital operations on the signal, a technique called **scrambling**.

It is difficult, if not impossible, to build a truly secure analog scrambler which provides acceptable voice quality, delay etc. Speech that has been scrambled by a stationary process using frequency domain techniques alone or time domain techniques alone can usually be understood by a persistent human listener. Even the more complex types can be broken by using Fourier analysis to reconstruct the sequence of transpositions and inversions.

Brunner, Kirchhofer, and McCalmont [56]-[58] present more detailed treatments. One can summarize the situation by saying that analog scramblers are not currently able to achieve the high levels of security possible with digital cryptosystems, and there is some doubt they ever will. If voice or some other analog signal must be conveyed with a high level of security then it should be digitized (A/D converted) and digitally encrypted. Analog scramblers still find a place in the security business because of their lower cost and lower bandwidth requirements, and one reason for the military's push to obtain low bit rate (2400-bit/s) digitized speech [59] is to allow use of voice grade telephone lines for secure voice transmission. Dial up lines are limited to 2400-4800-bit/s data rates, with the higher rates requiring expensive modems. Reasonable quality 2400-bit/s speech digitizers are now available but are expensive (≈$10 000) and require low noise, high quality input speech. Their cost should fall markedly in larger quantity production.

The simplest analog scramblers are **frequency inverters** which merely invert the spectrum of the voice signal through a heterodyne operation. If a 300-2800 Hz filtered voice signal is mixed with a 3100-Hz tone a 2800-Hz tone is converted to a 300-Hz tone and vice versa. The resultant speech in unintelligible to a casual eavesdropper, but is easily descrambled with the same device. Because there are no keys, a "cryptanalyst" need only buy one of the scramblers to understand all that is said. More distressingly, people can learn to understand, and to an extent speak, inverted speech much as if it were a new language.

More complex **band splitting** scramblers break the speech into approximately five frequency bands and permute these. While 5! = 120 permutations are possible, many of these are almost equivalent, and even 120 keys would allow exhaustive search of the keyspace with a human listener deciding when the correct key is being used. Proper placement of even one of the 500-Hz bands is often sufficient for reproducing intelligible speech.

Rolling code scramblers change the permutation several times per second, and also frequency invert a changing subset of the 5 bands. The sequence of inversion subsets and permutations is determined by a pseudorandom number generator which uses the key as a seed. While the number of keys can be very large, these systems are still breakable. Because the

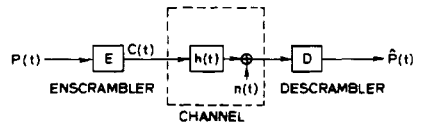


Fig. 14. An analog scrambler and link.

speech spectrum does not change instantaneously, it is possible to track a frequency band across a transition, thus determining how the new permutation is related to the previous permutation.

According to McCalmont [58], an even simpler approach is possible. Although there are $2^5 \times 5! = 3840$ distinct encoding transformations, these can be grouped into about five classes with one decoder for each class. While the decoder for a class will perfectly descramble only one of the encoding transformations in the class, it will partially descramble the others to the point where most of the meaning can be recovered. By summing the outputs of these five decoders and listening carefully, McCalmont says it is possible to extract about 60 to 80 percent of the meaning.

Time division scramblers break the speech into short segments and permute the segments within blocks, much as in a transposition cipher. Permuting both time and frequency segments is also possible and produces a **two-dimensional** scrambler. If the duration of each block is long enough (>10 s), and the duration of each segment short enough (<50 ms), time scrambling alone would seem to produce a highly secure system. Because the descrambled speech is delayed by twice the duration of a block, such long delays are impractical, and to the best of our knowledge none of the systems now in use is very secure.

Why are highly secure digital scramblers so much easier to design? Much of the answer lies in Fig. 14 which shows that transmission distortion and noise affect the scrambled signal $c(t)$ before it is descrambled. While E and D are inverse operations, they will not undo each other unless they commute with the channel. Further, unless D is a linear operation the effect of the additive noise may be disastrous.

Limitation to linear operations is as damaging to the security of any scrambler, whether analog or digital (Section III-H and III-I). Digital scramblers can largely neglect distortion and noise because on most digital channels these cause only very low bit error rates. Digital scramblers can therefore be highly nonlinear and are more easily made secure.

Aaron D. Wyner of Bell Labs has recently developed an analog scrambling approach (A. D. Wyner, "An analog scrambling scheme which does not expand bandwidth, part I," *IEEE Trans. Inform. Theory*, vol. IT-25, May 1979) which appears to overcome a number of these problems by using a time varying, distance preserving, linear operation (rotation) on an analog pulsetrain representation of the waveform. It uses optimal bandwidth efficient signals (discrete prolate spheroidal wave functions) to convey the scrambled pulseheights with complexity comparable to a high-speed telephone modem.

K. Public Key Distribution Systems

One public key distribution system [15] makes use of the apparent difficulty of computing logarithms over a finite (Galois) field GF(q) with a prime number q of elements (the numbers $\{0, 1, \cdots, q-1\}$ under arithmetic mod-q). Let

$$Y = \alpha^X \text{ mod-}q, \quad \text{for } 1 \leqslant X \leqslant q-1, \tag{31}$$

where α is a fixed primitive element of GF(q) (that is the powers of α range over the nonzero elements $1, 2, \cdots, q-1$ of GF(q)), then X is referred to as the logarithm of Y to the base α, over GF(q):

$$X = \log_\alpha Y \text{ over GF}(q), \quad \text{for } 1 \leqslant Y \leqslant q-1. \tag{32}$$

Calculation of Y from X is easy, taking at most $2 \times \log_2 q$ multiplications. For example

$$\alpha^{18} = (((\alpha^2)^2)^2)^2 * \alpha^2. \tag{33}$$

Computing X from Y, on the other hand can be much more difficult, and for certain carefully chosen values of q requires on the order of $q^{1/2}$ operations, using the best known algorithm [60].

Each user generates an independent random number X_i chosen uniformly from the set of integers $\{1, 2, \cdots, q-1\}$. He keeps X_i secret, but places

$$Y_i = \alpha^{X_i} \text{ mod-}q \tag{34}$$

in a public file with his name and address. When users i and j wish to communicate privately they use

$$K_{ij} = \alpha^{X_i X_j} \text{ mod-}q \tag{35}$$

as their key. User i computes K_{ij} by obtaining Y_j from the public file and letting

$$K_{ij} = Y_j^{X_i} \bmod q \tag{36}$$

$$= (\alpha^{X_j})^{X_i} \bmod q \tag{37}$$

$$= \alpha^{X_j X_i} = \alpha^{X_i X_j} \bmod q. \tag{38}$$

User j obtains K_{ij} in a similar fashion

$$K_{ij} = Y_i^{X_j} \bmod q. \tag{39}$$

Another user must compute K_{ij} from Y_i and Y_j, for example, by computing

$$K_{ij} = Y_i^{(\log_\alpha Y_j)} \bmod q. \tag{40}$$

Therefore, if logs over GF(q) are easily computed, the system can be broken. While no one currently has a proof of the converse (i.e., that the system is secure if logs over GF(q) are difficult to compute), neither has anyone come forth with a way to compute K_{ij} from Y_i and Y_j without first obtaining either X_i or X_j.

If q is a prime slightly less than 2^b, all quantities are representable as b bit numbers. Exponentiation then takes at most $2b$ multiplications over GF(q), while taking the logarithm requires $q^{1/2} = 2^{b/2}$ operations, using the best currently known algorithm. The cryptanalytic effort therefore grows exponentially relative to encryption or decryption. If $b = 200$, at most 400 multiplications are required to compute Y_i from X_i, or K_{ij} from Y_i and X_j, yet taking logs over GF(q) hopefully requires 2^{100} or approximately 10^{30} operations.

L. RSA Public Key Cryptosystem

Discrete exponentiation has been employed in a different way by Rivest, Shamir, and Adleman [17] to produce a public key cryptosystem. They make use of the fact that finding large (e.g., 100 digit) prime numbers is computationally easy, but that factoring the product of two such numbers appears to be computationally infeasible.

A user A selects two very large prime numbers, P and Q at random, and multiplies them together to obtain a number N. The number N is made public, but its factors, P and Q are kept secret. Using P and Q, A can compute the Euler totient function $\Phi(N)$ (the number of integers less than N and rela-

tively prime to N) as

$$\Phi(N) = (P - 1)(Q - 1). \tag{41}$$

He then chooses another number E at random from the interval 2 through $\Phi(N) - 1$. This number is also made public.

A message is then represented as a sequence of integers $M_1, M_2, \cdots$ with each M an integer between 0 and $N - 1$ (roughly 700 bits). Enciphering is carried out on each block M using the public information E and N, as

$$C = M^E \text{ mod-}N \tag{42}$$

where C represents the enciphered block.

Using the secret number $\Phi(N)$ user A can easily [61, vol. 2 p. 315 ex. 15] calculate a number D such that

$$ED = 1 \bmod \Phi(N). \tag{43}$$

(If E has a common factor with $\Phi(n)$ then D does not exist, but the algorithm will indicate this and another E can be chosen.) Equivalently, $ED = k\Phi(N) + 1$. Then, because

$$X^{k\Phi(N)+1} = X \text{ mod-}N \tag{44}$$

for all integers X between 0 and $N - 1$ and for all integers k, deciphering is easily accomplished by raising C to the Dth power

$$C^D = M^{ED} = M^{k\Phi(N)+1} = M \text{ mod-}N \tag{45}$$

As a very small example, suppose $P = 17$ and $Q = 31$ are chosen so that $N = PQ = 527$ and $\Phi(N) = (P - 1)(Q - 1) = 480$. If $E = 7$ is chosen then $D = 343$. ($7 \times 343 = 2401 = 5 \times 480 + 1$). If $M = 2$ then

$$\begin{aligned} C &= P^E \text{ mod-}N \\ &= 2^7 \text{ mod-}527 \\ &= 128. \end{aligned}$$

Note that only the public key (E, N) is needed to encipher M. To decipher, the private key D is needed to compute

$$\begin{aligned} M &= C^D \text{ mod-}N \\ &= 128^{343} \text{ mod-}527 \\ &= 128^{256}\ 128^{64}\ 128^{16}\ 128^{4}\ 128^{2}\ 128^{1} \text{ mod-}527 \\ &= 35\ 256\ 35\ 101\ 47\ 128 \text{ mod-}527 \\ &= 2 \bmod 527. \end{aligned}$$

Unless surprisingly large improvements are made in factoring, or methods of inverting M^E without calculating D are discovered, this system will remain secure. Simmons and Norris [62] analyzed a potential method for carrying out the latter approach, but found that its probability of success was too small to be of value. Rivest [63] has noted that a minor modification to the system removes even this small threat.

Note added in proof: Michael O. Rabin of the Hebrew University, Jerusalem, Israel, has shown that a variation of the RSA scheme in which $E = 2$ has cryptanalytic difficulty equivalent to factoring the modulus. (Michael O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," submitted to *Commun. ACM.*)

M. Trap Door Knapsacks

Merkle and Hellman [18] have devised a public key cryptosystem using a well-known problem in combinatorics known as the knapsack problem. Given a vector of integers $a = (a_1, a_2, \cdots, a_n)$ and an integer S, the knapsack problem is to find a subset of the $\{a_i\}$ such that the sum of the elements of the subset is equal to S. Equivalently, given a and S find a binary n-vector x such that $a \cdot x = S$.

The knapsack problem is believed to be extremely difficult in general, belonging to a class of problems that are thought not to be solvable in polynomial time on any deterministic computer. Some cases of the knapsack problem are quite simple, however, and Merkle and Hellman's technique is to start with a simple one and convert it into a more complex form.

The vector a can be used to encipher a message by dividing the message into n-bit blocks $x_1, x_2, \cdots$ and forming the dot products $S_i = a \cdot x_i$. The S_i form the ciphertext. Recovery of x_i from S_i involves solving a knapsack problem and is thus believed to be computationally infeasible if a and x are randomly chosen.

