1981

Cryptography and its application to operating system security

Michelle Painchaud

CRYPTOGRAPHY AND ITS APPLICATION TO OPERATING SYSTEM SECURITY

by

Michelle Painchaud

Submitted to the faculty of

the School of Computer Science & Technology

in partial fulfullment

of the requirements for

the degree of

Master of Science

Master of Science in Computer Science

Thesis Approval Form

This is to certify that _Michelle_Painchaud_ has submitted a thesis entitled:

Cryptography and its Application to Operating System Security

to the faculty of the School of Computer Science and Technology in partial fulfillment of the requirements for the degree of Master of Science.

Approval: Peter H. Lutz/ (thesis advisor) 6/12/81 (date)

Wiley McKinzie/ (committee member) 6/12/81 (date)

Peter Anderson/ (committee member) 6/12/81 (date)

Table of Contents

Growing Interest in Security and Privacy

Privacy and security are emerging as two vital issues and, as a result, these two topics are appearing in a growing number of computer related discussions (Conway, et. al., 1972). This current interest was not always the case for, up until the last decade, these issues were of minimal concern and only information which was vital for national security was deemed relevant for protection purposes (Martin, 1970)(Feistel, et. al., 1971). The advent of computer systems awakened the commercial, private, and legal sectors to the problem of ensuring privacy and security (Dinardo, 1978). Due to the increasing use of computers as a means of providing cheap and efficient communications plus their ability to process tremendous amounts of sensitive information, coupled with the vulnerability of these systems and their data files to penetration, the need for privacy and security within computer systems became increasingly apparent (Kolata, 1977). Thus, the culmunation of legislative and societal pressures combined with technological innovations resulted in the current interest in computer security and privacy (Burns, 1977)(Konheim, 1978).

Security versus Privacy

Security and privacy are related. Often time discussions involving these issues lump the two together thereby failing to distinguish their differences. As a result, privacy and security are frequently confused with one another (Davida, 1978).

Privacy encompasses not only the legal and ethical issues

of data collection, but the moral ones as well. It safeguards the individual's right to control the collection, dissemination, and use of all personal information. In contrast, the technical means of controlling access to, modification, and dissemination of, data is dealt with by security which ensures the enforcement of all privacy decisions (Conway, et. al., 1972).

Environmental Threats

Sensitive information stored within the computer network is highly vulnerable to disclosure thus making the task of data privacy and security extremely difficult (Sykes, 1976).

Data is always susceptible to monitoring, modification, misrouting, or substitution during its transferral over communication links. But perhaps an even greater and more insidious danger is the insertion of spurious data designed to confuse or misdirect the operation of the system (Benedict, 1974). Whenever a storage device such as a disk is used to store sensitive data, there is always the danger that this information will be accessed by unauthorized individuals. If the data is stored on removable media such as disc packs or magnetic tapes, the additional threat of the physical removal or theft of the information exists (Sykes, 1976).

If the network is not suitably protected then the perpetration of any of these threats can lead to considerable damage which may result in rendering the network useless for communication purposes and/or the storage of sensitive information (Popek & Kline, 1979). Consequently, in order to securely protect a network, the failures, errors, omissions, and vulnera-

bilities of that particular network must be considered (Browne, 1977).

Cryptography and Data Protection

The use of cryptography to protect information within the network is one means by which the privacy, security, and integrity of information exchange can be safeguarded (Popek & Kline, 1979). Cryptography is both the method and the process through which communicated and stored data can be protected against theft or misuse regardless of whether or not the disclosure is legal or illegal and the attempt is deliberate or accidental. Although cryptography by itself is not sufficient to protect a system, it can be used to supplement other data security countermeasures (Keys & Clamons, 1974).

What is Cryptography?

Cryptography is a highly technical science which deals with the techniques by which the originator of the information can transform the message in such a way that it is unintelligible if intercepted by someone other than the intended receiver (Dinardo, 1978). It is this transformation of original text or plaintext, usually on a bit or character level, into meaningless jumble or ciphertext which is referred to as encryption (Benedict, 1974). Decryption is the opposite of encryption and is the reversal of the transformation process, enabling the unintelligible message to be reconverted back into meaningful form. Both encryption and decryption are carried out by means of an algorithm referred to as a cipher system (Katzan, 1977). Associated with each cipher system is a key or pattern of bits. It is this key

which is the sole means of determining the transformation used for the encryption/decryption process (Walker & Blake, 1977).

The primary objective of cryptography is to produce complex data transformations such that the amount of time and resources required to break the ciphertext by far exceeds the value of the information which is being protected. The "time factor" enables the sensitive information to be kept secure long enough so that it may be out-dated and therefore useless by the time the intruder finally breaks the cipher. The "cost factor", on the other hand, results in the intruder paying a higher price for the information than it is actually worth to him (Hellman, 1979)(Mellen, 1978).

Cryptography in the Past

Cryptography, which literally means "hidden writing", initially evolved as an art and is believed to have developed independently in various ancient civilizations such as Egypt, India, and Mesopotamia (Lempel, 1979). It was not until much later with the systematization of cryptographic techniques by the Arabs that cryptography was finally transformed from an art into a science (Branstad, 1977).

Cryptography in Transition

Throughout history, cryptographic techniques were constantly changing as a result of the pursuit for a perfect cipher system which was impenetrable to cryptanalytic attack. One of the earliest cryptographic techniques devised replaced each letter of the plaintext message with one of several substitutes called homophones. For example, if the homophones for the letter "e"

were the values 16, 20, 36, or 75, then two possibilities of the resulting encryption of the word "secret" might have appeared as 07 16 27 55 75 99 or 07 20 27 55 36 99. Notice that while the homophones for the letter "e" varied, the substitutions for the other letters remained constant. Thus encryption of the same word two or more times within the same message enabled the cryptanalyst to reduce the homophones to equivalent terms and to the eventual cracking of the cipher. This technique, therefore, was soon discarded since it relied on too obvious a substitution technique (Kahn, 1966).

It was not until 1466 in Italy that the foundations for modern cryptography were laid. Leo Battista Alberti devised a cryptographic technique referred to as polyalphabeticity. Whereas in previous cipher systems, only a single alphabet was used to encipher the message, thus the term monoalphabetic, this technique employed several different cipher alphabets or substitution alphabets which were used periodically to encipher the message (Diffie & Hellman, 1979). These cipher alphabets were generated from a single, primary alphabet by varying the starting letters of the alphabets in relation to the primary alphabet. For example, if the primary alphabet was ABC...XYZ then a cipher alphabet could be created by starting with the third letter thus yielding CDEF...XYZAB. Each cipher alphabet had associated with it a unique keyletter or starting letter. Therefore, in the preceding example, the keyletter for the cipher alphabet would be "c" since this letter signified the starting point of the alphabet (for further examples, please refer to

figure 1). By inserting the keyletters at various intervals in the cryptogram, Alberti was able to indicate which cipher alphabet was to be used to encipher the following words (Kahn, 1966) (Simmons, 1979).

Through the use of numerous cipher alphabets to encipher a single message, the cipher equivalents for the plaintext letters varied thus enabling this method to withstand cryptanalysis more easily than the previous monoalphabetic methods. Cryptanalysis was by far more difficult since the intruder had to know which alphabet was used for each letter in order to decipher the information (Diffie & Hellman, 1979)(Lempel, 1979).

In the years that followed major improvements were made to this cipher system. However, it was not until 1553 that Giovanni Battista Belasco devised an efficient and secure means of indicating the current cipher alphabet in use. An easily remembered and interchangeable keyword was repeatedly written over the letters of the plaintext message. Each letter of the keyword was then used to designate the cipher alphabet that was to be used to encipher the corresponding plaintext letter (see figure 2).

Although this system remained essentially impregnable over the next 300 years, it was not widely used. Due to the fact that this system took some effort to employ and was subject to error, elaborate homophonic systems which did not require multiple cipher alphabets were used instead. However, with the invention of the electric telegraph, a flexible cipher system was needed which would permit prompt error correction. The

ability of polyalphabetic systems to fulfill this need produced an increased interest in this type of ciper (Kahn, 1966).

In 1863, Friedrich Kasiski published a method for solving polyalphabetic ciphers that was based upon repeating keys. The repetition of a frequent plaintext trigram such as "ing" or "the" in conjunction with the same portion of the repeating key produced equivalent ciphertext. If enough repeated ciphertext was made available to the cryptanalyst, it was possible to determine the length of the key used for encryption and the number of times the key was repeated between repetitions of ciphertext. The cryptanalyst could then proceed to sort the letters of the cryptogram into groups based upon key length and subsequently, analyze each group according to the principles of letter frequency (see figure 3)(Diffie & Hellman, 1979).

The discovery of the vulnerability of polyalphabetic ciphers spurred interest in the development of more ingenious enciphering schemes. One of the outgrowths of this period was the proposal that a single, continuous nonrepeating key, such as the text of a book, be used in place of repeatedly using a single word to encipher the plaintext (see figure 4). In spite of the fact that the running key prevented the periodicity exploited by the Kasiski solution, this proposal was short lived for Auguste Kerckhoff demonstrated that this cipher system was no more secure than the previously used polyalphabetic ciphers (Lempel, 1979). A technique referred to as superimposition permitted the isolation of two or more identical ciphertext fragments thereby, indicating that a repeated fragment of the

plaintext message had been enciphered by the same portion of running key. Given enough ciphertext, the cryptanalyst was able to obtain sufficient information to solve the cipher (Diffie & Hellman, 1979). In this way, Kerckhoff's superimposition method could be used to break any polyalphabetic cipher devised up until this time (Kahn, 1966)(Simmons, 1979)(Mellen, 1978).

In the early 1900's the discovery was made that only one cipher could resist the superimposition technique. This cipher system employed a nonrepeating key which was both meaningless and without pattern (see figure 5). The key consisted of a totally random numerical string which was added to the numerical representation of the plaintext (Feistel, 1973). The security of this system was dependent upon the key used to encipher the message being nonreusable and the same length as the message (Diffie & Hellman, 1979).

Although this "one-time system", as it was called, was unbreakable, practical difficulties existed. The extremely long key which was required resulted in this system being prohibitively expensive for most applications (Diffie & Hellman, 1976). This system was also plagued by key management problems which resulted from the advance preparation and distribution of the key. Consequently, the "one-time system" was impractical for universal use (Lempel, 1979).

The failure of the "one-time system" as a flexible cipher system led to the search for alternative cipher systems. Thus the period from 1917 through World War II witnessed the development of several types of cryptographic machines which marked

the beginning of a revolutionary new trend (Diffie & Hellman, 1976).

The rotor machine, first developed by Edward H. Hebem in 1917, was probably one of the best examples of this modern system. This machine consisted of a hard rubber disk or rotor which contained twenty-six contacts on each side. By randomly wiring the twenty-six contacts on one side to the twenty-six contacts on the other side and through the use of up to eight rotors which rotated on the same axis, the set of rotors provided an electrical path comparable to a series of cipher alphabets. By depressing keys on the machine's typewriter keyboard, a plaintext letter was transformed into a cipher letter which was illuminated on the panel. After each letter was enciphered, one or more of the rotors rotated to a new space thereby, establishing a new electrical path and consequently, substituting a new cipher alphabet (Kahn, 1966)(Simmons, 1979).

Another polyalphabetic system called the Hagelin device was invented in 1934 by Boris C. W. Hagelin. This device consisted of a variable toothed gear which was used to drive a cipher alphabet through a variable number of places. In comparison to the rotor machine, this device was simpler to use and much cheaper, however, it was not as secure. Despite this drawback, the Hagelin device was used quite frequently during the years that followed (Kahn, 1966).

All of the systems discussed so far enciphered only one letter at a time, thus the term monographic systems. The enciphering of two or more letters simultaneously was also pos-

sible through the use of polygraphic cipher systems. One of the best known polygraphic system was the Playfair Cipher (Katzan, 1977).

The Playfair Cipher was based upon digraphic substitution which permitted two characters to be enciphered simultaneously through the use of a 5x5 substitution matrix. The plaintext message was first divided into groups of two characters. The occurrence of any double letters within the groups was corrected by the insertion of an infrequently used character, such as "x" or "z" thereby, eliminating pairs of identical characters. By referring to the matrix, each pair of plaintext characters was transformed into an equivalent ciphertext pair according to the following rules: 1) if the pair of plaintext letters were in the same row, then the corresponding ciphertext letters were found to the immediate right of the plaintext letters, 2) if the two plaintext characters were in the same column, the equivalent ciphertext characters were the letters immediately below the plaintext ones, and 3) if the pair of plaintext characters were neither in the same row nor the same column, the ciphertext equivalent of each character was the letter found at the intersection of the corresponding row and column, with the row letter of the first plaintext letter being first (see figure 6)(Katzan, 1977)(Kahn, 1966).

The polygraphic system was by far superior with respect to security than the monographic system. This system was better equipped to resist frequency analysis by making available a greater number of letter pairs as well as disguising the

characteristics of the letters. As a result, the task of the cryptanalyst was made exceedingly difficult (Kahn, 1966)(Simmons, 1979)(Katzan, 1977).

Cryptography in the Age of Automation

The discovery that digraphic substitution provided greater security enhancements naturally led to the assumption that systems which enciphered several letters at a time would offer even greater security measures. Lester S. Hill was responsible for inventing an algebraic polygraphic system that was capable of enciphering any number of letters simultaneously (Simmons, 1979). With his discovery, the seeds of modern algebraic cryptography were laid (Diffie & Hellman, 1979).

Algebraic ciphers transformed letters to numbers by means of a table look-up operation. Encryption and decryption were then performed through the use of conventional mathematical methods (Katzan, 1977). In Hill's system, after the letters were converted to numerical values, these numbers were used as variables in a set of simultaneous equations. The cipher's key consisted of the constants which formed the equations. Although theoretically no limits were set on the number of letters that could be enciphered simultaneously, practical restrictions limited the encipherment to only a few letters (Simmons, 1979)(Katzan, 1977).

Hill's system was never seriously used primarily because of this system's vulnerability to cryptanalytic attack. Despite the failure of this system, the theory behind it was valuable and led to the search for more complex families of trans-

formations (Diffie & Hellman, 1979). Thus, the years that followed saw the development of greater sophistication in cipher systems and witnessed the corresponding growth in complexity with regard to the simultaneous equations. This implementation of more ingenious enciphering schemes was directly related to the invention of the computer. The computer increasingly began to play a more vital role in the design of highly technical ciphers and the computational power that became available with the use of the computer had a significant impact on cryptography (Lempel, 1979).

Prior to the development of electronic computers, only simple electromechanical systems existed thus restricting the type of cryptographic operation which could be performed. The computer, which did not rely upon the use of gears for its computing power, was able to break through these restrictions. As a consequence, the available computing power increased substantially and for the first time, the search for better encryption methods according to purely cryptographic criteria was made possible (Diffie & Hellman, 1976).

The introduction of the digital computer was responsible not only for increasing the available computing power but also for permitting a more adequate means of testing the strength of the cipher system. During the 16th and 17th centuries, mathematical arguments were often invoked to verify the strength of the cryptographic systems. These mathematical methods relied heavily upon the use of counting methods in order to show that an astronomical number of possible keys existed for a particu-

lar system. As systems whose strength had been so argued were repeatedly broken, the notion of giving mathematical proofs for the security of the systems was abandoned. This resulted in this technique being replaced by certification through cryptanalytic assault which tested the ability of the cryptosystem to withstand attack by a skilled cryptanalyst (Diffie & Hellman, 1976). This approach remained dominant until the availability of the computer led to the formation of a mathematical theory of algorithms which permitted an estimate of the computational difficulty involved in cracking the cipher system. The cryptographer was thus able to design various mathematical models and to employ the computer to determine which of the models was the most secure (Diffie & Hellman, 1979)(Mellen, 1978).

The introduction of electronic computers profoundly influenced cryptography and was responsible for ushering in the modern age of cryptography. However, only the practice of cryptography has been affected by the use of computers, for the underlying principles of this science have remained unchanged (Mellen, 1978). Cryptography has gained a valuable ally in the computer and it is increasingly apparent that the computer itself may well benefit from this alliance. With the evergrowing need for security and privacy, digital computers may well find themselves in need of cryptography (Girsdansky, 1971).

Codes versus Ciphers

Encryption can be achieved in one of two ways - through the use of ciphers or through the use of codes. All of the systems described thus far in this paper have used ciphers. Although

theoretically, there is little difference between the two, in practice ciphers and codes are quite distinct (Kahn, 1966). Ciphers are composed of plaintext letters of constant length and are usually character oriented or character group oriented. Encryption is performed by mapping a character, or group of characters, into a character or group of characters. Since ciphers assign substitute symbols to some given set of alphabetic letters, any message can be enciphered by a properly designed cipher (Davida, et. al., 1978). Consequently, a cipher can be used to encrypt any information or message that was never said before nor was ever anticipated as needing to be stated (Feistel, 1973).

Codes, on the other hand, are composed of plaintext letters of variable length and are not as flexible as ciphers. Unlike ciphers which ignore linguistic structure, codes are semantic in nature. They are collections of prearranged substitutions for words and phrases (Feistel, et. al., 1971). The process of encryption is accomplished by referring to a preordained table, usually called a code book, which consists of a list of letters, words, and/or phrases together with the corresponding random groups of numbers or letters called codegroups (Diffie & Hellman, 1979). As a result of codes being prearranged substitutions, only meanings thought of in advance and which can be composed from prearranged messages can be used. Therefore, the type of message which may be encoded is severely limited (Davida, et. al., 1978).

Codes are generally not well suited for computer use. Their

failure to be easily automated and more importantly, the difficulty in changing the key, or code book, if the key is compromised has resulted in codes not being widely used (Diffie & Hellman, 1979).

Basic Components of Cryptography

Essentially cryptography consists of only two types of operations: substitution and transposition (Simmons, 1979). The various enciphering schemes in use today are merely variations in the complexity of these two operations.

Substitution, which is the most elementary of the two, involves the replacement of plaintext characters with other characters, numbers, or arbitrary symbols as illustrated below.

C A N C E L R E N D E V O U S plaintext
X M A X L E P L A Q L G B Y R ciphertext

In this technique, the various characters comprising the plaintext message retain their position but lose their identity upon encryption (Katzan, 1977)(Diffie & Hellman, 1979).

In contrast to substitution, transposition techniques actually rearrange the characters in the plaintext message. This results in the characters retaining their identity but losing their position (Katzan, 1977).

T O M O R R O W plaintext
W M T O O R O R ciphertext

Substitution Techniques

Cipher systems which depend upon substitution methods are much more diverse and more widely used than those systems which employ transposition techniques. Perhaps the primary reason

for their universal appeal is their relative ease of application (Diffie & Hellman, 1979). Despite their ease in implementation, substitution ciphers offer minimal security enhancements since they are easily solved by means of frequency analysis. The cryptanalyst need only make a table listing the frequencies of the various letters, letter pairs (digrams), and letter triples (trigrams). Careful study of this table reveals those letters which occur quite frequently (the letters "e" and "t"), those which occur rather infrequently (the letters "q" and "z"), and finally, frequently occurring letter pairs which usually reflect the pairing of a vowel with a consonant. Thus through frequency analysis, the cryptanalyst is able to identify plaintext letters and their corresponding ciphertext equivalents (Diffie & Hellman, 1979).

Although substitution ciphers are vulnerable to cryptanalytic attack, the size of the alphabet plays a vital role in the ease with which the substitution cipher is broken. Too short an alphabet permits frequency analysis to be easily carried out while too long of one introduces other difficulties. If the alphabet size is increased, the compilation of a frequency table becomes a tedious task thus making this technique too prohibitive both with respect to time and expense for the cryptanalyst. Therefore, for substitution techniques to be relatively secure the key must be limited to a reasonable size - one that is neither too short and therefore vulnerable nor so long as to introduce compilation problems (Diffie & Hellman, 1979) (for further information on substitution ciphers see figure 7).

Transposition Ciphers

As with substitution ciphers, transposition ciphers are also relatively easy to decipher, however, they are not quite as vulnerable (see figure 8). Frequency analysis of the letter pairs and triples reveals the breakup of common letter pairs (i.e. the "i" and "e" of the word "believe"), thereby allowing the plaintext to be reconstructed through the search for permutations which rejoin them. This permits the recovery of the key used to transform the plaintext into ciphertext and subsequently, the plaintext itself from the cryptogram alone (Diffie & Hellman, 1979). As a result of the letter frequencies in the plaintext being invariant in cipher form, transposition techniques are seldom used by themselves due to their vulnerability to cryptanalytic attack (Katzan, 1966).

Applicability for Computer Use

Both transposition and substitution techniques are ideally suited for cryptographic processing by the computer. These two techniques can be easily implemented by means of the binary number system. The use of "n" binary digits results in the generation of 2^n distinct binary codes. Thus a block of 5 binary digits can generate 32 distinct combinations - more than adequate to encode the twenty-six letters of the standard alphabet. More combinations are made possible by increasing the size of the digit block (Katzan, 1966).

In addition to the ease with which binary digits can be represented in electronic circuitry, they have all the advantages of decimal numbers in that they can be added, subtracted,

multiplied, etc. It is this arithmetic manipulation which plays a vital role in cryptographic techniques designed for the computer (Feistel, 1973).

Message-Oriented Systems

Substitution and transposition ciphers are applicable for situations in which only short and transient messages are being transmitted. Since this type of message is characteristic of message-oriented systems, these two ciphers are well adapted for this type of system (Hsiao & Kerr, 1978).

Information-Oriented Systems

Situations which deal with large databases and long messages require the use of longer keys than are required by message-oriented systems. Although multiple short keys have been used to produce a long compounded key, this solution is not always adequate. Situations which require the easy detection of any errors need a transformation whose ciphertext is rich in bits and, therefore, sensitive to any change of a single digit position. Transformations which fulfill these characteristics provide high message confidentiality plus error detection. Substitution and transposition ciphers are unable to fulfill these needs and are therefore, inadequate. As a result, other ciphers which are better suited must be sought out (Hsiao & Kerr, 1978) (Diffie & Hellman, 1979).

Product Ciphers

Transposition and substitution ciphers are seldom used by themselves primarily due to their vulnerability to cryptanalytic attack. However, the combination of these two techniques with

other methods allows them to become important components in more complex cipher systems (Katzan, 1966).

Product ciphers consist of both transposition and substitution techniques. If transposition and substitution ciphers are properly combined and iterated a number of times, they provide a high degree of security, much more than they would otherwise provide by themselves (Feistel, 1971)(Diffie & Hellman, 1979). Product ciphers enable the construction of a strong system from simple, individually weak components. Thus the elementary schemes of substitution and transposition, which act as low-cost building blocks, can be efficiently combined to form complex ciphers (Lempel, 1979).

Principles of Confusion and Diffusion

Diffusion and confusion are two other techniques by which complex cipher systems can be generated. They are especially useful in information-oriented systems which require a flexible cipher that is capable of error detection.

The purpose of diffusion is to diffuse or spread local statistics throughout the length of the message. Diffusion attempts to eliminate some of the correlations and dependencies among the variables and in this way an intruder has to intercept larger amounts of enciphered material in order to decipher even a small portion of the message (Lempel, 1979).

The principle of confusion, on the other hand, is based upon nonlinear substitution. This principle dictates that even though large amounts of enciphered material may be intercepted, the relationship between the original message and the enciphered

version is so complex that a cryptanalyst would still find it extremely difficult to make any headway in cracking the cipher. Confusion relies upon making the functional dependencies among the related variables as complex as possible, so as to maximize the time required for cryptanalysis (Lempel, 1979). As a result, the work factor is very high and consequently, this principle offers good security measures (Difffie & Hellman, 1979).

Shannon, who was responsible for important contributions during the early theoretical development of cryptography, has argued that the alternate sandwiching of simple ciphers with the principles of confusion and diffusion, produces a cipher that is highly resistant to cryptanalytic attack (Walker, et. al., 1977). Thus the use of both of these principles is capable of producing a very flexible and secure cipher system (Diffie & Hellman, 1979).

Principles Underlying Present-Day Ciphers

Cryptograhic systems can be classified into two broad types: stream ciphers and block ciphers (Diffie & Hellman, 1976). The use of either type of cipher affects not only the strength of the algorithm but also has strong implications for computer use as well (Popek & Kline, 1979).

stream ciphers

Stream ciphers process the plaintext message bit by bit or character by character. The entire preceding portion of the message, as well as the key and the current bits are used to determine how to encipher the next bits of the message (Popek & Kline, 1979)(Lempel, 1979). Consequently, the incoming charac-

ters are not treated independently (Diffie & Hellman, 1979). In many cases, a sufficiently long substring of the key uniquely determines the succeeding elements of the key string. This allows the stream cipher to be vulnerable to cryptanalytic attack. This vulnerability, however, can be offset if special care is taken to ensure that the key string bits are a complex function of their predecessors (Lempel, 1979).

block ciphers

In contrast to stream ciphers, the basic structure of a typical present day block cipher is an iterated product cipher, with transposition and substitution as its main components (Lempel, 1979). Block ciphers divide the plaintext message into blocks, usually of fixed size with the block length ranging from 32 to 128 bits or characters. Encryption is performed by enciphering each successive block of the message on the basis of that block alone and the given key (Popek & Kline, 1979). Therefore, unlike stream ciphers, each block is operated on independently of the previous one (Diffie & Hellman, 1979).

stream versus block encipherment

Whereas it is easier to construct strong stream ciphers as compared to strong block ciphers, an unfortunate characteristic of stream ciphers deals with error propagation. A single error in any given block results in all subsequent blocks being undecipherable, and therefore, in error (Popek & Kline, 1979). This is directly related to the fact that the encipherment of the plaintext is dependent upon all of the preceding bits or characters (Diffie & Hellman, 1979).

In general, stream ciphers are less acceptable for computer use than are block ciphers. In order to update any portion of a lengthy ciphertext, not only must the relevant bits be reencrypted but also all subsequent bits of the stream as well. Block ciphers, on the other hand, facilitate updating. All that is required is to decrypt the relevant block of ciphertext, update it, and reencrypt that particular block. No other blocks need to be changed. Primarily due to their ease in application, block ciphers are usually preferred (Popek & Kline, 1970).

A Block Cipher Called DES

A prime example of a block cipher is the Data Encryption Standard or DES. Essentially, the DES is composed of transposition and substitution operations. After the information to be enciphered undergoes the transposition and substitution operations, the resulting cipher is reentered into the same enciphering process. Thus the cipher that is produced is called a recirculating product cipher. Since the DES operates only on a fixed length data block, it may be precisely referred to as a recirculating block product cipher (Branstad, 1977)(Bright & Enison, 1978).

In November 1976, the Department of Commerce approved the Data Encryption Standard as a Federal Information Processing Standard (FIPS). The adoption of the DES as a standard makes it mandatory for all Federal Government agencies to use this algorithm to ensure cryptographic protection of computer data. Only those Federal agencies which have been specifically exempted as well as nongovernmental institutions are not required

to use the Data Encryption Standard (Dinardo, 1978). With the adoption of this standard, it is hoped that this will significantly increase the use of encryption devices to protect computer systems in both the areas of communications and information storage (Branstad, 1977).

This algorithm employs a 64 bit key to encipher and decipher binary-coded data. The data is partitioned into blocks, which are 64 bits long, prior to the encryption/decryption process. Since the algorithm is made public, the key is of prime importance because it is the sole means by which the security of the data is maintained. It is this 64 bit key which is responsible for the generation of a unique block of ciphertext from each block of plaintext (Katzan, 1977). Decryption of the enciphered information is conducted through a reversal of the algorithmic process which was used to transform the original plaintext into ciphertext. The same key that was used for encryption must also be used for decryption. Therefore, unauthorized individuals who have intercepted the ciphertext and have knowledge of the algorithm can not recover the original plaintext message without knowledge of the unique key which was employed for encryption (National Bureau of Standards, 1977).

The Data Encryption Standard can be implemented in software or on a single LSI chip and can be used with any computer to encrypt and decrypt transmitted data (Hellman, 1979). Thus cryptographic transformations are made possible among different terminals, devices, etc., thereby allowing communications among diverse computer systems (Hsiao & Kerr, 1978).

Lucifer - A Model for the DES

Motivated by the growing need for data protection in their products, IBM initiated cryptographic research centering around nonlinear block ciphers in the late 1960's. This work significantly increased unclassified literature on cryptography and also produced several important cryptosystems (Diffie & Hellman, 1979). One of these, a cryptosystem named Lucifer, which was developed in 1972, was to later become the prototype for the Data Encryption Standard (Dinardo, 1978).

