IBM Research

CRYPTOGRAPHIC CODING FOR DATA-BANK PRIVACY

H. Feistel

March 18, 1970

RC 2827

[redacted]

Yorktown Heights, New York San Jose, California Zurich, Switzerland New York, N.Y.

Copies may be requested from:
IBM Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

CRYPTOGRAPHIC CODING FOR DATA-BANK PRIVACY

H. Feistel
IBM Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT: This report deals with a secure digital network concept in which computers communicate with computers. It is envisaged to be fully automatic in all aspects, including the switching at the nodes. All operations are protected by cryptographic coding techniques.

The security functions are broad in concept and include not only message secrecy in the usual sense but also high security verification. The latter provides assurance that all communications are of legitimate origin and rejects all data which are suspected as having been corrupted by accident or intent.

The cryptographic techniques employed are time independent ciphers which encrypt large blocks of binary digits. A special manner of using such block codes, called step coding makes possible a variety of super and multiple level enciphering techniques, previously not possible.

Although the concept is very broad in scope and applicability, it has been found useful to illustrate it in terms of a conceptual data bank scheme.

RC 2827 (#13260)
March 18, 1970

TABLE OF CONTENTS

1. INTRODUCTION

A Data Bank is essentially a machine to machine communications network in which input terminals are connected to a centrally located computer, the physically secured CPU.

The most outstanding feature of the kind of network structure we are talking about is that it must function reliably in a hostile environment. Secrecy in the usual sense, that is concealment of the meaning of the messages conveyed, would form the basic element of protection. This is required to insure the privacy of those forming the data bank community. But machine communications systems, in contrast to systems which can enlist the subtle filtering capabilities of the human brain are very sensitive to interference and deception. Without special protection computers are easily fooled and this can become an intolerable burden to a data bank operation if this remains unnoticed. Both accidental and intentionally designed errors must be detected with very large safety margins. A machine to machine communications network requires a properly secured method which assures the receiver that all incoming communications are of legitimate origin and uncorrupted. In military systems such methods are called authentication. We shall present a method called centralized verification. In contrast to military systems, where all participants have the same key, our system emphasizing individual privacy permits each individual member of the data bank to have his own private key.

It is perhaps surprising to find that a data bank network requires for its properly secured operation a highly sophisticated cryptographic system. To understand the deeper reasons for this let us take a look at the most important special requirements which a network like a data bank presents.

1) A data bank network can be tapped.

2) Both in principle and practice illicit elements can obtain the clear texts belonging to intercepted cryptograms. The designer can moreover not ignore the possibility of having cleverly designed clear texts "planted" at the terminal, for instance through collusion.

3) Legitimate communications may be recorded and retransmitted when they are no longer valid. This could be very harmful to the data bank and useful to the hostile element even through the meaning of the codes remain unknown.

4) The operation of a data bank will be seriously hampered if not made impossible if invalid data are furnished to the CPU, whatever the origin of the corruption.

5) A great deal of the traffic in a data bank will essentially be non-redundant. One false digit in such messages can completely alter their meaning. This cannot be tolerated. All traffic being suspected as having been corrupted by accident or intent must be rejected with very large safety margins.

The above observations along with certain operational aspects of cryptographic data processing lead to the following preliminary conclusions:

a) Stream ciphers are sensitive to time delays. A stream cipher which is out of step by a single bit position deciphers into complete garble.

b) Stream ciphers cause difficulties at network nodes. Since a node cannot handle more than one incoming cipher at a time, those being queued for later processing must be decrypted lest they produce garble at the receiver due to their delay. Schemes to account for such delays tend to be cumbersome and unreliable in the context of essentially untrained communicators, such as are to be found at the terminals of a data bank.

c) Conventional stream generators which encipher by using bit by bit addition mod 2 do not properly entangle password and data digits. An opponent must be prevented from being able to alter a digit representing data without also altering the password structure. To accomplish this with a stream generator would essentially involve the use of more than one cryptobox, which is undesirable. Moreover it would not remedy the basic inconveniences of a) and b).

d) To assure a high level of cryptographic security, a high degree of communications discipline must be maintained at all times over long periods of time, a requirement which cannot be expected from an ordinary data bank community.

e) Jamming is not of the same concern to a data bank as it would be in a similar military data network. A military system usually urgently requires the data right at the time when they are being transmitted and much complication and expense is required to make this possible. A possible jammer in a data bank network would be instantly detected as being present and his corruptions would be rejected with high probability. While the system would temporarily breakdown, the jamming source would have to be sought after and removed as any other troublesome source.

The above reasons and requirements have lead us to consider cyrptographic devices in which the coding process is not time dependent. We shall use cryptographic techniques which will encipher 128 binary digits as a single block. The fact the the cryptogram digits produced by such a system are all very involved and complicated functions of all input digits, i.e. password digits and data digits do not appear as individuals in the cipher, will be used as an agent of the very sensitive error detection which we require for effective verification. Fortunately, block cipher design, which faced almost unsurmountable difficulties in the age of electromechanical gadgets, has now almost unlimited possibilities even with the use only of common arithmetic computer components. In what follows we shall present the various aspects of our concept in more detail. In particular we shall be dealing with product ciphers, i.e. ciphers which build up cryptographic strength by repeated use of suitable basic cryptographic transformations, such as the classic systems of substitution and transposition.

2. PRODUCT CIPHERS

The concept of a product cipher is actually not very new. Before World War I various cumbersome ciphers using several steps of encryption operations were already considered. One of the first really successful simple product systems seems to have been the German ADFGVX system, making its appearance late in that war. (For details see commercially available texts on cryptography*.) This system, as later inventions, coupled so-called Fractionation with Transpositions and was quite successful in the environment of the day. In fact the ADFGVX had essentially the ingredients that appear to be necessary to produce what one might call a "strong" cipher. The intrinsic power of its underlying concepts, however, was hampered by the fact that the system was tailored for front line use. Moreover it was limited to a pencil and paper version because the technologies of the time could not be enlisted for help, for these were not ready for the task. In practice all this implied that the product operations of the basic transformations had to be restricted to essentially two rounds, with their consequent conquest by a most competent opponent.