If the vector a is chosen so that each element is larger than the sum of the preceding elements, its knapsack problem is very simple. For example, if $a' = (171, 197, 459, 1191, 2410)$ and $S' = 3798$ than x_5 must equal 1. If it were 0 then even if $x_1 = x_2 = x_3 = x_4 = 1$, the dot product $a \cdot x$ would be too small. Then, knowing that $x_5 = 1$, $S' - a'_5 = 3797 - 2410 = 1387$ must be a sum of a subset of the first four elements of a. Because $1387 \geqslant a'_4 = 1191$, x_4 must equal 1. Finally $S' - a'_5 - a'_4 = 196 = a'_2$ so $x_3 = 0$, $x_2 = 1$, and $x_1 = 0$.

This simple knapsack vector a' cannot be used as a public enciphering key because anyone can easily recover x from S. The algorithm for generating public keys therefore generates a random simple knapsack vector a' (with several hundred components) and keeps a' secret. It also generates a random number m which is larger than $\Sigma a'_i$ and a random pair w, w^{-1} such that $ww^{-1} = 1$ mod-m. It then generates the public knapsack vector or enciphering key a by multiplying each component of a' by w mod-m

$$a = wa' \text{ mod-}m. \tag{46}$$

When another user wishes to send the message x to A he computes

$$S = a \cdot x \tag{47}$$

and sends this to A. A uses his secret information, w^{-1} and m, to compute

$$\begin{aligned} S' &= w^{-1}S \text{ mod-}m \\ &= w^{-1}\Sigma a_i x_i \text{ mod-}m \\ &= w^{-1}\Sigma(wa'_i \bmod m)x_i \text{ mod-}m \\ &= \Sigma(w^{-1}wa'_i \bmod m)x_i \text{ mod-}m \\ &= \Sigma a'_i x_i \text{ mod-}m \\ &= a' \cdot x \end{aligned}$$

because $m > \Sigma a'_i$.

For example, if the secret vector a' is as above, then $w = 2550$ and $m = 8443$, results in the public vector, $a = (5457, 4213, 5316, 6013, 7439)$, which hides the structure present in a'.

The vector a is published by the user as his public key, while the parameters w^{-1} and m are kept secret as his private key. They can be used to decipher any message which has been enciphered with his public key, by computing $S' = w^{-1}S$ mod-m and then solving the simple knapsack $S' = a' \cdot x$.

This process can be iterated to produce a sequence of vectors with seemingly more difficult knapsack problems by using transformations (w_1, m_1), (w_2, m_2), etc. The overall

transformation is not, in general, equivalent to any single (w, m) transformation.

The trap door knapsack system does not meet condition 1′ of Section II-E because most elements S of the cryptogram space, $0 \leqslant S \leqslant \Sigma a_i$, do not have inverse images x. This does not interfere with the use of the system for sending private messages, but requires special adaptation when used to produce signatures [18], [64].

IV. Cryptographic Taxonomy

A. Block and Stream Ciphers

Since the plaintext to be enciphered is of arbitrary length it cannot be handled all at once by a computing device of fixed size. As shown by the examples of the previous section it is usually divided into chunks (bits, characters, or words) as is the input of other translating devices such as compilers. We shall often refer to these chunks as characters, treating the set of possible chunks as an alphabet regardless of their size, but sometimes use the suggestive term **block** to designate a chunk from an alphabet with a large size.

Block ciphers divide the plaintext into blocks, usually of a fixed size, and operate on each block independently. A particular plaintext block will therefore be carried into the same ciphertext block each time it appears in the text. Block ciphers are therefore simple substitution ciphers and must have large alphabets to foil frequency analysis. The 64-bit block size of the DES for example, represents an alphabet of 2^{64} characters.

The properties desired in a block cipher are much the opposite of those desired in a systematic error correcting code. No bit of the plaintext should ever appear directly in the ciphertext. Rather, each bit of ciphertext should be an involved function of all bits of the plaintext and the key. The cipher should be designed so that changing even a single bit of either the plaintext or the key causes approximately 50% of the ciphertext bits to change. As explained in Section V-D, this **error propagation** is useful in authentication and makes it improbable that an opponent can make undetected modifications to encrypted data, unless he has learned the key.

Stream ciphers, in contrast, do not treat the incoming characters independently. Every character accepted as input is enciphered into an output character in a manner which depends on the internal state of the device. After each character is enciphered, the device changes state according to some rule. Two occurrences of the same plaintext character will usually not, therefore, result in the same ciphertext character. Stream ciphers may be further subdivided into synchronous and self-synchronizing systems.

In **synchronous** stream systems the next state depends only on the previous state and not on the input, so that the progression of states is independent of the sequence of characters received. Such a system is **memoryless**, but **time varying**. The output corresponding to a particular input depends only on the input and its position in the input sequence, not on the characters enciphered before it or after it. In consequence, there is no error propagation at all. Each character corrupted in transmission will result in precisely one erroneous deciphered character. The loss of a character, however, causes loss of synchronization, and all text following the loss will be decrypted incorrectly.

The Hagelin machine, feedback shift registers, and rotor machines are examples of synchronous devices. The 43rd character in a message will be enciphered (or deciphered) in a manner which is dependent on its being the 43rd character, but independent of what 42 characters preceded it.

Another frequently encountered class of stream systems is called **self-synchronizing** or **ciphertext autokey**, and is typified by the operation of the DES in cipher feedback mode Fig. 13. Systems of this sort are self-synchronizing, because the deciphering device has limited memory so that an erroneous or lost ciphertext character causes only a fixed number of errors in the deciphered plaintext, after which correct plaintext is again produced.

A third class of stream systems are those which are both time varying and possessed of memory. The encryption of each plaintext character thus depends on the position of the character and the characters which preceded it. Some rotor machines were placed in this class by the addition of **influence letters**, specially selected characters (part of the key) which caused the machine to stutter in its usual motion whenever these characters occurred in the plaintext or ciphertext.

The encryption process which is internal to the DES is also of this character. In each round, the left half of the register is encrypted in a fashion which depends on the right half from the previous round (the memory) and on the key schedule which varies from round to round (the time-varying nature).

B. Cryptosystems as Finite Automata

The behavior of cryptographic processes is best studied by viewing them as finite automata [30], [65]. Using this model is illuminating and suggests extensions of the basic block and stream structures.

A **finite automaton** consists of three sets: an input alphabet I, an output alphabet O, and a set of states S, together with two functions and an initial state s_0. The next state function NS maps the input and the current state into a new state

$$NS: I \times S \longrightarrow S \tag{48}$$

and the output function OP maps the input and the current state into an output in the alphabet O

$$OP: I \times S \longrightarrow O. \tag{49}$$

Thus if at time n the device is in state s_n and receives an input i_n, it responds by producing an output $o_n = OP(i_n, s_n)$ and going to a new state $s_{n+1} = NS(i_n, s_n)$. Although no formal restriction is placed on the input and output alphabets, we will usually consider alphabets whose elements are n bit blocks of data.

The enciphering device can be thought of as an automaton with plaintext as input and ciphertext as output. The key determines the next state and output functions. Similarly, the deciphering device is a key dependent automaton with ciphertext as input and plaintext as output.

Important classes of cryptographic systems can be identified by placing suitable restrictions on the output and next state functions. The first distinction is between systems with only one state and those with more than one state. (If an automaton has more than one state but the next state function always maps onto a single fixed value, we disregard the unused states and say that there is only one.) In this way, cryptographic systems are divided into block and stream systems much as error correcting codes are divided into block and convolutional codes.

If $NS(i, s)$ does not depend on the external input i, the automaton is **free running** or in the **autonomous** mode. The state then varies with time, but in a memoryless fashion, and the cryptographic system is a synchronous stream system. Otherwise, the automaton is said to be **driven**. All systems,

other than block or synchronous stream, are of this type, but there are possibilities other than self-synchronizing ciphers. Self-synchronizing ciphers are driven, but the deciphering automaton has a **finite memory** with respect to the influence of past characters. The automaton model shows that there are other interesting cryptographic possibilities.

C. Structure of Some Synchronous Systems

One of the most common techniques for building a synchronous cryptographic system is to generate a pseudorandom sequence of characters, dependent on the key, which is combined in some invertible way with the bits of the plaintext. This sequence, called the **keystream**, must be long enough not to repeat during the life of a key, and must have most of the characteristics of statistical randomness. The Hagelin machine (Section III-F) and some shift register systems (Section III-H) are of this form.

With a synchronous system, the known plaintext assumption is equivalent to giving the opponent access to the keystream. For the system to be secure it must therefore be impossible for him to derive any portion of the keystream which he has not seen from the portion which he has seen. This precludes his being able to derive the key from which the keystream was generated, since if he had the key he would be able to derive any portion of the keystream that he wished.

The period of the keystream must also be long. If it is short compared with the length of the message, a Kasiski solution is possible (Section III-D), and even if the period of the keystream is long compared to the length of a message, a solution is often possible.

If 100 messages of 200 characters each are enciphered with the Hagelin M-209 with the same key, there are only 20 000 characters in the overall "message" while the M-209's keystream has a period of over 101-million characters. But, if the initial positions of the six keywheels are chosen at random, as they often are, there is a significant probability that at least two messages will use some overlapping portion of the keystream. (The "birthday problem" [66] shows that if more than $n^{1/2}$ people choose independent random integers between 1 and n, there is significant probability that at least two people will pick a common number, and twenty thousand characters is approximately twice the square root of 101 million.) Such overlaps can be detected through use of statistical tests. The two ciphertexts can then be aligned and subtracted to cancel the keystream yielding an easily solved running key cipher (Section III-D).

Guaranteeing that the keystream is of adequate length requires constructing a finite automaton whose autonomous behavior produces a long sequence before repeating. Most of the theory of free running automata is unfortunately confined to those whose change of state function is linear (linear feedback shift registers), or at least affine (linear congruential generators, counters). Linear or affine automata produce sequences which are readily predicted and thus cryptanalytically weak (Section III-H). They can be combined with nonlinear elements, however, to produce systems of much greater strength.

The output sequence of one of the simplest automata, a counter, has maximal period, but lacks the other required properties. It is statistically anything but random, and from any portion of the output, an opponent can determine any other portion by inspection. To counteract this, the output can be put through a nonlinear output transformation as shown in Fig. 15. The automaton model shows that all

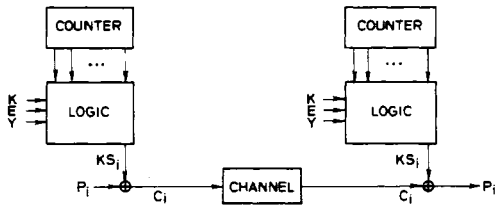


Fig. 15. Counter with nonlinear combinatorial output logic.

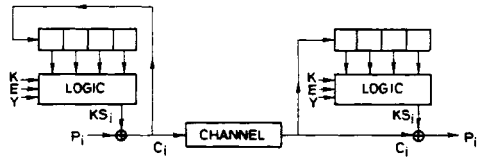


Fig. 16. Generic form of a self-synchronizing cipher.

synchronous systems can be thought of as being of this form, but the complexity of such an implementation may be excessive. Rotor machines with odometer motion are implemented directly in this form, and the Hagelin M-209 could be implemented this way too. (At time t, the wheel with n teeth is in position $i + t \bmod n$, where i is its initial position. It is therefore easy to compute the ith keystream character from the key and i, without computing the $i - 1$ intervening characters.) Algebraic study of linear feedback shift registers [67, ch. 6] shows that they also allow rapid calculation of the ith keystream character (bit), but rotor machines with complex motions and other complex systems (e.g., shift registers with varying feedback functions) cannot be efficiently implemented directly in the form of Fig. 15.

After a counter, the next simplest process for generating sequences of very long guaranteed period is the product of short cycles of relatively prime length. This is the basis of the Hagelin machine, where the short cycles were of lengths 17, 19, 21, 23, 25, and 26, and the resultant keystream cycle was of length $17 \times 19 \times 21 \times 23 \times 25 \times 26 = 101$ million. It is also used in the multiple Vigenere systems [24] of Section III-C.

More complex process with very long periods include linear feedback shift registers (Section III-H), and linear congruential random number generators [61, ch. 3]. Such known period processes may be made more complex by introducing **stutter**; detecting certain properties of the internal state and either pausing or skipping accordingly. A shift register can, for example, be tapped at some number of places independent of the feedback taps. A function computed on these tapped places then determines how many times the register will be cycled before its contents are again used as input to the output logic. This does not shorten the period unduly, and adds greatly to the complexity of the sequence.