The Lucifer algorithm enciphers or deciphers fixed length blocks consisting of 16 bytes or 128 bits. Each block is processed independently of all other blocks and is under the control of a 128 random bit key. This key can be furnished from a magnetic strip-card or from a plug-in 16 byte read-only store module (Girdansky, 1971).

Each 16 byte block is divided into two halves prior to encryption. The top half consists of the first 8 bytes of the plaintext message while the bottom half consists of the latter 8 bytes. The following functions play a vital role in the encipherment of the data.

confusion: Eight bits are selected from the key. The value of each of these 8 bits (be it a 0 or 1) is used to determine which of two unique one-to-one, nonlinear transformations will be performed on a copy of each of the 8 bytes in the top half of the message group. This operation is known as "confusion" and the resulting 8 bytes, which are subjected to the keyed nonlinear transformations, are referred to as "confused bytes".

key interruption: The eight "confused bytes" are then added to the eight selected bytes of the key by modulo-2 addition. The purpose of this operation is to effectively interrupt the cryptanalyst's attempts to decipher the encrypted message, thus the name key interruption.

diffusion: The resulting eight bytes of the modulo-2 addition are then permuted in a random fashion. The result of this permutation undergoes pairwise convolution with the eight bytes which comprise the bottom half of the message by modulo-2 addition.

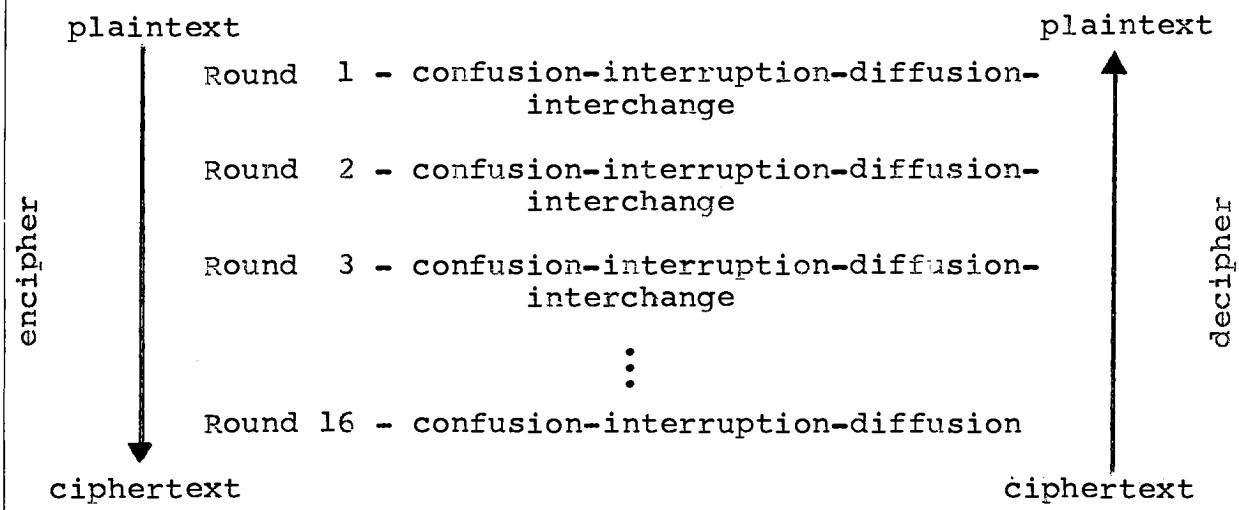
interchange: Finally, the topmost eight bytes are then interchanged with the eight bytes which form the bottom half (Feistel, et. al., 1971)(Smith, 1971).

The entire encryption process consists of sixteen rounds of confusion-interruption-diffusion operations, alternated with fifteen interchanges of the two halves. For each round, a different set of eight bits is selected from the key. The final result of the encipherment is a unique sixteen byte ciphertext which represents the original sixteen message bytes (Feistel, et. al., 1971)(Girsdansky, 1971)(Smith, 1971).

In performing these sixteen rounds, the selection of the key bits is such that each of the 128 bits, which make up the key, is used nine times - once as the control bit to govern the nonlinear transformations for confusion and eight times as a key interruption bit in each of the eight different bit positions (Smith, 1971).

In order for the entire encryption process to be reversed

for deciphering, the eight bytes which form the top half must be retained unaltered during any confusion-interruption-diffusion operation. In addition, the same key bits used for the encryption process must be retained. The only requirement is that the order of the operations be reversed. This means that the first confusion-interruption-diffusion operation executed for the decryption process must be the last one performed for the encipherment (see chart below)(Smith, 1971).



Introduction to the Data Encryption Standard

The Data Encryption Standard, which is a reduced modification of the Lucifer algorithm, operates on 64 bit blocks of data (Lempel, 1979). Like Lucifer, the DES requires that the same secret key be used for both encryption and decryption (Everton, 1978). Decryption is accomplished using the same key as that used for encryption, with the exception being that the schedule of addressing the key bits is reversed so that the deciphering process is the inverse of the enciphering process (Branstad, 1977). Thus, the invertible transformation can be

described by the following equations:

$$C = S_k(P) \qquad P = S_k\text{-}1(C)$$

In these equations, P denotes a 64 bit block of plaintext; C, a 64 bit block of ciphertext; k, the 64 bit key; S_k, the enciphering transformation when key k is used; and S_{k-1}, the deciphering process (Diffie & Hellman, 1977).

As with Lucifer, the Data Encryption Standard also relies upon sixteen rounds or iterations of a block enciphering process where each iteration is composed of the permutation of the block and the operation of some complicated function. In the DES, the complicated function involves a substitution process on small sub-blocks of the data. Both of these operations, the permutation and the complicated function, are controlled by a key which is dependent upon the round (Davida, et. al., 1978).

In actuality, only 56 of the 64 bits which make up the key are used for the encryption/decryption process. The remaining eight bits are used solely for parity error detection. Consequently, the key is divided into eight 8 bit bytes, where the first seven bits of each byte are used by the algorithm while the last or eighth bit is used to maintain odd parity (Lempel, 1979)(Diffie & Hellman, 1977).

Specifics of the Algorithm

The 64 bit block to be enciphered undergoes three major steps during the encryption process: an initial permutation, signified by IP, then to a recirculating block product cipher process which consists of sixteen rounds of a block enciphering process, and finally to a permutation which is the inverse of

the initial permutation, denoted as IP^{-1} (Branstad, 1977)(Katzan, 1977).

The first step in the encryption process is to divide the plaintext into blocks of 64 bits in length. Each 64 bit block is then subjected to an initial permutation (IP) which is performed according to the following table.

IP

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

This operation results in the 58th bit of the input block becoming the first bit of the permuted block; the 50th bit of the input block, the second bit of the permuted block; the 42th bit, the third; and so on, until the 7th input bit becomes the last bit of the permuted block. The resulting 64 bit permuted block is then used as input to a complex key-dependent computation or product transformation which consists of sixteen iterations (recirculating block product cipher process)(Branstad, 1977).

The recirculating block product enciphering process consists of a series of substitutions and transpositions. The substitutions are performed under the control of a cipher key while the transpositions are executed according to a predetermined sequence (Katzan, 1977). Prior to each iteration, the 64 bit block of plaintext is divided into two halves or data vectors. The first 32 bits of the input block make up the left

half while the remaining 32 bits form the right-hand vector, and are denoted by L_i and R_i, respectively (Branstad, 1977). In the next step, the right-hand data vector becomes the leftmost 32 bits of the output block. Thus $L_i = R_{i-1}$. The rightmost 32 bits of the input block, R_{i-1}, undergo a selection process which results in a 48 bit data block. Note that this selection process is not key dependent.

A 48 bit subkey, K_n, is associated with each iteration, thus there are 16 subkeys. This subkey is generated from the 64 bit key and represents a unique subkey which corresponds to the $_i$th iteration of the product transformation. This 48 bit subkey is then added to the resulting 48 bit data block obtained through the selection process as defined above by modulo 2 addition. The output of this step is a 48 bit block which is partitioned into eight groups consisting of 6 bits each. Each 6 bit group is further subjected to a unique substitution cipher that results in a 4 bit group. Thus the final result of this process is a 32 bit output consisting of eight 4 bit groups. This output then undergoes a permutation operation which yields a 32 bit permuted block. This 32 bit permuted block is added (modulo 2 addition) to the lefthand data vector, L_{i-1}, to yield R_i, which is the righthand data vector of the 64 bit output block (see figure 9c) (Branstad, 1977)(Katzan, 1977)(National Bureau of Standards, 1977).

The entire process outline above, is repeated 16 times and thus constitutes the major portion of the recirculating block product enciphering process. The final step in this

operation is the interchange of the left and right halves of the output from the last iteration. This result is frequently referred to as the preoutput block (Katzan, 1977).

Since the deciphering process is the exact reversal of encipherment, decryption occurs by applying the very same algorithm to the enciphered message block. The only requirement is that at each iteration of the computation, the same subkey is used for decipherment as was used during encryption. Thus the decryption process can be expressed as follows:

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

(Katzan, 1977)(National Bureau of Standards, 1977).

the cipher function

The cipher function, f, is the heart of the recirculating block product enciphering process. This function, which is used in each iteration of the product transformation, may be symbolically denoted by $f(R_i, K_n)$ where R_i represents a string of 32 bits (data) and K_n is the 48 bit subkey generated from the 64 bit key (Katzan, 1977). The output of this function is a 32 bit block obtained by the following process. The 32 bit data vector, R_{i-1}, is expanded into 48 bits as briefly described in the previous section. This 48 bit block is denoted by $E(R_{i-1})$, where E represents a function which takes a 32 bit block as input and produces a 48 bit block as output. The 48 bit block is written as eight groups of 6 bits and the bits are obtained by selecting the following input bits:

E Bit-Selection Table

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

Thus the 32nd bit of R_{i-1} becomes the first bit of $E(R_{i-1})$, and so on, until the first bit of R_{i-1} becomes the 48th bit of $E(R_{i-1})$ (for a schematic diagram of the cipher function, see figure 10)(Katzan, 1977).

Each of the functions S_1, S_2, S_3, ... , S_8, as shown in figure 11, is a unique selection function which has as its input a 6 bit block and yields a 4 bit block as output. These eight primitive functions are described below.

S_1

Column Number

Row #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

Column Number

Row #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Each selection function, S_i, takes a 6-bit block as input and yields a 4-bit result. Thus, input to the unique set of selection functions S_1 through S_8 is a 48 bit block, which is denoted as $B_1B_2B_3B_4B_5B_6B_7B_8$ where each B_i is composed of 6 bits. B_1 serves as input to the selection function S_1; B_2 to S_2; and so on. The resulting 4 bit block from a single selection function

may be denoted by $S_i(B_i)$ where S_i is the selection function and B_i is its argument (Katzan, 1977). The calculation of this 4 bit block $S_i(B_i)$ occurs as follows: The first and last bits of B_i represent a value in base 2 with a range from 0 to 3. This binary number is denoted by i. The middle four bits of B_i also represent a binary value. This value is denoted by j and is in the range 0 through 15. Using the table, S_i, which corresponds to the argument B_i, the intersection of the $_i$th row and the $_j$th column yields decimal value in the range from 0 through 15. This decimal number can be uniquely represented as a binary value or 4 bit block. It is this four-bit block which represents the output for $S_i(B_i)$ for the input B_i (Branstad, 1977)(National Bureau of Standards, 1977).

As an illustration of this operation suppose that the six bit input B_1 is the binary value 101010. The first and last bits of B_1 represent the value 2, therefore, the designated row of table S_1 is the second row and the variable i is equivalent to the value 2. The next step is to extract the middle four bits of B_i which is the value 0101. This value is the binary representation of 5, therefore, j=5 and the designated column is the fifth one. Referring to table S_1, the value found at the intersection of the second row and the fifth column is the decimal value 6. This represents the binary value 0110 and is thus the output for the selection function $S_1(B_1)$ (Katzan, 1977) (see figure 12).

The next operation to occur after the selection functions S_1 through S_8 is a permutation function, P, which uses a 32 bit

block as input to produce a 32 bit block as output. The permutation occurs according to the following table:

P

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

This 32 bit permuted block is obtained by taking the 16th bit of the 32 bit result of the set of selection functions as the first bit of the permuted block, the seventh bit as the second bit of the permuted block, and so on (National Bureau of Standards, 1977). This 32 bit block represents the output for the cipher function, f, for inputs R_i and K_n (Katzan, 1977).

the key schedule calculations

The key schedule calculations are responsible for generating the 16 subkeys, denoted as K_n, used during the encryption/decryption process. Permutation, selection, and shifting operations are employed to calculate the 48 bit subkey, K_n.

Not all of the 64 bits which comprise the key are used to calculate the subkeys. In fact, only 56 of the 64 bits that make up the key are actually used. Since the key is partitioned into eight 8-bit bytes, one bit in each byte is utilized for error detection. Consequently, the 8th, 16th, ..., and 64th bits of the key are used to assure the odd parity of each byte (Katzan, 1977)(National Bureau of Standards, 1977). These eight bits are not used in the key schedule calculations (Branstad,

1977).

The initial step in the execution of the key schedule computations is to subject the non-parity bits in the key to a permutation operation. This permutation operation may be described by the following table and yields two 28-bit blocks denoted by C_0 and D_0 (the leftmost and rightmost blocks, respectively).

Permuted Choice 1 (PC-1)

57	49	41	33	25	17	9		63	55	47	39	31	23	15
1	58	50	42	34	26	18		7	62	54	46	38	30	22
10	2	59	51	43	35	27		14	6	61	53	45	37	29
19	11	3	60	52	44	36		21	13	5	28	20	12	4
yields C_0								yields D_0						

(see figure 13) Thus, bits 57, 49, 41, ... , 60, 52, 44, and 36 specify the 28 bit block C_0 while bits 63, 55, 47, 39, ... , 28, 20, 12, and 4 represent D_0 (Branstad, 1977)(Katzan, 1977).

The generation of blocks C_0 and D_0 represent the starting point for computing the subkeys. Hereafter, the blocks C_{i-1} and D_{i-1} for i = 2,...,16 are generated according to a schedule of left shifts for the individual blocks (Katzan, 1977). Following the generation of blocks C_0 and D_0, the 28-bit blocks C_1 and D_1 are formed by circularly shifting C_0 and D_0 left one place. Subkey K_1 is then generated by selecting specified bits from C_1 and D_1. Blocks C_1 and D_1 are once again shift left one place to yield C_2 and D_2. Selected bits are chosen from these two blocks to form subkey K_2. This entire process continues for subkeys K_3 through K_{16} with blocks C_{i-1} and D_{i-1} being generated according to the following table:

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

The computation of each of the subkeys K_1 through K_{16} is done by means of a second permutation operation, denoted as permuted choice 2 or PC-2 (Katzan, 1977). Thus a particular key K_n is selected from the concatenation of C_n and D_n according to the permutation operation described by the following table:

PC-2

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

(for a schematic representation of this second permutation operation, please refer to figure 14 - a summary of the key schedule calculations is given in figure 15).

This completes a description of the Data Encryption Standard. Note that the choice of primitive functions K_n, S_1 through S_8, and P is critical to the strength of the encryption/decryption

process (Branstad, 1977)(National Bureau of Standards, 1977) (Morris, et. al., 1977).

Is the Data Encryption Standard Secure?

Since the development of the Data Encryption Standard and its subsequent adoption as a standard, much criticism has been raised as to the security of this algorithm. A current battle is raging over the question of whether or not the standard is a strong algorithm.

The alledged weaknesses of the Data Encryption Standard were first brought to the forefront a little over five years ago when Martin Hellman and Whitfield Diffie voiced their concern over the inadequacy of this standard to effectively protect sensitive data (Kolata, 1977). Dr. Hellman argues that the encryption standard is only marginally secure and consequently, is not adequate against commercial data thefts (Sugarman, 1979). Both Dr. Hellman and Whitfield Diffie believe that the standard will have to be redesigned in the next few years in order to permit it to withstand attacks from newly developed higher technology (Yasaki, 1976)(Morris, et. al., 1977).

The main criticism against the Data Encryption Standard is the belief that the key is too short. Since key size is the primary determinant of the ease with which a cipher can be broken, critics such as Dr. Hellman believe that the standard's 56 bit key (64 bits - 8 parity bits) is too short to provide protection against a brute force attack using a special large purpose computer (Branstad, 1977). Current technology makes it possible to design a computer, which employs a special search chip, that is

capable of testing one million keys per second. Thus, the implementation of a million of these search chips in parallel would make it feasible to break the Data Encryption Standard, with its $2^{56} \approx 10^{17}$ possible keys, in one day (Morris, et. al., 1977). Although the estimated cost of such a machine would be approximately twenty million dollars, due to the depreciation in the machine's cost over the next five years, the daily operating cost would drop to only ten thousand dollars. As computation and hardware costs drop even more in the near future, the cost of this code breaking machine would drop substantially (Kolata, 1977).

Hellman and Diffie claim that the security of the Data Encryption Standard can be substantially improved by increasing the key length from the current 56 bits (64 bits with the inclusion of the 8 parity bits) to 128 or even 256 bits. The use of a 128 bit key would increase the estimated cost for a brute force search to $\$2 \times 10^{25}$ with the added benefit that no foreseeable technological advances would allow the cost to be brought into reasonable range (Diffie & Hellman, 1977).

Those who support the Data Encryption Standard, such as Dr. Ruth Davis, director of the Institute of Computer Sciences and Technology at the National Bureau of Standards, argue that the current key size of the standard is more than adequate for today's technology and therefore, meets present-day needs (Kolata, 1977). Dr. Davis states that a standard's objectives are different from technological objectives and consequently, the predicted five year life-span of the DES algorithm is more than adequate. After

all, there is no guarantee that enlarging the key size to 128 bits will prevent future technological advances from cracking the cipher system. This plus the added cost of the semiconductor chip in which the algorithm is implemented as well as the number of chips required are major factors used in the argument against the implementation of a larger key (Yasaki, 1976).

Despite the fact that the arguments over the security level provided by the DES are far from resolved, it is generally felt that this standard is adequate in its simplest form for normal business applications. If a higher level of security is needed. the use of multiple encryption can be employed to compensate for the small key size (Yasaki, 1976)(Branstad, et. al., 1976).

Summary of Part One

Up until this point, this paper has attempted to provide the reader with the necessary background information needed to gain a more thorough understanding of cryptography. The previous sections have dealt with the development of cryptography from an art to a science and the subsequent transformations of cryptographic practices throughout history. Two cryptographic algorithms, Lucifer and the Data Encryption Standard, were presented so as to enable the reader to observe what is involved in the design of a cipher system. Although most cipher systems such as these, are composed of two elementary cryptographic operations, substitution and transposition, the combination of these operations produces a fairly complex algorithm.

Part Two deals with the use of encryption as a means of enhancing operating system security. The use of encipherment

within the operating system is used primarily for authentication purposes. This means ensuring that the user is who he claims to be and that only authorized individuals have access to data stored within the system. This and other related topics will be covered in the subsequent sections.

Operating System Security

Operating system security ultimately involves a multi-user, multi-programming environment (Hsiao & Kerr, 1978). Thus the primary goals of operating system security include the prevention of any unauthorized access to data stored within the system, the safeguarding of users from undesirable results triggered by their own actions, the assurance that various user programs will not interfere with one another, and the ability of different users to have different rights and abilities to cooperate with each other (Davida, et. al., 1978).

As a result of the operating system being responsible for the management and control of all computer hardware resources, operating system security is of vital importance. For this reason, the issues which deal with operating system security are critical and far reaching (Hsiao & Kerr, 1978).

Network Security Threats

A computer network is usually a rather large, single system which has been created from numerous individual computer systems. As a result, the data processing functions must now be distributed among a set of distinct systems thus decentralizing the control of data storage and processing. In addition, information which must be transmitted between the various computers within the network is subject to exposure. Forged user identification and unauthorized access to stored data by legitimate users are also problems which plague a multi-user, multi-resource environment. Consequently, these factors combine to complicate the problem of ensuring a high degree of security

within the network and may present formidable pitfalls (Bright & Enison, 1978).

Encryption and Network Security

Security problems such as monitoring of communications and user authentication may be alleviated through the use of encryption. In the past, protection against the monitoring of communication lines was guaranteed by the use of physically secure lines. This technique, however, proved to be extremely expensive and, often times, impractical. Since that time, it has been discovered that data encryption may be used as a viable alternative to secure lines (Bright & Enison, 1978)(Davida, et. al., 1978).

With regard to the problem of identification and authentication of users, user identification has usually been performed by an identification number and a secret password known only to the user. Presently, a method which relies upon the one-way encipherment of passwords is being tested. This method has been proven to be extremely useful in verifying user identification while maintaining the secrecy of the password (Bright & Enison, 1978).

Thus it can be seen that encryption may be employed as a solution to computer network security problems and consequently, a valuable tool in operating systems security (Davida, et. al., 1978). Cryptographic technology is, therefore, a relatively inexpensive and highly effective process by which sensitive data may be protected against disclosure (Bright & Enison, 1978).

Implementation of Encryption within the Network

Link encryption and end-to-end encryption are two means by which cryptography can be applied to computer networks. The decision as to which of the two methods will be implemented is dependent upon whether or not encryption is regarded as a responsibility of the network or of the users (Popek & Kline, 1979).

Link encryption is a low-level encryption which is used primarily for packet switching. All information sent through the network is encrypted and decrypted at each node through which the information passes. Consequently, all data including even address information, is encrypted (Popek & Kline, 1979).

End-to-end encryption, on the other hand, is of a higher level of integration than link encryption. In this process, the information is encrypted only once at its source with decryption occurring only after the data has arrived at its final destination. Unlike link encryption, this technique has the advantage of protecting the data throughout its entire journey (Diffie & Hellman, 1979)(Sykes, 1976).

The level of integration used within the computer network significantly influences the number of keys which must be generated and distributed as well as the amount of software which must be employed (Popek & Kline, 1979). In general, the higher the level of integration required, the greater the need for matched key pairs which must be separately distributed and/or the number of previously arranged secure channels. Although a higher level of integration frequently entails additional cost and greater complexity, a significant reduction in the amount

of properly functioning software results (Diffie & Hellman, 1979).

Network Applications of Encryption

Within the computer network, encryption may be used for both message and user authentication, private communications, network mail, and/or digital signatures (Popek & Kline, 1979)(Diffie & Hellman, 1976). Each of these areas is discussed in further detail below.

authentication

One of the foremost functions of network security is the authentication of both messages and users. User authentication permits secure communications among various participants by ensuring that the individuals are who they claim to be (Diffie & Hellman, 1976). Consequently, this eliminates the possibility of another individual masquerading as a valid system user (Popek & Kline, 1979).

Message authentication verifies the legitimacy of the message. Encryption ensures message authentication by assuming that the possession of the correct key is a primary prerequisite to participation in message exchanges while knowledge of the proper password ensures user authentication (Diffie & Hellman, 1976).

private communication

Encryption plays a vital role in permitting secure communications to occur where an insecure transmission medium is being employed. As a result, encryption has been used solely for communication purposes in the past (Bartek, 1974).

Two prerequisites for private communication are a secure

channel and the presence of all participating parties. During the initial stages of establishing a secure channel, overhead is frequently incurred. This overhead takes the form of a fairly complex key distribution algorithm which requires several messages and the interaction of all participants (Popek & Kline, 1979).

network mail

The overhead required for private communications may be impractical for network mail primarily due to the transmission of short messages which is characteristic of electronic mail. Unlike private communications, this situation does not require the receiver of the message to be present at the time of the transmission. Thus it may be possible to get lower overhead at the cost of increased queuing delays (Popek & Kline, 1979).

digital signatures

A digital signature is a means of providing evidence to a third party that a specified communication is exactly as received from a particular sender (Needham & Schroeder, 1978). Thus the author of a digitally represented message may "sign it" in such a manner that the "signature" has properties similar to a hand-written signature (Rivest, et. al., 1978). Digital signatures protect against forgery and repudiation of authorship while providing authentication at a relatively low cost (Popek & Kline, 1979).

Limitations of Encryption

Although in most cases encryption can be used to enhance security measures, there are practical limitations to encryption's

viability. These limitations are discussed below.

processing in plaintext

Most of the arithmetic operations require the data to be supplied in plaintext form. Since the data may not be in encrypted form, additional emphasis is placed upon the internal controls of the operating system to maintain adequate security measures for the plaintext data. Various suggestions have been made for solving this problem including one such solution which employs an encoding algorithm which is homomorphic with respect to the desired arithmetic operations. Although this technique would permit the desired operations to be performed on the encrypted values, known encoding schemes which fulfill the necessary properties are not very secure algorithms. It is feared that strong algorithms with these properties can not be constructed. Therefore, since data must be processed in plaintext, other means are necessary to protect data from being compromised while the data is under the control of the operating system (Popek & Kline, 1979).

revocation

The methods used for selective revocation of access to data are very complex. Currently, the only known means of revoking access to a particular piece of data is to render the corresponding key void. This can only be accomplished by decrypting the data and re-enciphering it under a different key. This action, however, is not very selective since all old keys are invalidated. Hence new keys must be redistributed to all individuals for whom access is still permitted (Popek & Kline,

1979).

protection against modification

Despite the fact that encryption can not guard against inadvertent or intentional modification of data, it can serve as a means of detecting that modification. The inclusion of a number of check bits within the encrypted data permits the comparison of these bits (upon decryption) with expected values. If a match does not occur, then the data is invalid due to modification. The use of encryption to detect data modification, however, is troublesome in situations where a long period of time has elapsed before a particular data item is referenced. In cases such as this, data modification may occur unnoticed until long after the incident has been carried out. Thus, detection of modification may not provide adequate protection for sensitive information (Popek & Kline, 1979).

key storage and management

The problem of key storage arises in those situations in which a specific data item needs to be protected independently of all other data items. This requires the use of a unique key to encrypt the particular data item. This problem becomes quite troublesome when it is necessary to protect numerous long-lived data items separately. In situations such as this the key storage and management problem becomes formidable (Diffie & Hellman, 1976)(Popek & Kline, 1979).

Public-Key Cryptosystems versus Conventional Cryptosystems

The security of cipher systems in the past, relied solely upon the secrecy of the entire encryption process. Modern

cipher systems, however, have dispersed this shroud of secrecy thus enabling the algorithm to be made public without compromising the system's security (Diffie & Hellman, 1976). These ciphers - the conventional cryptosystems of today - consist of a key which is supplied along with the plaintext message as input to the enciphering algorithm. The security of this system resides in maintaining the secrecy of the key. Although the conventional cryptosystems have provided adequate data protection in the past, the development of an even newer encryption procedure provides an even greater promise for added security. This new class of encryption procedures is based on a group of mathematical problems which are characterized by computational intractability. This type of system was first proposed by Ralph Merkle, Whitfield Diffie, and Martin Hellman, and is called a public-key cryptosystem (Hellman, 1979). Unlike any previous cipher system, the public-key cryptosystems permit the revelation of not only the algorithm but the key used for encipherment as well (Diffie & Hellman, 1976)(Hellman, 1979).

conventional encryption

The encryption/decryption process within conventional cryptosystems can be described by the following equations:

$E = F(D,K)$ - data encryption $\qquad D = F'(E,K)$

D signifies the data to be encoded, K is the key, E is the resulting ciphertext, and F represents a function. The second equation permits the recovery of the original plaintext data from the ciphertext and thus signifies decryption. In this

case, F' represents the inverse of the function F. Since the security of conventional cryptosystems is dependent upon maintaining the secrecy of the key, the use of the functions F and F' is valuable only if it is impractical to recover the original plaintext message from the ciphertext without knowledge of the corresponding key (Popek & Kline, 1979).

To gain a better understanding of this type of cipher system, equate the conventional cryptosystem with a strongbox having a resettable combination lock. Prior to any message exchanges, a key must be mutually agreed upon by both the sender and the receiver. This key, which is actually a sequence of numbers, serves as the combination of the lock. Once a key has been selected, the sender may then place his message in the strongbox, set the combination, and lock the box. Assuming that the cryptosystem (or strongbox) is secure, no third party who intercepts the box during its journey to its final destination will be able to break the lock and discover the contents. In this way, conventional cryptosystems prevent the extraction of information from an insecure channel and also the modification of data within the channel (Hellman, 1979).