Interest from the military direction in product transformations of the n-gram variety seems to have almost totally disappeared with the advent of so-called rotor or wired-wheel machines. These served the military interests extremely well. It is hence not surprising that much of the research and development effort in cryptographic design went into their perfection.

The subject matter next reappeared in a scientific journal with the

*For instance David Kahn, The Code Breakers, McMillon Co., N. Y., 1967

publication of the Lester Hill ciphers.* These again were simple product systems, involving a linear transformation (matrix multiplication) and a non-affine mapping. Lester Hill's ciphers are of special interest because the inventor strongly emphasized the need for mechanization, which in his case meant "mechanical" indeed. At least the possibility for iterated products with many rounds in practical systems became now apparent. Historical fact, however, indicates no further development of the ideas, at least as far as the open literature is concerned. The concept of the rotor systems and their cipher streams seem to have better suited all major users concerned.

A renewed interest in product systems was potentially sparked with the publication of Shannon's well known paper on Secrecy Systems.** In his section on Practical Cipher Design he introduced the notion of a Mixing Transformation, which involved products of transformations in a somewhat special way. Shannon also outlined certain intuitive guides to the design of (possibly) strong ciphers and introduced the notions of Confusion and Diffusion and described their role in Cipher Design and Analysis.

Although Shannon merely suggested possibly promising classes of mixing transformations, he did not fail to open up almost unlimited possibilities to invention, design, and research. Shannon in particular suggested repeated products of Linear Transformations (presumably matrix multiplications), simple substitutions, additions mod 26 of neighboring letters and

*Lester S. Hill, "Cryptography in an Algebraic Alphabet", The American Mathematical Monthly", June-July, 1929.
**Communication Theory of Secrecy Systems, C. E. Shannon, Bell Systems Technical Journal, pp. 47-59.

letter transpositions. He seemed to envisage systems in which the "mixes" are not actually keyed, but part of the "System". They were bracketed at beginning and end and in between mixes with one of the simple standard ciphers, which were the ones to be keyed.

In a modern binary digital computer environment, Shannon's suggestions inspire almost automatically their adaptation to relevant arithmetic computer gadgetry which is readily available. In a later section we shall describe in detail a specific product system which we propose to investigate and analyze as a basis for actual use in data bank applications. For the present it may suffice to say that we shall devise a system which will encipher binary digits block by block (each block being perhaps of dimension 128--but there is nothing very critical about that, except that the block must be sufficiently large to prevent such brute force attacks as for instance complete cataloguing). The component mappings of the product system will involve general-substitution (on blocks of dimension four, five, or six) coordinate permutation (corresponding to the classical "transposition") addition mod 2 and similar common arithmetic operations. A detailed description of the secrecy system will be given in Section 5.

Independent of any particular definition, all cryptographic systems may be regarded as a very large set of possible transformations, out of which the key selects the specific mapping in use.

Now, that the reader has had a preliminary introduction to the possibilities in product cipher design we are ready to discuss their operational

uses in actual applications. Having clarified their basic utilization, we shall then proceed to assemble the building blocks into a complete Data Bank Concept.

3. CENTRALIZED ADDRESS IDENTIFICATION AND DATA VERIFICATION

3.1 Introduction

An automatic network in which machines communicate directly with machines is potentially wide open to deception. Thus, it would have catastrophic consequences to a data bank if unauthorized elements could deposit, withdraw or alter the data stored in the central file.

It is hence clear that our coding procedures must not only be able to provide cryptographic protection to hide the meaning of our data transmissions, but must also guarantee a very high degree of safety against communications deception in the broadest sense possible. Just as in military networks, two basic approaches to verification are commonly used for computers, passwords, and so called handshaking. Either method involves the transmission of identification information.

In the password method the information is locally available and prearranged, in the handshaking method it is generated on the spur of the moment as a challenge and must be returned in prearranged format to the originator of the challenge. In its simplest form verification could be based on a sufficiently long block of randomly generated digits, known only to the sender and the receiver. It becomes at once apparent that in a hostile environment of even minimal sophistication, such a password could be used once only, for a single transmission amounts to publication, making it available to anyone who might want to misuse it. Moreover, it is also apparent that a password which is isolated from the data which are to be verified is essentially useless, since anyone familiar with the general arrangement of

such codes could tamper with the data part of the codes leaving the password unharmed and thus deceive the computer, which without special protection is easily fooled. In contrast to military identification systems in which all participants are required to be in possession of the same key, a data bank can be realized only if each member of the of the community can have a private key. In the data bank concept here described this is made possible by a scheme called "Centralized Verification".

The verification process must protect not only against forged codes designed by a possibly highly sophisticated intruder, but also against any attempt to alter our own communications, including retransmission. The process must reject any communications corruption at the slightest suspicion with enormously high safety factors.* We shall first illustrate such a technique on the basis of the handshaking method.

3.2 Centralized Verification by Handshaking Method

In Fig. 1 we present a flow diagram in which a typical terminal (left-hand side of the diagram) attempts mutual identification with the physically secured computer location, the vault.

The typical terminal subscriber with address "A" has a cryptographic device π which makes it possible for him to encipher a large block of binary digits (say 128) as a unit. The respective Terminal A has been assigned a private key K_A. The central location, or vault, possesses the same cryptographic device as the terminals and in addition a listing of all subscribers

*This is automatically accomplished by the use of block ciphers, which are so sensitive to transmission interference, that the slightest alteration turns the decipherment including password into garble, providing us with what one might term to be cryptographically protected error detection.