D. Stream Systems derived from Block Systems

A secure block system can be used to construct either a synchronous or a self-synchronizing cipher system.

Just as Fig. 15 represents the general form of a synchronous stream system, Fig. 16 represents the general form of a self-synchronizing cipher. The finite memory of the decoder, and the associated limited error propagation and self-synchronization are seen clearly.

Comparing Fig. 16 with Fig. 13, shows that DES in cipher feedback mode has the form of Fig. 16, with DES's logic replacing the arbitrary Boolean logic of Fig. 16. Any block

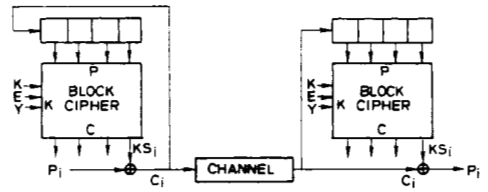


Fig. 17. A self-synchronizing cipher derived from a block cipher.

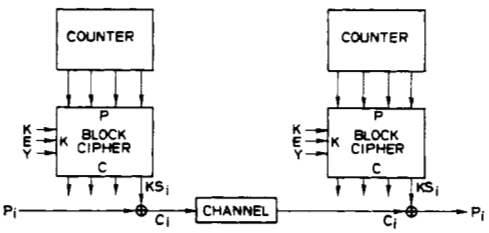


Fig. 18. A synchronous cipher derived from a block cipher.

cipher can be used to generate a self-synchronizing system by employing cipher feedback, as shown in Fig. 17, and it is natural to wonder about the relative security levels offered by the original block system and the derived cipher feedback system of Fig. 17.

If the block system is secure against recovery of the key under a chosen text attack, then the derived cipher feedback system is also secure against recovery of the key under a chosen text attack. A chosen text attack against a block system is equivalent to giving the cryptanalyst a cipher device loaded with a key which he cannot see. He can apply any input he wishes to either the plaintext or ciphertext port of the device and observe the resulting output at the other port. If the cipher feedback system of Fig. 17 were insecure and the cryptanalyst were trying to break the associated block system, he could simulate the cipher feedback system and use its solution for the key as the solution to the block system. In this sense, the cipher feedback system is at least as secure as the block system.

Fig. 17 can be modified in an obvious way to produce a synchronous keystream system from a block system, as shown in Fig. 18. The initial value assumed by the counter could be 0, part of the key, or a random value sent as an indicator (Section V-B). The initial load of the shift register in the cipher feedback system of Fig. 17 is not as useful as part of the key, because of the limited error propagation at the receiver. If only entirely correct messages were accepted by the receiver (losing the main motivation for self synchronization), the initial shift register load could be either the date and time (to foil the playback threat discussed in Section V-C), or part of the key.

The counters in Figs. 15 and 18 could be replaced with a feedback shift register (as suggested in Section III-H) or other generalized counter, but it is not clear whether any real advantage is gained. While a MLSRS has more bits changing per "count," error propagation in the block system should allow even single bit changes in the input to suffice.

The same argument used to demonstrate the security of a cipher feedback system derived from a block system can also be applied to show that if a block system is secure against recovery of the key in a chosen text attack, so is the derived synchronous system of Fig. 18. NBS has approved the cipher feedback mode of operation for DES, but has not mentioned the possibility of using it in synchronous fashion.

The above arguments cannot be viewed as a complete answer to the question of the security of the derived systems, because, in pathological cases, it may not be necessary to recover, the key to break them. In the synchronous case, for example, if successive keystream characters are derived as

$$KS_i = \alpha^{K+i} \bmod q \tag{50}$$

where α and q are public, then the keystream is easily extended using

$$KS_{i+j} = KS_i \alpha^j \bmod q \tag{51}$$

whereas computing the key K involves finding a discrete logarithm (Section III-K) and is a much more difficult problem.

V. Cryptography in Practice

A cryptographic system requires far more than a mathematical description of the effects of the encryption algorithm on the message. It must be realized in electronic hardware, packaged, tested for reliability, usability, and security, and maintained in the field. In addition, keys must be distributed to all installations, and cryptography must be integrated with other systems features such as error correction.

A. Key Management

Providing for the proper distribution of keys to the senders and receivers of enciphered messages is one of the major problems in building a working cryptographic communication system. Keys must be produced and distributed not once, but constantly. In some systems they must be changed with the passage of time, or with the amount of traffic, and in all systems they must be changed when they are feared compromised. Frequent key changes limit the amount of data compromised if an opponent does learn a key. Keys must be provided to new users of the system and old keys must be retired as users withdraw. The consideration of all these problems forms the subject of **key management**.

The classical military application of cryptography involves only a few independent 'sides' and a restricted flow of information. A colonel in one division does not need to be able to talk privately to a sergeant in another. Messages normally flow up and down the chain of command in a predictable pattern and are seen by a variety of intermediaries.

Cryptographic protection of a large, randomly accessed, communication network, like the telephone system, has entirely different problems. It would be hard to predict in advance which pairs of users would wish to talk privately on a given day and complete privacy from all other parties is often desired.

Resorting to the predistribution of keys to all potential user pairs by some physically secure channel such as a private courier or registered mail is clearly infeasible. A system with n subscribers has $(n^2 - n)/2$ such pairs, and requires almost 500-billion keys for a system with only a million users. Other solutions must, therefore, be sought.

A partial solution, known as **link encryption** is discussed further in the next section. Each user has only one key which he uses for communicating with a local network node. The message is decrypted and reencrypted as it passes through each successive node. The compromise of any node on the network, however, will cause the compromise of all messages passing through it.

In military networks where each node is a physically secure facility staffed by cleared personnel, link encryption is an attractive solution. It is much less attractive for a commercial network with a smaller budget whose nodes may be unmanned.

Some commercial networks even place nodes on their customer's property in cities where they have no facilities of their own. In such an arrangement, link encryption would provide little or no protection.

There is an alternative which permits the vulnerability to be concentrated in one place rather than distributed through the net [68]. Each user, instead of sharing a key with the local node, shares a 'master' key with a special network resource called the **key distribution center (KDC)**. Whenever user A wishes to converse with user B, he contacts the KDC which generates a key for this specific conversation. The KDC then sends this key to A, encrypted in A's master key, and to B, encrypted in B's master key, then withdraws, allowing A and B to communicate directly with their conversation specific key. The security of this conversation depends only on the security of the KDC, rather than on the security of each node through which messages between A and B must pass.

The security of this system can be improved substantially by the use of multiple KDC's [69]. Each user must have several different master keys, each of which he shares with a different KDC. When starting a conversation, A and B receive a conversation key, properly encrypted, from each of the KDC's, and combine all of these keys, for example by XORing them, to create the key they will actually use. For such a network to be compromised, all of the KDC's would have to be subverted.

A more satisfactory solution to the key distribution problem is given by public key cryptography as described in Section II-E and Sections III-K through III-M. Until public key systems are adequately certified and widely available, however, key distribution will continue to be a major problem of cryptography.

B. Indicators

Closely related to the problem of key management is the subject of **indicators**, data which are used to vary the enciphering and deciphering process from time to time or from message to message. An indicator is sent as part of a cryptogram and, combined with the key, tells the receiver how to decrypt the message. Indicators may be sent in clear or enciphered, and may either be placed at the beginning of a cryptogram or hidden within it. In any case, the legitimate receiver must know how to locate or decipher the indicator in order to decipher the rest of the message.

In using a stream cipher it is important to guarantee that no portion of the keystream is reused, since if this were to happen the cryptanalyst might be able to locate two ciphertexts C_1 and C_2 which had been enciphered with the same piece of keystream KS. For a binary stream cipher, he would then be in a position to form

$$C_1 \oplus C_2 = (P_1 \oplus KS) \oplus (P_2 \oplus KS) = P_1 \oplus P_2, \qquad (52)$$

which is a running key cipher with P_1 as the plaintext and P_2 as the key (or vice versa). This could then be solved by the methods described in Section III-D.

A similar concern applies to the M-209 Section III-F. If the keywheels were always started in the same position, for example, AAAAAA to encipher the first letter, the cryptanalyst would be able to read messages by solving a derived running key cipher. Instead, for each new message the operator chooses a random initial setting (e.g., XFRLAM) for the keywheels and sends this as an indicator. As long as the key is changed frequently enough, this will suffice, but if too much traffic is enciphered in any one key, then the birthday problem discussed in Section IV-C will lead to overlapping keying sequences.

For a self-synchronizing system, the indicator is the initial contents of the shift register. This is chosen at random by the transmitter and sent at the beginning of the cryptogram.

A form of indicator which IBM has used with the DES is the **session key**, a key which is used only for one "session" or conversation. A session key may either be supplied by a KDC (in this case it is synonomous with a conversation key) or it may be employed by a pair of users who already hold a "master" key in common. At the beginning of the session, the session key is arranged between the two parties by using messages enciphered in the "master" key. After this, all traffic during that session is enciphered in the session key.

C. Traffic Analysis and Playback

As described in Section II, the basic model for the application of cryptography consists of communications which are subject to interception or modification by an eavesdropper. If the eavesdropper can cryptanalyze the messages he receives and determine the key, he can obtain any information which flows in the communication system, or inject false information at will. If he cannot determine the key, he is much more restricted in his actions, but not totally stymied.

In some circumstances, the mere existence of traffic is important. The pattern of messages in a communication network can reveal the amount of business being transacted between different users, the hours of peak load, the flow of authority, etc. The determination of such information from traffic flow is called **traffic analysis**. If a traffic analyst observing a commercial network were to notice a sudden increase in traffic between a large conglomerate and a young but promising company, he would suspect that a takeover was in the planning.

Since it is often impractical to reduce traffic artificially in order to mislead a traffic analyst, the usual practice is to add additional traffic. This is called **padding** and the data added are called **pads** or **nulls**. If dummy traffic is sent whenever the channel is idle, the channel will appear constantly busy. This technique must be used with care, since the cost of a channel often depends on the amount of traffic it carries.

Even if the eavesdropper is unable to cryptanalyze the system, he can record properly encrypted messages and play them back later. While he cannot be certain of the results, he hopes this will sabotage the system. If, for example, a telephone authorization for electronic funds transfer were recorded and then played back several times, it would wreak havoc with the account being debited. This threat of **playback** shows that authentication procedures must establish not only the origin, but the timeliness of messages.

One way to guarantee timeliness is to employ a synchronous system driven by an external clock, but this requires absolute clock synchronization, and is inconvenient for many purposes. Another is to include the date and time with each message. In the latter case, authentication must tie the message together so that an opponent cannot alter a current message by substituting a part of an older message into it.

Messages encrypted with synchronous stream ciphers or cipher feedback systems are tied together in this way. With block ciphers, however, explicit steps must be taken to foil playback, either by setting aside information in each block which connects it with the adjacent blocks, or by introducing time variance into the system. The first technique is typified

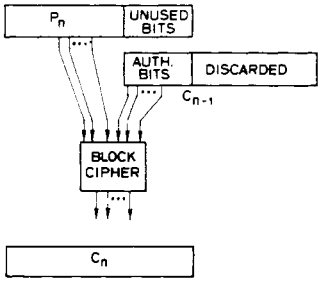


Fig. 19. Block chaining.

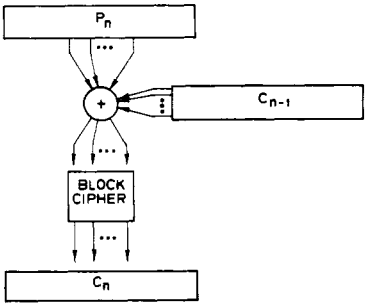


Fig. 20. Cipher block chaining.

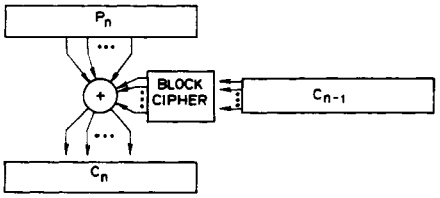


Fig. 21. Cipher feedback on whole blocks.

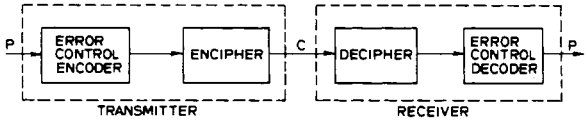


Fig. 22. Internal error control.