Despite the fact that conventional cipher systems are capable of protecting sensitive data, several disadvantages of this type of system are particularly troublesome to users. Maintaining the secrecy of the key has proved to be the major drawback of this system. Since both the receiver and the sender must agree upon a key prior to any message transmissions plus the fact that the entire security of this system is dependent

upon the secrecy of the key, it is vital that the key be transmitted by means of a secure channel (i.e. a trusted courier). This key distribution problem compounds itself in situations where the sender and receiver have had no prior communication or where the network is extremely large. If a large network must rely upon couriers to distribute the needed keys, this technique soon becomes too prohibitive and thus, impractical (Hellman, 1979)(Diffie & Hellman, 1976).

Another disadvantage of conventional cipher systems involves message authentication. As a result of the same key being used for both encryption and decryption, there is no guarantee that the receiver will not send himself messages which appear to have been generated by the sender. Forgeries such as this, could create serious problems within the network. Consequently, conventional cryptosystems can not ensure the authenticity of the message in the same manner that the exchange of signed documents can. For this reason, conventional cryptosystems are being abandoned in favor of public-key cryptosystems which are capable of solving both the key distribution problem and the problem of message authentication (Hellman, 1979).

public-key encryption

Public-key cipher systems define the encryption/decryption process according to the following equations:

$E = F(D,K)$ - for encryption $\quad D = F'(D,K')$ - decryption

As before, D signifies the data to be encoded, K is the key, E is the resulting ciphertext, and F' is the inverse function of F. Note that although the equations which describe the process

by which the data is enciphered or deciphered, are similar for both public-key and conventional cryptosystems, there does exist a distinct difference between the two techniques with regard to the decryption process (Diffie & Hellman, 1978). Unlike conventional cryptosystems, the same key is not employed for both encryption and decryption. In public-key cipher systems, the key K' is used for deciphering purposes. This key is not equivalent to key K which is used to encipher the data. Furthermore, K' can not be derived from K. As a result, public-key cryptosystems are a variation of conventional encryption methods which are believed to have certain advantages over standard algorithms (Hellman, 1979).

Unlike the conventional cryptosystem where both the sender and receiver must mutually agree upon a common key, the public-key cipher system generates two distinct keys - an enciphering key,K, and a deciphering key, K'. It is this generation of two keys which is said to be the major advantage of this type of system over the standard or conventional cipher system (Diffie & Hellman, 1976)(Diffie & Hellman, 1979).

In contrast to the conventional system where the security of the system is solely dependent upon maintaining the secrecy of the key, this system's security is dependent upon the computational infeasibility of deriving K' from K. If an extremely high work factor is associated with the calculation of the deciphering key, K', given the encryption key, K, then each user can safely publish his enciphering key in a public file and still ensure the secrecy of his deciphering key which is private

information(Denning, 1979). Consequently, anyone who wishes to transmit a message to a particular individual merely enciphers the information with that person's public enciphering key K and sends the ciphertext. Only the intended receiver is capable of deciphering the message since he is the only individual who has knowledge of corresponding secret deciphering key K' (Hellman, 1979).

As an illustration of how this type of system functions, consider once again a strongbox, only this time the box has two combinations - one to lock the box and one to unlock it. By permitting all combinations which lock the strongbox to made public, any individual can transmit a message or lock the information in the box. The information stored within the box is guaranteed to safely reach its destination since only the individual who owns the strongbox and who has knowledge of the two combinations will be able to open the box. Thus this system's simplicity eliminates the need for key distribution prior to any message exchanges as well as the possible construction of "digital signatures" which would prevent the forgery of messages (Hellman, 1979)(Diffie & Hellman, 1976).

Many public-key cipher systems permit the use of either the function F of F' for encryption purposes while the inverse function is used for decryption. Thus, the data can be encrypted using F' and decrypted using F or vice versa. This property plays an important role in both key distribution and "digital signatures". The RSA algorithm, which exemplifies this characteristic, is proof the use of either function for encryption

results in a strong ciphertext (Rivest, et. al., 1978).

Key Management

Within the network, various users are able to communicate with one another only if they possess matching keys for both the encryption and decryption process. Thereby, admission to the communications channel is determined solely by the possession of the appropriate key. Without the key, the channel is unavailable. Since the task of the network is to provide as many communication channels as is possible, the manner in which the keys used to access the channels are generated is of vital importance (Popek & Kline, 1979). As a result of keys being distributed over the same channels as the data is transmitted, the only secure means of distributing the keys is through the use of encryption. This practice is viable only if the number of keys to be distributed is limited in number so as to ensure that the security of the keys may be maintained (Popek & Kline, 1979)(Matyas & Meyer, 1978).

Conventional-Key Distribution

The most frequently employed technique for conventional-key distribution can be described as follows. Prior to the transmission of data, the potential participants mutually agree upon a pair of matched keys. A single host machine is designated as the key distribution center or KDC for the time period of the desired connection. One of the potential participants in the communication, A_i, requests the transmission of the matched key pairs to all participants in the communication, including himself. This serves as the first step in initiating the connection.

It is then up to the KDC to determine whether or not to authorize the connection. If the desired connection is approved, secure messages containing the key and status information are then sent to each participant over prearranged channels. After this step has been completed, data can then be transmitted over the newly established communications channel (Popek & Kline, 1979). Variations of this approach are described in the subsequent sections of this paper.

centralized key control

This approach employs a single KDC for the entire network and for this reason is the simplest key distribution technique possible. The number of matched key pairs required for this method is equal to the number of entities comprising the network. Thus, if the network is composed of n distinguishable entities, then n prearranged matched keys are required for communication purposes (Popek & Kline, 1979).

One drawback to this type of key management method is the fact that network reliability may be endangered if communication with the key distribution center is not possible. This situation may occur if the node on which the KDC is located is not functioning properly or if the network itself breaks down. Therefore, in order for a single KDC to be applicable for distributed networks, the underlying communications topology must be that of a star with the KDC located at the center of the communication network. However, there still exists the possibility that the KDC may malfunction thus causing a complete breakdown. The only alternative solution to this drawback involves the maintenance

of redundant KDC's which are used solely during failures of the main facility. These redundant facilities can be located at any site which supports a secure operating system and which provides appropriate key generation facilities.

An illustration of centralized key control may be described as follows. Suppose that X, a member of the network, wishes to communicate with Y, who is also a member of the network. Prior to any message exchanges, both participants X and Y must select their own unique, secret keys. These two keys, designated as K_x and K_y respectively, are known only to the KDC and to themselves. In order to establish a connection, X sends a message to the KDC requesting the construction of a communication channel with Y. Enclosed within this request X also transmits an identifier of some sort, such as a number, which will be used by the KDC at a later time. If the KDC determines this request as being a valid one, a new key K_c will be sent to participant X along with the identifier (which was initially transmitted by X), a copy of the request, and information which will aid in identifying X as a legitimate participant to Y. This new key K_c is used solely for communication purposes with Y. To ensure that the secrecy of this key is maintained, the transmission from the KDC to X revealing the new key is encrypted with the key selected by participant X, K_x. This guarantees that only X can receive the message and consequently, serves to verify the authenticity of the message. By having the KDC return X's identifier, this permits X to check the identifier in order to ensure the transmission is authentic and not a forged message thus verifying that the original request

as made by X was not altered before reception by the KDC (Popek & Kline, 1979).

After X has received the new key from the KDC, X may then proceed to transmit data to participant Y. The first step is for X to send to Y the data from the KDC intended for Y. This data includes not only the connection key K_c but proof of X's identity as well. To ensure that only Y is able to decipher the information, the entire transmission to Y is enciphered in Y's secret key, K_y. Upon decryption of the message, Y now knows that K_c is the communication key, X is the other participant in this communication, and that this information came from the KDC. To ensure that the message just received by Y is authentic and not a replay of some previous message, Y sends a unique identifier to X encrypted by key K_c. Upon receipt of the identifier, X performs some operation and obtains a result which is in turn returned to Y. This permits participant Y to verify that the message is indeed a current one and is not a replay of some previous message (see figure 16)(Popek & Kline, 1979)(Matyas, 1978).

fully distributed key control

This type of key management technique permits every "intelligent" node in the network to serve as a KDC at one time or another. In the situation where there are three participants, A_1, A_2, and A_3 which reside at nodes N_1, N_2, and N_3, then only the key distribution centers at each of these nodes need to be involved in the establishment of a communication channel for this particular case. Of the three KDC's, only one is responsi-

ble for selecting the key used for communication purposes. This KDC will then transmit the necessary information to the remaining two KDC's. Each of these KDC's will, in turn, decide whether or not to authorize the desired connection. Upon reaching a decision, each KDC will transmit its reply to the originating KDC. If the connection is permitted, the keys will be distributed to the potential participants.

With this method, only those nodes which are directly involved in the communication link (i.e. those nodes which support the intended participants) need to be properly functioning. As a result, this is a major advantage for this key management scheme. For this scheme to function effectively, each KDC must be able to communicate with the participants at its own node in a secure fashion. This means that if the user software is forced to use the network only through those channels which have been encrypted, that each host has the authority to enforce its own security policy. Thus, this method is particularly useful for decentralized organizations (Popek & Kline, 1979).

hierarchical key control

This technique distributes the key control function among "local", "global", and "regional" controllers. In the case of the "local" controller, he is permitted to communicate only with those entities which are located in his immediate locale. If communication is desired with entities that are located outside of his locale, the "local" controller must contact the "regional" controller for that particular area. Thus each local KDC is responsible for prearranging channels for the potential partici-

pants in his area while regional controllers are responsible only for secure communications with local controllers.

If all of the participants in a channel are within the same region, then the connection procedure is the same as for centralized key control since only the local controller is involved. If the participants are not located within the same region, then it is necessary for the appropriate regional controllers to request permission from the global controller for the construction of a communication channel.

This technique is quite flexible in that it generalizes to multiple levels as in the case of a very large network and is analogous to national telephone exchanges where the exchanges play a role very similar to the key distribution center. As a final point, let it be noted that any of the three levels, be it local, regional, or global, has the ability to select the keys to be used during communication.

Unlike the previous key management schemes, the failure of any component of the distributed key control facility results in only the users local to the failed component suffering hardship. Due to the considerable importance of the regional and global controllers to the architecture of the network, these two types of controllers are frequently duplicated so that the crash of a single node will not segment the network.

After briefly describing three types of key management schemes, it can be noted that there are strong similarities among them. The differences which do exist between the various methods can be reduced even further by designing hybrid schemes

which take on some of the advantages of each. Whereas centralized key control is a degenerate case of hierarchical control, fully distributed key control can be viewed as a variant of hierarchical key control. In the case of fully distributed key control, each host's KDC acts as a local key controller for that host's entities and communicates with other local key controllers to establish connections. In this instance, the communication is direct and a regional controller is not required (Popek & Kline, 1979).

Public-Key Based Distribution Algorithms

It is believed by many that public-key algorithms may be better suited for key distribution methods than conventional algorithms primarily due to their simplicity. Since the key K', which is used for decipherment, can not be derived from the enciphering key K, a user A after obtaining a matched key pair (K,K'), can publicize his enciphering key K. Thus another participant, B, who wishes to send a message to A can do so by employing the publicly available key K. In order for A to respond to B's message, A merely employs B's public key.

Although this technique appears to be much simpler than the previous means used to establish secure communication channels, in actuality, it is not. Despite the fact that no secure dialogue is required with the key distribution center as a preliminary step in establishing a channel and the appearance that no key distribution problem exists nor that any central authority is required to establish the channel, some form of central authority is indeed needed (Popek & Kline, 1979)(Needham & Schroeder,

1978). Thus the protocol involved is no simpler nor any more efficient than the one based on conventional algorithms.

In public-key algorithms, the safety of the public-key scheme is totally dependent upon the selection of a correct key by the sender. Thus, if the key listed in the public directory is incorrect, then the public-key encryption scheme can not adequately protect the data. Furthermore, since the directory listing the public keys for all of the system users will need to be constantly updated, maintenance of the directory will be time consuming. This requires a means of carefully maintaining the directory while at the same time ensuring that all changes have been carefully authenticated and that the correct public key is sent out upon request (Popek & Kline, 1979).

To illustrate how public-key algorithms can be incorporated in key distribution centers, a modified version of the previous example for this section is presented below. In this example, participants A and B each have a public key which is known only to the authority and a private key known only to themselves. The authority also has a public key which is known to all participants and a private key known only to the authority (Popek & Kline, 1979)(Needham & Schroeder, 1978).

In order to initiate a communication channel with user B, A sends to the authority a time-stamped message requesting communication with B. The authority responds by sending A the public key of B, a copy of the original request, and the time stamp. This entire transmission is encrypted using the authority's private key. Participant A proceeds to decrypt this message to

verify the authenticity of the message. The time stamp guarantees that this is not an old message from the authority containing a key other than B's current public key, and the copy of the request permits A to verify that the original plaintext message was not modified in any way (Popek & Kline, 1979).

Now that A has knowledge of B's public key, communications between the two participants, A and B, can now begin. The first step is for A to properly identify himself to B. In order to do this, A sends his name and an identifier to B. This entire transmission is encrypted in B's public key. B then repeats the first two steps described in the paragraph above with the authority to retrieve A's public key. B then sends to A the identifier just transmitted along with an additional identifier, both of which are encrypted with A's public key. A can then proceed to decipher the message and verify that he is communicating with participant B. A now sends back the new identifier to B so that B can be sure that he is talking to participant A (see figure 17).

The use of certification has been proposed as one means of eliminating the need to request the public key from the central authority for each communication. In this scheme, a certificate is nothing more than a user/public-key pair plus some additional information used for certification purposes. Thus, each user's public key can be transmitted to him via a certificate. Consequently, if the user/public-key is stored as a signed message from the central authority and the user wishes to communicate with other users, it is only necessary for the individual to

send the certificate to each of the desired participants. This permits each user to check the validity of the certificate, using the certifying information, and the eventual retrieval of the public-key. As a result, this scheme requires that the central authority be used only once during the request of the initial certificate (Popek & Kline, 1979).

Although certification initially appears to be the perfect solution, drawbacks do exist. Prior to using the certificate, the user must decrypt it in order to verify the authenticity of the signature. This also makes it mandatory for the recipient of the certificate to have a secure and correct way of storing the key. Thus, even with certificates, an internal authentication mechanism is required (Popek & Kline, 1979).

Unlike conventional encryption algorithms, public-key systems are not as flexible in permitting the merging of protection policy issues with key distribution. This results in making the implementation of protection policy checks more difficult. In conventional systems, if two users are not authorized to communicate with one another, the key controller merely refuses to distribute the keys. This is not the case with public-key systems where the enciphering keys are public knowledge. As a result, if the addition of a protection check is so desired, modifications must be made to the existing system (Popek & Kline, 1979)(Needham & Schroeder, 1978).

Public-Key versus Conventional-Key Distribution for Private Communication

As mentioned previously, the security of the conventional-

key distribution scheme is totally dependent upon maintaining the security of the secret key used for encryption and decryption whereas for the public-key distribution method the safety of the deciphering key is vital. In both techniques an equivalent amount of secure storage is mandatory. Both public-key and conventional-key algorithms require approximately the same amount of overhead to establish a connection.

At first glance it may appear that the software required to implement the public-key authority would be less complex than that needed for the KDC. The simplicity of the software would make it much easier to certify the correct operation of this scheme, which in turn would give the public-key method a significant advantage over the conventional-key method. This is not the case, however. In spite of the fact that the contents of the authority need not be protected against unauthorized reference in public-key encryption since the enciphering keys are made public, the keys used in the authentication protocol between the KDC and the user must be protected against reference (Popek & Kline, 1979). In reality, the amount of software needed to ensure the proper functioning of the authority is not substantially different from that required for a secure KDC. If all of the KDC keys are stored in encrypted form using a KDC master key and are decrypted only when needed, the security of the KDC is reduced to maintaining the secrecy of the master key and of the individual keys when they are in use (Popek & Kline, 1978)(Popek & Kline, 1979). This situation is identical to the secure storage and protection required by the deciphering key during

its use in the public-key encryption scheme (Popek & Kline, 1979).

Unlike the public-key method, it is possible for the KDC to eavesdrop or even generate supposedly valid messages in the conventional-key scheme. This is made possible by the fact that the KDC is issued a conversation key. This problem can be remedied through the addition of certified code to the KDC which oversees the destruction of the conversation keys immediately following their distribution. This eliminates the minor distinction between the two schemes. Consequently, both the conventional- and public-key algorithms are quite similar - moreso than was initially expected at first glance (Popek & Kline, 1979)(Hellman, 1979)(Needham & Schroeder, 1978).

Public-Key Cryptosystems

Public-key systems are based on trapdoor one-way functions. A one-way function is characterized by the fact that it is invertible and although easy to compute, it is computationally infeasible to solve the equation $y = f(x)$ for almost all x in the domain of f. This means that given a complete description of the function f unless certain private information which was employed in the design of the function is made available, it is computationally infeasible to compute the inverse of f, f^{-1}. Thus the characteristics of a trapdoor one-way function f and its inverse f^{-1} make it ideally suited to fulfill the requirements of a matched key pair (for encryption and decryption) as needed for a public-key cryptosystem (Hellman, 1979)(Lempel, 1979).

One particular class of trapdoor one-way functions on which

to base public-key cryptosystems is believed to be quite promising in terms of providing strong algorithms. This class is referred to as NP or nondeterministic, polynomial-time problems and is characterized by the fact that as the size of n increases, the number of computational steps required to solve the problem increases in proportion to an exponential function of n (i.e. 2^n) in contrast to the number of steps required to check a possible solution increasing only in proportion to a polynomial function of n (i.e. n^2). Since the exponential function increases far more rapidly than the polynomial one, any solution which requires exponentially increasing amounts of computer time is impossible to implement for even moderate-sized problems. The two public-key cryptosystems described below are based on NP problems (Hellman, 1979).

Merkle-Hellman Scheme

The Merkle-Hellman scheme is based on the knapsack of subset sum problems where given a set of numbers $a_1, a_2, \ldots, a_n$, and a sum C, the solution involves determining which of the numbers add up to C (Lempel, 1979).

Messages in this scheme can be described as binary n-vectors. Thus, $M = (b_1, b_2, \ldots, b_n)$ where M represents the message and each b_i signifies either a value of 0 or 1. The initial step in this method is to convert the message into a string of binary numbers. The message is then divided into blocks of n bits each with each block being designated by $X = (x_1, x_2, \ldots, x_n)$. Since each key is a trapdoor knapsack n-vector symbolized by the equation $A = (a_1, a_2, \ldots, a_n)$ where a_i is a positive

integer, the public directory consists of a listing of the ordered set of n values which comprise each user's key. Note that the number of elements in vector A is the same as the number of bits per block or the length of vector X (Hellman, 1979).

The next step is to form the dot product C = A•X where A represents the enciphering vector A and X designates the message block. Thus, $C = a_1x_1 + a_2x_2 + \ldots + a_nx_n$. The resulting sum C is the information that is transmitted over the insecure channel. Should the message be intercepted, it is the intruder's task to recover not only X from result C, but A as well. The recipient of the transmission is confronted with the same task as the intruder, however, his task is simplified since he has additional information concerning the design of the function and knowledge of the deciphering key (Lempel, 1979)(Hellman, 1979). As an illustration of how the knapsack problem works, the steps which comprise this NP problem will be described in detail below. Please refer to figure 18 for further information on this example.

Suppose that the following message is to be encoded: 'MERGER SET'. In order to transform this message into a binary string, a 5-bit binary alphabet is used for this example. Thus, 5 bits are allocated for each letter as illustrated below.

A	B	C	D	E	F	G	H	I
10110	01010	10111	01101	01001	01011	11011	11010	00101
J	K	L	M	N	O	P	Q	R
10101	11110	00011	01110	11000	11100	10011	00001	10001
S	T	U	V	W	X	Y	Z	
11111	10100	00100	10000	11001	00110	11101	00000	

By substituting the proper 5-bit value for the specific

letter, the message is transformed into the following binary string:

M	E	R	G	E	R	S	E	T
01110	01001	10001	11011	01001	10001	11111	01001	10100

Upon consulting the public directory listing the various users' enciphering keys, suppose that the intended receiver's enciphering key is represented by the vector defined as A = (2292, 1089, 211, 1625, 1283, 599, 759, 315, 2597, 2463). Thus a_1 = 2291, a_2 = 1089, a_3 = 211, and so on. Since there are ten elements comprising vector A, n = 10. Therefore, the first block of information, which consists of the first n bits of the binary plaintext, is equivalent to X_1 = 01110 01001, the second block is designated as X_2 = 10001 11011, and so on with the final block X_5 = 10100 00000. (Note that only five bits remain to fill the last block. Zeroes are used to round out the remaining five bits).

Encipherment of the data proceeds according to the equation $C = a_1x_1 + a_2x_2 + \ldots + a_nx_n$. This results in C_1 = (2292 x 0) + (1089 x 1) + (211 x 1) + (1625 x 1) + (1283 x 0) + (599 x 0) + (759 x 1) + (315 x 0) + (2597 x 0) + (2463 x 1) = C_1 = 6147. Proceeding in a similar fashion for the remaining blocks of data C_2 = 1089 + 1283 + 599 + 759 + 2597 + 2463 = 9993; C_3 = 1089 + 1283 + 599 + 2463 = 5434; C_4 = 2292 + 1089 + 211 + 1625 + 1283 + 759 + 2463 = 9722; and C_5 = 2292 + 211 = 2503. Thus encipherment of the plaintext message "Merger Set" produces the numerical string 6147 9993 5434 9722 2503 (Hellman, 1979).

All of the known methods for solving the knapsack problem

require adding up all of the 2^n possible subsets of the a_i's to determine which subset yields sum C. As the number of elements n increases, the computational difficulty of solving this problem also increases. For this reason, the knapsack problem is classified as belonging to the class of NP problems (Lempel, 1979) (Hellman, 1979).

If the elements of vector A are randomly chosen, then the recovery of X is impossible, even for the receiver of the transmission. In this respect, the knapsack problem appears to be a one-way function. In order to be able to easily derive X from C, it is necessary to specifically design vector A so that the knowledge of some additional information facilitates the solution of the knapsack problem (Hellman, 1979). However, it must be noted that not all NP problems lend themselves to the insertion of a trapdoor. The knapsack problem does not fall into this category and therefore, a trapdoor can be designed since there exist certain vectors for which a solution can be easily calculated (Simmons, 1979). One of these special vectors, designated as A', can be disguised so as to appear an an ordinary-looking vector A in the public file of enciphering keys. Thus, the trapdoor information concealed in this vector enables a relatively easy knapsack problem involving vector A' to be disguised as a complex knapsack problem involving A (Hellman, 1979)(Lempel, 1979).

The creation of such a trapdoor can be illustrated as follows. During the generation of vector A, suppose that A' is chosen so that $A' = (a_1', a_2', \ldots, a_n')$ and each element a_i' is greater

than the sum of the preceding elements $a_1' + a_2' + \ldots + a_{i-1}'$. If A' = (3, 5, 11, 20, 41, 83, 169, 340, 679, 1358) then a_2' = 5 is greater than a_1' = 3; a_3' = 11 is greater than $a_1' + a_2'$ = 8. If C' = 1563, then C' = A'•X' for some binary vector X' = $(x_1', x_2', \ldots, x_n')$ or $1563 = 3x_1' + 5x_2' + 11x_3' + 20x_4' + 41x_5' + 83x_6' + 169x_7' + 340x_8' + 679x_9' + 1358x_{10}'$.

The problem of deciphering the ciphertext would be equivalent to solving a knapsack problem if it were not for the fact that the special property of vector A' permits the easy solution of X'. Element a_{10}' = 1358 which is less than the value 1563 or C'. According to the special property of vector A' the sum of the nine remaining elements of A', a_1' through a_9', must be less than 1563 or C'. Since this condition holds, a_{10}' = 1358 is a member of the subset sum and x_{10}' = 1. Given x_{10}' = 1, the equation C' = A'•X' can be rewritten as $1563 = 3x_1' + 5x_2' + 11x_3' + 20x_4' + 41x_5' + 83x_6' + 169x_7' + 340x_8' + 679x_9' + 1358$. Subtracting 1358 from both sides of the equation results in the following: $205 = 3x_1' + 5x_2' + 11x_3' + 20x_4' + 41x_5' + 83x_6' + 169x_7' + 340x_8' + 679x_9'$. The problem is now reduced to determining which of the 9 remaining elements, x_1' through x_9', add up to the sum 205. Continuing as before, a_9' = 679 which is greater than 205, therefore, this element is not included as a member of the subset sum and x_9' = 0. Element a_8' = 340 which is also greater than 205 and the value of x_8' = 0. Element a_7' is less than 205. This means that a_7' is a member of the subset sum. Consequently, x_7' = 1. This process continues until all values have been calculated and X' = (0, 1, 1, 1, 0, 0, 1, 0, 0, 1).

This represents the original message block X (Hellman, 1979).

The design of a simple knapsack vector A' is relatively easy. The difficulty arises when the recipient of the data attempts to get from vector A' to A and back again. To accomplish this task two large random numbers, w and m, are selected. The vector A is then generated according to the equation $a_i = a_i'w$ modulo m for each i from 1 to n. If w = 764 and m = 2731 and if a_4' of vector A' is equal to the value 20, then $a_4'w = 15280$. Dividing the value 2731 into 15280 results in the value 5 with a remainder of 1625. Therefore a_4 of $a_4'w$ modulo m = 1625. Using this equation, it is possible to derive vector A' from vector A and vice versa (Hellman, 1979).

Merkle and Hellman recommend the use of a knapsack vector whose length n is greater than or equal to 100. For the case where n = 100, it is suggested that the value for m be chosen uniformly from the integers between $2^{201} + 1$ and $2^{202} - 1$. The a_i values should be chosen uniformly from the integers between $(2^{i-1} - 1)2^{100} + 1$ and $2^{100 + i-1}$ where i = 1, 2, ... , n (Hellman, 1979).

There is some hesitation to accept the MH scheme as a secure algorithm solely on the basis of its membership within the class of NP complete problems. Eligibility for membership in the NP complete class is determined on a worst-case basis. Since it is not clear how difficult a worst-case sample of an NP-complete problem is, it has been argued that this scheme may not be as strong an algorithm as some believe it to be (Hellman, 1979) (Lempel, 1979).

In conclusion, it can be seen that modular arithmetic plays a vital role in the generation of public-key cryptosystems, primarily because this type of arithmetic can turn a continuous function into a discontinuous one. This permits the introduction of a great deal of confusion into the calculation of the inverse functions. Thus, the introduction of modularity into the generation of the knapsack vector A prevents the recovery of the vector A' for anyone not possessing knowledge of the private transformation parameters w and m (Hellman, 1979).

Rivest-Shamir-Adleman (RSA) Scheme

This scheme is also an NP problem. It is based on the problem of discovering all prime numbers which evenly divide a very large number. Each user is responsible for randomly selecting two large random prime numbers p and q along with a pair of positive integers E and D, respectively. A value n is obtained by taking the product of p and q. This value n and the positive integer E are then placed in the public file as the user's enciphering key. Thus, the encryption key consists of the pair (n,E) while the corresponding decryption key is defined as (n,D). In the case of the decryption key, the value of D is kept secret (Hellman, 1979).

Both messages and cryptograms in the RSA scheme are represented as integers in the range 0 and n-1. After the message has been converted into a string of numbers, the sender then proceeds to divide the string into blocks P_1, P_2, ... , P_n where each P_i is an integer value between 0 and n-1. The receiver's public key (n,E) is located in the directory and the sender

computes the ciphertext value $C_i = P_i^E$ modulo n for each plaintext number P_i. Given p = 5, q = 11, and E = 3, the enciphering key for this particular example is (3, 55). If the value of P_1 is 2, then according to the equation $C_i = P_i^E$ modulo n, $C_1 = 2^3 = 8$ modulo 55 (Lempel, 1979)(Hellman, 1979).

The strength of the RSA algorithm is based upon the computational infeasibility of factoring the product of two large prime numbers. In order to decipher the ciphertext message, the receiver must employ n and a secret deciphering key D which is derived from the prime factors p and q (Lempel, 1979).