CENTRALIZED
ADDRESS IDENTIFICATION
AND
DATA VERIFICATION

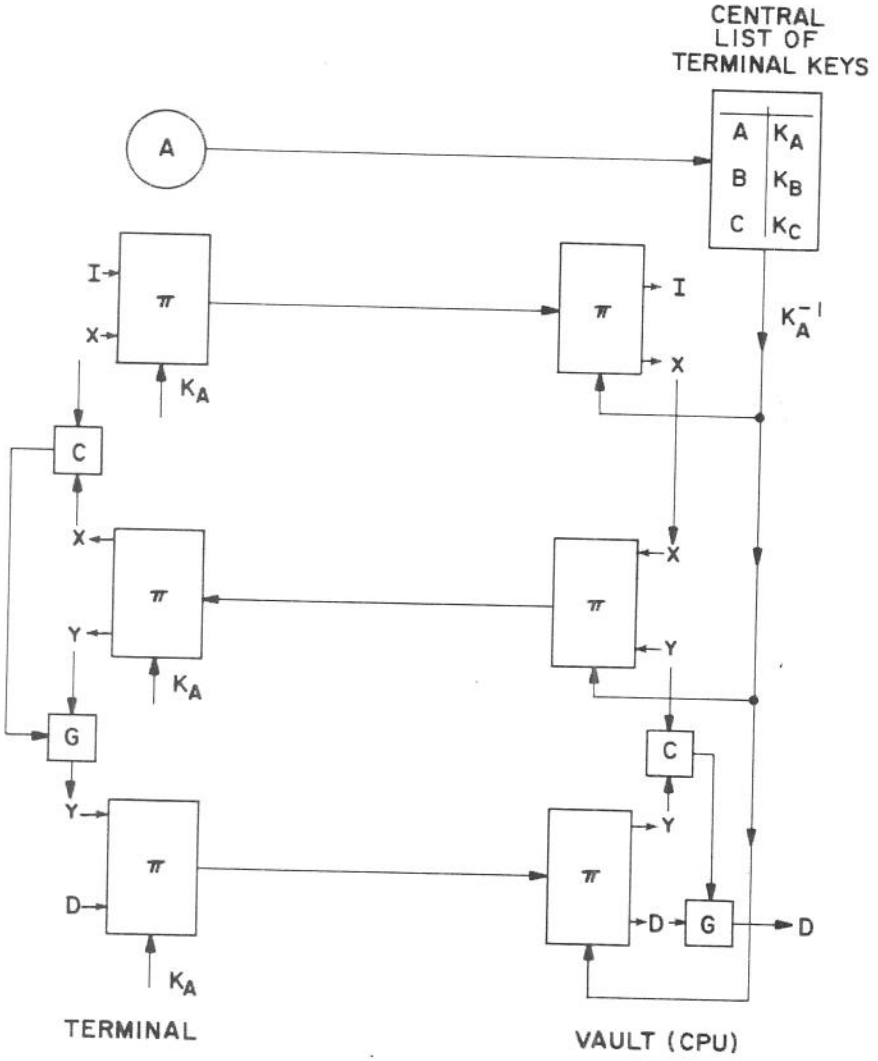


Figure 1

and the keys corresponding to their address-code. The system operates as described below.

The terminal selects code I which is to inform the vault that this subscriber wishes to initiate a verified data transaction with the vault. Along with the code group I the terminal inserts random digits X, which it also remembers for later comparison and verification upon receipt of the return communication. Code groups I and X are enciphered as a block, resulting in a code which can no longer be read or interpreted by anyone not in possession of the key K_A. This cryptographic code is now preceded by the individual address of the terminal in question. This address code will not be enciphered, i.e. it will be transmitted in the clear. The entire code ensemble is now transmitted to the vault.

Upon reception of this communication the vault first reads the address specified by the terminal. It then locates in its central file the cryptographic key K_A assigned to terminal with address A. We here assume that each terminal is assigned its private key, known to no one else but the vault. The vault on the other hand has the complete key listing for all subscribers of the data bank community. The keys are, of course, held in complete confidence and are stored and protected under physical security. Moreover, the key listing could be enciphered by a special vault key, only accessible to the corporation.

At this stage of the identification cycle the vault still views the request for data transfer with suspicion. Having used key K_A^{-1} to decipher the

communication, it finds request I and digits X. A test for secure identification is now prepared. The vault reasons that if the terminal is indeed a legitimate member of the data bank community, it must be in possession of key K_A and hence be able to interpret the communication which the vault will now prepare.

For this purpose, the vault takes random digits X just received from A and lines these up with random digits Y generated in turn by the vault. Just as the terminal A stored its digits X for later comparison, the vault will memorize digits Y for its own verification purposes. Digits X and Y are now enciphered by the vault as a block with the key K_A^{-1} and returned to terminal A.

Upon reception terminal A will decipher the communication from the vault and look for digits X stored in its memory. If they are present and in agreement with those stored in the memory, the terminal has now a high degree of assurance that the communication just received came indeed from the legitimate vault. The terminal has identified the vault. The terminal will now prepare digits for return to the vault and line up block Y with a block D of data digits, representing the information to be deposited in the vault for storage or processing. The resulting cryptogram is now transmitted to the vault. The vault which is awaiting this reply communication will, after decipherment with K_A^{-1} look for the presence of Y. If digits Y received match digits Y stored, the communication is regarded as having originated by a legitimate terminal A. Terminal and vault have mutually identified each other and the vault has verified the data as being of legitimate origin for subsequent storage or processing.

The method as here described differs from customary handshaking methods in that it is based on the centrally located and physically secured subscriber key list. Handshaking methods have the desirable property of not being sensitive to transmission delays, say in switching centers. They do, however, require a back and forth checking procedure which may in some applications be undesirable.

We observe that for a data-transaction involving many contiguous blocks the handshaking operation for identification and verification need to be performed once only, i.e. at the beginning. The only requirement which has to be fulfilled is that each block is tied together with its neighbors by a suitable redundancy structure anchored within the cipher-block. A typical possibility is the following:*

$$(D_3 \; ; \; D_2)\, S_A \; ; \; (D_2 \; ; \; D_1)\, S_A \; ; \; (D_1 \; ; \; R)\, S_A$$

i.e. each code contains a repetition of the data from its preceding neighbor.

Such a data transaction thus literally involves a data train consisting of a lead-code and data trailers. The computer can, of course, establish the end of such a train automatically because the redundancy structure will no longer be repeated if the codes which follow belong to a different data train. We also note that such a data train can be delayed and switched at various nodes of the data net, without loosing its potential for verification at the vault.