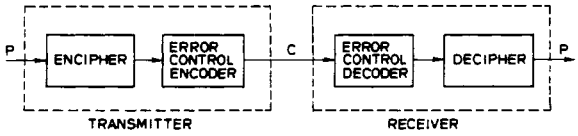


Fig. 23. External error control.

by **block chaining**, which was described by Feistel [41]. The first block contains the date and time and, as shown in Fig. 19, a portion of each successive data block is given over to authentication information, which connects it to the previous block. Each plaintext block contains in addition to data, a number of bits of the previous ciphertext block. To be considered valid, a deciphered block must agree with the previous block. If a 20-bit authentication field is included, the chances are only one in a million that a block will mistakenly be accepted if it occurs out of place [41].

Unless the blocks are very large, this technique is inefficient, since at least 20 to 30 bits in each block must be set aside for authentication. In order to overcome this inefficiency, a technique known as **cipher block chaining** has been proposed for use with the DES [70]. Cipher block chaining is very similar to a cipher feedback mode in which the whole block is fed back every time (Fig. 21). As shown in Fig. 20, each block of the message is XORed with the previous ciphertext block and then enciphered prior to transmission.

D. Error Control

Error control codes are used in digital communications either to detect or correct transmission errors [31], [71]. Such codes are usually publicly known and offer no cryptographic protection.

When cryptography and error control coding are used together, either operation can be performed first, but with different results.

If, as in Fig. 22, the error control coding is performed first then, at the receiver, decryption is preformed first. With a block cipher, this method has the advantage of allowing automatic authentication of received messages because an opponent does not know how to generate a cryptogram which, when decrypted, will possess valid error control bits (even though he does know what the modified error control bits should be). If instead the error control code is the outer code, as in Fig. 23, an opponent can inject messages which will pass the error control decoder. Although he cannot know into what messages they will be deciphered, he can still hope to create confusion and disrupt the system. Automatic authentication therefore requires internal error control.

If error correction, rather than just error detection, is desired then external error control must be used. If internal error control is used, error propagation in the deciphering operation introduces too many errors for the error control code to correct. If both automatic authentication and error correction are needed, two error control codes can be used, one before and one after encryption.

Some thought shows that error expansion in decryption, which prevents error correction with internal error control, is necessary for automatic authentication. A synchronous stream cipher has no error expansion and so provides no authentication when used with a fixed linear error control code, since an opponent knows which error control bits to change when he changes information bits in the message. (The use of a keyed or nonlinear error control code could prevent this.) Conversely, error correction can be implemented either internally or externally with a synchronous stream cipher.

Because error detection with retransmission of erroneous messages is usually more efficient than error correction and because automatic authentication is needed in most applications, we expect internal error control, shown in Fig. 22, to predominate. The advantage given to the cryptanalyst by his knowledge of the redundant error control bits would make this scheme risky for use with any cipher which was not secure against a known plaintext attack [18], but should be of little concern today.

E. Operation and Maintenance

Reliability and ease of operation are crucial to the security of cryptographic systems. Many systems that are secure when operated properly have been penetrated because of either mechanical failure or operator error.

Consider, for example, a rotor machine with three rotors (Section III-G). Since the arrangement of the rotors in the machine is typically part of the key, the operator must select the right rotors and place them in the correct positions in the machine. It is easy for him to make a mistake in this process

and insert one rotor correctly, but interchange the other two, resulting in an incorrectly encrypted message which cannot be decrypted properly by the receiving station. When the receiver sends back a message saying so, the operator has the opportunity to make a very serious mistake. He can correct the error in rotor position and retransmit the message. If he does so, he will put a pair of cryptograms into the hands of his opponents whose underlying plaintexts are identical and whose keys differ only by an interchange of rotors. Furthermore, the opponents may recognize that this is what has happened from their observation of the traffic pattern and their knowledge of the machine and the errors to which its operation is prone. An error in the operation of a system which allows a cryptanalytic break is called a **bust**.

To avoid such errors, the operator is instructed to encipher, but not transmit, a test message, such as 'AAAAAAAAAA . . .', prior to enciphering any real traffic. Some portion of the resulting cryptogram is then compared with a correctly enciphered version included with the keying material. This technique is called a **letter check**. A 26 letter check was used with the Hagelin machine and a 36 through 48 letter check was used with some rotor machines.

Returning to a three rotor machine, there is another possibility that may weaken it substantially: a failure of the mechanism which moves one of the rotors could reduce the effective number of rotors to two. This indicates that mechanical failure can be as disastrous as operator error.

In modern systems human failures are minimized by automating encryption and making it as transparent to the user as possible. The details of key setup, selection of indicators etc. are handled by the machine.

In order to achieve high levels of reliability in modern cryptographic equipment, encryption is often done in duplicate. Only if the outputs of two independent encryption devices are identical will the cryptogram be transmitted. This is coupled with a careful analysis of the effects of failures, to guarantee that no single failure can result in loss of system security. If a failure is detected, the equipment shuts itself off to prevent insecure transmissions.

F. Integration with Other Security Measures

To offer a high degree of security, cryptography must be employed in coordination with other security measures. Effective communication security requires that an opponent be prevented from bypassing the cryptanalytic problem by either intercepting unintended leaks of radiation, or physically violating the cryptographic equipment. In addition, the use of data by authorized personnel must be monitored.

The output of an electronic encryption device is a sequence of pulses which represent abstract zeroes and ones. In any electronic equipment such pulses must conform to certain standards so they can be recognized by other parts of the system; in cryptography, there is the additional requirement that pulses representing the same value must be as uniform as possible. The output of an imperfect XOR gate, for example, may allow an opponent to distinguish not just two types of signals, but four: those corresponding to $0 \oplus 0$, $0 \oplus 1$, $1 \oplus 0$, and $1 \oplus 1$. Although $0 \oplus 1$ should produce the same output as $1 \oplus 0$, if the two waveforms can be distinguished, even a one time tape system using this gate could be broken by an opponent who carefully examined the waveform representing the ciphertext. Even this extreme case has occurred [23, p. 714], and substantially more subtle failures are possible.

Cryptographic equipment, like any other computing equipment is prone to radiation leakage resulting from the short pulses employed. Since radiation leaking from a cryptographic device can be picked up and analyzed for useful information, it is necessary to shield the equipment very carefully against both conductive and radiative losses. Techniques for preventing such losses are known in the military by the term TEMPEST.

The first step in the shielding process is a careful division of the device into red and black regions, that is secure regions in which plaintext is present and insecure regions in which only ciphertext is present. The design must guarantee that all data paths between red and black are known, and that these data paths are protected, (for example with optical isolators) to insure that only the intended data travel along them.

An opponent may also mount an active tempest attack by transmitting high energy radiation at a communications node, while monitoring its output, in hopes of gaining information about its internal state.

Similar attention must be paid to leakage of sound, light, or any other potential carrier of information. Sound is a less serious problem with purely electronic machines than with electromechanical ones. Rotor and Hagelin machines, for example, are both susceptible to analysis of their motions when the sound of their operations is overheard.

If cryptographic equipment is to be placed in an unguarded area, it must be packaged in a container that protects it against physical tempering. Such a container must be made resistant to surreptitious entry, without interfering with either cooling or shielding requirements. In order to prevent the surreptitious extraction of the key, devices are included that reset the key to zero if the container is opened or if tampering is detected.

Physical and electronic security measures for use with DES are the subject of a new federal standard [72] which is still under development.

A cryptosystem is of no value if a person, authorized to have access to plaintext documents, sells or otherwise divulges them. Personnel security is the obvious first line of defense and audit trails [73], which record every access to sensitive information, provide a valuable second line. Authorized personnel hesitate to access information not needed for their assigned tasks for fear they will be discovered during an audit. If every access of an income tax return were recorded it is probable that the illegal uses of this information brought out during the Watergate investigations would not have occurred. In addition, a computer can be programmed to look for unusual activity on the part of a user or of a file, and to print out warnings.

G. Certification

Certification of security systems has always been fraught with unseen hazards. Designers of systems have often overlooked weaknesses, which were later found by opponents. The success of the opponent can result from a larger budget, a cleverer staff, better luck, or a fresh point of view.

We expect that provably secure systems will be developed as computer science progresses, but until that time, the current process of certification by mock attack will remain the most reliable test of a system's strength.

Certificational attacks should give the attackers a much greater advantage than is expected in practice. This provides both a conservative measure of security and an economic lever. That is, if a small-scale certificational attack is not successful under extremely favorable circumstances, then a larger scale attack by an opponent will probably not succeed

if carried out under more usual, less favorable, conditions. Use of the chosen text attack in certification is one method of providing such leverage. Another is to assume the attackers know some of the key as well.

Contrary to common practice, it is desirable for the design, including any underlying principles, to be published. An attempt to keep the design secret is likely to guarantee only that the designers will be unaware if the system is broken by opponents who have obtained it surreptitiously. If the design is published, especially if a reward is offered for its solution, it is much more likely that the manufacturer will learn if it is broken. Systems with interesting mathematical structure have a built in reward, since the solution is publishable and will bring prestige to its discoverer.

If a system is in use by many organizations, it may even be economical to offer a larger reward than any one solution would produce through illegal use. If a thousand organizations were to offer a thousand dollars each in a pooled reward, each would carry only a small risk and yet could feel safe protecting a million dollars worth of information, because an opponent rarely knows how to use a solution against more than one target.

VI. Applications of Cryptography

A. Timesharing Systems

Most of the study of timesharing system security has been devoted to noncryptographic techniques [74]–[76]. There are, however, several ways in which cryptography can augment other protection measures in a timesharing system, particularly in the areas of data storage and authentication.

In the usual authentication procedure for computer login, the computer demands a password from the user and compares the response with an internal password table. It is vital that the internal table be extremely well protected since anyone who gains access to its contents can impersonate any user perfectly.

This threat can be countered by using a **oneway function** f, a function which is easy to compute in one direction, but computationally infeasible to compute in the other [15], [77], [78]. The passwords are "encrypted" and only the images of the passwords, under f, rather than the passwords themselves, are stored in the password table. The system can now judge the validity of a login request by operating with the function f on the password given by the user before comparing it with the table. An opponent who steals the password directory cannot use this information to impersonate other users, because the image of a password is not a password, and inverting f to find the password is computationally infeasible. For a function to be truly oneway, it must have a large domain, so this technique is secure only with large passwords. It is the most widespread computer security application of cryptography to date.

An obvious application of cryptography to computers is the protection of files by encryption. This presents many problems and, although many timesharing systems have cryptographic programs available, no system known to the authors offers a satisfactory implementation. The proper integration of file encryption into system functioning will probably await the advent of machines with built in enciphering hardware.

A virtue of cryptographic file protection is the degree to which it frees the user from the need to trust the system managers. Noncryptographic protection measures can always be bypassed by the system authorities. With cryptographic protection, however, the system need not know the user's key. To gain access to an encrypted file, without resorting to cryptanalysis, the system programmers would have to place a tap in the system which could copy the plaintext file while it was being processed. This action would be more difficult, more likely to be observed, and thus less likely to succeed.

Encryption also frees the system managers from any moral dilemma associated with surrendering a user's data in response to a subpoena. They may surrender encrypted data without fear of compromising its owner, and thereby protect themselves from citation for contempt. Any effective legal demand for the data must, as it properly should, be levied against the data's owner.

Cryptography cannot replace standard file protection measures but, properly integrated into system functioning, it can greatly increase the overall security of the system. A file is most vulnerable during the long periods when it resides idly on disk or tape. If the file is encrypted at such times, an opponent can only compromise it during the brief periods when it must be converted to plaintext for processing.

In order for encrypted files to be processed conveniently, the cryptographic system must be integrated directly into the file access mechanism in such a way that excessive encryption and decryption times are avoided. On read access, excessive decryption may seriously slow down system response. When a file is modified, however, a delicate balance must be maintained between the inefficiency of reencrypting the entire file and the danger that the opponent will learn which part of the file has been changed, even though the precise change may be unknown.

Another major application of cryptography to computer security lies in the protection of secondary storage, such as tapes and disk packs. Encrypting the contents of such portable memory devices, protects them from being read on drives other than those on which they were written. In this way their security may be brought under the control of the system and an opponent prevented from bypassing system security by means of physical theft. This also lowers the cost of providing backup against physical destruction. Encrypted backup tapes may be prepared in duplicate and stored at remote locations without being guarded, while each copy of an unencrypted tape must be guarded at considerable expense.