Euler's tolient function, which can be expressed as $\emptyset(n) = (p-1)(q-1)$ when n = pq, plays an important role in understanding the principle behind the derivation of the deciphering key. All the functions within the RSA system are calculated modulo n with the sole exception of the exponent which is determined by modulo $\emptyset(n)$ (Lempel, 1979).

The arithmetic properties of Euler's tolient function always guarantees that a multiplicative inverse D of E modulo $\emptyset(n)$ exists. Thus, ED modulo (p-1)(q-1) is equal to the value 1. It is this inverse D which is the secret deciphering key for the RSA scheme. To decipher a ciphertext message, C_i^D modulo n is computed for each ciphertext number C_i. Since $C_i = P_i^E$ modulo n, C_i^D modulo n is equivalent to $(P_i^E)^D = P_i^{ED}$ modulo n. Since the exponent is calculated by modulo $\emptyset(n)$ and ED modulo $\emptyset(n) = 1$, P_i^{ED} modulo n equals P_i^1, or simply P_i. This illustrates the transformation of ciphertext into plaintext by raising the ciphertext to the Dth power and reducing modulo n (Hellman, 1979).

Modularity plays a dual role in this scheme. It not only thwarts the recovery of the secret deciphering key D from the public enciphering key (E,n) but it also is responsible for preventing a direct recovery of the plaintext from the ciphertext. The difficulty with which D can be computed from the enciphering key (E,n) is determined by the difficulty of factoring n into p and q. To ensure that no one but the intended receiver will be able to recover the plaintext message from the ciphertext, it is recommended that p and q be chosen such that n is approximately 200 digits long (Hellman, 1979)(Lempel, 1979).

As an illustration of the RSA scheme suppose that the following data is available: p = 47, q = 59, D = 157, and E = 17. Since $\emptyset(n) = (p-1)(q-1)$, $\emptyset(n) = (46)(58) = 2668$ while $n = pq = (47)(59)$ or 2773. The message to be enciphered is "MERGER SET" and the transformation of the plaintext message into a string of numbers proceeds according to the following substitution.

∅(blank)	A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12	13
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	14	15	16	17	18	19	20	21	22	23	24	25	26

The plaintext message "MERGER SET" is converted into the numerical string listed below.

M	E	R	G	E	R		S	E	T	plaintext
13	05	18	07	05	18		19	05	20	numerical conversion

The numerical string is then broken into blocks of equal length $P_1, P_2, \ldots$, so that each P_i is less than the value of n. If each block is four digits in length, then the maximum value of

P_i is 2626 (i.e the numerical representation of the letters ZZ). Since 2626 is less than the value of n which is equal to 2773, two letters will be enciphered per block. This results in $P_1 = 1305$, $P_2 = 1807$, $P_3 = 0518$, $P_4 = 0019$, and finally, $P_5 = 0520$. To encipher the first block, P_1 is raised to the Eth power and then reduced modulo n. Therefore, $C_1 = P_1^E$ modulo n = 1305^{17} modulo 2773. The same procedure is followed for the remaining blocks P_2 through P_5. The resulting values obtained for C_1 through C_5 represent the ciphertext for this particular message. Recovery of the original plaintext message is easily accomplished by raising each ciphertext block to the power D = 157 modulo 2773 (see figure 19 for further details concerning this example) (Hellman, 1979)(Lempel, 1979)(Simmons, 1979).

Whereas the MH scheme did not possess a straightforward digital signature feature, the RSA system permits direct generation of a digital signature. The RSA public-key cryptosystem allows the sender to generate his unique signature $S_i = P_i^D$ modulo n for each P_i. Note that the use of the user's secret deciphering key plays an important role in the creation of the digital signature. After all of the signatures $S_1, S_2, \ldots, S_n$ have been calculated, each signed message block (P_i, S_i) is then enciphered by the sender using the public enciphering key of the receiver. This step ensures the privacy of the communication (Lempel, 1979)(Simmons, 1979).

Upon receipt of the ciphertext, the receiver recovers the signed message block (P_i, S_i) using his secret key D. He then looks up the sender's public key (E,n) and computes S_i^E modulo n

for each i. The order of application of D and E is unimportant. Since D and E are inverse operations and $S_i = P_i^D$ modulo n, then S_i^E modulo n = P_i. This guarantees that the message is authentic and guards against repudiation of authorship. The digital signature is dependent upon both the sender and the message being transmitted and thereby ensures that neither the receiver nor a third party can modify the message without destroying the validity of the signature (Hellman, 1979)(Rivest, et. al., 1978) (Lempel, 1979).

User Authentication

Within every network there must exist some means by which user authentication is guaranteed. This authentication mechansim guarantees that the would-be user is in fact the individual he claims to be. At the present time, the most frequently employed means of establishing user authentication is to issue a unique and secret password to each user of the system. This technique requires the maintainance of a hidden Password Table which is nothing more than a listing of each system user and his corresponding password. Only the authentication program is permitted access to this table. Since the system accepts the individual as an authentic user if and only if he is able to present the proper password, it is vital that the security of the passwords be maintained (Purdy, 1974). Therefore, it is the duty of the operating system to ensure that all system users, with the exception of the System Administrator, are prohibited access to the Password Table (Evans & Kantrowitz, 1974).

There are several disadvantages to this scheme, however.

The major drawback is the dependence upon the entire access control mechanism for the security of the system. Thus, the success of this scheme is dependent upon the correct operation of a very large part of the operating system. Another drawback is that anyone who can obtain physical access to the computer, may be able to obtain a listing of the Password Table. Any listing of the table, even if it is obtained for a valid purpose, may be inadvertently seen by an unauthorized individual. Finally, in order to implement this authentication scheme within a network, safeguards must already exist which protect against the unauthorized reading of files (Purdy, 1974).

A proposed scheme which permits authentication without compromising security is outlined below. As in the previous method, the potential user of the system requests access to the system by presenting his name and secret password P. The validation program applies some function H to the given password. This results in the value of function H being computed at some point P or H(P). This computation is then followed by a look-up operation in the Password Table for the entry E which corresponds to the individual's name. If a match exists between E and H(P), then the individual is accepted as an authorized user of the system and the user is said to be authenticated (Purdy, 1974).

This authentication process is based on the assumption that only the individual issued the password P knows the value of P that gives rise to the tabulated E. Thus, this scheme is based on using a function H which the would-be-intruder is unable to invert. Even if the intruder has knowledge of function

H and has access to the Password Table, he is still unable to penetrate the system unless he can invert H to determine the input value which produces the desired result. The use of a function H of this type makes it necessary to use an approach other than mere brute force to penetrate this scheme (Purdy, 1974) (Evans & Kantrowitz, 1974).

There are two possible ways in which the function H may be chosen. One approach is to select a function H which is very hard to invert mathematically. The validity of this scheme is dependent upon mathematical analysis. The alternative approach is to select a function that is computationally hard to invert. This scheme can not be analyzed mathematically nor can the function be analyzed. The subsequent paragraphs describe an authentication scheme which employs a function that is computationally hard to invert (Purdy, 1974).

An extremely large family of functions is a prerequisite for this scheme. The password determines which of these functions F_p will be chosen. This results in F_p being dependent upon password P in an extremely complex manner (Purdy, 1974).

The computation of the function H is performed in J cycles with each cycle having as input a value derived from the previous cycle. The first cycle uses the original password P to generate a new input value for the second cycle. In this manner, each cycle converts a value from the password space into a new value. This process continues J times with the final cycle producing the value H(P). It is this value H(P) which is compared with entry E in the Password Table.

In this scheme, x designates the input value or parameter for each cycle while F_x denotes the function calculated by the cycle. The parameter for each successive cycle is derived by applying the function NEXT X to the parameter of the previous cycle. As a result of each cycle employing a different input value, each cycle is computationally unique (Purdy, 1974).

To further ensure the security of this scheme, each cycle, in turn, is composed of successive applications of K scrambling functions f_k. Each scrambling function replaces the current value by a new one and is parametrized by x, the input value for a particular cycle, and/or by the original password P. Furthermore, each scrambling function is repeated m times, where m is determined both by the current x and P (Evans & Kantrowitz, 1974). (Please see figure 20 for further details on this scheme).

This scheme is particularly desirable primarily because it is extremely difficult for any intruder to penetrate the system. Given that the table entry is E, g was the last scrambling function that was used, Q was the parameter for g, and that g was executed m times, then $E = g^m (Q)$. Even if the intruder has knowledge of E,g, and g^{-1}, which is the inverse function of g, he still lacks any knowledge of m. Since m is dependent on the original password P and is independent of Q and E, the intruder can not gain any further insight to the problem. Successive application of the inverse function g^{-1} to E sheds no light on the situation. The resulting values are of absolutely no help to the intruder. Even if the intruder was fortunate enough to obtain Q he probably would not recognize it. Consequently,

there is no easy manner in which one can go backward from E to the original password P. The problem becomes increasingly more difficult as both the number of different scrambling functions f_k and cycles increases (Purdy, 1974).

Various possibilities exist for the possible families of functions which comprise the scrambling functions f_k. This paper will consider the implementation of a polynomial over a prime modulus for f_k.

Given n, a_1, a_2, ... , a_n are integers, then the polynomial p(x) can be defined by the following equation:

$$p(x) = x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$$

If the value of x is greater than or equal to 1 and less than or equal to P, where P represents a large prime number, then the function f(x) represents a unique number. Thus $f(x) \equiv p(x) \pmod{P}$ for all values f(x) which are greater than or equal to 1 and less than or equal to P. Note that '≡' symbolizes congruence. This means that f(x) - p(x) is exactly divisible by P. The function f(x) is said to be congruent p(x) modulo P. Given the value of f(x) is 63, p(x) = 1, and P = 31, then $63 \equiv 1 \pmod{31}$. Since 63 - 1 is exactly divisible by 31, f(x) is congruent p(x) modulo P (Purdy, 1974).

The degree of degeneracy of f(x) at no time exceeds the degree n of the polynomial p(x). This is a very desirable characteristic of the function f(x). As an illustration of this property suppose the $f(x_i) = Y$ for all values of i greater than or equal to 1 and less than or equal to d. Given the

congruence $x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n - Y \equiv 0 \pmod{P}$, d solutions exist (as determined by x_i). If P is a prime number then no more than n solutions are possible. This is related to the fact that polynomials of degree n can not have more than n roots (for real numbers only).

As a result of polynomial root-finding modulo P being an active research area, it is necessary that some precautions be taken to maintain the security of the scheme. Currently, all of the algorithms devised to determine the root of a polynomial modulo P require at least $cn^2(\log P)^2$ operations where c is greater than or equal to 1 and the polynomial is of degree n. To ensure the security of the password scheme relying upon a polynomial to a prime modulus, it is necessary to choose an astronomically large degree n while at the same time avoiding special forms of polynomials (such as polynomials in x^K where K is greater than 1). The selection of a large value for n may result in the computation time for f(x) increasing. By using a sparse polynomial $p(x) = x^n + b_1x^{n_1} + b_2x^{n_2} + \ldots + b_kx^{n_k} + a_{k+1}$, where $n > n_1 > n_2 > \ldots > n_k \geq 1$, the computation time for f(x) can be substantially reduced (Purdy, 1974).

Given x, f(x) can be calculated by a sequence of operations consisting of the multiplication of two numbers (i.e. U and V), the addition of two numbers, or reduction modulo P.

If the prime P is of the form $2^q - a$, then the types of operations dealing with the multiplication of two numbers and reduction modulo P can be effectively combined by observing that $(U = 2^qV)(Y + 2^qZ) = UY + (2^q - a)(VY + UZ) + a(VY + UZ) +$

$(2^q - a)(2^q + a)VZ + a^2VZ \equiv UY + a(VY + UZ) + a^2VZ$. Added benefits result if q is related to the word size of the computer. In general, fewer than ck log n operations will be required if the size of constant c is small. This is based on the assumption that the powers of x are computed by the binary method (Knuth, 1969). Since multiplication usually requires c $\log^2$ P basic machine operations, the total computation time required is c'k log n $\log_2$ P. The value of c' is dependent upon the type of computer being used. The computation time can easily be reduced while at the same time the time required to penetrate the scheme, c $n^2(\log P)^2$ and cP/nm, can be made astronomically large (Purdy, 1974).

The polynomial $f(x) \equiv p(x) = x^n + a_1x^{n_1} + a_2x^3 + a_3x^2 + a_4x + a_5 \pmod P$, where $P = 2^{64} - 59$, $n = 2^{24} + 17$, $n_1 = 2^{24} + 3$, and a_i are randomly selected 19 digit numbers, illustrates a polynomial to a prime modulus which has been tested on an IBM 360/75 and a PDP-10.

If the number of assigned passwords is 1000, then the likelihood of cracking the system by trial and error is relatively small. It has been estimated that approximately 10^3 attempts must be made before the system can be compromised. The threat of compromise by a polynomial root-finding modulo P algorithm is also very small. The number of operations required to crack the scheme is $n^2(\log P)^2 \approx 10^{14}(19 \log 10)^2 > 10^{16}$. If it can be assumed that a machine operates at the speed of 10^6 operations per second, then 10^{10} seconds or 400 years would be needed (Purdy, 1974).

File Encryption

Although used primarily for communications purposes, encryption techniques are also applicable to data stored on removable or fixed media. Unlike communications applications where the keys are changed frequently, file encryption requires the enciphering key to be maintained for as long as the data is considered to be sensitive. This requires the addition of extra security measures which will guarantee that the keys are being properly protected (Sykes, 1976). Other distinctions between encryption techniques used for stored data and private communications are outlined below.

1. Only a single copy of the key is needed for file encryption. Encryption techniques when applied to private communications require as many keys as there are participants.

2. Relatively short keys must be used for file encryption since more time is needed to encipher and decipher the data. Archival files of previously used keys may have to be maintained.

3. Special control characters do not present a problem like they did with data transmission.

4. Data that is being transmitted over a channel may not be modified at any time during transit. File encryption, on the other hand, permits the updating of files at any time (Walker & Blake, 1977).

Media or file encryption is primarily used to guard against physical theft of the data as well as ensuring that authorized users will be allowed access to only those files they are priviledged to see. Access protection below the file level can also be implemented by enciphering those fields of the record which are confidential. This technique guarantees that only authorized users will be able to view the sensitive areas (Sykes,

1976)(Walker & Blake, 1977).

protection of removable media

Essentially there are three ways of providing protection for removable media. Encryption devices may be installed in each tape or disk drive, in the peripheral control units, or in the input/output controller (see figures 21a, 21b, and 21c). Placing the encryption devices in the peripheral control units may be far superior to installing these devices in each drive since fewer encryption devices are required. A substantially large number of encryption devices are required when they are placed in the disk and tape drives (Keys & Clamons, 1974).

When the encryption devices are placed in the peripheral control unit, this control unit is responsible for setting, enabling, and/or disabling the corresponding encryption device. Since transfers in the peripheral control unit are tagged as data or control, only data which is to be recorded by a peripheral device is enciphered. All control and status information is ignored. This is in direct contrast to the sitation where each drive has its own encryption device. In this scheme everything contained on the tape or disk is enciphered. In figure 21b the peripheral control unit does not encipher the record identifier and key fields found on magnetic disks since this information must be interpreted by the control unit during search operations (Keys & Clamons, 1974).

Placing the encryption device in the input/output controller may be even less desirable since this results in the encryption device being more difficult to use. Enabling and disabling

the encryption device is now dependent upon whether control, status, or data is transmitted as well as which peripheral is receiving. Transfers for unit record devices (i.e. line printer, card reader, console, etc.) should not be encrypted (Keys & Clamons, 1974).

encryption for internal processing

Unlike the aforementioned methods which provide protection against physical theft of the tape or disk, the techniques described in this section prevent unauthorized attempts to read the tape or disk. Protection is guaranteed by a software-loadable key which is stored in a portion of the software system. This section can be accessed only by authorized users (see figure 22a)(Keys & Clamons, 1974)(Bartek, 1974).

Initially a user requests access to a specified data segment. This request passes through the access control checks and is either accepted or rejected. If access is granted, the key is obtained from a key list along with a pointer to the desired data segment. The key is transmitted to a decoder/encoder mechanism which deciphers the data, performs the proper arithmetic operation, and then enciphers the information and once again places it back into the file. One advantage of this scheme is that it can be easily modified to permit sharing among various users of the system. The individual wishing to share a portion of his file merely encrypts that portion under a different key. The key is then distributed to all users authorized to access that portion of the file. Thus through double encipherment certain portions of the file are accessible only by

individuals who possess the double key (see figure 22b)(Bartek, 1974).

This method's security is dependent upon the correct operating system routine obtaining the key. If the key is compromised through hardware errors or covert means, then the scheme fails. The intruder is then able to decipher the encrypted information. One means of eliminating this problem is to permit the user to supply a portion of the key whenever he logs-on to the system. Thus even if the intruder has knowledge of the encrypted file and the key through breach of access control, he is still unable to recover the plaintext data (Bartek, 1974).

A variation of this scheme would require a different key to be used each time a user logs-on. This "one-time" key would enhance the security benefits of this scheme while at the same time requiring additional hardware to be installed. The hardware would be responsible for re-enciphering the file according to the new key after the arithmetic operation is performed (see figure 22c)(Keys & Clamons, 1974)(Bartek, 1974).

Data stored within the system must be able to be easily updated. Re-encipherment of the entire file must not be necessary each time a change is made to the file. This implies the use of a cipher technique which facilitates the updating of selected portions of any file. A partial solution for this problem is the selection of a new starting key for each 1000-word block or page. This method not only significantly reduces the amount of re-enciphering needed for any changes to the file but also permits a simple normal iterative encrypting algorithm to

be employed (see figure 22d). Any updating that does occur would require only a single page to be re-enciphered (Bartek, 1974).

data encryption in the main store

Encryption of data in the main store can be carried out by installing an encryption device between the CPU and the main store (see figure 23). The handling of control information for the I/O devices and the data for the printer can be done in any of the following ways.

1. The channel programs and data can be stored in encrypted form in the main store. The software needed for this approach may be more complex as a result of the program being responsible for turning the encryption device on and off as needed. The CPU must also be able to distinguish between data destined for the printer and data destined for tape, to be printed at a later time.

2. The encryption devices can be installed within the I/O controllers. These devices are responsible for decoding the channel programs and data when needed (see figure 23b).

3. The encryption device may be placed in the main store unit. This permits the keys and control information to be transmitted at the same time that the addresses are sent (see figure 23c).

Placement of the encryption device between the CPU and storage (as in figure 23a) presents some difficulties since it may be difficult to control the encryption device. This scheme restricts the encryption device to operating solely on small fields. This in turn, limits the type of encryption algorithm that can be used. Since it is characteristic of the CPU to make one storage request after another using different keys for each request, implementation of several "key registers" would be beneficial. Each "key register" would be associated with the instruction counter and address register. Whenever the corre-

sponding address registers are changed the appropriate keys would be loaded and unloaded. Such a mechanism could also be installed in the I/O controller (Keys & Clamons, 1974).

The use of encryption techniques within the main store has several advantages. In paging systems information is frequently transferred between the main store and the secondary store. Consequently, if encryption is used to protect the data in the main store, these systems will not find it necessary to encrypt or decrypt the data on each transfer. The paging mechanism will no longer be required to keep track of the keys. Another added benefit of this scheme is the simplification of the security kernel. The tables associating user identification with keys must still be protected regardless of the use of encryption. It is also necessary to protect programs that manipulate this table along with other programs and tables (Bartek, 1974)(Keys & Clamons, 1974).

Network Mail

Network mail essentially consists of short messages which are to be transmitted as soon as possible to the recipient. Unlike private communications, the recipient of the message need not be present. If the intended receiver is not currently logged on to the system, then a system process or daemon is responsible for storing the message until it can be delivered to the proper person. To ensure the security of the message, it is desirable for the daemon never to access to the plaintext message. This means that the encrypted message is transmitted to the daemon which in turn places the encrypted data directly into the

recipient's mailbox. At log-in time, the receiver may then proceed to decrypt his mail (Popek & Kline, 1979).

The conventional-key algorithm described earlier may also be used for network mail. The only modification needed is the deletion of the last two messages which guarantee that a current channel has been obtained. Since the recipient of the message may not be present these messages are superfluous. In order to send mail to another party, the originator of the message requests a key K. The KDC (key distribution center) responds by sending key K plus a copy of K encrypted with the receiver's secret key. This transmission is appended to the encrypted mail and sent to the desired receiver. The daemon delivers both the encrypted message and key to the recipient's mailbox. The receiver is thereby able to decrypt the message at his earliest convenience (Popek & Kline, 1979).

As with the conventional-key algorithm, the authentication process may be dropped from the public-key algorithm. In this situation, the originator of the message obtains the receiver's public key. The message is enciphered and transmitted to the desired party. The daemon is responsible for delivering the encrypted mail to the specified individual. Only the recipient of the message can decipher the information since no one else knows his secret key (Popek & Kline, 1979).

In both the conventional- and public-key protocols, a previously transmitted message may be re-transmitted as current mail. Since the authentication steps have been eliminated from both protocols, the possibility of an individual receiving

duplicate mail is quite possible. If this situation is not acceptable, then records must be kept of previous mail. These records will aid in detecting the transmission of previously sent messages (Popek & Kline, 1979).

Although both protocols guarantee that only the intended recipient of the mail will be able to recover the original message, the receiver can never be certain of the identity of the sender. The use of digital signatures is one means of ensuring the authorship of the message (Popek & Kline, 1979).

Digital Signatures

Digital signatures may be generated by either a public-key or conventional-key algorithm. The paragraphs which follow outline the methods used to produce digital signatures.

With regard to the public-key algorithm, the originator of the message enciphers the information with his private key. The data is then transmitted to the desired party. Prior to decryption, the recipient of the message requests a copy of the sender's public key from the central authority. The key is transmitted to the receiver who then proceeds to recover the plaintext by using the sender's public key to decipher the information. In this scheme, the central authority is responsible not only for retrieving the appropriate public key but also for securely storing the values of all the public keys. This includes any public keys which are not currently in use but which may be needed should a dispute arise over an old signature (Popek & Kline, 1979).

Rabin proposes the use of any strong conventional-key

algorithm to create a digital signature. One of the initial steps in this scheme involves an explicit agreement between the two parties prior to the exchange of signed correspondence. If one wishes to eliminate the need for this explicit contract, it is possible to include authentication protocols in this scheme which serve as a central authority. As with public-key algorithms, an adjudicator is required for all challenges (Rabin, 1978).

Both public-key and conventional-key algorithms as described above suffer from the problem of repudiation of authorship. This means that the author of the signed message may at any time disavow his signature merely by making public his secret key. The result of this action causes all previously signed messages displaying that particular signature to be invalid. Since the private key is known, any individual is capable of creating a message with that individual's signature. Rabin's scheme limits the amount of damage which can be caused by repudiation of authorship via key compromise. Since his scheme employs a different key for each message, no one key is ever repeated. In this manner, even if a key is compromised only the corresponding signature is affected. The security of the signatures based on the other keys is still maintained. This feature need not be unique to only the conventional-key algorithm. The addition of a protocol capable of changing the keys for each message would provide the same benefits to any public-key method as it does for Rabin's scheme (Rabin, 1978)(Popek & Kline, 1979).

It is apparent that the problem of repudiation exists any

time private information needed to validate a signature is in danger of being compromised. Thus, alternate solutions must be developed which can eliminate this problem. One proposal involves the inclusion of a time-stamp. This would prevent the author of the signature to disavow earlier signed correspondence should he later reveal his private key. Such modifications to both the conventional-key and public-key algorithms are discussed below.

a conventional-key approach using network-registry-based signatures

This scheme is based upon the insertion of a trusted interpretive layer (i.e. a software and/or hardware "unit") between the author and his signature keys. All units in the network are collectively organized to provide digital signature facilities. The total collection of units is referred to as a distributed network registry or NR. Communication between the various components of the registry is made possible through a low-level link-style encryption protocol. A digital signature may then be generated as follows. The originator of the signature identifies himself to a local component of the network registry. After authentication procedures have been carried out the message to be transmitted is sent to the NR. This transmission includes the message, a request for a digital signature, and the name of the desired recipient of the message. The NR, in turn, computes a characteristic function of the message, author, recipient, and current time. The resulting characteristic value is then enciphered with a key known only to the NR. This produces a

"signature block" which is transmitted to the desired recipient. (The characteristic function operates on the plaintext message, author, recipient, and current time to produce a characteristic value. This function is characterized by the fact that given the plaintext message, the function, and the resulting characteristic value, it is extremely difficult to determine another plaintext message which produces the same characteristic value. Consequently, this function is very similar to the one-way cipher functions used to protect passwords.) (Needham & Schroeder, 1978).

This scheme must have adequate safeguards built into the system which can ensure the safety of the keys used to encrypt the signature blocks. The characteristic function which operates on the plaintext message, author, recipient, and current time should require the compromise of multiple components before the validity of the signature is affected (Popek & Kline, 1979).

notary-public -based signatures

The use of a public-key algorithm to create digital signatures requires the implementation of a number of notary public mechanisms. Each mechanism must operate independently of all others. In this scheme, the originator of the message appends his signature to the data and transmits it to one of the notary public mechanisms. Here the message is time-stamped and the entire message plus time-stamp is enciphered a second time. The second encryption process is the public notary's way of signing the "signature block". The resulting block is then returned to the originator of the message. He then places the

necessary plaintext information around the doubly signed correspondence. The block is now ready to be transmitted to the desired party (Popek & Kline, 1979)(Popek & Kline, 1978).

Upon receipt of the encrypted message, the receiver verifies the notary's signature. This is accomplished by deciphering the block using the notary's public key. The signature block is then deciphered using the public key of the originator of the message thus permitting recovery of the plaintext message (Popek & Kline, 1979).

Repudiation of authorship is prevented in this scheme by having each notary time-stamp his signature. Thus the originator of the message can not invalidate his signature by publicizing his key. If the notary public returns a copy of each notarized message it processes to the author's permanent address, then the possibility of a forged message being transmitted is eliminated. This prevents the author of the signature from claiming that his key was compromised without his knowledge and selective messages forged (Popek & Kline, 1979).

Success of both the network-registry-based scheme and the notary-public solution is dependent upon the use of multiple facilities. If only one notary exists, then this approach is no more secure than the use of a single NR. Redundant facilities reduce the danger of key compromise (Popek & Kline, 1978).

notary-public versus network-registry-based signatures

Both schemes are similar with respect to their reliance upon a trusted interpretive layer. It is this trusted mechanism which is the basis of both signature algorithms. As with previous

signature algorithms, the security of the signatures is dependent upon the ability of the two methods to protect the keys in the future (Popek & Kline, 1979).

The notary-public and network-registry-based signature algorithms represent a definite improvement over previous schemes. Unlike earlier protocols, the originator of the signature can not repudiate the signature at will. The addition of time-stamps prevents this from happening. As a result of these modified schemes being composed of several components which collectively provide digital signature facilities, the failure of several of the components is necessary before a signature becomes invalid. Earlier protocols resulted in the signature being invalid when a single failure occurred (Popek & Kline, 1979).

Execution Time Requirements for Encryption Algorithms

Despite the fact that encryption techniques are deemed valuable for protecting sensitive data, little information has been gathered concerning the cost in terms of CPU time required for the encryption/decryption process. For this reason, Friedman and Hoffman performed a study dealing with execution time requirements for specific encryption algorithms. The purpose of their study was to provide more extensive and replicable data on the cost of encryption techniques (Friedman & Hoffman, 1974) (Walker & Blake, 1977).

In this experiment, the information to be enciphered was a portion of text which was entered in alphabetic form from cards. The binary representation of the text was then added by means of modulo two addition with the corresponding bits of the key.

The encryption algorithm was coded in both assembly language and Fortran for purposes of comparison. In order to minimize the effect of any external influences upon the experiment, the entire encryption process was performed in main core memory (Friedman & Hoffman, 1974).

Using a CDC 6400 computer five different tests were performed to measure the CPU time. Initially a test was made to measure the time required to transfer the test data from one location to another without encipherment. These results were later used as a base for determining the execution time overhead incurred by the other four tests. The remaining four tests included encipherment with a one-word or constant key (a 60-bit word was added modulo two to each 60-bit word of data), encipherment with a 125-word key (a periodic key 125 words in length was added modulo two to each successive 125 words of data), double key encipherment (two periodic keys, 125 and 123 words in length, were added modulo two to the plaintext data), and encipherment using a pseudo random key (Friedman & Hoffman, 1974).