The need for handshaking arises, of course, solely from the fact that our communication must convey to the receiver some information about its

*The symbolism $D_3 \; ; \; D_2$ denotes a direct sum, i.e. digits D_3 lined up with digits D_2. S_A means encipherment with key A.

origin, i.e. verification information. In some applications the handshaking procedure may not be acceptable. In this case, we could use the password method. The code train could then be arranged as follows:

$(D_3 ; P) S_A ; (D_2 ; P) S_A ; (D_1 ; P) S_A$ or $(D_3 ; D_2) S_A ; (D_2 ; D_1) S_A ; (D_1 ; P) S_A$,

where P is an ever changing password, different for each data train.

3.3 Centralized Verification by the Password Method

In the password method of verification the random challenges, which play the role of the password in the handshaking method, are replaced by a locally generated pre-agreed password. The advantage of not requiring a return communication brings with it the need for password synchronization. The random challenges in the handshaking method provide automatic protection against repeat-deception, i.e. any attempts by unauthorized elements to deceive the vault by retransmission of outdated traffic. This feature must now be replaced by some other means of outdating and updating the password so that communications of the past cannot be recorded and reused when they are no longer valid. The fact that the messages which dominate our data bank traffic are frequently highly non-redundant, necessitates, as already explained, that password and data bits are mathematically thoroughly interlocked and bundled so that after encryption the output digits represent no longer individual data or password digits but rather very involved functions of each other. The cryptogram must not offer the opponent any information as to how data and password are reflected in the cipher.* The simplest way, and perhaps the best, is to use some counting procedure or coded clock time as the ever changing password, i.e. a new password for each new cipher-block to be transmitted. The time standard would, of course, keep advancing even if no messages are transmitted. In Fig. 2 we illustrate a typical pass-

*This we accomplish, as in the handshake method, by the use of block ciphers.

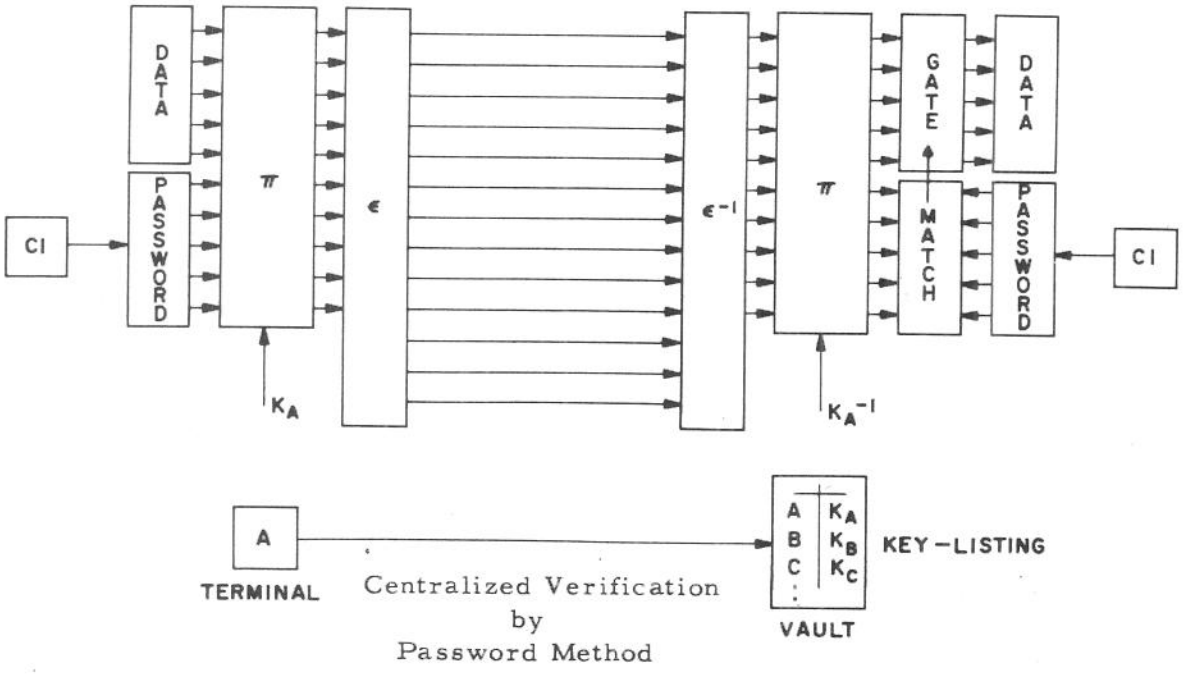

DATA
PASSWORD
CI
π
ε
K_A
ε^{-1}
π
K_A^{-1}
GATE
MATCH
DATA
PASSWORD
CI
A
TERMINAL
Centralized Verification
by
Password Method
A
B
C
K_A
K_B
K_C
KEY-LISTING
VAULT

Figure 2

word verification system.

D represents the data, P the time controlled password. Both are enciphered as a block by the Cryptosystem π. After encryption the cipher block is encoded by an error-correction code, matched to the noise characteristics of the channel in use. Whenever extraordinarily severe noise bursts on the channel occur, the error correction code will break down. This will cause no harm, for the receiving end of the system provides an optimal degree of error detection, as we shall now demonstrate.

As shown in Fig. 2 the receiving center, i.e. the vault, has a password generator identical to the one used in the transmitting terminal and in absolute synchronism with it. We observe that in the absence of interference the transmission from the terminal will be deciphered correctly, resulting in a decipherment yielding the password and data as originally enciphered. The password unfolded after decipherment will thus match the corresponding password locally generated. This causes the comparator C to open the gate G to accept the data D as being of legitimate origin. We have silently assumed that the communication was, as before, preceded by the address indicator of the terminal (in the clear) just as in the handshaking method, allowing the vault to look up the key K_A corresponding to terminal A. If the communication is subjected to natural interference beyond the capabilities of the error correcting code, the decipherment will turn into garble, causing a mismatch at the comparator which in turn locks the gate causing the data received to be rejected as being of suspicious origin. The same will obtain if the corruption

is introduced by any attempt to alter the communication by clever design, brute force, or repeat of past traffic.