Ideally, each disk or tape drive would be equipped with its own cryptographic device and its own key. This key is unique and is neither transmitted nor shared with other devices. The encryption is completely transparent: all data stored on the tape or disk is automatically encrypted in this key, and all data read is automatically decrypted. The physical theft of a tape would not, therefore, entail the theft of the information it contained.

B. Communication Cryptography

Communication between the computer and a remote user is currently one of the most vulnerable aspects of a computer system. In order to secure this, cryptographic equipment must be built into the user terminal, and suitable protocols developed to allow the computer and the user to recognize each other upon initial contact and maintain continued assurance of each other's identity [41], [79].

Password logins are vulnerable to eavesdropping, even when a one-way function is used to protect the password table. Therefore, when the computer answers the user's login request, it should initiate the conversation by sending the user a challenge which is guaranteed never to repeat, for example the date and time. The terminal receives the challenge and en-

crypts it under a key supplied by the user. The computer also knows this key and can authenticate the user's identity by comparing this response with a correctly encrypted version of the challenge. Because the key itself is never sent, an eavesdropper must cryptanalyze the system in order to impersonate a user. Once the user's identity has been established, the user's specific key can be used to encrypt all further transmissions.

Because the user's login request is sent to the system in clear, a traffic analyst can observe the user's identity. It is possible to avoid any exchange in clear at the expense of greater login overhead. When the computer receives the encrypted challenge, it decrypts the response under each of the keys in the user key table until one is found which produces the original challenge as plaintext.

This protocol allows the user to convey his identity to the computer without ever having to transmit it in clear. A traffic analyst observing the interchange would probably recognize it as an initial connection, but would be unable to learn the identity of the user.

This **challenge and response** procedure may be combined with the use of one-way functions to offer both protection against eavesdropping and against the possibility that compromise of system passwords will permit forged logins.

Both problems can be solved at once by the use of public key cryptography. The user transmits a login request containing both his name and the date and time, encrypted first in his own private key and then in the system's public key. The system first decrypts this using its own private key then, acting on the basis of an internal header giving the user's name, transforms it using the user's public key to check its authenticity. The activities of traffic analysts are foiled, because the header is encrypted in the system's public key. Safety against compromise of system password tables is also provided because there are no secret tables of user passwords in the system, only public tables of users' public keys.

Cryptography is even more important in teleprocessing networks than in timesharing systems. The most important data paths in a timesharing system are centralized and can therefore be physically protected. In a geographically distributed network, by contrast, data vital to system functioning must be transmitted over communication links whose physical protection would be uneconomical. For such a network to function securely, therefore, these links must be protected cryptographically.

There are two basic ways in which cryptography can be applied to networks, depending on whether encryption is regarded as a responsibility of the net or a responsibility of the users. In **link encryption** a message passed along the net is decrypted and reencrypted at each node through which it passes. This allows all data flowing on the links, even the address information, to be encrypted. Since the data are decrypted at each node, they are available in clear for the node to decide where to forward them.

An alternative to link encryption is **end-to-end encryption**, in which a message is encrypted at its source and decrypted only at its final destination. This has the advantage that the data are protected throughout their entire journey and will not be compromised if a node has been subverted. End-to-end encryption by itself, however, does not permit message addresses to be encrypted in a store and forward network, as each node through which the message passes must have access to the address in order to decide how to forward it.

End-to-end encryption using conventional cryptosystems also has the disadvantage of requiring a key to be exchanged securely between each pair of users who wish to communicate. With link encryption, each user needs only one key for communicating with his local node.

VII. Selected Bibliography

The following bibliography is intended to provide the interested reader with a guide to the open cryptographic literature. Unfortunately, although this literature is extensive, it is fragmentary, frequently hard to obtain, and highly variable in quality. We have listed an assortment of both the best cryptographic work and some of the more readily available.

The cryptographic literature has recently been enriched by the appearance of both a new press

Aegean Park Press
P.O. Box 2837
Laguna Hills, CA 92653

and a new journal

Cryptologia
Albion College
Albion, MI 49224.

The Aegean Park Press has concentrated on reprinting classic works in cryptography, which has made out of print works available again. *Cryptologia* is a journal devoted to all aspects of cryptography. It has published articles ranging from history and biography to technical details of public key cryptography.

The bibliography has been divided into sections by subject, and the subjects have been arranged alphabetically. Items which straddled more than one category have been placed according to their most central themes. In some places this has been a close decision, and the reader interested in applications will want to consult some of the papers on system implementation and vice versa.

A. Analog Scrambling

E. R. Brunner, "Efficient speech scrambling: An economic solution to the secure voice communication problem," in *Rec. Int. Conf. Communications Equipment and Systems* (Brighton, Sussex, London, England, pp. 336–339, June 8–11, 1976.

Kirk H. Kirchhofer, "Secure voice communication—Cryptophony," *Int. Defense Rev.*, vol. 9, pp. 761–767, May 1976.

A. M. McCalmont, "Communication security for voice-techniques, systems, and operations," *Telecommun.*, pp. 35–42, Apr. 1973.

A. M. McCalmont, "How to select and apply various voice scrambling techniques," *Commun. News*, pp. 34–37, Jan. 1974.

Aaron D. Wyner, "An analog scrambling scheme which does not expand bandwidth, part I," *IEEE Trans. Inform. Theory*, vol. IT-25, May 1979.
A promising new approach to analog scrambling using n-space rotations of vectors of sample points as the keying elements, and discrete prolate spheroidal wave functions to encode the signal in an optimally bandwidth efficient manner.

B. Applications

Paul Baran, "On distributed communications: IX. Security, secrecy, and tamper-free considerations," The Rand Corporation, Santa Monica, CA, RM-3765-PR, Aug. 1964. (Reprinted in Lance J. Hoffman, Ed. *Security and Privacy in Computer Systems*. Los Angeles, CA: Melville Publ., 1973, pp. 99–123.)
This is a fine discussion of the application of stream cryptography to computer networks. It is part of a series of papers which laid the foundations for packetized computer networks.

D. K. Branstad, "Security aspects of computer networks," in *Rec. AIAA Computer Network Systems Conf.* (Huntsville, AL), AIAA Paper no. 73–427, Apr. 16–18, 1973.

Dennis Branstad, "Encryption protection in computer data communications," in *Proc. 4th Data Communications Symp.* (Quebec City, Canada), Oct. 7–9, 1975.

Ehud Gudes, Harvey S. Koch, and Fred A. Stahl, "The application of cryptography for data base security," in *Proc. Nat. Computer Conf.* (New York, June 7-10, 1976), pp. 97-107, 1976.

Steven T. Kent, "Encryption-based protection protocols for interactive user-computer communication," M.I.T. Lab. Computer Science, Tech. Rep. 162, May 1976.

This thesis developes protocols for the use of block cipher systems in protecting and authenticating messages sent over networks with asynchronous and full duplex line protocols.

Richard R. Keys and Eric H. Clamons, "Security architecture using encryption," in *Approaches to Privacy and Security in Computer Systems* (Proceedings of a conference held at the National Bureau of Standards), pp. 37-41, Mar. 4-5, 1974.

Richard R. Keys and Eric H. Clamons, "File encryption as a security tool," *Honeywell Comput. J.*, vol. 8, no. 2, pp. 90-93, 1974.

"Design alternatives for computer network security," National Bureau of Standards, Special Publication 500-21, vol. 1, Jan. 1978.

"The network security center: A system level approach to computer security," National Bureau of Standards, Special Publication 500-21, vol. Jan. 1978.

Pierre E. Schmid, "Review of ciphering methods to achieve communication security in data transmission networks," in *Proc. Int. Zurich Seminar on Digital Communications*, Mar. 11, 1976.

J. L. Smith, W. A. Notz, and P. R. Osseck, "An experimental application of cryptography to a remotely accessed data system," in *Proc. ACM Nat. Conf.* (Boston, MA), pp. 282-297, Aug. 14-16, 1972.

A description of the construction of an experimental system using Lucifer to communicate with an IBM 360/67 timesharing system.

C. Authentication

Horst Feistel, "A survey of problems and systems in authenticated communication and control," M.I.T. Lincoln Lab., May 20, 1958.

Roger M. Needham and Michael D. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. ACM*, vol. 21, pp. 993-999, Dec. 1978.

Walter R. Widmer, "Message authentication, a special identification requirement in one-way digital data transmission," in *Proc. Int. Zurich Seminar on Digital Communications*, Mar. 11, 1976.

D. Bibliographies

John Arthur Scherf, "Computer and data security: A comprehensive annotated bibliography," M.I.T. Project MAC, Tech. Rep. 122, Jan. 1974.

This bibliography is less comprehensive on cryptography than the present one, but its extensive coverage of other aspects of computer security will be of use to those interested in applications of cryptography.

David Schulman, *An Annotated Bibliography of Cryptography*. New York: Garland Publishing, 1976.

This bibliography has almost no overlap with the present one. It offers the most extensive coverage of the classical literature available.

E. Classical

Wayne G. Barker, *Cryptanalysis of the Hagelin Cryptograph*. Laguna Hills, CA: Aegean Park Press, 1977.

Charles Eyraud, *Precis de Cryptographie Moderne*, Paris, France: Editions Raoul Tari, 1st ed., 1953; 2nd ed., 1959.

This is the most recent major public treatise on cryptanalysis, and as such touches on some relatively modern devices such as the Hagelin machine which are omitted in earlier works.

William Frederick Friedman, *Military Cryptanalysis*, Washington, DC: U.S. Government Printing Office, 1944. 4 volumes, I—Monoalphabetic Substitution Systems. II—Simpler Varieties of Polyalphabetic Substitution Systems, III—Aperiodic Substitutions, IV—Transposition Systems.

This is the most detailed course in cryptography which has slipped into public hands, and one of the most readable. Although out of print, it is available in several major libraries.

Helen F. Gaines, *Cryptanalysis, A Study of Ciphers and Their Solution*. New York: Dover, 1956.

This is the most readily available work on cryptanalysis, and gives extensive coverage of classical systems.

Solomon Kullback, *Statistical Methods in Cryptanalysis*. Laguna Hills, CA: Aegean Park Press, 1976.

This reprint of a 1938 paper is an extensive exposition of the use of correlation in cryptanalysis.

Hans Rohrbach, "Mathematische und maschinelle Methoden biem chiffrieren und dechiffrieren," *Fiat Review of German Science, 1939-1946 Applied Mathematics*, Part I, Article IX. Office of Military Government for Germany, Field Information Agencies Technical, Wiesbaden, Germany, pp. 233-257, 1948. (Translated by B. Hardie in two parts, *Cryptologia*, vol. 2, pp. 20-37, Jan. 1978 and *Cryptologia*, vol. 2, pp. 101-121, Apr. 1978.)

This paper describes some of the systems attacked by the German cryptanalysts and discusses their successes and failures.

Luigi Sacco, *Manuale di crittografia, 2e edition riveduta e aumenta*, Rome, Italy 1936. (Translations, French—*Manuel de Cryptographie*, from 3rd Italian edition by J. Bres, reviewed by the author with a preface by R. Leger, Paris, France: Payot, 1951; English—*Manual of Cryptography*. Laguna Hills, CA: Aegean Park Press, 1977.)

This work is noted for being particularly readable. It is similar in scope to Eyraud or Friedman, but slightly earlier in date.

Abraham Sinkov, *Elementary Cryptanalysis, A Mathematical Approach*. New York: Random House, New Mathematical Library no. 22, 1968. (Now published by the Mathematical Association of America).

This is the best elementary cryptanalysis book available.

F. Cryptanalysis

Don Coppersmith and Edna Grossman, "Generators for certain alternating groups with applications to cryptography," *SIAM J. Appl. Mathe.*, vol. 29, pp. 624-627, Dec. 1975.

This paper examines the family of permutations generated by schemes which are in some ways a generalization and in other ways a specialization of the method used in the IBM Lucifer family of ciphers.

Daniel James Edwards, "OCAS—On-Line Cryptanalytic Aid System" M.I.T. Project MAC Tech. Rep. 27, May 1966.