To aid in interpreting the results of this experiment, the concept of an "enciphering time coefficient" was introduced. This coefficient was defined to be the ratio of time required to encipher the data versus the time needed solely to transfer the test data from one location to another without encipherment. Encryption of data with a one-word key in assembly language resulted in an enciphering time coefficient of 1.00 - a very small time penalty. The pseudo-random key cipher, on the other hand, turned out to be the worst assembly language case with an

enciphering time coefficient of 4.21. This method was four times slower than the initial test which merely fetched and stored the test data. The coefficients for the Fortran encryption algorithm were found to be significantly higher than those for the assembly language routines (see figure 24). Thus the results of this study indicated that Fortran took approximately four times as long as those same techniques written in assembly language. Not only is more time required to encipher data using a Fortran program as compared to an assembly language program, but the ratio of encipherment time to merely transfer the test data from one location to another is significantly greater in Fortran than in assembly language. The results of this experiment therefore indicate that the programming language must be considered to have a significant effect upon the time and cost of enciphering techniques (Friedman & Hoffman, 1974)(Walker & Blake, 1977).

Conclusion

The low operational cost of small computers as compared with larger centralized ones coupled with a more advanced technology dealing with the interconnection of these small computers has resulted in a significant increase in the growth of computer networks. These networks easily facilitate organizational growth as well as permitting the decentralization of computing resources and information. Both of these factors are highly desirable in meeting organizational needs (Popek & Kline, 1979).

The security of such computer networks is now being questioned since the underlying hardware can not be assumed to be secure. Since the communication channels used by the network are not

under the physical control of the user, precautions must be taken to ensure the safe transmission of the data. Care must also be taken to guarantee that files stored within the network have not been modified. Only authorized users should have access to those data segments they are priviledged to access. As a result the growth of computer networks has led to concern over the issues of privacy, security, and integrity of information exchange (Popek & Kline, 1979).

The government's success in using cryptographic techniques to protect sensitive data has led commercial and private sectors to take a closer look at data encryption. As a result encryption techniques are being used to transmit and store data over media which are believed to be insecure. Data encryption techniques are being applied to communications, removable media, and internal processing. Encryption algorithms are currently taking on a vital role within computer networks. A strong encryption algorithm that can not be easily compromised is therefore a prerequisite to the development of a secure network. The ability to integrate encryption methods into the operating system and applications software which are part of the network are other important areas which must be considered very carefully (Popek & Kline, 1979).

Encryption techniques may be incorporated within the network through hardware, firmware, or software. If the computer network is small and only a few communication lines are needed to transmit data between computers, a separate cryptographic device can be installed on each individual line. In contrast,

data that is transmitted to and from the user and the computer will probably be deciphered only upon receipt of the message. In general, the time required to encipher/decipher the message at the terminal is faster than that required for transmission rates. Any encryption/decryption techniques performed at the computer are usually carried out by means of a software program. Thus the means of implementating the encryption/decryption process is dependent upon the type of application required (Walker & Blake, 1977).

A strong and flexible encryption/decryption algorithm should make the intruder's task as difficult as possible. This means that the time and money required to penetrate the system should by far exceed the value of the information being protected. Given knowledge of the algorithm the intruder should still be unable to penetrate the system without possession of the appropriate key. A flexible key should be used. Thus if a key is ever compromised, another key can be easily substituted. The use of changeable keys also permits each user to employ his own separate code. Simple encryption/decryption techniques should be used so that excessive hardware and/or software can be eliminated. Any transmission errors that result should not cause the entire message to be re-transmitted. Error detection and correction schemes must be employed which do not interfere with security measures. Finally, the encryption/decryption process must be transparent to the user. No special operations should be required by the user to initiate the process (Bartek, 1974).

In the past encryption has been used primarily for data

communication applications. Cryptographic techniques with regard to file protection have received little attention primarily because this area is more problematic than communication applications. The principal difficulties encountered with file encryption deal with the maintenance of the cryptographic keys for different files. The integrity of the key must be ensured for as long as the data it is protecting is considered to be sensitive (Anderson, 1972). Thus file encipherment is effective only where the integrity of the key can be maintained and where the cryptographic technique is itself reasonable complex (Anderson, 1972) (Walker & Bruce, 1977).

Prior to employing encryption techniques to preserve the security of information within the computer, one must carefully evaluate the degree of security encryption will provide as well as its limitations. Those parts of the computer network which are capable of performing encryption and decryption must be determined along with the cost/performance penalty introduced by cryptographic techniques (Keys & Clamons, 1974).

File encryption while protecting data from being accessed by unauthorized users does have its disadvantages. Encryption does not guarantee that the files are "write protected". This means that the danger does exist that the files may be written over and thus destroyed. A back-up file system is still required. Nor is file encryption able to completely remove the possibility that the system will be penetrated by a professional code breaker. The sole means of eliminating the danger of key compromise is to periodically change the keys. This requires

that all files be copied and re-enciphered using the new keys (Keys & Clamons, 1974).

The Future of Cryptography

Encryption techniques promise to enhance security measures within computer networks. A secure environment is possible only through the inclusion of encryption techniques with other security measures. Encryption by itself is not sufficient to protect a network (Bartek, 1974).

As a result of the software implementation of enciphering techniques being relatively low in cost plus the low overhead of fast hardware devices, data encryption is a highly satisfactory cost-effective means of providing data security (Bartek, 1974) (Keys & Clamons, 1974). At the present time inexpensive high-speed devices when properly included within the computer network are capable of providing data protection for tapes and disks. Encryption in the main store may be feasible within the near future as hardware technology improves. The development of large scale integration (LSI) will enable data encryption to occur within the CPU. The design of more superior encryption techniques and faster, more inexpensive circuits promises to broaden the future of cryptographic techniques (Keys & Clamons, 1974).

As a result of encryption technology being used primarily by the government, the majority of information on this topic is classified information. The ultimate success of security architecture using encryption is dependent upon the willingness of government agencies to help develop the algorithms necessary for commercial applications (Keys & Clamons, 1974)(Bartek, 1974).

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

Figure 1: This example illustrates the manner in which cipher alphabets can be generated from a plaintext alphabet. Initially, the first cipher alphabet is obtained by means of substitution. The remaining

cipher alphabets are generated from the initial cipher alphabet by sliding them in relation to that primary alphabet.

Each cipher alphabet is designated by a keyletter as shown in the furthermost lefthand column. The primary purpose of the keyletter is to designate which alphabet is to be used to encipher each successive letter of the plaintext message (Katzan, 1977)(Kahn, 1966).

H A R D W A R E H A R D W A R E H A R D W A R E H A R D	key word
D E M O N S T R A T I O N T O B E H E L D T U E S D A Y	plaintext message
P Z N N B W O J H G L M B G P V X D V W N G J U E J R B	ciphertext message

Figure 2: Each letter of the keyword designates the particular cipher alphabet which is to be used to encipher the plaintext message. For example, referring to the cipher alphabets generated in figure 1, the letter "D" is to be enciphered using the "H" cipher alphabet, the letter "E" using the "A" cipher alphabet, and so on until finally the letter "D" is enciphered using the "Y" cipher alphabet. This is how the ciphertext message is formed (Kahn, 1966).

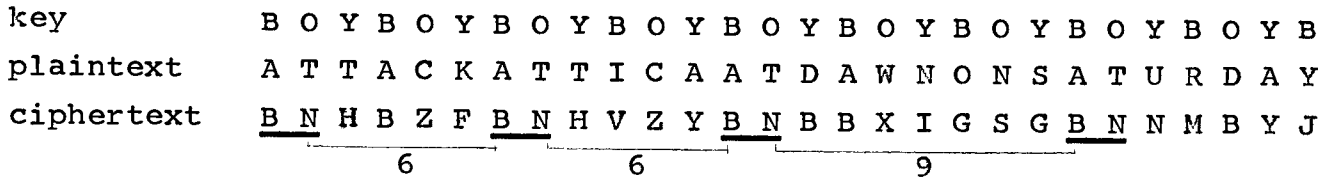


Figure 3: In this illustration, the word "AT" occurs four times in the plaintext message. In all instances, "AT" is enciphered by the cipher alphabets designated by the keyletters "BO". Referring to figure 1, the "B" alphabet transforms the letter "A" into the letter "B" and the "O" alphabet, the letter "T" into the letter "N".

In this example, the repetitions of the ciphertext occur every six letters for the first three encipherments of "AT" and at nine letters for the last encipherment. Since these values represent multiples of three, the keyword is three letters long.

Thus, by determining the number of letters between repetitions of ciphertext and their common factor, the keylength can be calculated (Kahn, 1966).

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

Figure 4: The plaintext message and the text of the book are written one over the other as shown above. Referring to figure 1, the message is enciphered - i.e. the "T" cipher alphabet is used transform the letter "A" into the cipher letter "T" and so on.

	7	2	8	4	6	0	3	5	8	4
	E	N	V	E	L	O	P	E		
9	A	B	C	D	F	G	H	I	J	K
1	M	Q	R	S	T	U	W	X	Y	Z

plaintext	M	E	E	T	I	N	G		T	O	M	O	R	R	O	W		A	T		D	A	W	N
	17	07	07	16	95	02	90		16	00	17	00	18	18	00	13		97	16		94	97	13	02
key	1	31	4	77	90	6	2		17	25	19	97	42	37	31	15		9	11		16	80	52	21
	18	08	01	83	85	08	92		23	25	26	97	50	45	31	28		96	27		00	77	65	23

ciphertext 18080 18385 32526 97504 53128 96270 07765 23

Figure 5: The "one-time system" uses a nonrepeating series of digits as the key in this illustration. Initially the plaintext is transformed into numbers according to a keysquare (as shown at the top of the page). The keysquare converts the eight letters of the keyword "ENVELOPE" into single digits and the remaining twenty letters of the alphabet into pairs of digits. The key is then added to the cipher numbers, by means of noncarrying addition, in order to form the ciphertext. In the final step, the ciphertext is broken into groups of five digits for transmission of the message (Kahn, 1966).

O U T L A

W B C D E

F G H I/J K

M N P Q R

S V X Y Z

plaintext	M E R G E R T O O C C U R F R I D A Y A T N O O N
	ME RG ER TO OC CU RF RI DA YA TN OX ON
ciphertext	RW NK KZ LU TW BT MK QK EL ZL UP TS UM

Figure 6: This example of the Playfair Cipher demonstrates how two letters can be enciphered simultaneously. The first step in encrypting the plaintext message is to divide the message into groups consisting of two characters. Any pairs of identical characters are separated by an infrequently used character, such as "X" or "Z".

Substitution is performed according to the following rules:

1) If the pair of plaintext letters is in the same row, then the equivalent ciphertext letters are to the immediate right of the plaintext

$(OA)_{plaintext}$ becomes $(UO)_{ciphertext}$

2) If the pair of plaintext characters are in the same column, the corresponding ciphertext letters are those letters immediately below

$(WM)_{plaintext}$ becomes $(FS)_{ciphertext}$

2) If the plaintext characters are neither in the same row nor column, then the ciphertext equivalent of each letter is that letter found at the intersection of its own row and the column of the other character

$(LS)_{plaintext}$ becomes $(OY)_{ciphertext}$

(Katzan, 1977)(Kahn, 1966)

Figure 7: Substitution ciphers employ a permuted or ciphertext alphabet as the key (Diffie & Hellman, 1979). Encryption is then accomplished by substituting the letters which make up the permuted alphabet with the corresponding letters of the plaintext or primary alphabet. Various ways of generating a ciphertext alphabet are discussed below.

method 1 The most common means of generating a ciphertext alphabet is through the use of a keyword of phrase. Any ciphertext or permuted alphabet derived by this means is referred to as a mixed alphabet (Katzan, 1977).

In the following example, the keyword "doughnut" is selected. All duplicate letters are then eliminated from the keyword, thus "doughnut" becomes "doughnt". All letters which appear in the modified keyword are then eliminated from the plaintext alphabet and the revised primary alphabet is affixed to the modified keyword. The resulting alphabet is the used to encipher the plaintext message.

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

method 2 Another means of generating a ciphertext alphabet is by means of a matrix of characters. The primary objective of this matrix of characters is to mix the alphabet. As in the previous method, a keyword or phrase is selected and all repeated letters are eliminated. The letters of the modified keyword are then used to form the first row of the matrix. The letters of the plaintext alphabet from which the letters of the keyword have been purged form the successive rows of the matrix. Using the same keyword as above, the matrix is formed as follows:

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

By affixing all of the letters in each of the succeeding columns, the mixed alphabet generated by this technique is illustrated below.

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

method 3 The ciphertext alphabet can be generated merely by selecting a new starting point as illustrated in figure 1(Katzan, 1977).

Figure 8: This illustration discusses three transposition methods which are easily implemented on a digital computer (Katzan, 1977).

Rail-Fence Cipher

In this technique, the first half of the plaintext message is written on one line, while the remaining portion of the message is written directly below. The ciphertext is formed by selecting the columns from left to right and recording the text in fixed-sized groups as shown below.

plaintext D E M O N S T R A T I O N D E L A Y E D

D E M O N S T R A T

I O N D E L A Y E D

ciphertext DIEOM NODNE SLTAR YAETD

Note that no key is required. Although this may have a slight advantage, the fact that the letter frequencies are invariant in the ciphertext results in minimal protection for the system.

Route Cipher

In this technique, the characters of the plaintext are arranged in a matrix according to a prescribed sequence. The characters in the matrix are then selected according to columns and recorded in fixed-size groups.

D E M O N

S T R A T

I O N D E

L A Y E D

ciphertext DSILE TOAMR NYOAD ENTED

As with the rail-fence cipher, no key is required, however, the size of the matrix must be known. Additional matrices may be utilized in cases where the plaintext exceeds the size of the matrix. In the situation where the number of characters to be enciphered is less than the size of the matrix, blank or null characters, such as "x", can be inserted to fill the matrix.

Keyed-Columnar Ciphers

This technique uses both a keyword and a mixed number to encipher the plaintext message. The plaintext message is inscribed by rows in a matrix where each column is numbered by a mixed number. The ciphertext is then formed by selecting the columns in numerical order using the mixed number.

In this example, the keyword is "FEBRUARY". The letters which comprise the keyword are numbered in accordance with their relative order of appearance in the standard alphabet. In the case of repeated letters, they are numbered in sequence from left to right. If the plaintext message to be encrypted is "NEW MODEL TO BE DEMONSTRATED FRIDAY" then encryption proceeds as follows:

keyword	F E B R U A R Y
mixed numbers	4 3 2 5 7 1 6 8
	N E W M O D E L
plaintext	T O B E D E M O
	S T R A T E D F
	R I D A Y

ciphertext DEEWB RDEOTI NTSRM EAAEM DODTY LOF

Although the last row of the matrix can be filled with null characters, it is not recommended to do so since the security level of the cipher is increased if the null characters are not added (Katzan, 1977).

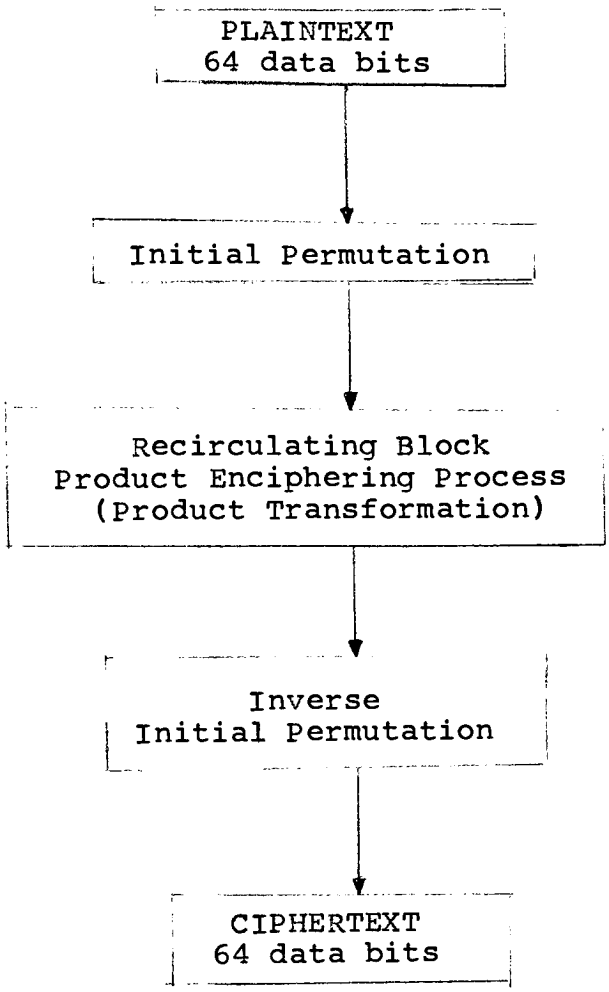


Figure 9: This diagram illustrates the three major steps which the Data Encryption Standard incorporates during the encipherment of a 64 bit data block.

Step 1: The initial permutation, IP, is actually a transposition operation. This operation manipulates only the bits that comprise the 64 bit input block and does not utilize the 64 bit key.

Step 2: The recirculating block product enciphering process is a complex key-dependent product transformation, the major portion of which consists of 16 iterations of substitution and transposition operations.

Step 3: The inverse initial permutation, IP^{-1}, which is the final transposition operation and is the actual reversal of step 1. (Katzan 1977)

(Figure 9 continued on next page)

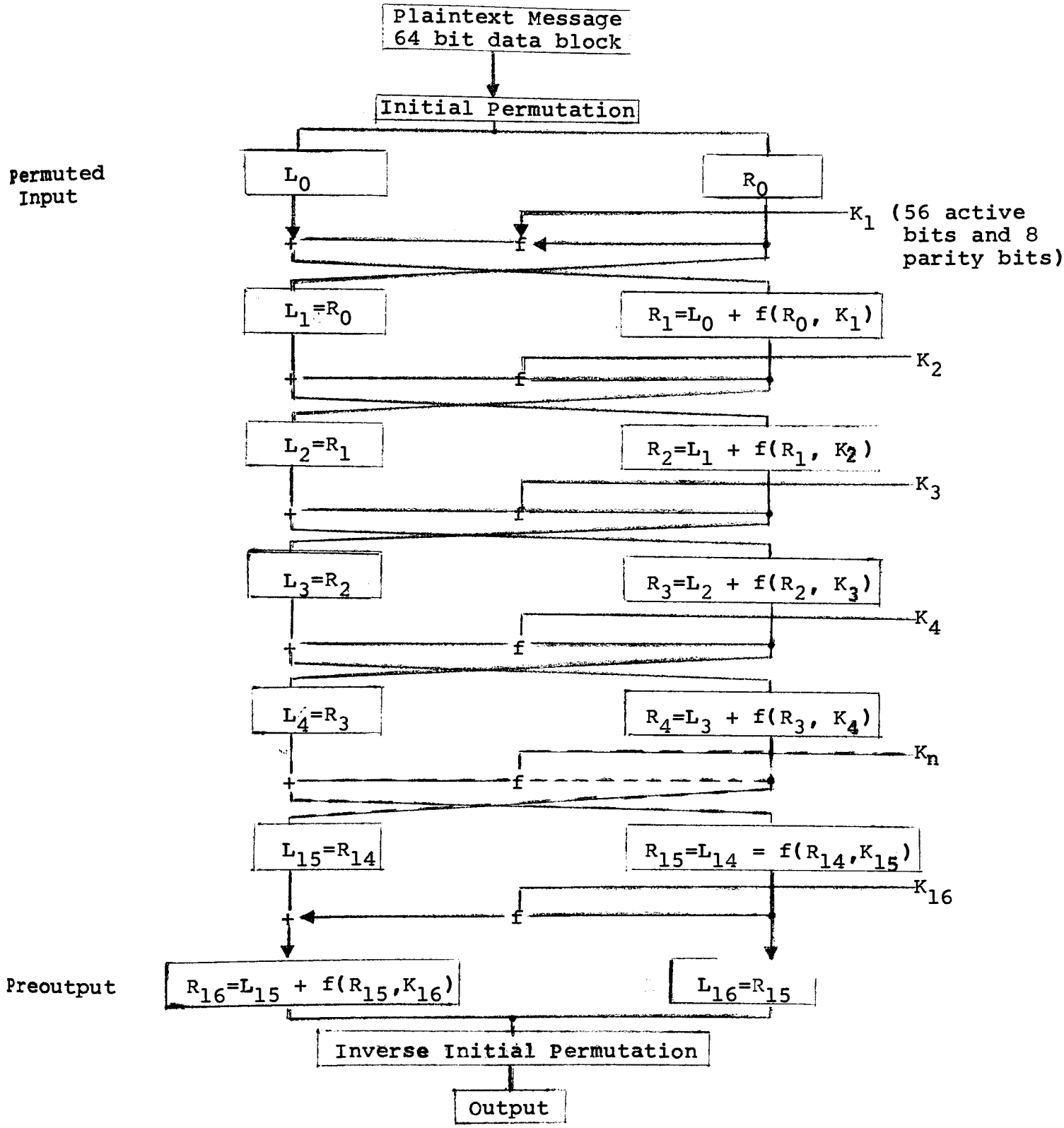


Figure 9: Schematic diagram of the Data Encryption Standard (Bright & Enison, 1978)

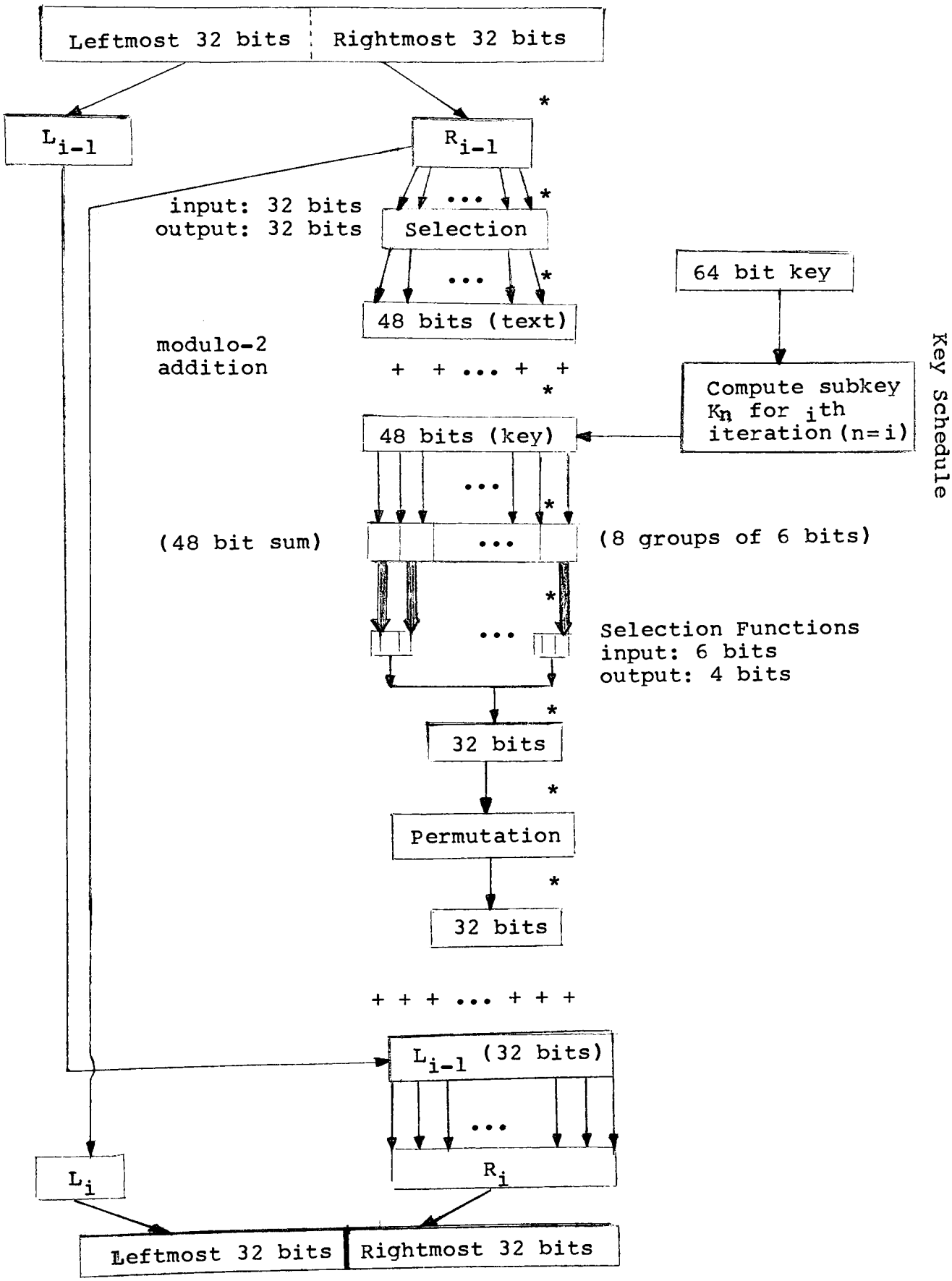


Figure 9c: This diagram illustrates the steps involved in one iteration of the computations in the product transformation of the DES (recirculating block product enciphering process). Note that the "*" denotes steps in the cipher function (Katzan, 1977).

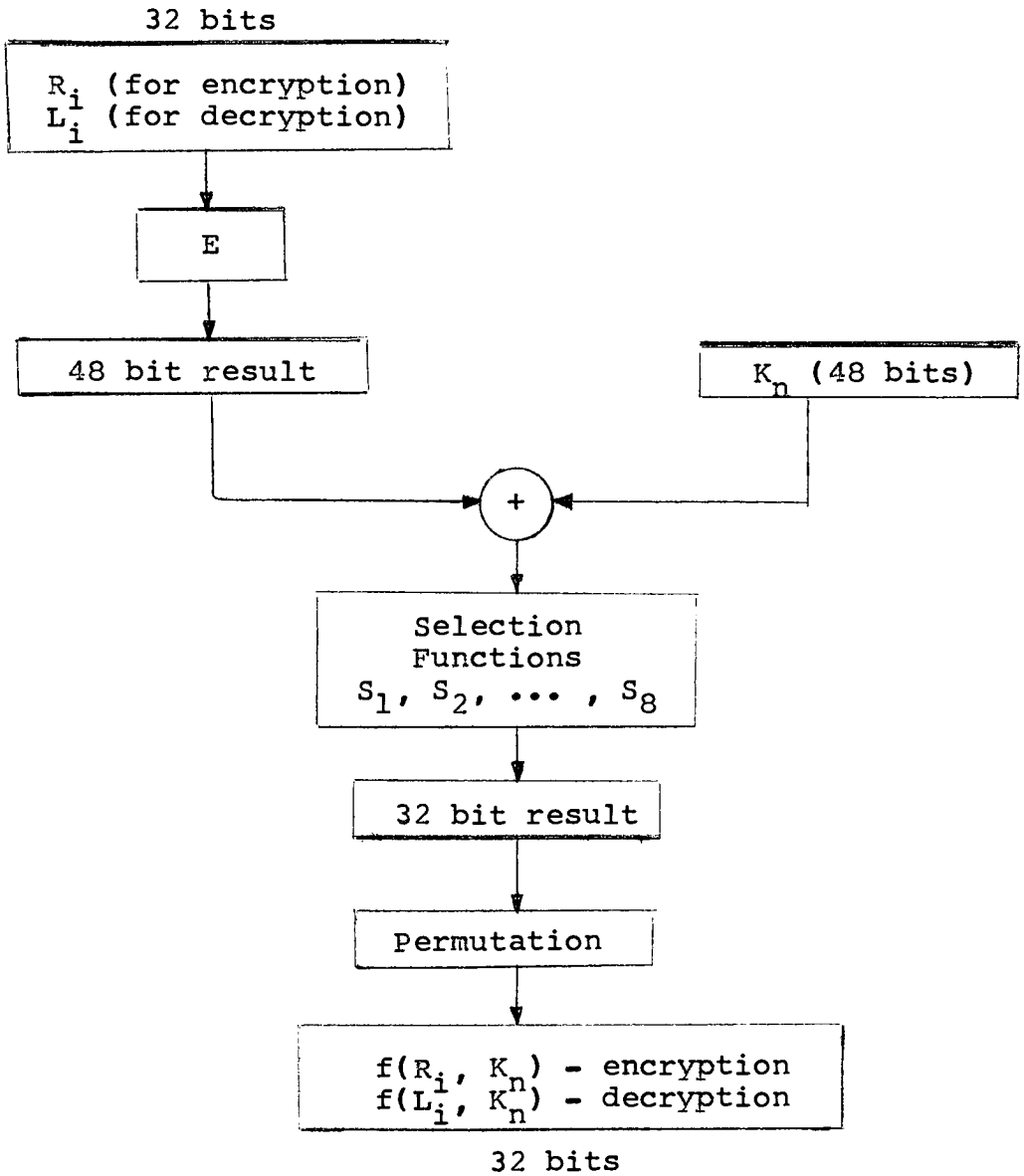


Figure 10: Overview of the cipher function (Katzan, 1977)

The selection operation E takes a 32 bit block as input and yields a 48 bit result. This result is then added (modulo-2 addition) to a 48 bit subkey, K_n, on a bit-by-bit basis yielding a 48 bit result. This result is in turn converted to 32 bits by means of a set of selection functions ($S_1,\ldots,S_8$). In the final step, this 32 bit result undergoes a permutation operation which produces a 32 bit result.