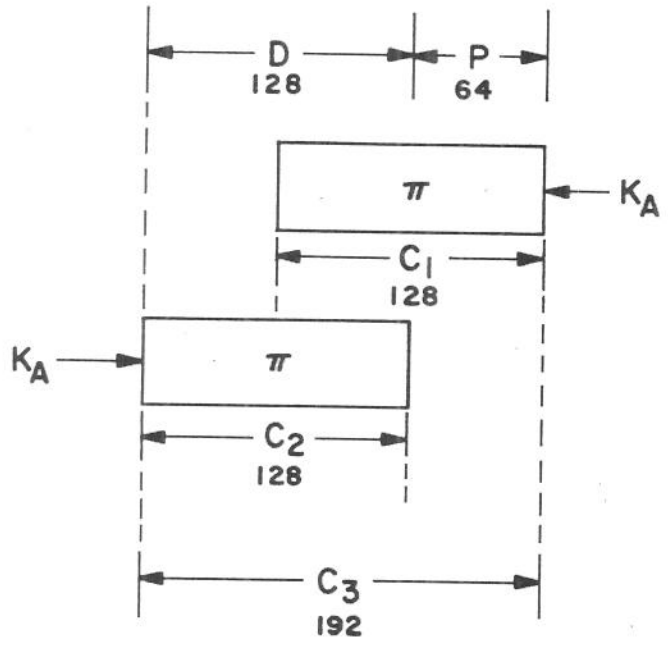
For a password n digits long an opponent has only one chance in 2^n to deceive the system. We also observe that the relative message expansion introduced with the password is simply a function of the dimension of the data vector attached to it, i.e. a box design problem. Once the degree of verification protection has been decided upon, the efficiency of the code is simply a function of the input dimension to S which available technology can provide.

3.4 Stepped Products and Multiple Level Encipherment

The cryptographic coding and verification routines just described suffice for very simple types of data transactions only, say between point A and point B. For a more sophisticated system serving a whole community of subscribers, requiring various computational services, and individual needs, we shall have to make more effective use of our cryptographic techniques and their inherent potential.

To this end we introduce the notion of multiple level step-coding of block ciphers. Our concepts are standardized on the use of one and only one type of cryptographic device which will process blocks of binary digits of a fixed dimension, say 128 or perhaps 256 (a number which will be largely determined by the needs of the practical user). Stepped coding refers simply to a special method of sliding the encipherment operation along a sequence of message digits, generally longer than the basic block dimension, to obtain cryptographic interlocking by a suitable overlap. To explain the basic technique, we refer the reader to Fig. 3 .

Here D represents a block of 128 data digits, P represents a 64 digit password. The boxes π represent the cryptographic system, and K_A the key of subscriber A. The step coding procedure operates as follows. Box π which operates on a block of 128 binary digits, is lined up with D and P in such a manner that it will first encipher the 64 digits of P and 64 of the 128 digits of D, resulting in the cryptogram C_1. Box π is next stepped to the left (now represented by the second, lower Box π) so that it will now encipher



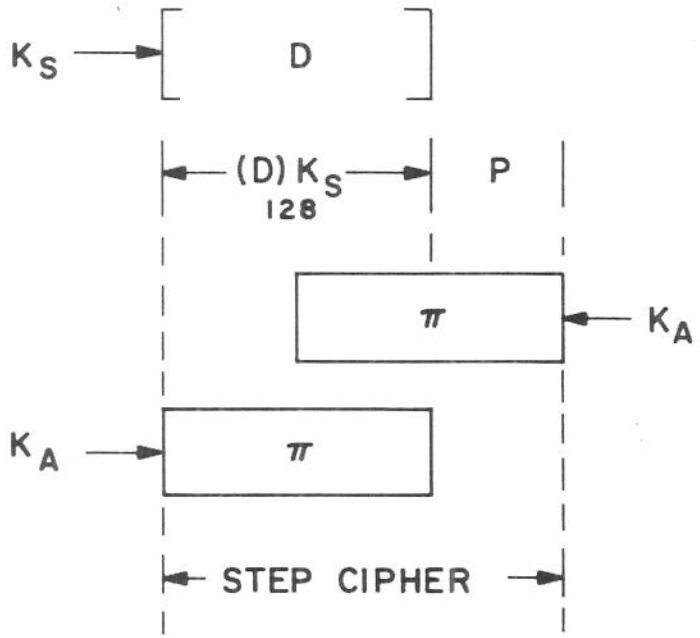
MULTIPLE LEVEL ENCIPHERMENT
WITH STEPPED BLOCK-CIPHER

Figure 3

the remaining 64 digits of D together with the lefthand 64 digits of the cryptogram C_1. This will result in a cryptogram C_2, which is now a function of both the D and P. The total cipher C_3 now comprises C_2 and the right half of C_1. We observe that the system is reversible and that any corruption introduced anywhere along the cipher C_3 will garble not only the data but also the password, which is what we require. For C_1 this is obvious; for C_2 it is true because any corruption introduced there will upon decipherment of C_2 infect C_1.

Such a stepped code opens many possibilities for very flexible operation of a secure network through multiple level superencipherment In Fig. 4 which is essentially like Fig. 3 , we have bracketed the Data Vector D. This is to symbolize that the arrangement would make possible a preliminary encipherment of the Data digits D with a special key, K_S, known only to the individual subscriber and no one else using, of course, the same standardized box π. This operation amounts to a superencipherment and per se is nothing new except that our present arrangement makes it possible for the receiver to verify the communication as being of legitimate origin <u>without</u> having to have access to the data themselves which can remain encrypted and cannot be read by anyone but the subscriber A. K_S would offer the subscriber a special service of <u>complete privacy</u>, shared with no one, for data which are highly confidential and simply to be filed for complete security.