A primitive system for computer aided cryptanalysis, including a language with integer arithmetic and strings over definable alphabets. Routines to find the period of a polyalphabetic cipher, do frequency counts, and locate repeated sequences are shown, but not built in.

Edna Grossman, "Group theoretic remarks on cryptographic systems based on two types of addition," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC 4742, Feb. 26, 1974.

This paper determines the class of transformations generated by a system which alternates addition mod 2 with addition mod 2^n.

Edwin L. Key, "An Analysis of the structure and complexity of nonlinear binary sequence generators," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 732-736, Nov. 1976.

K. Kjeldsen, "On the cycle structure of a set of nonlinear shift registers with symmetric feedback functions," *J. Combinatorial Theory* (A), vol. 2, pp. 154-169, 1976.

Robert Morris, "The Hagelin cipher machine (M-209), reconstruction of the internal settings," *Cryptology*, to appear.

Cryptanalysis of the M-209 with known plaintext.

Steven C. Pohlig and Martin E. Hellman, "An improved algorithm for computing logarithms in $GF(p)$ and its cryptographic significance," *IEEE Trans. Information Theory*, vol. IT-24, pp. 106-110, Jan. 1978.

A discussion of a cryptographic system with a high degree of mathematical structure and simplicity.

James Reeds, Dennis Ritchie, and Robert Morris, "The Hagelin cipher machine (M-209), cryptanalysis from ciphertext alone," *Cryptologia*, to appear.

Cryptanalysis of the M-209 from ciphertext only.

Bryant Tuckerman, "A study of the Vigenere-Vernam single and multiple loop enciphering systems," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC 2879, May 14, 1970.

Tuckerman develops cryptanalytic techniques for solving these ciphers from either rather small amounts of matched plaintext and ciphertext or larger amounts of pure ciphertext.

Bryant Tuckerman, "Solution of a substitution-fractionation-transposition cipher," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC 4537, Sept. 21, 1973.

Solution of a cipher which is similar to one round of the system shown in Fig. 10.

G. Data Encryption Standard

Dennis K. Branstad, Jason Gait, and Stuart Katzke, "Report of the workshop on cryptography in support of computer security," National Bureau of Standards, NBS IR 77-1291, 21-22 Sept. 1976.

Herbert S. Bright and Richard L. Enison, "Cryptography using modular software elements," in *Proc. Nat. Computer Conf.* (New York, NY), pp. 113-123, June 7-10, 1976.

This paper describes the implementation and testing of the DES on several computers.

Whitfield Diffie and Martin E. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard," *Computer*, vol. 10, no. 6, pp. 74-84, June 1977.

This paper gives a detailed cost estimate for the construction of a highly parallel special purpose computer to break the DES in one day.

W. F. Ehrsam, S. M. Matyas, C. H. Meyer, W. L. Tuchman, "A cryptographic key management scheme for implementing the data encryption standard," *IBM Syst. J.*, vol. 17, no. 2, pp. 106–125, 1978.

Jason Gait, "Validating the correctness of hardware implementations of the NBS data encryption standard," National Bureau of Standards, Special Pub. 500-20.

"Telecommunications: Compatibility requirements for use of the data encryption standards," Proposed Federal Standard 1026, General Services Administration, Oct. 13, 1977.

"Telecommunications: Security requirements for use of the data encryption standard," Proposed Federal Standard 1027, General Services Administration, 25 Aug. 1977.

M. E. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, "Results of an initial attempt to Cryptanalyze the NBS data encryption standard," Electrical Engineering Dep., Stanford Univ., Stanford, CA, SEL 76-042, (available from NTIS), Sept. 9, 1976.

Donald L. Heaton and Howard O. Wright, "LSI implementation of the proposed data encryption standard," presented at the Nat. Computer Conf., New York, NY, June 7-10, 1976, and circulated but not included in the proceedings.
A detailed description of the Collins implementation of DES on a chip.

David Kahn, "Tapping computers," *New York Times*, p. 27 (op. ed.), Apr. 3, 1976.

Gina Bari Kolata, "Computer encryption and the National Security Agency connection," *Science* vol. 197, pp. 438–440, July 29, 1977.

"An evaluation of the NBS data encryption standard," Lexar Corporation, Los Angeles, CA, Sept. 1976.
This report is remarkably similar to the above report of Hellman *et al.*

Robert Morris, N. J. A. Sloane, and A. D. Wyner, "Assessment of the National Bureau of Standards proposed federal data encryption standard," *Cryptologia*, vol. 1, pp. 281–291, July 1977.

"Data encryption standard," National Bureau of Standards, Federal Information Processing Standard (FIPS) Publication No. 46, Jan. 1977.

"Report of the 1976 workshop on estimation of significant advances in computer technology," held at the National Bureau of Standards, Aug. 30–31, 1976.
This workshop concluded that exhaustive search of DES was infeasible until at least 1990, in disagreement with the conclusions of the present authors.

"Guidelines for implementing and using the NBS data encryption standard," National Bureau of Standards (draft), Nov. 10, 1975.

"Computer security and the data encryption standard," National Bureau of Standards, Special Pub. 500-7, Feb. 15, 1978.

E. Stephan, "Communication standards for using DES," in *Proc. COMPCON*, Sept. 8, 1978.

David J. Sykes, "Implementation of the NBS encryption standard," presented at the 1977 Honeywell Computer Security and Privacy Symp., Apr. 19–20, 1977.

W. L. Tuchman, "Efficacy of DES in data processing," in *Proc. COMPCON*, Sept. 78.

Edward K. Yasaki, "Encryption algorithm: Key size is the thing," *Datamation*, vol. 22, no. 3, pp. 164, 166, Mar. 1976.

H. Encyclopedia Articles

Lambros D. Callimahos, "Cryptography," *Colliers Encyclopedia* vol. 7, pp. 513–530, 1973.

Lambros D. Callimahos, "Cryptology," *Encyclopedia Britanica*, 15th ed., *Macropaedia*, vol. 5, pp. 322–333, 1976.

William Frederick Friedman, "Cryptology," *Encyclopedia Britanica*, vol. 6 pp. 844–851, 1967.

David Kahn, "Cryptology," *Encyclopedia Americana*, vol. 8, pp. 276–285, 1976.

I. Glossaries

David Kahn, *Plaintext in the New Unabridged: An Examination of the Definitions on Cryptology in Websters Third New International Dictionary, Section 2 of Handbook of Cryptography*. New York: Crypto Press, 1963.

J. Historical

Linda Flato, "NSA's computer story," *Datamation*, vol. 24, no. 3, pp. 207–208, 212, Mar. 1978.

Bian Jones, *The Secret War*. New York: Methuen, 1978.
Chapter 6 contains a good acount of British cryptanalysis during the World War II.

David Kahn, *The Codebreakers, The Story of Secret Writing*. New York: Macmillan. 1967.
This is the most complete history of cryptology available, containing within its thousand pages material sufficient to reward several careful readings. As a technical reference, it leaves much to be desired, because it is concerned more with the influence of cryptology than with its technical development, though much of the latter comes through. Despite its nontechnical orientation, a lot can be learned from a careful study of the many details assembled. In addition, Kahn conveys a valuable feel for the subject, through the experiences of past cryptologists.

Herbert O. Yardly, *The American Black Chamber*. Indianapolis, IN: Bobbs-Merrill, 1931.
This volume gives an excellent feeling for the flavor of cryptanalysis.

K. Homophonic Systems

Wesley W. Chu and Charlie Neat, "A new computer cryptography: The expanded character set (ECS) cipher," *Advances in Computer Communications* (Sect. XIII—Computer Communications Security) Dedham, MA: Artech House, 1976.

Fred A. Stahl, "A homophonic cipher for computational cryptography" in *Proc. Nat. Computer Conf.* (New York, NY), pp. 565–568, June 4–8, 1973.

Fred A. Stahl, "On computational security," Ph.D. dissertation, Univ. Illinois, Urbana, Rep. R-637, UILU-ENG 73-2241, Jan. 1974.

L. Information Theoretic Papers

Aydano B. Carleial and Martin E. Hellman, "A note on Wyner's wiretap channel," *IEEE Trans. Information Theory*, vol. IT-23, pp. 387–390, May 1977.
This paper presents a slight generalization of Wyner's results on wiretap channels and suggests a way to quantify Shannon's concept of cryptographic diffusion.

E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane, "Codes which detect deception," *Bell Syst. Tech. J.*, vol. 53, pp. 405–424, Mar. 1974.
This paper considers an interesting authentication problem which arises in monitoring a nuclear test ban treaty. The problem is closely related to digital signatures, which were not yet known.

Ingemar Ingemarsson, "Analysis of secret functions with application to computer cryptography," in *Proc. Nat. Computer Conf.* (New York, NY), pp. 125–127, June 7–10, 1976.

S. K. Leung-Yan-Cheong and Martin E. Hellman, "The Gaussian wiretap channel," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 451–456, Sept. 1978.

C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, pp. 656–715, Oct. 1949.
This is a classic paper in which Shannon develops the information theoretic foundation for classical cryptanalysis, characterizes the systems which are unbreakable for information theoretic reasons, and outlines the construction of product ciphers.

Aaron Wyner, "The wiretap channel," *Bell Syst. Tech. J.*, vol. 54, pp. 1355–1387, Oct. 1975.
This paper evaluates the rate at which information can be conveyed to a legitimate receiver and kept secret from a wiretapper whose channel is noisier, introducing a new concept to both cryptography and information theory.

M. Key Distribution

S. M. Matyas and C. H. Meyer, "Generation, distribution, and installation of cryptographic keys," *IBM Syst. J.*, vol. 17, no. 2, pp. 126–137, 1978.

M. Sendrow, "Key management in E. F. T. environments," in *Proc. COMPCON*, Sept. 8, 1978.

N. Miscellaneous

Whitfield Diffie, *Data Security for EFT and Automated Business*. San Jose, CA: SBS Publishing, 1978.
Prognosis for the development and business applications of communication security over the next twenty years, followed by technical and

business advice for companies interested in making cryptographic products or employing cryptographic security.

Whitfield Diffie, "The outlook for computer security," *Mini-Micro Syst.*, vol. 2, no. 9, pp. 42–44, Oct. 1978.
Fairly accurate brief summary of the above.

Solomon W. Golomb, *Shift Register Sequences*. San Francisco, CA: Holden-Day, Inc., 1967.
Shift registers are one of the best studied approaches to the generation of pseudorandom sequences. This volume discusses the randomness characteristics of sequences from both linear and nonlinear shift registers.

James Clinton Halsey, "Finite automata admitting inverses with some applications to cryptography," Ph.D. dissertation, Dep. Mathematics, North Carolina State Univ., Raleigh, 1970.

John B. Kam and George I. Davida, "A structured design of substitution-permutation encyrption network," in R. A. De Millo, D. P. Doblin, A. K. Jones, and R. J. Lipton, Eds., *Foundations of Secure Computation*. New York: Academic Press, 1978.
Design of complete *S-P* networks (every output depends on every input) given complete *S*-boxes.

Paul Kinnucan, "Data encryption gurus: Tuchman and Meyer," *Mini-Micro Syst.*, vol. 2, no. 9, pp. 54, 56–58, 60, Oct. 1978.
History of DES from the IBM point of view.

R. E. Lennon, "Cryptography architecture for information security," *IBM Syst. J.*, vol. 17, no. 2, pp. 138–150, 1978.

Carl H. Meyer, "Design considerations for cryptography," in *Proc. Nat. Computer Conf.* (New York, NY), pp. 603–606, June 4–8, 1973.

Carl H. Meyer and Walter L. Tuchman, "Putting data encryption to work," *Mini-Micro Syst.*, vol. 2, no. 9, pp. 46, 48–50, 52, Oct. 1978.

Stephen Carl Pohlig, "Algebraic and combinatoric aspects of cryptography," Ph.D. dissertation, Electrical Engineering Dep., Stanford Univ., Stanford, CA, SEL-77-038, Oct. 1977.

O. Popular

Horst Feistel, "Cryptography and computer privacy," *Scientific Amer.*, vol. 228, no. 5, pp. 15–23, May 1973.
This article gives the most readable account of the operation of the IBM Lucifer cipher.

David Kahn, "Modern cryptology," *Scientific Amer.*, vol. 215, no. 1, pp. 38–46, July 1966.
This article presents a superficial discussion of cryptology in roughly the World War II period. Rotor and Hagelin machines are mentioned, along with a discussion of one time pads, but neither mathematical techniques in cryptanalysis nor the use of computers are covered.