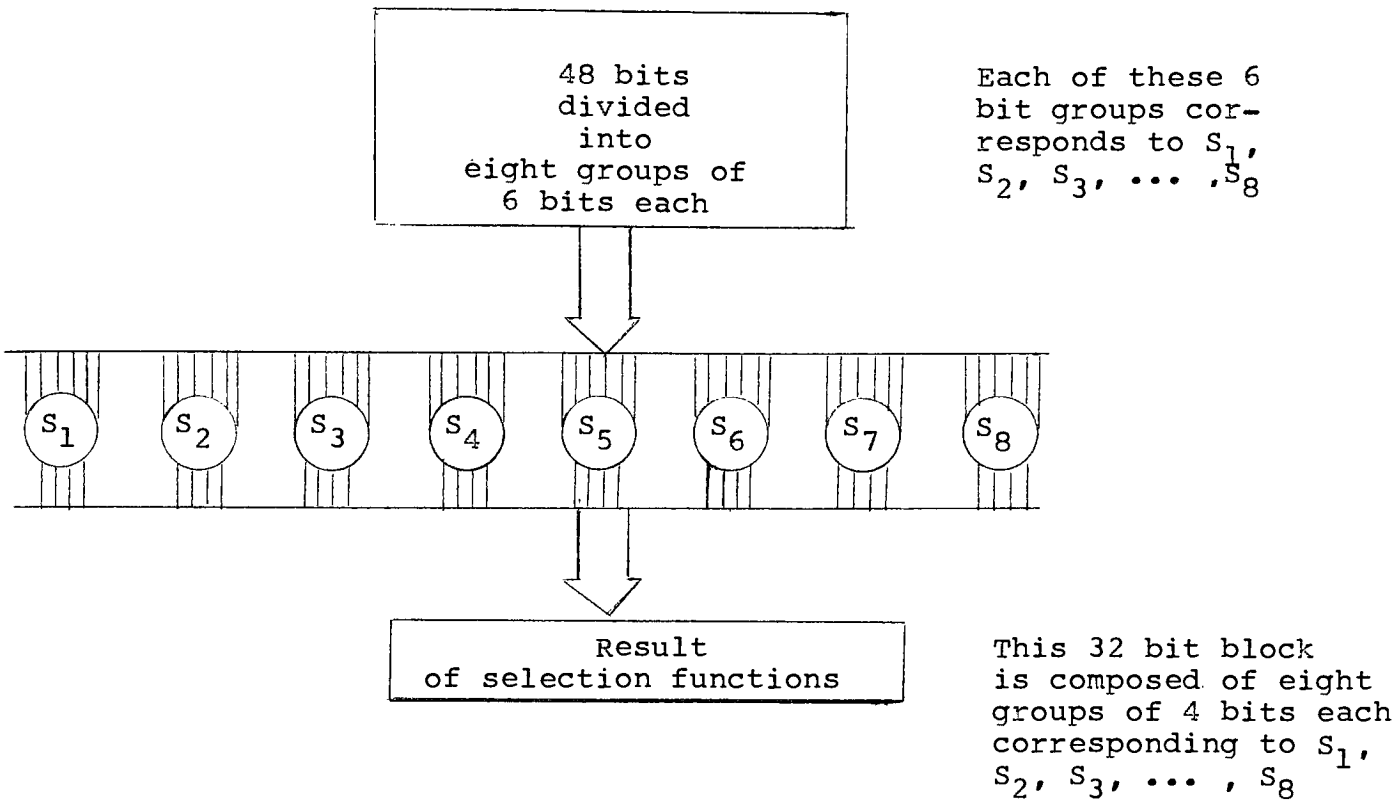


Figure 11: This diagram illustrates the unique set of selection functions used in the cipher function.

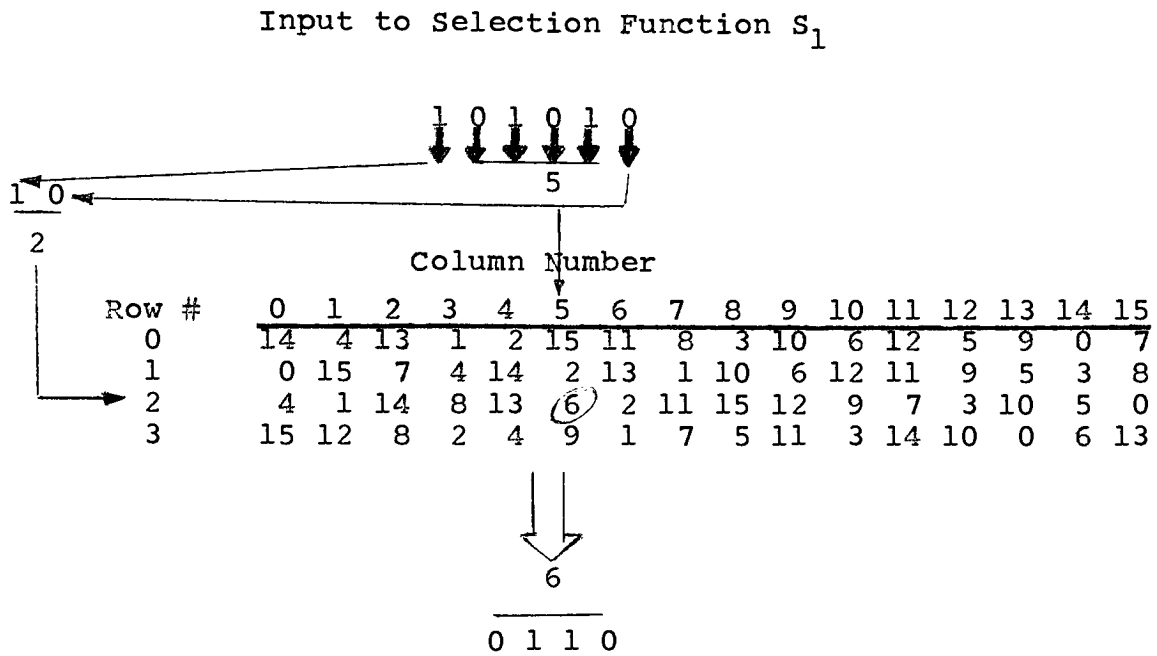


Figure 12: This diagram illustrates the use of the selection function S_1. Input to the function S_1 is the binary string 101010. The corresponding output is the binary value 0110.
(Katzan, 1977).

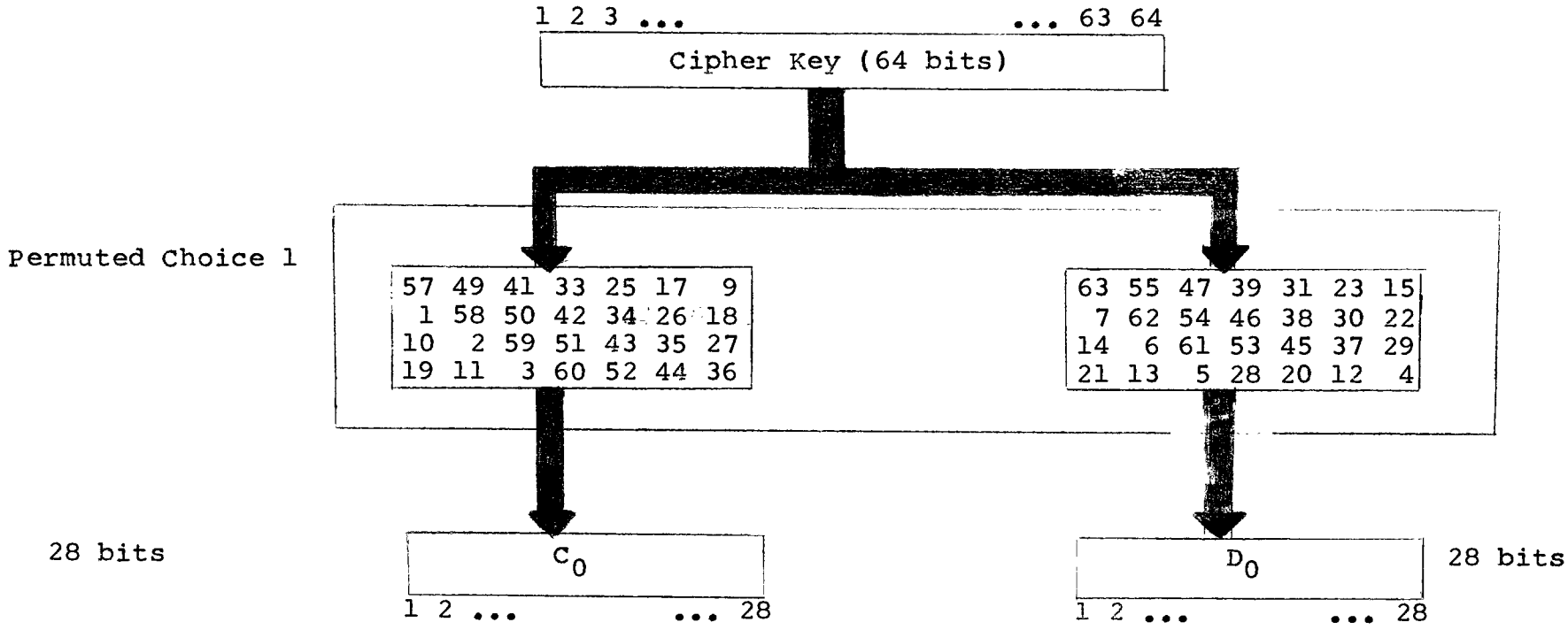


Figure 13: This diagram represents the permutation operation, permuted choice 1, used in the calculation of the 28 bit blocks C_0 and D_0 (Katzan, 1977).

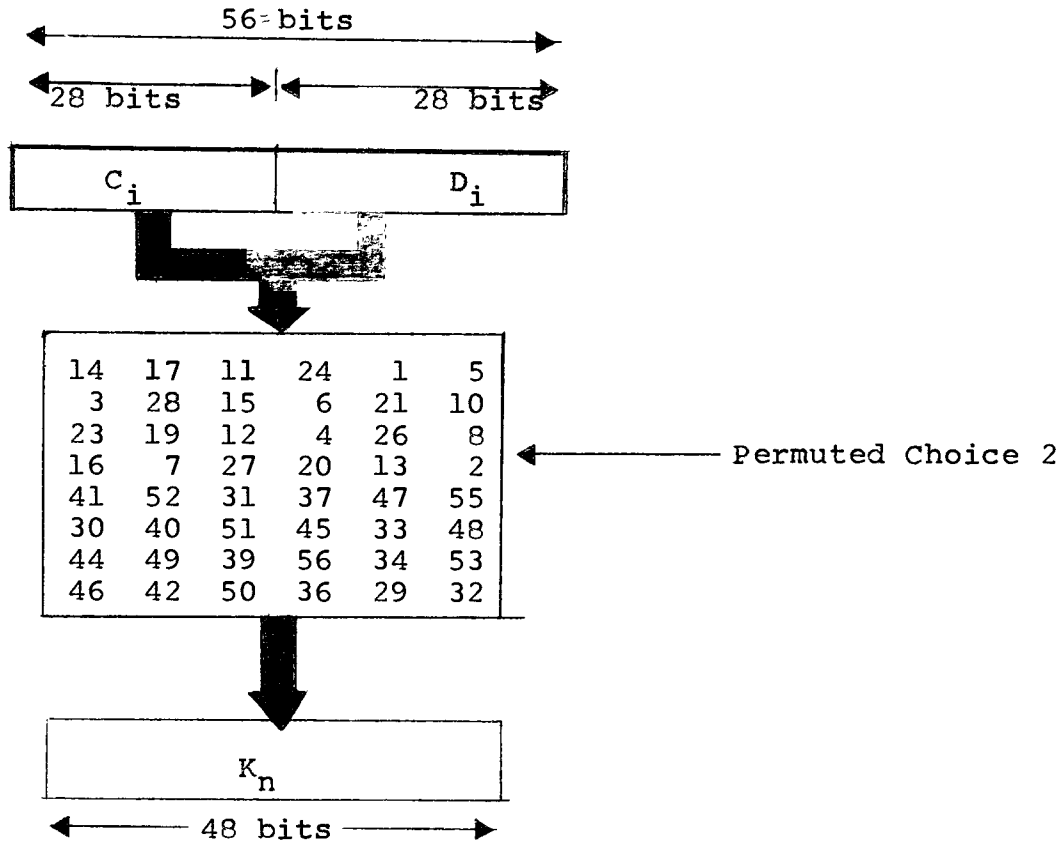


Figure 14: This diagram represents the permutation operation, permuted choice 2, used in the calculation of each of the sixteen subkeys, K_n (Katzan, 1977).

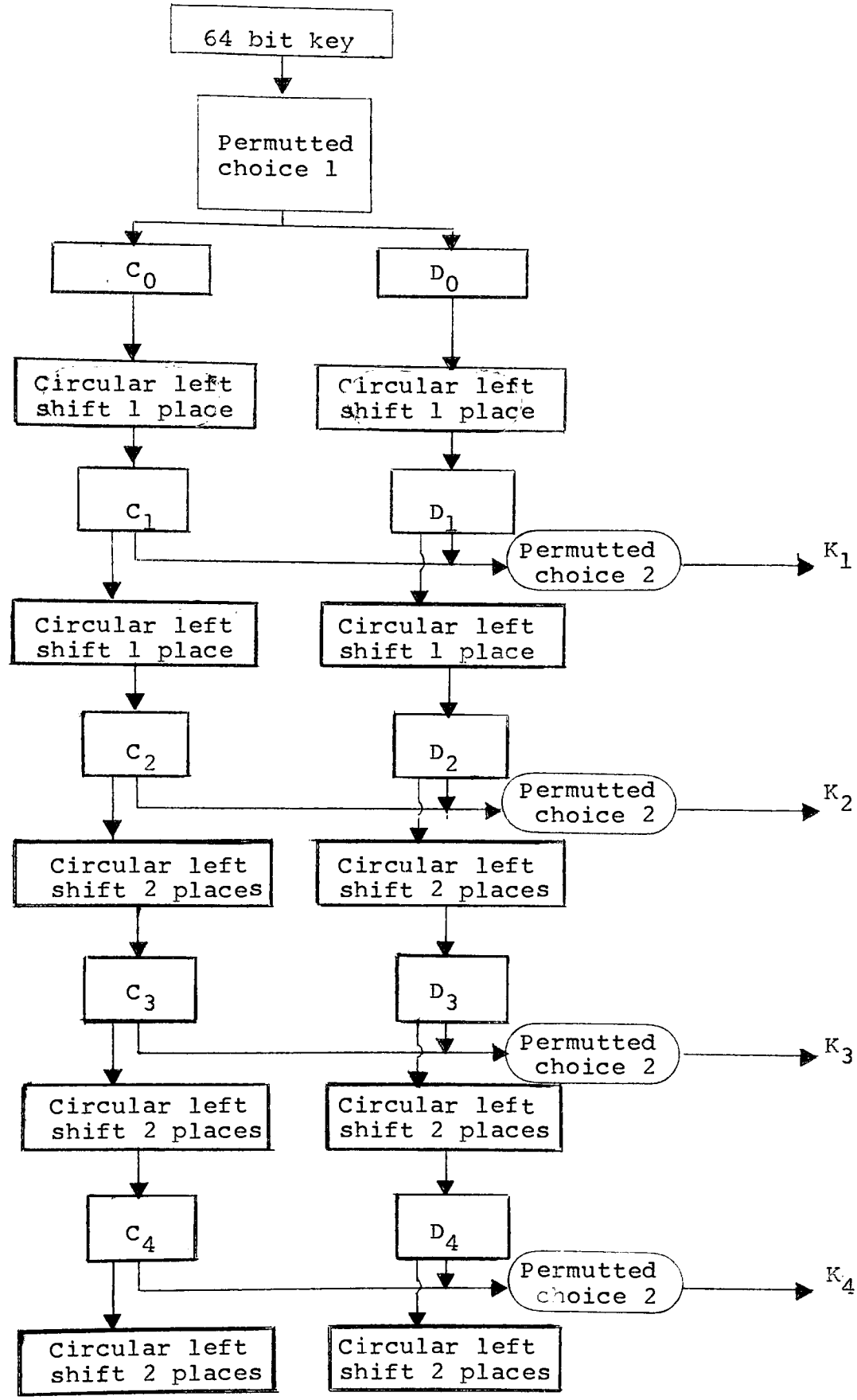


Figure 15: Summary of the key schedule calculations (Katzan, 1977).

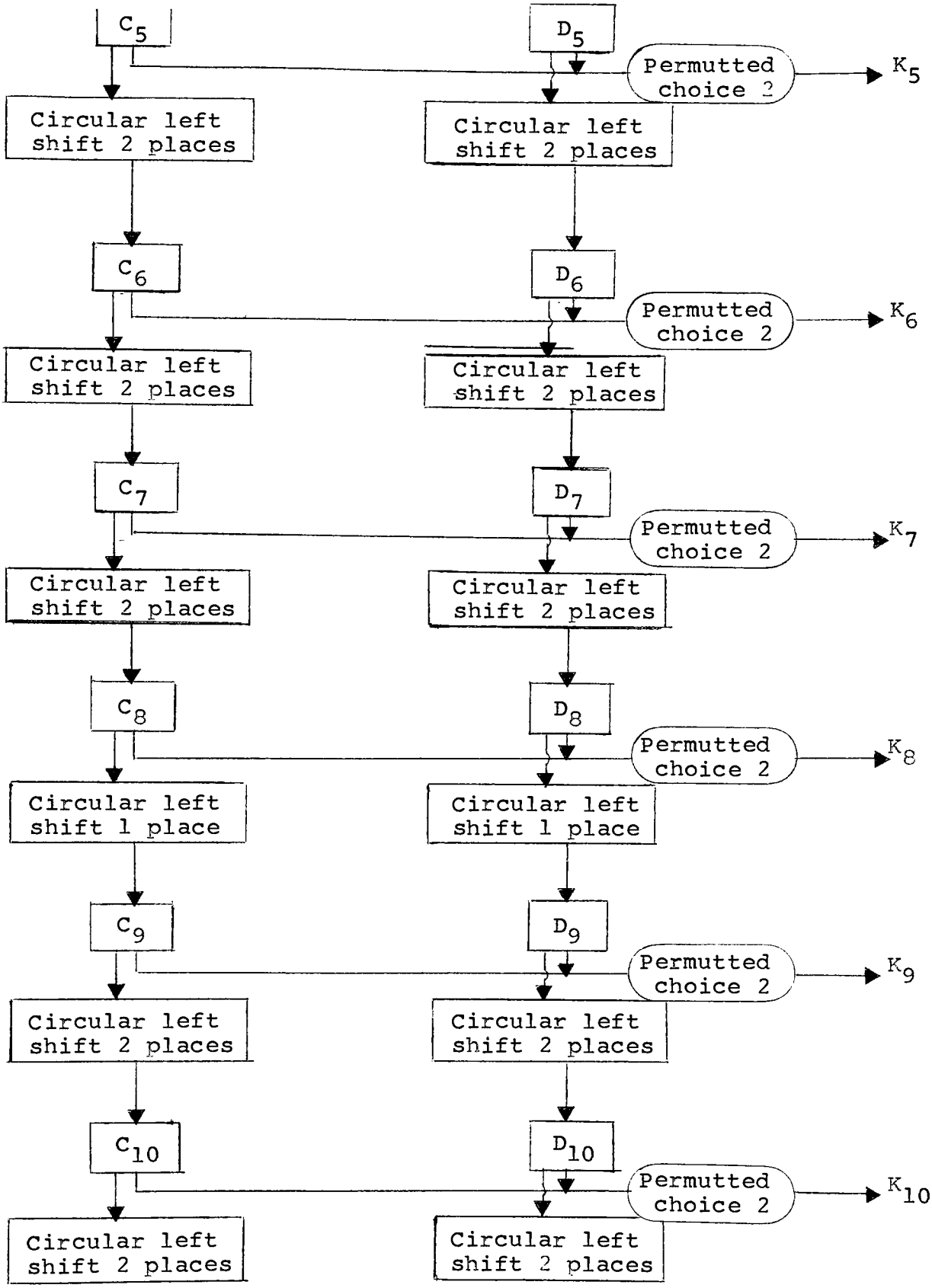
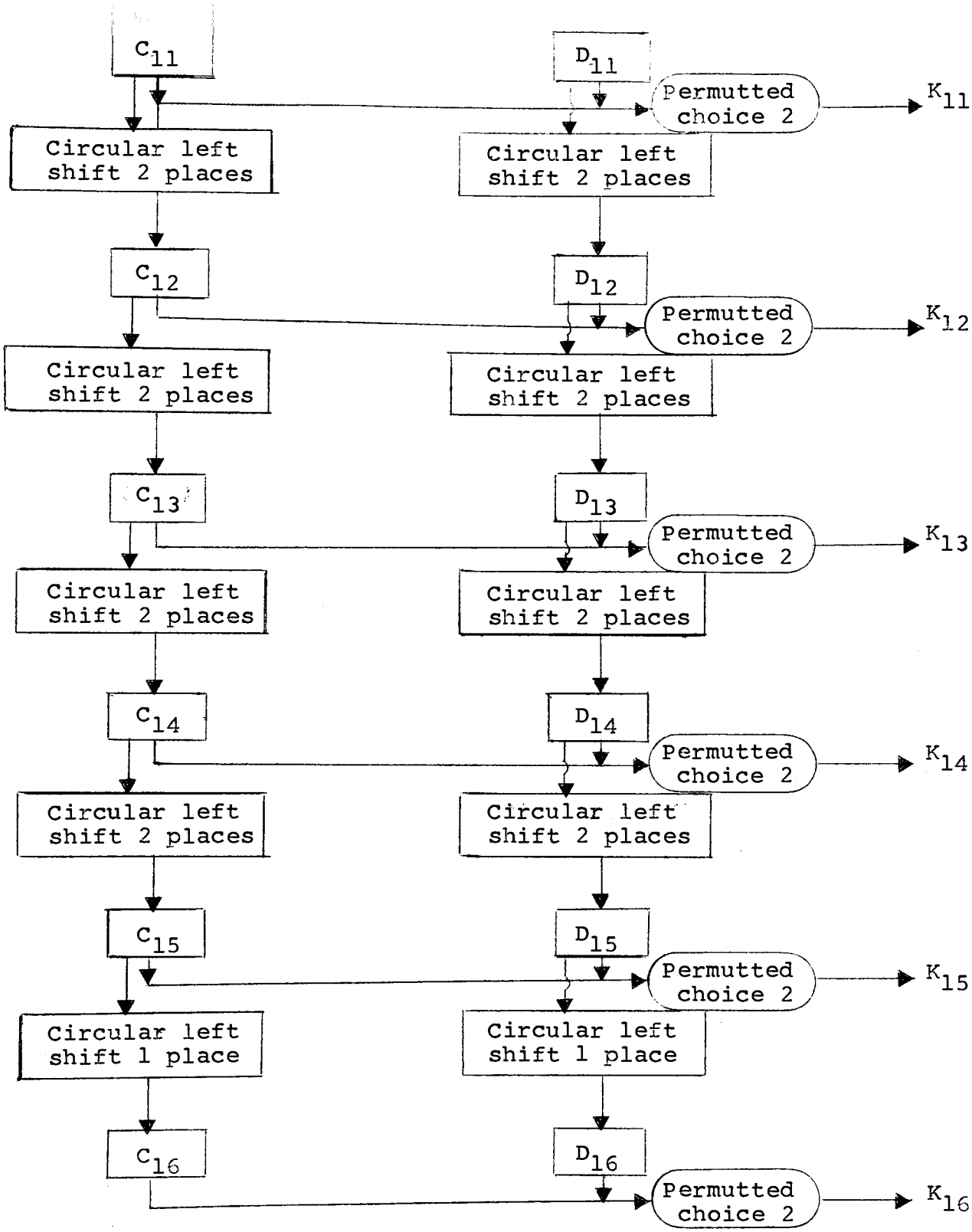


Figure 15: Key calculation schedule continued.


C_{11}
D_{11}
Permutted choice 2
K_{11}
Circular left shift 2 places
Circular left shift 2 places
C_{12}
D_{12}
Permutted choice 2
K_{12}
Circular left shift 2 places
Circular left shift 2 places
C_{13}
D_{13}
Permutted choice 2
K_{13}
Circular left shift 2 places
Circular left shift 2 places
C_{14}
D_{14}
Permutted choice 2
K_{14}
Circular left shift 2 places
Circular left shift 2 places
C_{15}
D_{15}
Permutted choice 2
K_{15}
Circular left shift 1 place
Circular left shift 1 place
C_{16}
D_{16}
Permutted choice 2
K_{16}

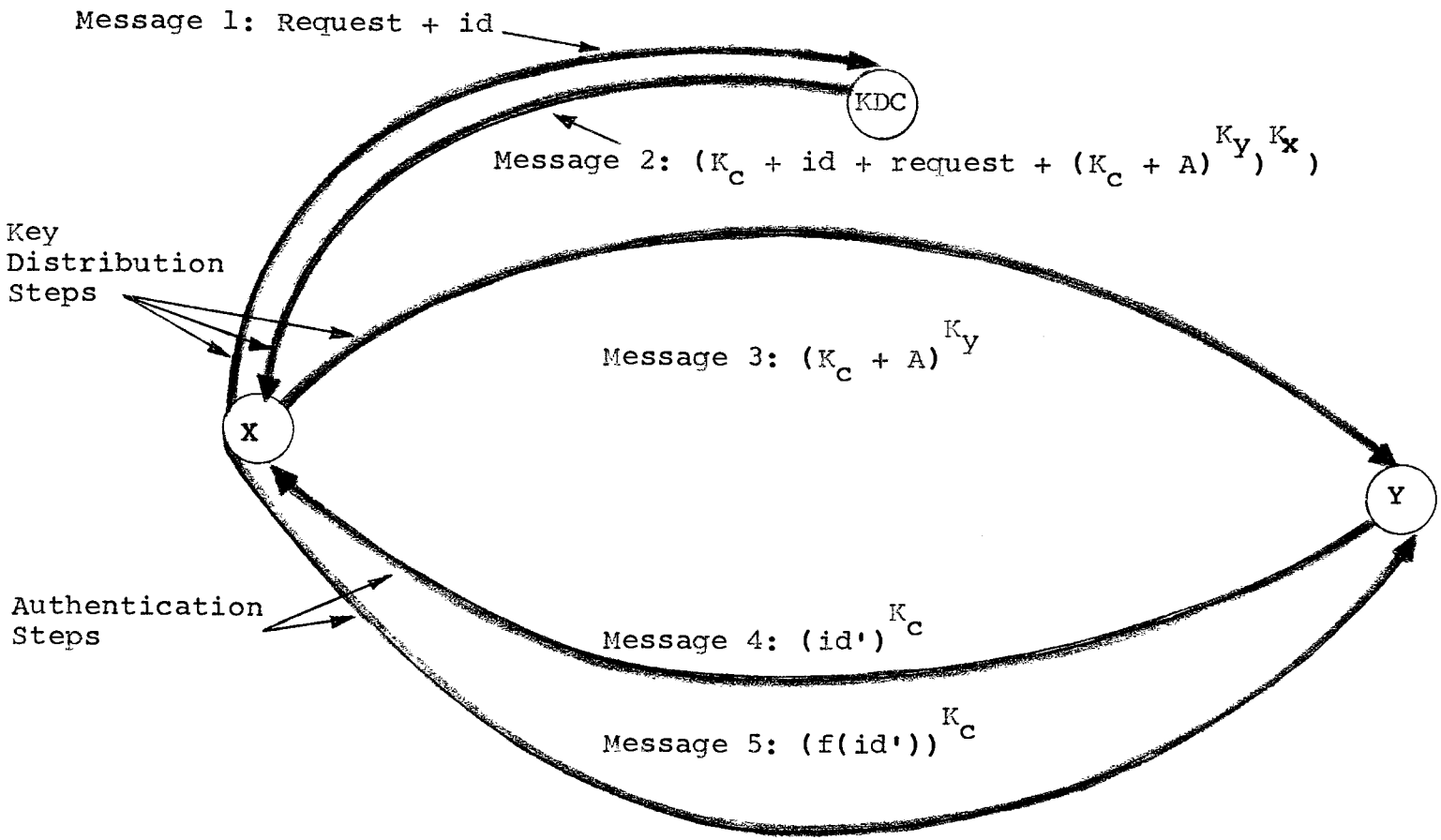


Figure 16: Summary of the steps required for key distribution and the establishment of a communication channel for conventional key algorithms.