We have now assembled the basic ingredients necessary for the automatic operation of a secure data bank. Before we return to various other ways



Super-Encipherment with K_S
for
Absolute Privacy

Figure 4

of using the stepping principle, we shall now proceed to illustrate the operation of the basic Data Bank concept.

4. THE VAULT AND ITS ASSOCIATED NETWORK

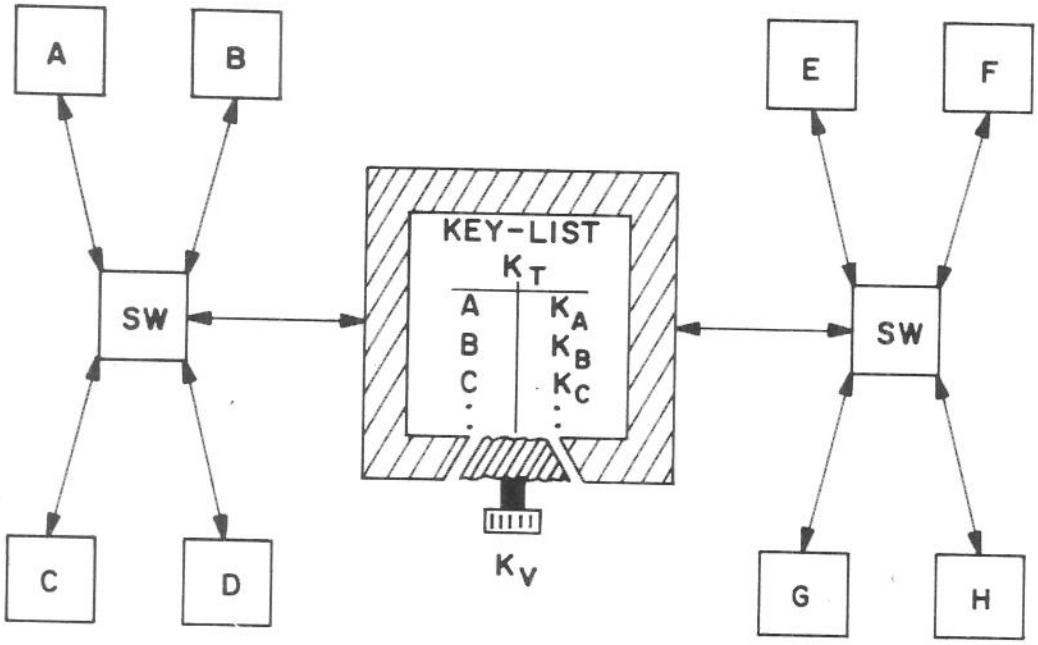
The heart of our Data Bank Network is the so called Vault, which is properly secured physical location of the central data processing facility consisting of a time sharing CPU and appropriate storage or filing facilities. In Fig. 5 we illustrate the basic concept as a simple network with two switching centers SW and eight subscribers, numbered one to eight. The vault is assumed to have in its possession a listing of all subscribers and their associated keys, i.e. secrecy and verification are to be accomplished on a centralized and individual basis. A terminal desiring to perform a data transaction with the vault will proceed exactly as described in Section 3. The request to be verified and accepted as a legitimate member of the data-bank is preceded by the address, in the clear, of the terminal. As soon as a terminal has established itself as a bona fide member, many other security functions, beyond those already indicated by the different key classes, can be accomplished.

The system would operate with seven distinct types of keys, indicated in the diagram. We shall now proceed to explain their functions

K_T is the basic key. To each terminal a distinct key K_T will be assigned and listed in the vault along with the address associated with each key. This is, of course, the key which we used to illustrate the centralized verification scheme. K_T is known only to the individual subscribers and the vault.

K_{SW} is shared by the switching nodes and the vault only. We shall describe its role later on.

The Vault and Subscriber Network



THE SEVEN SEALS!

K_T : SUBSCRIBER & VAULT

K_S : SWITCH & VAULT

K_S : SUBSCRIBER ONLY

K_P : PARTNERS ONLY

K_F : FILE

K_C : CORPORATE ONLY

K_V : PHYSICAL VAULT

Figure 5

K_S is specially assigned to a terminal upon request and would be known to the individual subscriber only. Its role is, as we already remarked, to assure complete and absolute privacy for those requiring it

K_P is a partnership key shared by two or more partners who desire to communicate among themselves without anyone else, including the vault, being able to interpret their traffic. Functionally, it is exactly the same as Key K_S. K_P is intended to be unknown

K_F is a filing key; known only to the vault and to be described later.

K_C is a special key for corporate use only and K_V is the physical key to the Vault Facility.

We are now ready to illustrate the operation of the vault. Of basic importance are the verification techniques described earlier. The vault must be able to verify the terminal requesting a data transaction as being legitimate. This then can be done either by handshake or reference standard (password) methods.

The most striking feature of this centralized network is the fact that each terminal can securely communicate with any other terminal, without knowing its key! Thus, if terminal A wishes to communicate with some other terminal, say G, it simply sends a request to this effect to the vault The vault will in the usual manner establish the communication as being of legitimate origin It then looks up the Key of G and reenciphers the data with the key of G. The reenciphered data are then transmitted to G who can read and verify them, since he is, of course, in possession of his own key The same

procedure can be arranged between any pair or group of terminals as long as they agree and desire this type of special service.

K_S and K_p, the private and partnership keys respectively, would be used to either deposit with absolute privacy data in the central files or would make it possible for partners to exchange data, all transactions being verifiable by themselves and the vault, the latter nevertheless not being able to read these specially secured communications, as desired by the terminal.