Gina Bari Kolata, "Cryptography: On the brink of a revolution?" *Science*, vol. 197, pp. 747–748, Aug. 19, 1977.

P. Public Keys

Whitfield Diffie and Martin E. Hellman, "Multiuser cryptographic techniques," in *Proc. Nat. Computer Conf.* (New York, NY), June 7–10, 1976.
This paper describes two approaches to supplying secure first contact between users of a computer network, without sending keys along outside channels, and some related problems.

Whitfield Diffie and Martin E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.
This paper examines several new possibilities in cryptography: systems which require no prior key channel, unforgeable digital signatures, and the use of complexity theory to study the security of cryptographic systems.

Ralph Merkle, "Secure communication over insecure channels," *Commun. ACM*, vol. 21, pp. 294–299, Apr. 1978.
This paper introduced the concept of public key distribution independently of Diffie and Hellman.

Michael O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," submitted to *Commun. ACM*.
A variation of the RSA scheme with $E = 2$ is shown to have cryptanalytic difficulty equivalent to factoring.

Ronald L. Rivest, Adi Shamir, and L. Adleman, "On digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, Feb. 1978.
This paper describes a public key cryptosystem based on exponentiation in modular arithmetic.

Gustavus J. Simmons and Michael J. Norris, "Preliminary comments on the M.I.T. public key cryptosystem," *Cryptologia*, vol. 1, pp. 406–414, Oct. 1977.

Q. Surveys

M. B. Girsdansky, "Cryptology, the computer and data privacy" *Compu. Automation*, vol. 21, no. 4, pp. 12–19, Apr. 1972.
This article gives an overview of cryptographic work at IBM, including a discussion of the weakness of linear feedback shift registers, Tuckerman's attack on the Vigenere–Vernam systems, and Lucifer.

G. E. Mellon, "Computers, cryptography and common sense," in *Proc. Nat. Computer Conf.* (New York, NY), pp. 569–579, June 4–8, 1973.
A rather pessimistic survey of the application of cryptography.

Rein Turn, "Privacy transformations for data banks" in *Proc. Nat. Computer Conf.* (New York, NY), pp. 589–600, June 6–8, 1973).
Survey of the fundamentals of cryptography with a view to data bank applications.

R. System Design and Implementation

G. Gordon Benedict, "An enciphering module for multics," M.I.T. Project MAC, Tech. Memo. 50, July 1974.
This masters thesis describes the computer implementation of the IBM Lucifer system as described by Smith.

J. M. Carroll and P. M. McLelland, "Fast 'Infinite-key' privacy transformation for resource sharing systems," in *Proc. Fall Joint Computer Conf.* (Houston, TX), pp. 223–230, Nov. 17–19, 1970.
This paper describes a stream cipher based on an additive random number generator.

Arthur Evans Jr., William Kantrowitz, and Edwin Weiss, "A user authentication system not requiring secrecy in the computer," *Commun. ACM*, vol. 17, pp. 437–442, Aug. 1974.
This paper describes a oneway function obtained by using a transformation similar to the IBM Lucifer system and using the argument as both input and key.

Horst Feistel, "Cryptographic coding for data-bank privacy," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC-2827, Mar. 18, 1970.

Horst Feistel, William A. Notz, and J. Lynn Smith, "Some cryptographic techniques for machine to machine data communications," *Proc. IEEE*, vol. 63, pp. 1545–1554, Nov. 1975.
These two papers describe various ways of building block cipher systems and shows their application to a variety of privacy and authentication problems in computer systems and networks.

Philip R. Geffe, "How to protect data with ciphers that are really hard to break," *Electronics*, vol. 46, no. 1, pp. 99–101, 4 Jan. 1973.
A scheme for constructing a keystream by using a linear random number generator to select bits alternately from the streams produced by two other linear random number generators.

Marvin Perlman, "Generation of key in cryptographic system for secure communications," NASA Technical Brief 75-10278, Feb. 1976.
Discussion of some nonlinear shift registers with known periods.

Vera Pless, "Encryption schemes for computer confidentiality," *IEEE Trans. Computers*, vol. C-26, pp. 1133–1136, Nov. 1977.
An interesting discussion of the use of *J-K* flip flop networks to produce maximal period keying sequences.

George B. Purdy, "A high security log-in procedure," *Commun. ACM*, vol. 17, pp. 442–445, Aug. 1974.
This paper studies the use of a polynomial of very high degree over a finite field as a one-way function. This function has subsequently been used to protect the passwords on the BBN TENEX system.

R. O. Skatrud, "A consideration of the application of cryptographic techniques to data processing," in *Proc. Fall Joint Computer Conf.* (Las Vegas, NV), pp. 111–117, Nov. 18–20, 1969.
Describes two systems: a substitution system similar to a double loop Vigenere, and a matrix transposition system.

J. Lynn Smith, "The design of Lucifer, a cryptographic device for data communications," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC 3326, Apr. 15, 1971.
This working paper gives a detailed discussion of the functioning, hardware design, and costs of the IBM Lucifer system.

J. Lynn Smith, "Hardware implementation of a cryptographic system," *IBM Tech. Disclosure Bull.*, vol. 13, pp. 1004–1008, Aug. 1971.
This describes the implementation of a system different from the above.

REFERENCES

[1] J. Squires, "Russ monitor of U.S. phones," *Chicago Tribune* pp. 123, June 25, 1975.
[2] R. Davis, "Remedies sought to defeat Soviet eavesdropping on

microwave links," *Microwave Syst.*, vol. 8, no. 6, pp. 17-20, June 1978.
[3] *Supplementary Detailed Staff Reports on Intelligence Activities and the Rights of Americans*, Final Report of the United States Senate Select Committee to Study Governmental Operations with Respect to Intelligence Activities, Report No. 94-755, Book 3 (Washington, DC: U.S. Government Printing Office), 1976.
[4] J. L. Flanagan, *Speech Analysis, Synthesis, and Perception, 2nd ed.* Berlin; Germany: Springer-Verlag, 1972.
[5] D. Raj Reddy, *Speech Recognition.* New York: Academic Press, 1975.
[6] *Proc. IEEE, Special Issue on Packet Communication Networks*, Nov. 1978.
[7] *Data Encryption Standard.* National Bureau of Standards, Federal Information Processing Standard (FIPS) Publication no. 46, Jan. 1977.
[8] W. Diffie and M. E. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard," *Computer*, vol. 10, no. 6, pp. 74-84, June 1977.
[9] W. L. Tuchman, "Efficacy of DES in data processing," in *Proc. COMPCON*, Sept. 1978.
[10] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, pp. 656-715, Oct. 1949.
[11] W. F. Friedman, "Cryptology," *Encyclopedia Britanica*, vol. 6, pp. 844-851, 1967.
[12] M. E. Hellman, "An extension of the Shannon theory approach to cryptography," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 289-294, May 1977.
[13] C. E. Shannon, "Prediction and entropy of printed English," *Bell Syst. Tech. J.*, vol. 30, pp. 50-64, Jan. 1951.
[14] T. M. Cover and R. C. King, "A convergent gambling estimate of the entropy of English," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 413-421, Aug. 1978.
[15] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644-654, Nov. 1976.
[16] R. Merkle, "Secure communication over insecure channels," *Commun. ACM*, pp. 294-299, Apr. 1978.
[17] R. L. Rivest, A. Shamir, and L. Adleman, "On digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21 no. 2, pp. 120-126, Feb. 1978.
[18] R. C. Merkle and M. E. Hellman, "Hiding information and signatures in trap door knapsacks," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 525-530, Sept. 1978.
[19] R. J. McEliece, "A public key cryptosystem based on algebraic coding theory," JPL DSN Progress Report 42-44, pp. 114-116, Jan.-Feb. 1978.
[20] A. Sinkov, *Elementary Cryptanalysis, A Mathematical Approach.* New York: Random House, New Mathematical Library, no. 22, 1968.
[21] H. F. Gaines, *Cryptanalysis, A Study of Ciphers and their Solution.* New York: Dover Publications, 1956.
[22] W. F. Friedman, *Military Cryptanalysis.* Washington DC: U.S. Government Printing Office, 1944.
[23] D. Kahn, *The Codebreakers, The Story of Secret Writing.* New York: Macmillan, 1967.
[24] Tuckerman, "A study of the Vigenere-Vernam single and multiple loop enciphering systems," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC 2879, May 14, 1970.
[25] W. F. Friedman, *Methods for the Solution of Running-Key Ciphers.* Geneva, IL: Riverbank Laboratories, Department of Ciphers Publication no. 16, 1918.
[26] R. Morris, "The Hagelin cipher machine (M-209), reconstruction of the internal settings," *Cryptologia*, to appear.
[27] J. Reeds, D. Ritchie, and R. Morris, "The Hagelin cipher machine (M-209), cryptanalysis from ciphertext alone," *Cryptologia*, to appear.
[28] W. G. Barker, *Cryptanalysis of the Hagelin Cryptograph.* Laguna Hills, CA: Aegean Park Press 1977.
[29] S. C. Pohlig, "Algebraic and combinatoric aspects of cryptography," Ph.D. dissertation, Electrical Engineering Dep., Stanford Univ., SEL-77-038, Oct. 1977.
[30] S. W. Golomb, *Shift Register Sequences.* San Francisco, CA: Holden-Day, 1967.
[31] W. W. Peterson and E. J. Weldon, *Error Correcting Codes, Second Edition*, Cambridge, MA: M.I.T. Press, 1972.
[32] R. C. Tausworthe, "Random numbers generated by linear recurrence modulo two," *Math. Computation*, vol. 19, pp. 201-209, 1965.
[33] E. J. Watson, "Primitive polynomials (mod 2)," *Math. Computation*, vol. 16, pp. 368-369, 1962.
[34] E. J. Groth, "Generation of binary sequences with controllable complexity," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 288-296, May 1971.
[35] E. L. Key, "An analysis of the structure and complexity of nonlinear binary sequence generators," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 732-736, Nov. 1976.
[36] N. J. A. Sloane, *A Handbook of Integer Sequences.* New York: Academic Press, 1973.
[37] J. L. Massey, "Feedback shift register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122-127, Jan. 1969.
[38] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw-Hill, 1968.
[39] L. S. Hill, "Cryptography in an algebraic alphabet," *Amer. Math. Monthly*, vol. 36, pp. 306-312, June-July 1929.
[40] H. Feistel, "Cryptographic coding for data-bank privacy," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC-2827, Mar. 18, 1970.
[41] ——, "Cryptography and computer privacy," *Scientific Amer.*, vol. 228, no. 5, pp. 15-23, May 1973.
[42] H. Feistel, W. A. Notz, and J. L. Smith, "Some cryptographic techniques for machine to machine data communications," *Proc. IEEE*, vol. 63, pp. 1545-1554, Nov. 1975.
[43] J. L. Smith, "The design of Lucifer, a cryptographic device for data communications," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC 3326, Apr. 15, 1971.
[44] J. L. Smith, W. A. Notz, and P. R. Osseck, "An experimental application of cryptography to a remotely accessed data system," in *Proc. ACM Conf.*, pp. 282-297, 1972.
[45] M. E. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, "Results of an initial attempt to cryptanalyze the NBS data encryption standard," Electrical Engineering Dep., Stanford Univ. SEL 76-042 (available from NTIS), Sept. 9, 1976.
[46] D. K. Branstad, J. Gait, and S. Katzke, "Report of the workshop on cryptography in support of computer security," National Bureau of Standards Rep. NBSIR 77-1291, 21-22 Sept. 1976.
[47] E. K. Yasaki, "Encryption algorithm: Key size is the thing," *Datamation*, vol. 22, no. 3, pp. 164, 166, Mar. 1976.
[48] D. Kahn, "Tapping computers," *New York Times*, p. 27, Apr. 3, 1976.
[49] G. B. Kolata, "Computer encryption and the National Security Agency connection science," vol. 197, pp. 438-440, July 29, 1977.
[50] Unclassified summary: Involvement of NSA in the Development of the Data Encryption Standard, Senate Committee on Intelligence, 1978.
[51] F. Faggin, "The role of technology in microcomputer design and evolution," *IEEE Trans. Circuits Syst.*, vol. CAS-7, pp. 4-13, Feb. 1975.
[52] ——, "Microprocessor technology: Yesterday, today and tomorrow," *EDN*, Nov. 20, 1975.
[53] "Pentagon to fund major IC program," *Electronics*, pp. 81-82, Sept. 14, 1978.
[54] "VHSI proposal finds willing audience," *Electronics*, pp. 89-90, Sept. 28, 1978.
[55] W. L. Tuchman, Talk presented at the Nat. Computer Conf., Anaheim, CA, June 1978.
[56] E. R. Brunner, "Efficient speech scrambling: An economic solution to the secure voice communication problem," in *Rec. Int. Conf. Communications Equipment and Systems* (Brighton, Sussex, London, England), pp. 336-339, June 8-11, 1976.
[57] K. H. Kirchhofer, "Secure voice communication—Cryptophony," *Int. Defense Rev.*, vol. 9, pp. 761-767, May 1976.
[58] A. M. McCalmont, "Voice privacy (scrambling) for more effective communications: Techniques, systems, and operations," Technical Communications Corp., Jan. 1973.
[59] B. Gold, "Digital speech networks," *Proc. IEEE*, vol. 65, pp. 1636-1638, Dec. 1977.
[60] S. C. Pohlig and M. E. Hellman, "An improved algorithm for computing logarithms in $GF(p)$ and its cryptographic significance," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 106-111, Jan. 1978.
[61] D. E. Knuth, *The Art of Computer Programming.* Reading, MA: Addison-Wesley, 1969.
[62] G. J. Simmons and M. J. Norris, "Preliminary comments on the M.I.T. public key cryptosystem," *Cryptologia*, vol. 1, pp. 406-414, Oct. 1977.
[63] R. L. Rivest, "Remarks on a proposed cryptanalytic attack on the MIT public key cryptosystem," *Cryptologia*, vol. 2, no. 1, pp. 62-65, Jan. 1978.
[64] A. Shamir, "A fast signature scheme," MIT/LCS/TM-107, July 1978.
[65] H. S. Stone, *Discrete Mathematical Structures.* Chicago, IL: Science Research Assoc., 1973.
[66] E. Parzen, *Modern Probability Theory and Its Applications.* New York: Wiley, 1960.
[67] R. G. Gallager, *Information Theory and Reliable Communication.* New York: Wiley, 1968.