Note that (i^j) denotes the ciphertext resulting from the encryption of plaintext i with key j.

(Popek & Kline, 1979).

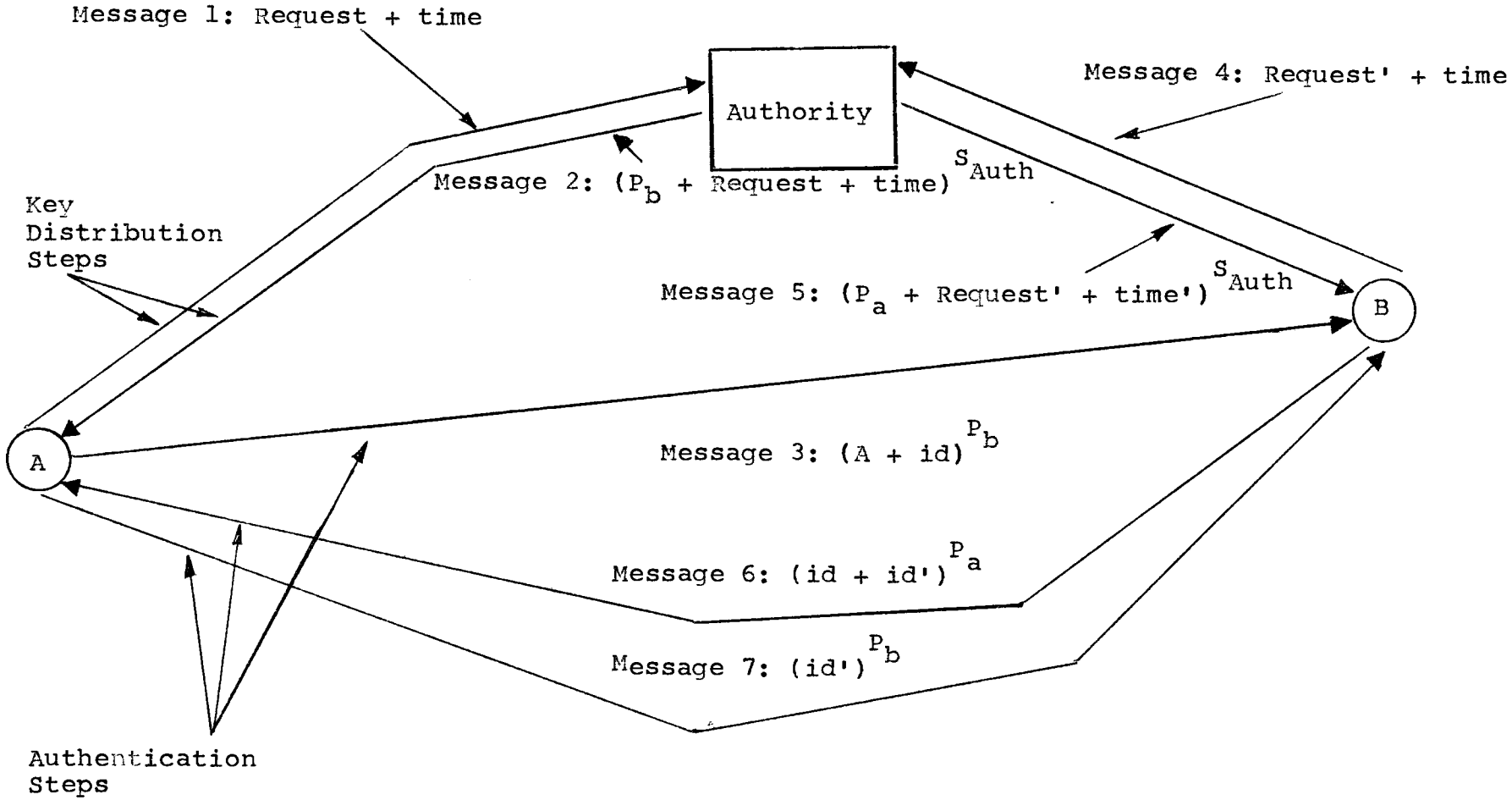


Figure 17: Summary of the steps required for key distribution and the establishment of a communication channel for public-key algorithms.

Note that P_i is the public key for i while S_i is the secret key for i.
(Popek & Kline, 1979)

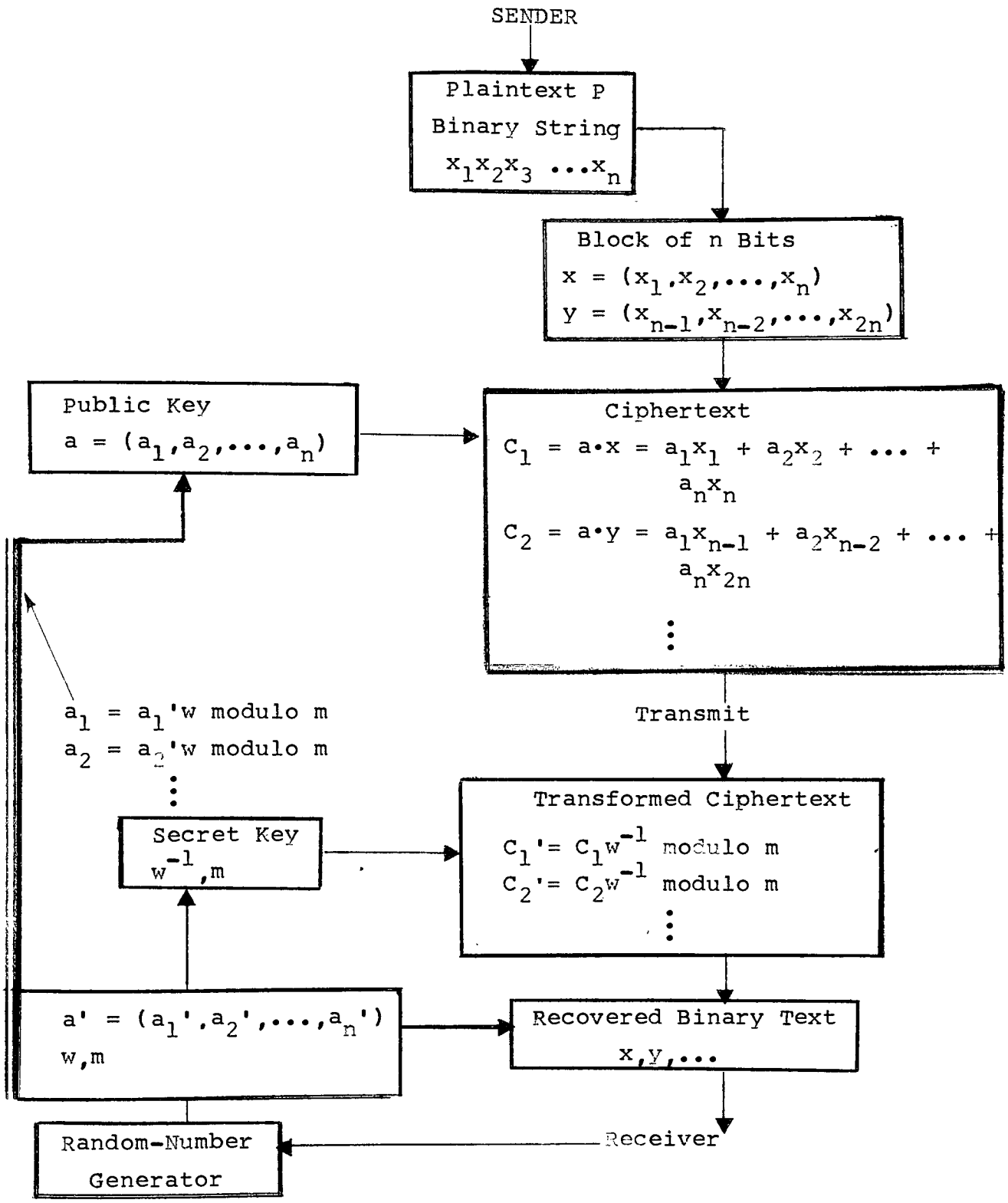


Figure 18: This diagram illustrates the flow of information in the trapdoor knapsack cryptosystem.

The random-number generator is used to select the secret vector a'. Each element of this vector is characterized by the fact that it is greater than the sum of the preceding elements. (continued on next page)

Figure 18 (continued)

After the receiver has selected his secret vector a' he also selects two large random numbers w and m. Neither w nor m must share a common factor. These two values are the trapdoor parameters and are responsible for transforming the secret and relatively easy vector a' into the more difficult public vector a. This transformation of vector a' into vector a is accomplished by using the equation $a_1 = a_1'w$ modulo m, $a_2 = a_2'w$ modulo m, and so on. Vector a is then transmitted over an insecure channel to the sender or is listed in the public directory as the receiver's public enciphering key.

The first step in enciphering the plaintext message P is for the sender to convert the message into a binary string. The sender then refers to the public directory to look up the receiver's public key. The number of elements which comprise the receiver's public key determines the length of each block into which the binary string will be divided. For example, if there are 10 elements in the public vector, then the binary string will be divided into blocks of 10 bits each.

Each block is then enciphered by forming the dot product with vector a. If $x = (x_1,x_2,\ldots,x_n)$ represents a block of information, then $C_i = a \cdot x = a_1x_1 + a_2x_2 + \ldots + a_nx_n$. The ciphertext values are then transmitted over an insecure channel.

Upon receipt of the ciphertext, the receiver proceeds to recover x. This is done by calculating w^{-1} followed by the value $C_1' = C_1w^{-1}$ modulo m. Since w^{-1} is the inverse of w modulo m and $a_1 = a1'w$ modulo m, $a_2 = a_2'w^{-1}$ modulo m, etc., then $a_1' = a_1w^{-1}$ modulo m, $a_2' = a_2w^{-1}$ modulo m, and so on. Therefore, $C1'$ or C_1w^{-1} modulo m equals $a_1x_1w^{-1} + \ldots + a_n'x_nw^{-1}$, or $a_1w^{-1}x_1 + \ldots + a_nw^{-1}x_n$, modulo m, equals $a_1'x_1 + \ldots + a_n'x_n$. This merely states that C1' equals $a' \cdot x$. This then permits the difficult recovery of x from C_1 & a to be transformed into the easy problem of recovering x from C_1' and a'. Only the receiver, who has knowledge of the necessary trapdoor information w and m and the secret vector a', is able to recover x without too much effort (Hellman, 1979).

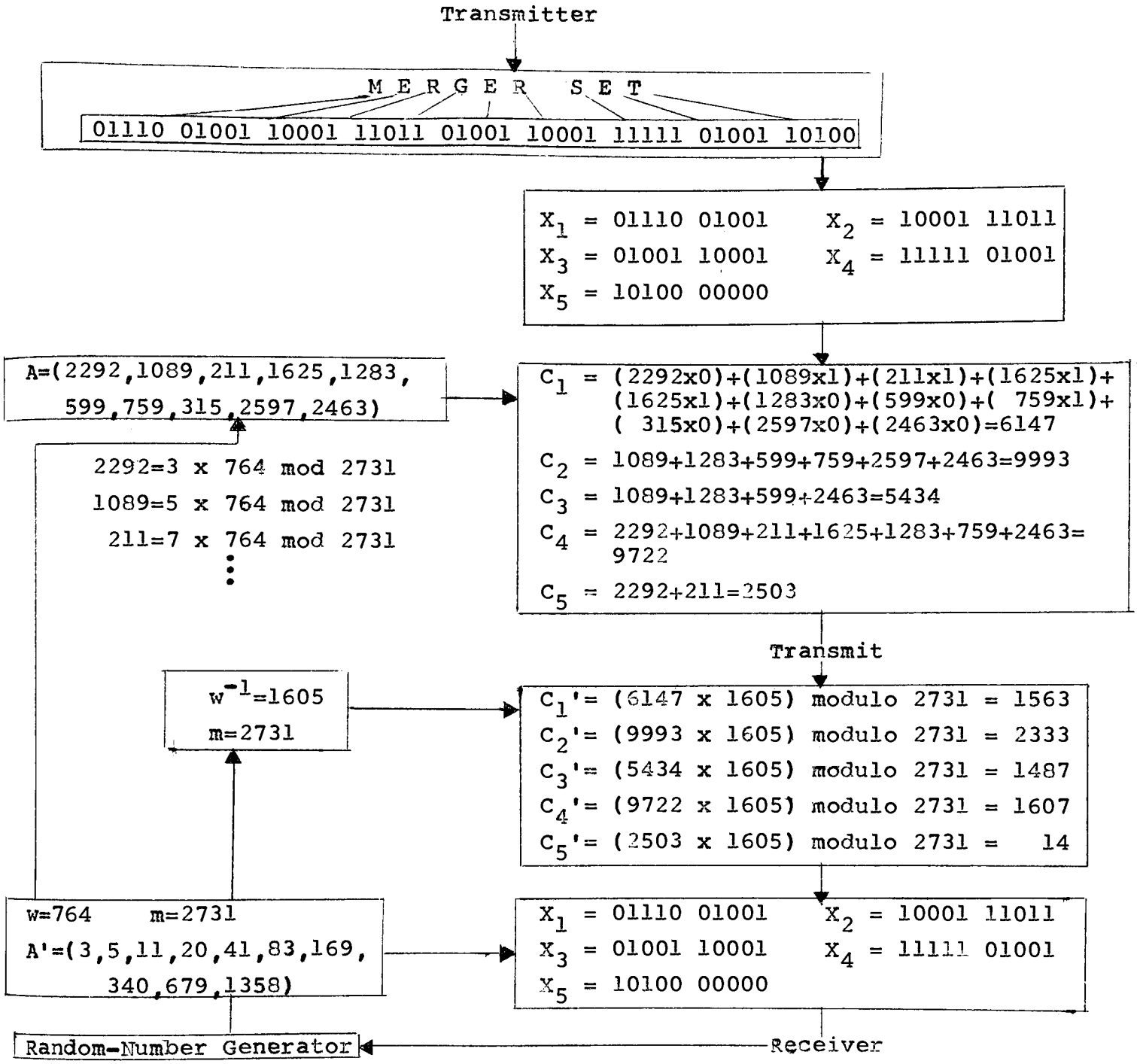


Figure 18 (Continued): Schematic illustration of the trapdoor knapsack cryptosystem described on pages 65-71.

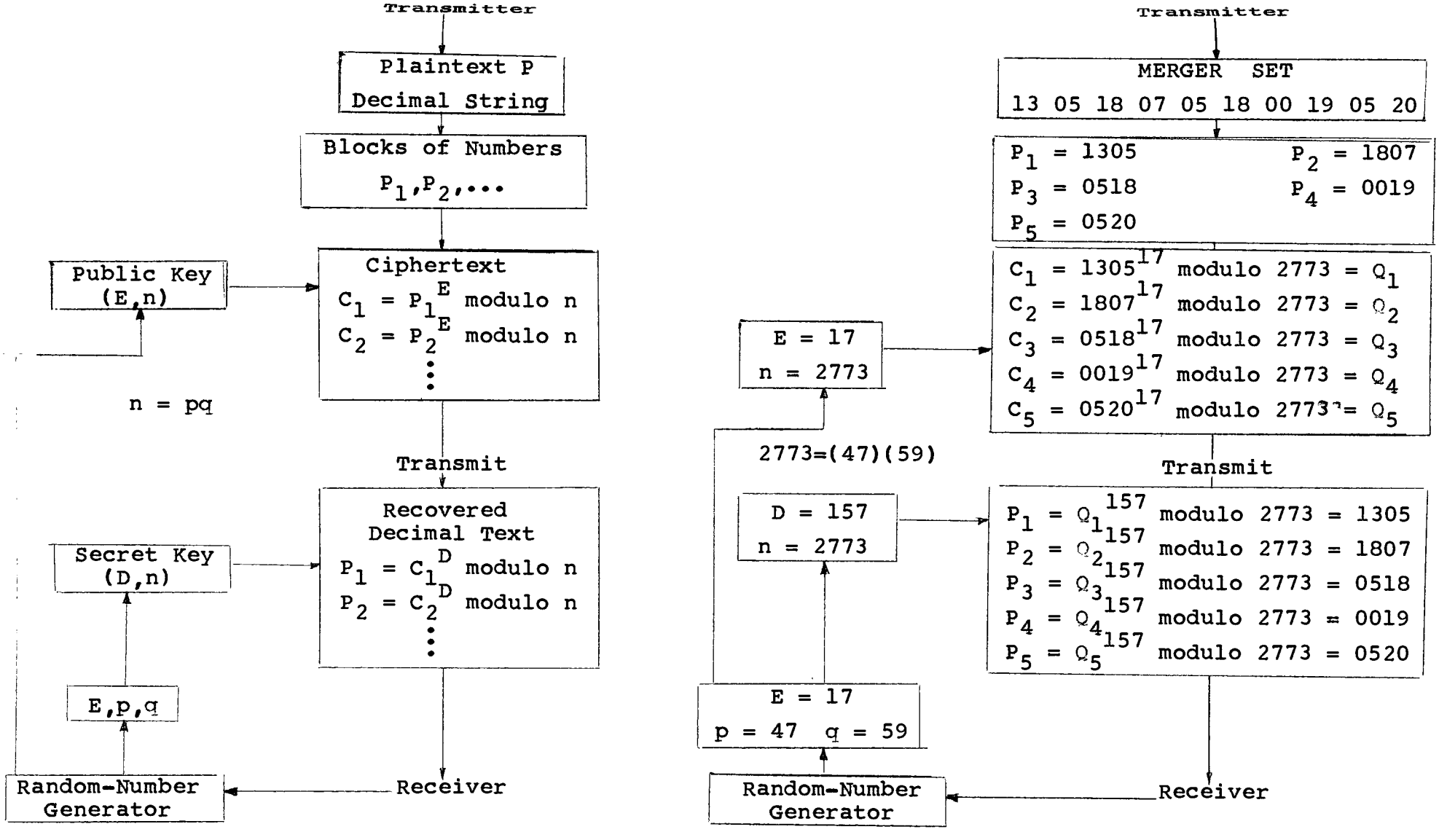


Figure 19: RSA public-key cryptosystem (Hellman, 1979).

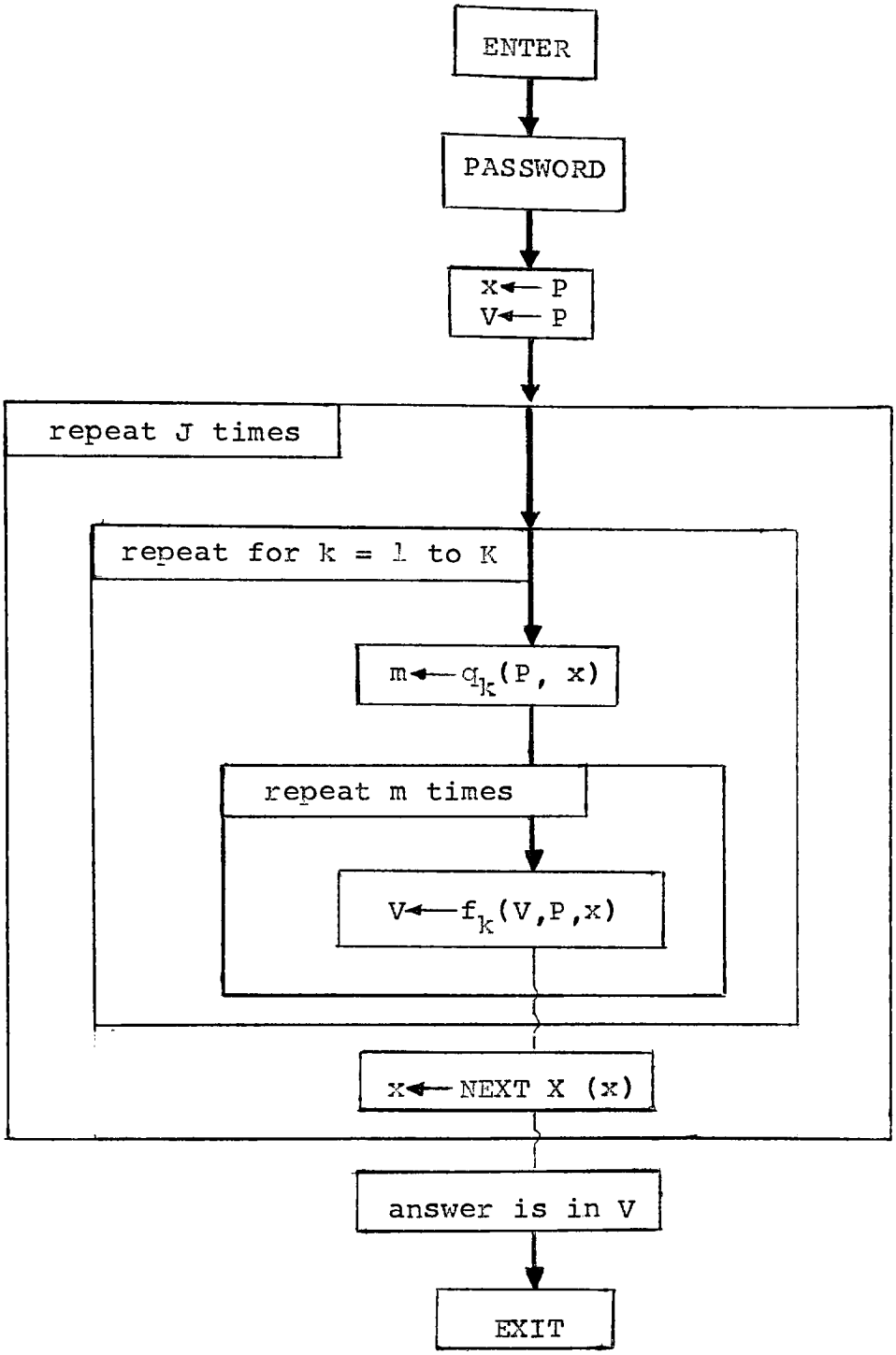


Figure 20: This diagram illustrates a user authentication scheme which does not require secrecy in the computer. The overall transformation is described by the flowchart outlined above.

The variables used in the diagram are described on the following page.

V The current value to be scrambled. Each application of a scrambling function f_k replaces V by a new value.

m The number of times a particular scrambling function f_k is to be repeated.

k A counter used to designate K unique scrambling functions.

NEXT X; A function which when applied to the parameter of the previous cycle results in the parameter for the next subsequent cycle.

P The original password entered by the user.

q_k A function which is applied to P and the current value of x to determine the number of times to iterate the current f_k.

f_k The scrambling functions.

K The number of distinct scrambling functions.

J The number of cycles.

x The value used to parametrize the current cycle.

Figure 20 (continued)

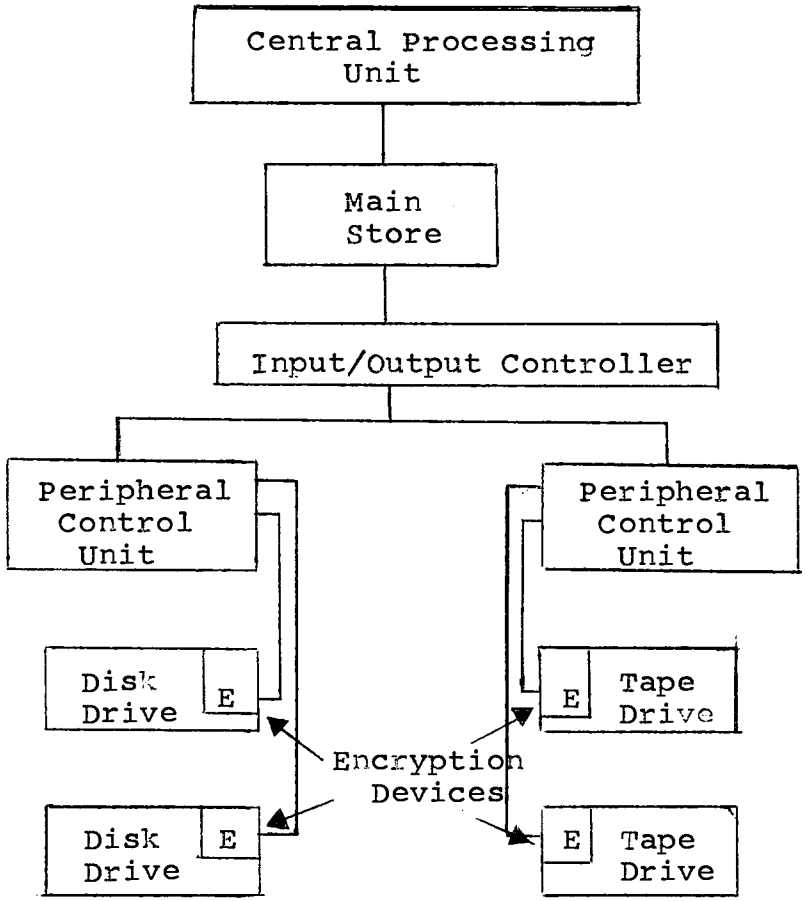


Figure 21a: Each disk or tape drive is installed with an encryption device thus requiring a large number of these devices.