5. THE CRYPTOGRAPHICALLY KEYED DATA PROTECTION SYSTEM

In this section we present a product cipher usable to encrypt any kind of information whatsoever, represented in terms of binary digits (such as (EBCDIC). The device, or the software system representing it, is initially loaded with the clear text message, presumably a computer output, by shifting the digits into a special type of shift-register. Once this shift register is loaded, the system will encipher the register contents according to a keyed schedule. Since the method involves sequential product operations, a specified number or block of information-digits, to be explained in detail later, will be read and subsequently enciphered and thence returned to the message register. The arithmetic operations involved in this transformation process will be repeated several times. These repetitions are called rounds. Each round uses a new, nonsystematic pattern. The rounds, controlled by the basic Keying schedule, are hence non-iterative. Having completed the desired number of rounds, the message register will then contain the final cryptogram, ready for transmission to the destination The emptied register is now again available to receive and to encrypt the next block of 128 binary digits, which will be processed in accordance with the same key and routine. The system can also be operated in an inverse manner. To accomplish this we load the message register with a cryptogram (i.e. into the same digit positions from which the cryptogram was read out). The device will then be operated with a reversed key schedule.

For purposes of illustration we have chosen the dimension of the basic message block to comprise 128 binary digits. Nevertheless, considerably larger dimensions, or somewhat smaller dimensions could be equally well encrypted with a suitably adapted device The goal of the system is to entangle in a very involved manner, on the average, all of the message digits and all of the key digits, so that any analysis of an unlimited amount of intercepted ciphers, or illegitimately derived clear-text cipher pairs, will to the best of our understanding be economically and practically extremely difficult, if not hopeless and unacceptable. In what follows we describe the details of the processing of one 128 digit block and refer to the accompanying data flow diagram (Fig. 6).

As already indicated, the 128 digit block of arbitrary information is initially loaded into 64 2-bit shift registers (in our illustration 24 registers). (This may be done in parallel inserting all 64 digits at the same time or serially by bytes, serially by bits, or whatever may be most convenient.) Different manners of loading the 2-bit registers will, of course, require different gating and shifting schedules. (These are not indicated in our diagram, because such fixed support functions are not really part of the cryptographic technique employed.)

The 64 shift registers may be imagined as having been arranged side by side, as shown in the diagram. We shall refer collectively to the 64 binary

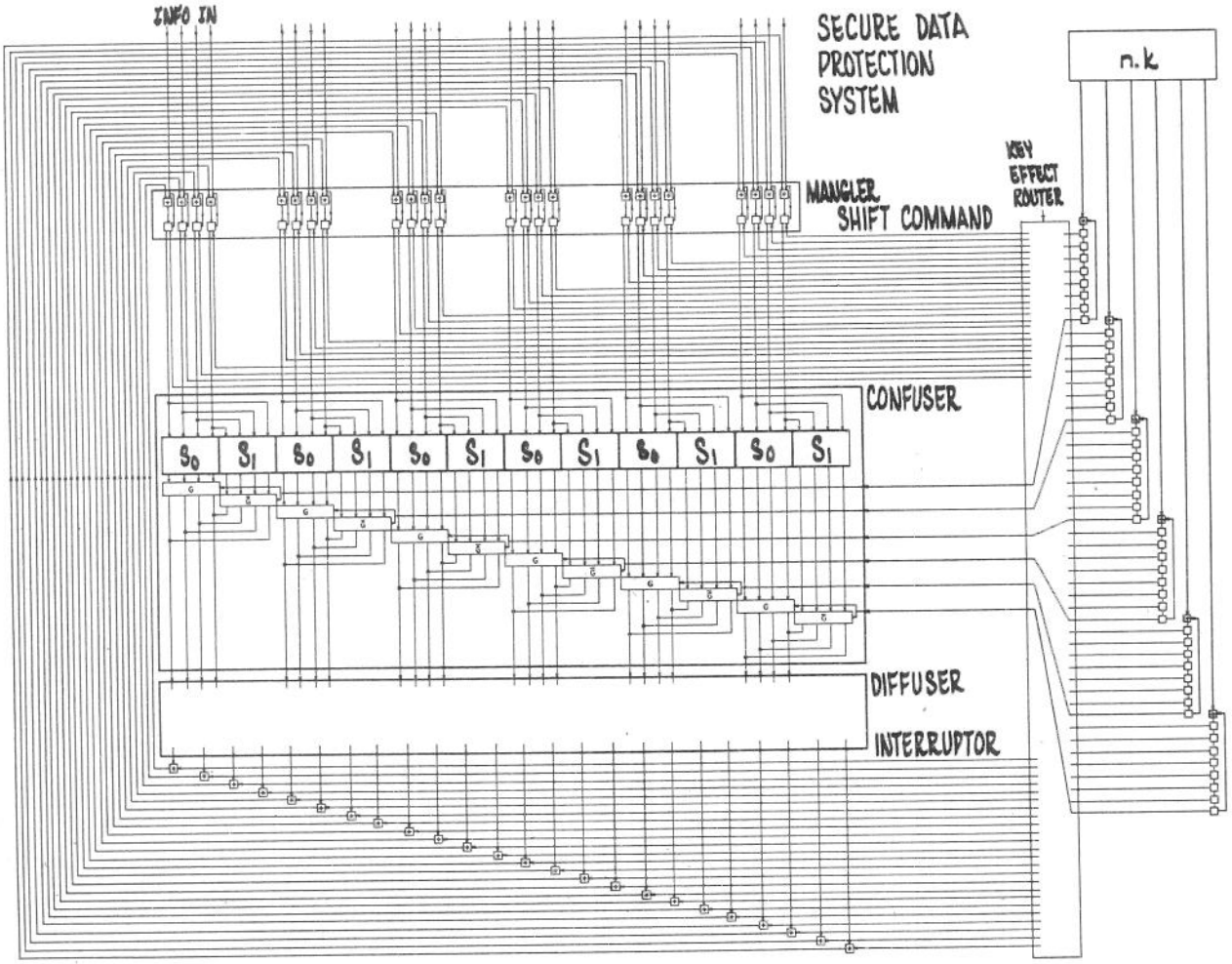


Figure 6

digits loaded into the upper section of each respective 2-bit register, as the top-row, and the contents of the lower-register-sections correspondingly as the bottom row. Shifting will be in the vertical direction, i.e. simply interchange the contents of the corresponding top and bottom sections of the registers. The registers of the bottom row are designed to be read without disturbing their respective bit contents. The top row of registers will be designed to serve for a writing operation. This writing operation entails a simple addition mod 2 onto their respective contents denoted by the ordinary + . (I.e. all additions in this system are mod 2, unless otherwise indicated.)