[68] D. Branstad, "Encryption protection in computer data communications," in *Proc. 4th Data Communications Symp.* (Quebec City, Canada), Oct. 7-9, 1975.
[69] W. Diffie and M. E. Hellman, "Multiuser cryptographic techniques," in *Proc. Nat. Computer Conf.* (New York, NY), June 7-10, 1976.
[70] "Telecommunications: Compatibility requirements for use of the data encryption standard," Proposed Federal Standard 1026, General Services Administration, Oct. 13, 1977.
[71] S. Lin, *Error Correcting Codes.* Englewood Cliffs, NJ: Prentice-Hall, 1970.
[72] "Telecommunications: Security requirements for use of the data encryption standard," Proposed Federal Standard 1027, General Services Administration, Aug. 25, 1977.
[73] R. A. De Millo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, Eds., *Foundations of Secure Computation.* New York: Academic Press, 1978.
[74] J. H. Saltzer and M. D. Schroeder, "The protection of Information in computer systems," *Proc. IEEE*, vol. 63, pp. 1278-1308, Sept. 1975.
[75] L. J. Hoffman, *Modern Methods for Computer Security and Privacy.* Englewood Cliffs, NJ: Prentice-Hall 1977.
[76] J. A. Scherf, *Computer and Data Security: A Comprehensive Annotated Bibliography*, M.I.T. Project MAC, MAC TR-122, Jan. 1974.
[77] G. B. Purdy, "A high security log-in procedure," *Commun. ACM*, vol. 17, pp. 442-445, Aug. 1974.
[78] A. Evans, Jr., W. Kantrowitz, and E. Weiss, "A user authentication system not requiring secrecy in the computer," *Commun. ACM*, vol. 17, pp. 437-442, Aug. 1974.
[79] *Design Alternatives for Computer Network Security.* NBS Special Publ. 500-21, 2 vols., Jan. 1978.

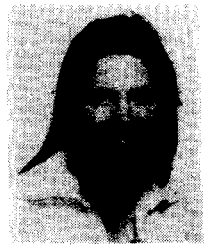
Contributors

Berthold G. Bosch (M'64-SM'67) was born in Bonn, Germany, on May 30, 1930. He received the Dipl.-Ing. degree in electrical engineering from Aachen Technical University, Germany, in 1956, the Ph.D. degree from Southampton University, England, in 1960, the Habilitation from Karlsruhe University, Germany, in 1969, and the D.Sc. degree from Southampton University in 1976.

From 1956 to 1957, he held an AEG Foreign Scholarship at the Electronics Department of the University of Southampton. During 1958-1960, as a Research Assistant at the same department, he was engaged in work on microwave-tube noise. From 1960 to 1972, he was with AEG-Telefunken, Ulm, Germany, where he occupied various posts in the Tube Works and in the Research Institute, eventually becoming Head of the Electronics Department in the latter institution. During that time, he carried out and was responsible for work on microwave tubes, parametric amplifiers, microwave semiconductors, and high-rate PCM circuitry. In 1969, he was made an Adjunct Staff Member (Privat-Dozent) in the Faculty of Electrical Engineering of Karlsruhe University, and in 1972, he became Professor of Electronics at the Ruhr-University, Bochum, Germany, and simultaneously Joint Director of the Institute of Electronics. During the academic year 1973-1974, he served as Dean of the Faculty of Electrical Engineering. His present research interests include devices and integrated circuits for high-speed electronics, integrated optics, and optical communications. He is coauthor (together with R. W. H. Engelmann) of *Gunn-Effect Electronics.*

Dr. Bosch was awarded the A.F.-Bulgin Premium of the British IRE in 1962 (jointly with W. A. Gambling), and in 1969, he received the Annual Prize of the Nachrichtentechnische Gesellschaft. He is a member of Verband Deutscher Elektrotechniker, Deutsche Physikalische Gesellschaft, and of the Administrative Committee of the European Society for Engineering Education.

*



Whitfield Diffie (M'77) received the B.S. degree in mathematics from Massachusetts Institute of Technology, Cambridge, in 1965.

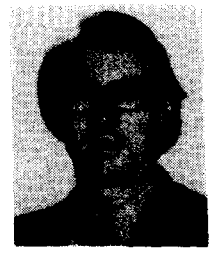
Subsequently, he worked on symbolic mathematical manipulation at the Mitre Corporation in Bedford, MA, and proof of program correctness at the Stanford Artificial Intelligence Laboratory, Stanford, CA. More recently, he was a graduate student in the Department of Electrical Engineering of Stanford University, Stanford, CA. He is a member of the Scientific Staff of BNR, Inc., where he is responsible for research on computer and communications security.

Martin E. Hellman (S'63-M'69-SM'78) received the B.E. degree from New York University, New York, NY, in 1966, the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, in 1967 and 1969, respectively, all in electrical engineering.

Currently, he is an Associate Professor of Electrical Engineering at Stanford University, doing research in cryptography, information theory, and communication theory. He is also acting as Associate Department Chairman during 1977-1979. He was an Assistant Professor at Massachusetts Institute of Technology Cambridge, from 1969 to 1971, and on the Staff of IBM T. J. Watson Research Center, Yorktown Heights, NY, from 1968 to 1969. He is the author of over thirty publications.

Dr. Hellman is a member of the Information Theory Group's Board of Governors and was an Associate Editor of the IEEE Transactions on Communications.

*



Geoffrey H. C. New was born in Shrewsbury, England, in 1942. He received the B.A. and D.Phil. degrees in physics at Oxford University, Oxford, England, in 1964 and 1967 respectively.

From 1967 until September 1973 he was a Lecturer at Queen's University, Belfast, Northern Ireland, U.K., and has recently moved to a lectureship at Imperial College, London University where he is presently a Reader in Nonlinear Optics. For his graduate work at Oxford, he conducted experiments both on optical rectification and on third-harmonic generation in gases which was first observed at that time. In Belfast, he took part in a research program dealing with various aspects of the generation, measurement, and properties of ultrashort light pulses. His recent work has been concerned with the theory of mode-locking in various types of laser and with the theory of two-photon resonant frequency mixing processes.

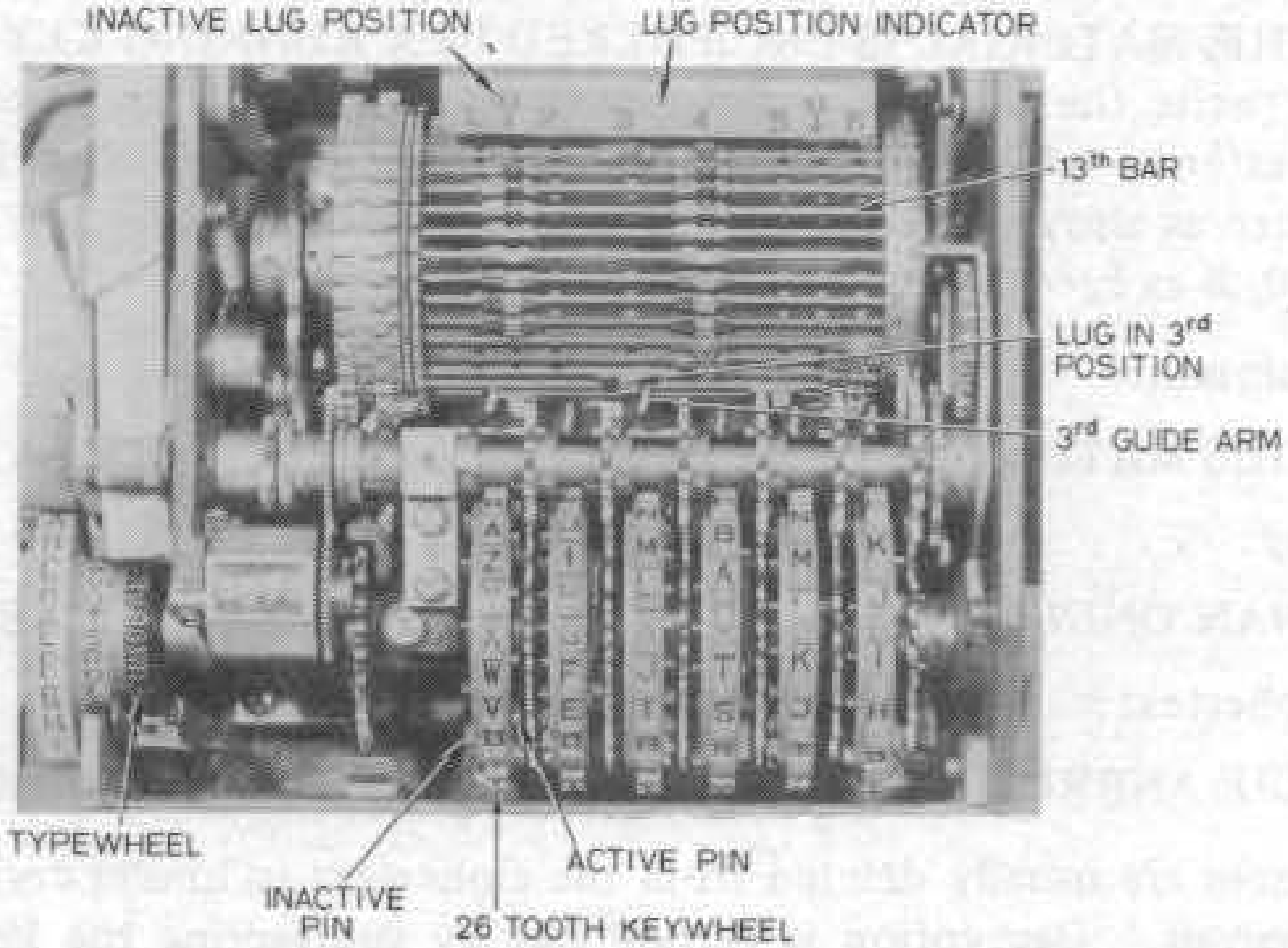


Fig. 4. Essential parts of Hagelin machine.