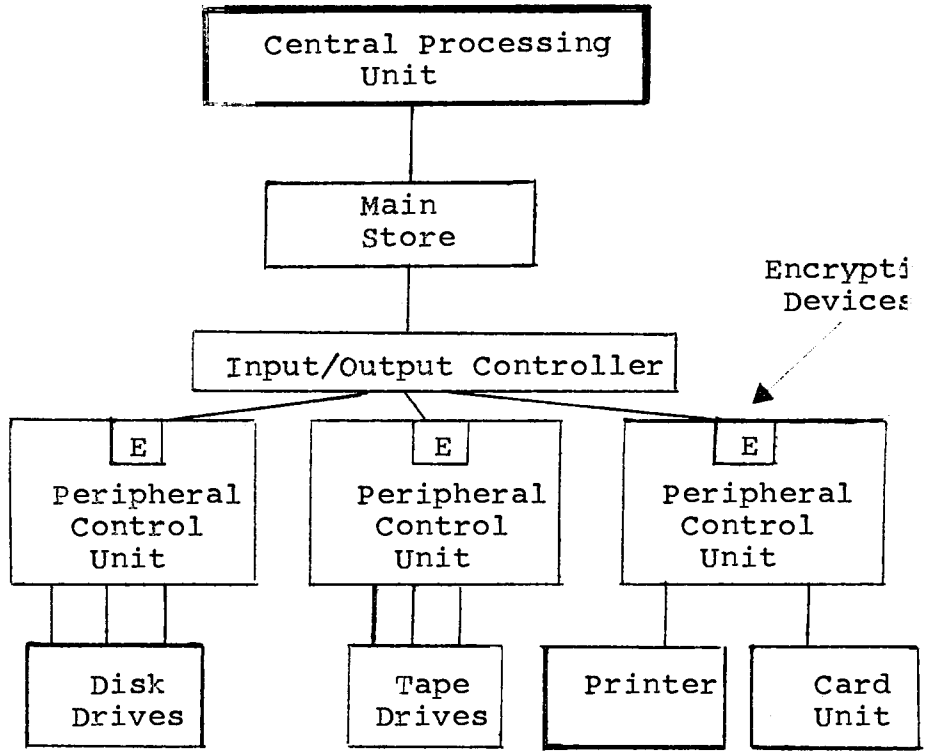


Figure 21b: Each peripheral control unit is installed with an encryption device. Unlike figure 21a, the record identifier and key fields can not be encrypted since they must be interpreted by the device during search operations. (Keys & Clamons, 1974)

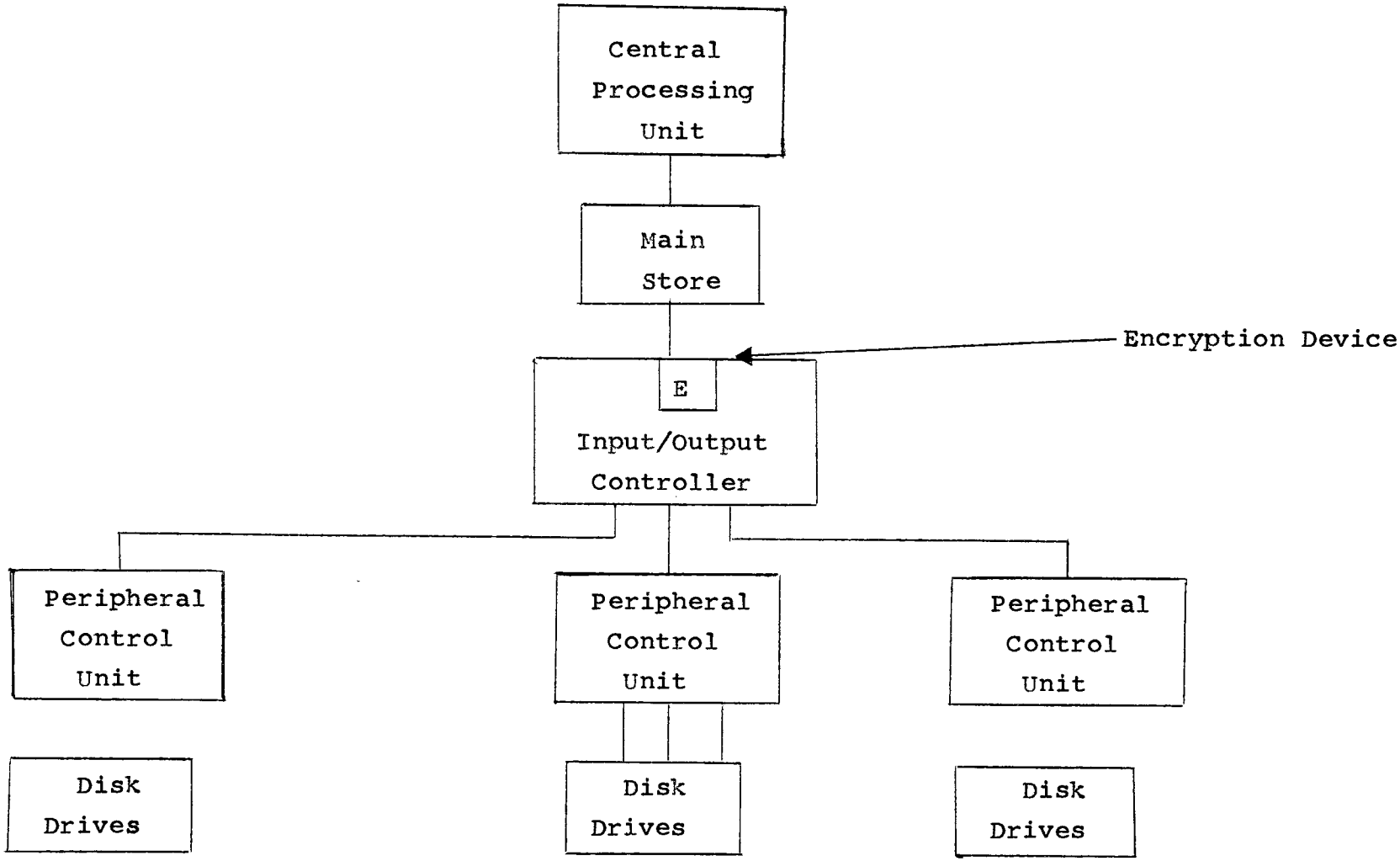


Figure 21c: The input/output controller is installed with an encryption device. The encryption device must now be enabled or disabled not only according to to whether control, status, or data is being transmitted, but also according to which peripheral is receiving (Keys & Clamons, 1974).

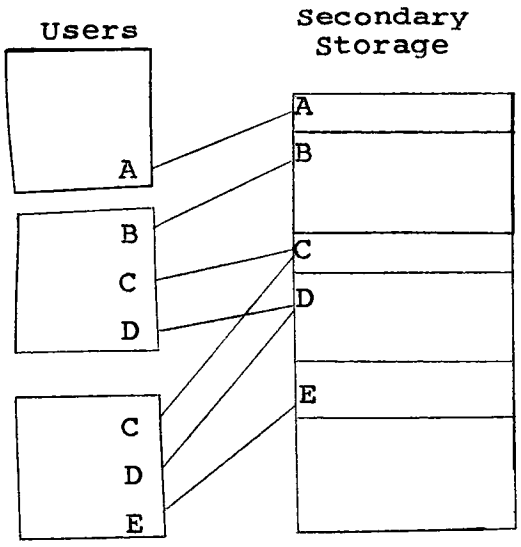


Figure 22a: Each user possesses keys to only those files he is priviledged to access.

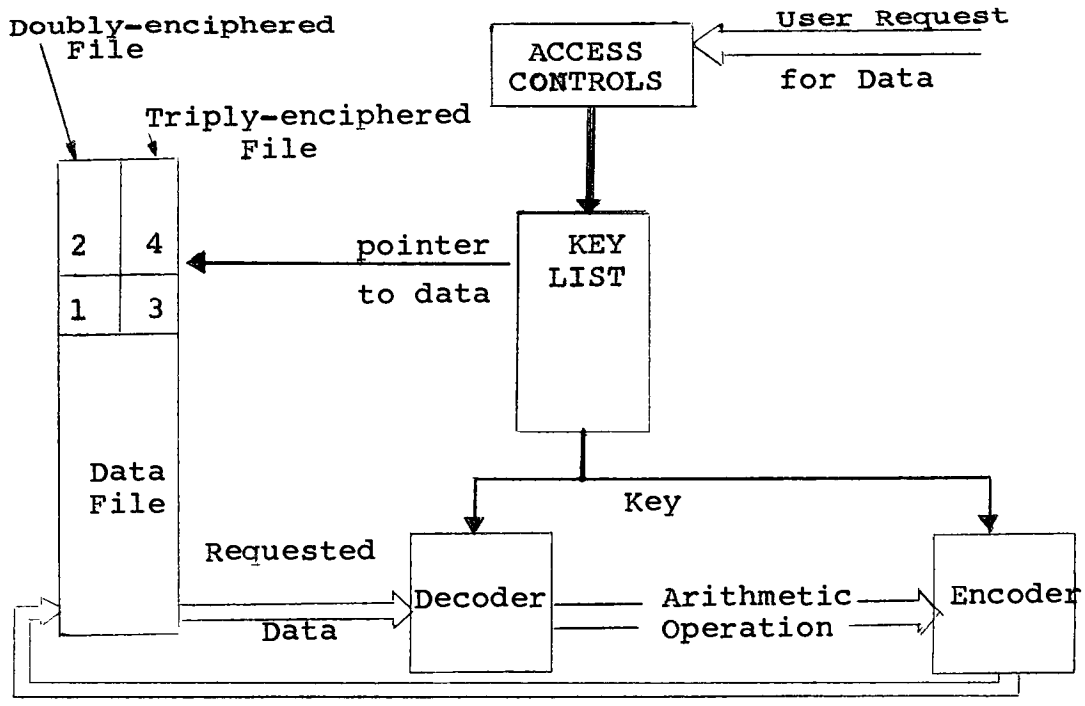


Figure 22b: In this illustration the protected key is stored in a portion of the software system which is not accessible to unauthorized users.

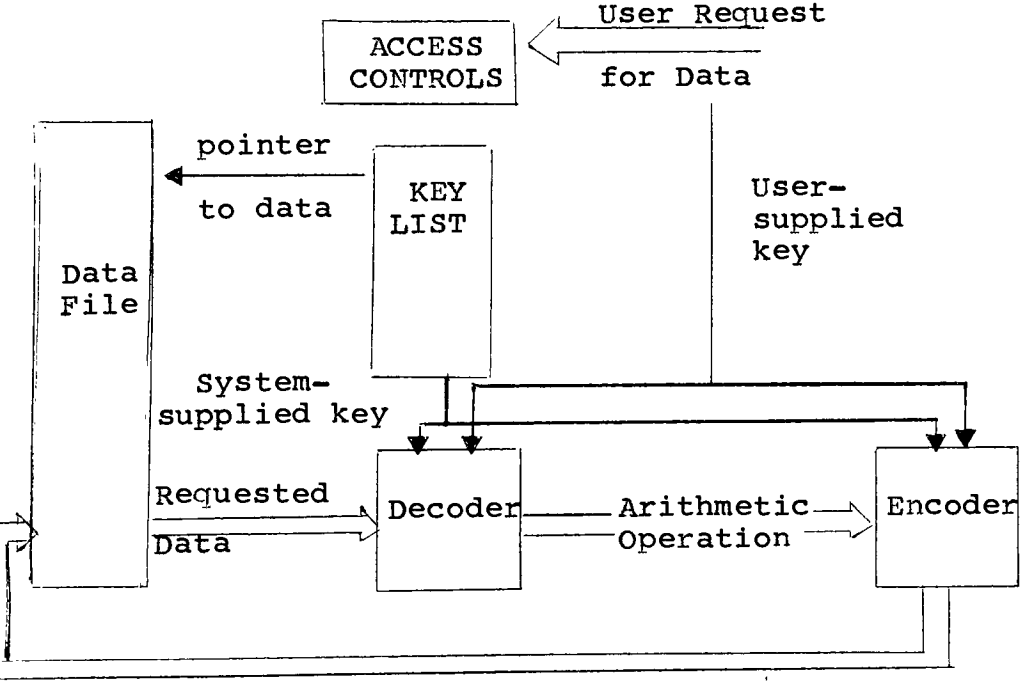


Figure 22c: This scheme illustrates how a "one-time" key could be implemented which requires a different key at successive log-ons (Bartek, 1974)(Keys & Clamons, 1974)

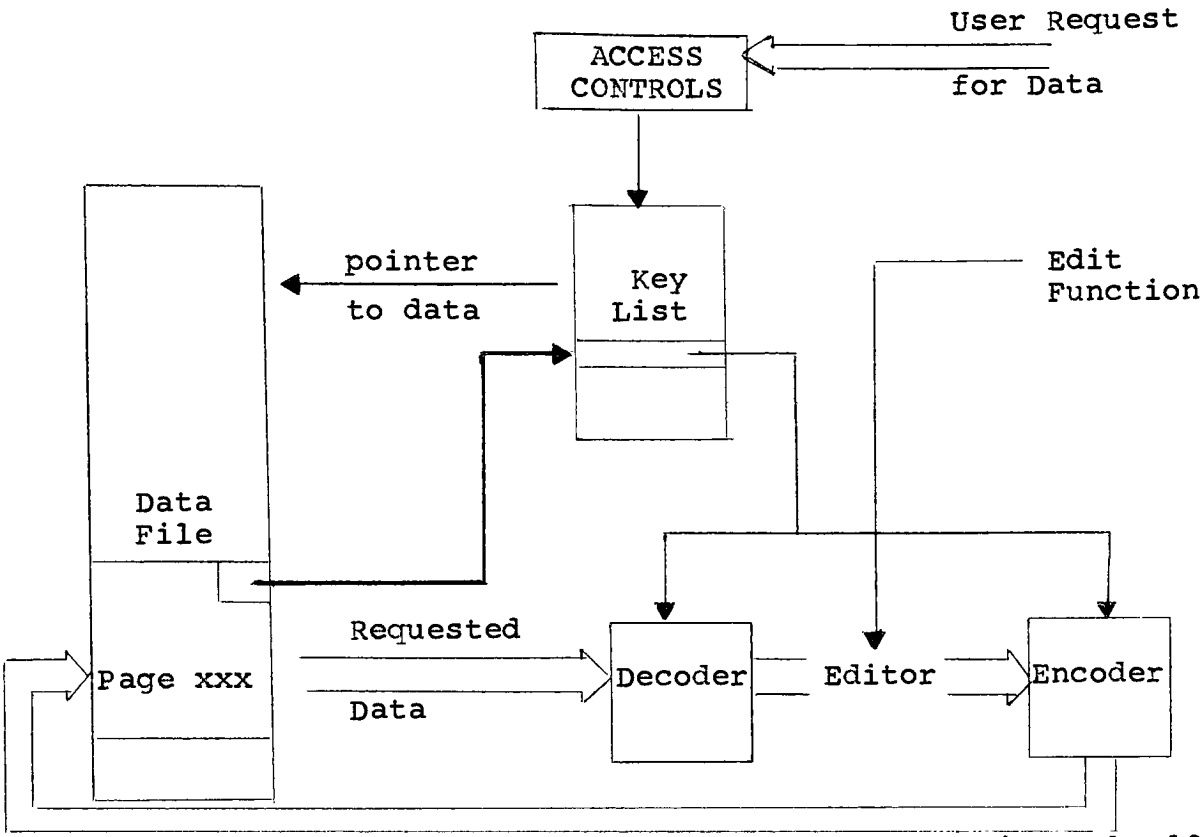


Figure 22d: Encryption with starting key per page (Bartek, 1974).

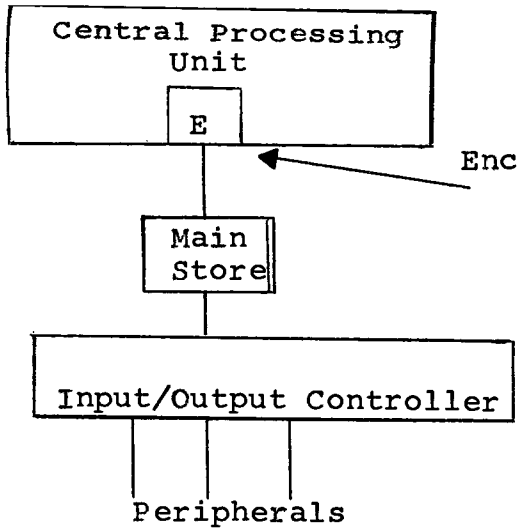


Figure 23a: The encryption device is placed between the CPU and main store.

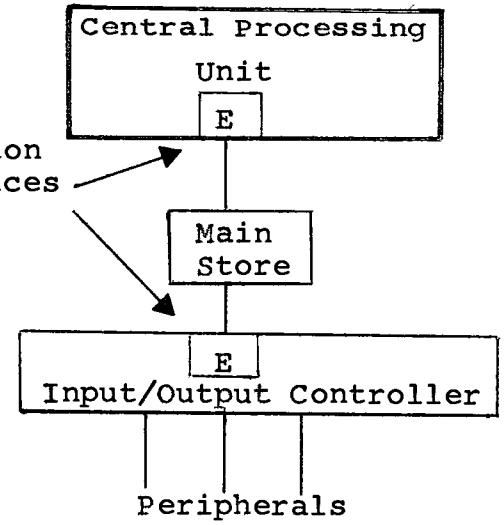


Figure 23b: The I/O controllers contain the encryption devices.

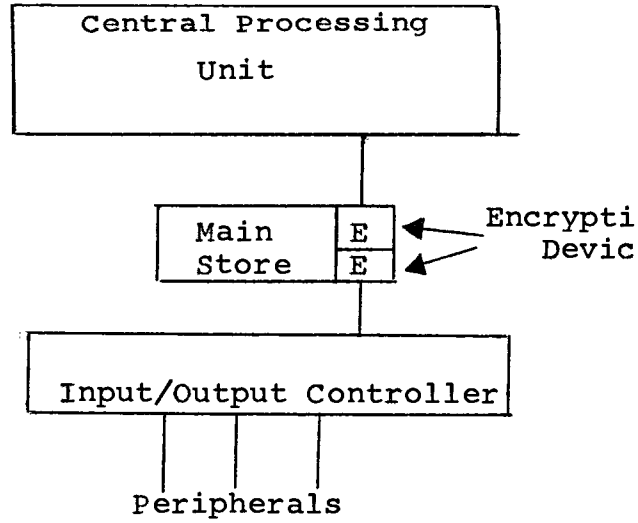


Figure 23c: The encryption device is placed in the main store.

These three diagrams illustrate the various means for providing encryption in the main store. Figure 23a may be fairly difficult to incorporate within the system due to complexities in controlling the encryption device. The decision to use configurations 23b and 23c is dependent upon the design of the main store/processor interface and the number of main store modules (Keys & Clamons, 1974).

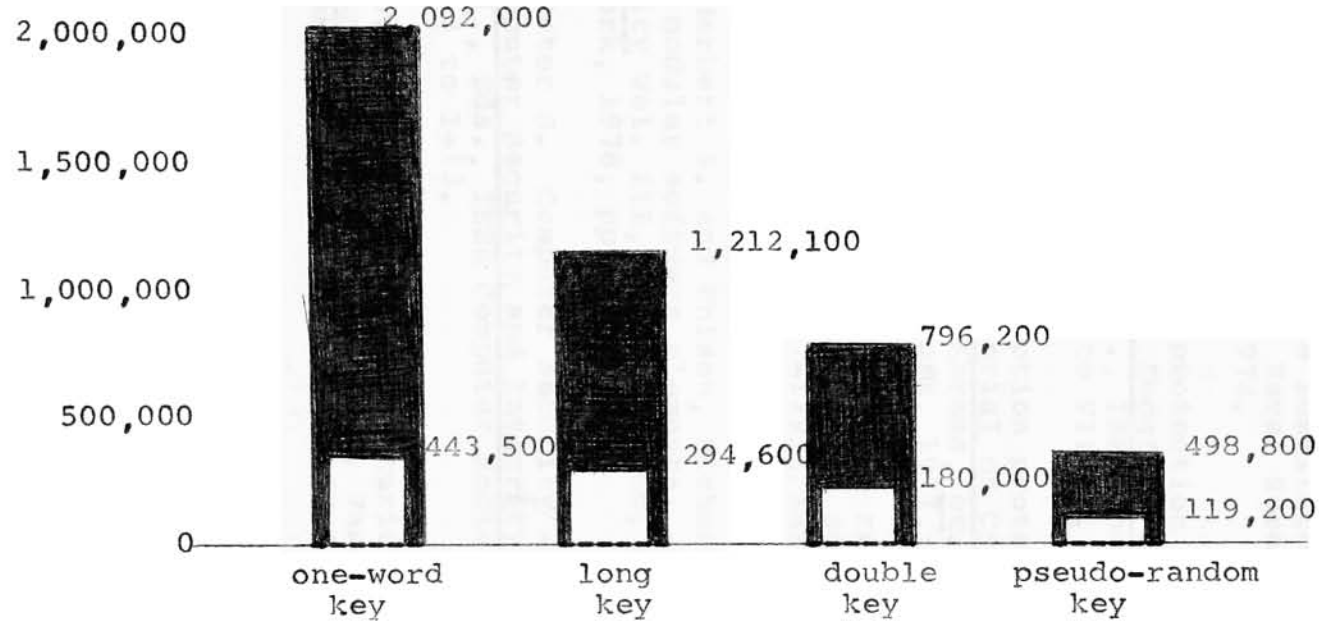


Figure 24: This diagram illustrates the encipherment speeds required for the various tests. Note that the dark colored bars represent assembly language while the light colored bars designate Fortran (Friedman & Hoffman, 1974).

Bibliography

1. Anderson, James P. Information security in a multi-user computer. Advances in Computers, Vol. 12, 1971, 30-36.

2. Bartek, D. J. Encryption for data security. Honeywell Computer Journal 8, 2(1974), 86-89.

3. Benedict, G. Gordon. An enciphering module for multics. MAC TM-50, Massachusetts Institute of Technology, prepared for the Office of Naval Research Advanced Research Projects Agency, July 1974.

4. Branstad, Dennis K. Data protection through cryptography. In Tutorial on Computer Security and Integrity, Marshall D. Abrams, et. al., Eds., IEEE Computer Society, New York, 1977, pp. VII-25 to VII-28.

5. Branstad, Dennis K. Encryption protection in computer data communications. In Tutorial on Computer Security and Integrity, Marshall D. Abrams, et. al., Eds., IEEE Computer Society, New York, 1977, pp. VII-29 to VII-34.

6. Branstad, Dennis K., Gait, J., and Katzke, S. Report of the workshop on cryptography in support of computer security. Rep. NBSIR 77-1291, National Bureau of Standards, Sept. 21-22, 1976.

7. Bright, Herbert S. and Enison, Richard L. Cryptography using modular software elements. In Computers and Security Vol. III, C. T. Dinardo, Ed., AFIPS Press, New York, 1978, pp. 181-191.

8. Browne, Peter S. Computer security - a survey. In Tutorial on Computer Security and Integrity, Marshall D. Abrams, et. al., Eds., IEEE Computer Society, New York, 1977, pp. I-3 to I-13.

9. Burns, Kevin J. Keys to DBMS security. In Computers: Auditing and Control, Elise G. Jancura, Ed., Petrocelli/Charter, New York, 1977, pp. 169-173.

10. Conway, R. W., Maxwell, W. L., and Morgan, H. L. On the implementation of security measures in information systems. Commun. ACM 15, 4(April 1972), 211-220.

11. Davida, George I. Security and privacy. In Proceedings of Very Large Data Bases: 4th International Conference on Very Large Data Bases, S. Bing Yao, Ed., West Berlin, Germany, Sept. 1978, p. 54.

12. Davida, George I., Wells, David L., and Kam, John B. Security and privacy. In COMPSAC 1978 Proceedings: Computer Software and Applications, IEEE Computer Society, New York, 1978, pp. 194-203.

13. Davis, Ruth M. The data encryption standard in perspective. In Computer Science and Technology: Computer Security and the Data Encryption Standard, Dennis K. Branstad, Ed., National Bureau of Standards, Washington, 1978, pp. 4-13.

14. Denning, Dorothy E. Secure personal computing in an insecure network. Commun. ACM 22, 8(August 1979), 476-482.

15. Diffie, Whitfield and Hellman, Martin E. Multiuser cryptographic techniques. In Computers and Security Vol. III, C. T. Dinardo, Ed., AFIPS Press, New York, 1978, pp. 193-196.

16. Diffie, Whitfield and Hellman, Martin E. New directions in cryptography. IEEE Transactions on Information Theory IT-22, 6(November 1976), 644-654.

17. Diffie, Whitfield and Hellman, Martin E. Privacy and authentication: an introduction to cryptography. Proceedings of the IEEE 67, 3(March 1979), 397-427.

18. Dinardo, C. T. Cryptology - introduction. In Computers and Security Vol. III, C. T. Dinardo, Ed., AFIPS Press, New York, 1978, pp. 151-153.

19. Ehrsam, W. F., Matyas, S. M., Meyer, C. H., and Tuchman, W. L. A cryptographic key management scheme for implementing the data encryption standard. IBM Systems Journal 17, 2(1978), pp. 106-125.

20. Evans, Arthur Jr. and Kantrowitz, William. A user authentication scheme not requiring secrecy in the computer. Commun. ACM 17, 8(August 1974), 437-442.

21. Everton, Joseph E. A hierarchical basis for encryption key management in a computer communications network. In Trends and Applications: 1978 Distributed Processing, National Bureau of Standards, Gaithersburg, MD, May 1978, pp. 25-32.

22. Feistel, Horst. Cryptography and computer privacy. Scientific America 228, 5(May 1973), 15-24.

23. Feistel, Horst, Notz, W. A., and Smith J. L. Cryptographic techniques for machine to machine data communications. Res. Rep. RC 3663, IBM Thomas J. Watson Res. Ctr., Yorktown Heights, New York, December 1971.

24. Friedman, Theodore D. and Hoffman, Lance J. Execution time requirements for encipherment programs. Commun. ACM 17, 8(August 1974), 445-449.

25. Girsdansky, M. B. Data privacy - cryptology and the computer at IBM research. IBM Research Reports 7,4(1971), 1-12.

26. Hsiao, David K. and Kerr, Douglas S. Privacy and security of data communications and data bases. In Proceedings of the IEEE 67, 3(March 1979), 397-427.

27. Hellman, Martin E. DES will be totally insecure within 10 years. IEEE Spectrum 16, 7(July 1979), 32-39.

28. Hellman, Martin E. The mathematics of public-key cryptography. Scientific America 241, 3(August 1979), 146-157.

29. Hellman, Martin E. and Diffie, Whitfield. Exhaustive cryptanalysis of the NBS data encryption standard. Computer 10, 6(June 1977), 74-84.

30. Kahn, David. Modern cryptology. Scientific America 215, (July 1966), 38-46.

31. Katzan, Harry Jr. The Standard Data Encryption Algorithm, Petrocelli Books, Inc., New York, 1977.

32. Keys, R. R. and Clamons, E. H. File encryption as a security tool. Honeywell Computer Journal 8, 2(1974), 90-93.

33. Knuth, Donald E. The Art of Computer Programming Vol. II, Addison Wesley, Reading, Mass., 1969.

34. Kolata, G. B. Computer encryption and the national security agency connection. Science 197, 29(July 1977), 438-440.

35. Kilata, G. B. Cryptography: on the brink of a revolution? Science 197, 29(August 1977), 747-748.

36. Konheim, Alan G. Cryptographic methods for data protection Res. Rep. RC 7026 (#30100), IBM Thomas J. Watson Res. Ctr., Yorktown Heights, New York, March 1978.

37. Lempel, Abraham. Cryptology in transition. ACM Computing Surveys 11, 4(December 1979), 285-303.

38. Martin, James. The Computerized Society, Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

39. Matyas, S. M. and Meyer, C. H. Generation, distribution, and installation of cryptographic keys. IBM Systems Journal 17, 2(1978), pp. 126-137.

40. Mellen, G. E. Cryptology, computers, and common sense. In Computers and Security Vol. III, C. T. Dinardo, Ed., AFIPS Press, New York, 1978, pp. 155-165.

41. Morris, Robert, Sloane, J. J. A., and Wyner, A. D. Assessment of the national bureau of standards proposed federal data encryption standard. Cryptologia 1, 3(July 1977), 281-291.

42. National Bureau of Standards. Data encryption standard. Federal Information Processing Standards Publication #46, Washington, 1977.

43. Needham, R. M. and Schroeder, M. D. Using encryption for authentication in large networks of computers. Commun. ACM 21, 12(December 1978), 933-999.

44. Popek, Gerald J. and Kline, Charles S. Encryption and secure computer networks. Computing Surveys 11, 4 (december 1979), 331-356.

45. Popek, Gerald J. and Kline, Charles S. Protocols, algorithms, and signatures in computer networks. In Foundations of Secure Computation, Richard A. Demillo, et. al., Eds., Academic Press, New York, 1978, pp. 133-153.

46. Purdy, George B. A high security log-in procedure. Commun. ACM 17, 8(August 1974), 442-445.

47. Rabin, M. Digitalized signatures. In Foundations of Secure Computation, Richard A. Demillo, et. al., Eds., Academic Press, New York, 1978, pp. 155-168.

48. Rivest, R. L., Shamir, A., and Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2(February 1978), 120-126.

49. Simmons, Gustavus J. Cryptology: the mathematics of secure communications. The Mathematical Intelligencer 1, 4(January 1979), 233-246.

50. Smith, J. L. The design of lucifer, a cryptographic device for data communication. Res. Rep. RC 3326, IBM Thomas J. Watson Res. Ctr., Yorktown Heights, New York, April 15, 1971.

51. Sykes, David. Protecting data by encryption. Datamation 22, 8(August 1976), 81-85.

52. Sugarman, Robert. On foiling computer crime. IEEE Spectrum 16, 7(July 1979), pp. 31-32.

53. Walker, Bruce J. and Blake, Ian. Computer Security and Protection Structures. Dowden, Hutchinson, & Ross, Inc., Stroudsburg, PA, 1977.

54. Yasaki, Edward K. Encryption algorithm: key size is the thing, Datamation 22, 3(March 1976), 164-166.