We now proceed to describe the actual cryptographic components. These consist of four parts:

1) The Confuser

2) The Diffuser

3) The Interruptor

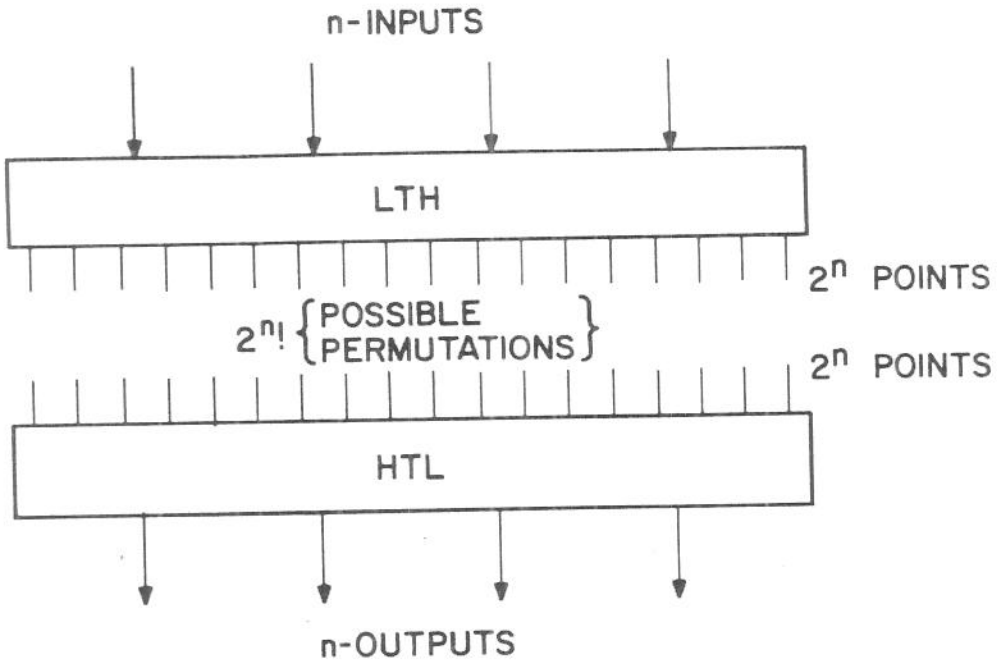
4) The Mangler

In a final device the first three operations would be incorporated together, so that we are dealing with a special Crypto Module henceforth denoted by CDI-Module. It is planned to make these modules easily exchangeable. For our present representation, the module would have 64 input and 64 output wires for the routing of the digits being processed by the device, along with 16 plus 64 input terminals to be connected with appropriate positions of the key registers.

The confuser is made up of 16 boxes S_0 and 16 boxes S_1, forming S_0 - S_1 pairs. (In our illustration there are only six S_0 - S_1 pairs.) These are "simple"

substitution devices, depicted in the next diagram (Fig. 7), with 4 or more inputs and outputs. The internal permutations selected must be different for the two S pairs. In fact, all S-permutations might be chosen at random, and could be acceptable as long as the just stated rule is not violated. However, the choice of these permutations is also subject to various other mathematical considerations. In any case, once decided upon, they would become an integral part of the CDI-module. The system per se would thus in principle be accessible and known to anyone. The S_0 - S_1 pairs are operated as a unit and have (for 4-digit S-boxes) four common input wires as illustrated in the next figure. Since each S_0 - S_1 pair will operate on a common four digit input, being derived by reading four register units of the bottom row, we may regard the 2-bit registers as being grouped into four tuplets (see Figs. 8 and 9).

Since the S_0 - S_1 pairs have together 8 outputs, while we need only 4 of these for addition mod 2 into the respective register units of the top row, we will now introduce a selection process, which chooses one S-output or the other, according to a keyed instruction. This is accomplished through the gating circuits G and $\overline{G}$, which are complementary in their action, i. e. if the substitution control signal from the respective key element is zero, G would open and $\overline{G}$ close; if the substitution control signal is a one, the opposite would be true (Fig. 10). Thus depending on the substitution control bit on the 16 substitution control wires connected to their respective units in the key registers, the output of a typical S_0 - S_1 pair and its gating device would be either the four digits coming out of S_0 or S_1. For an illustration


n-INPUTS
LTH
2^n POINTS
$2^{n!}$ {POSSIBLE PERMUTATIONS}
2^n POINTS
HTL
n-OUTPUTS

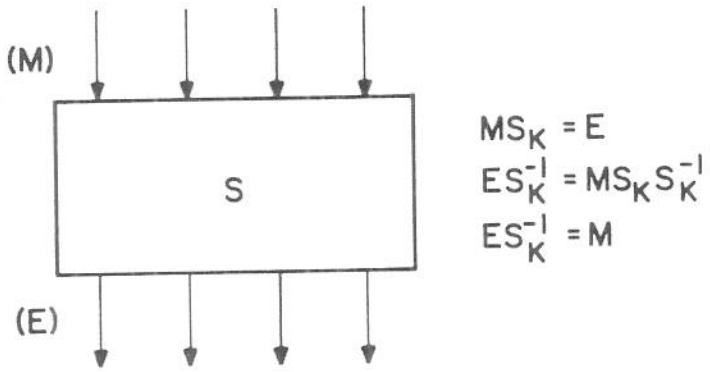

(M)
S
(E)
$MS_K = E$
$ES_K^{-1} = MS_K S_K^{-1}$
$ES_K^{-1} = M$

Figure 7

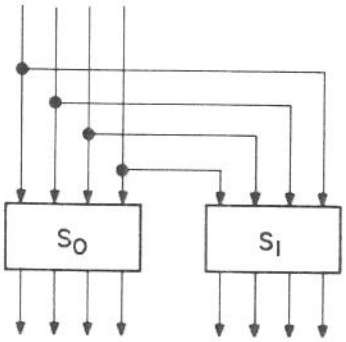


Figure 8

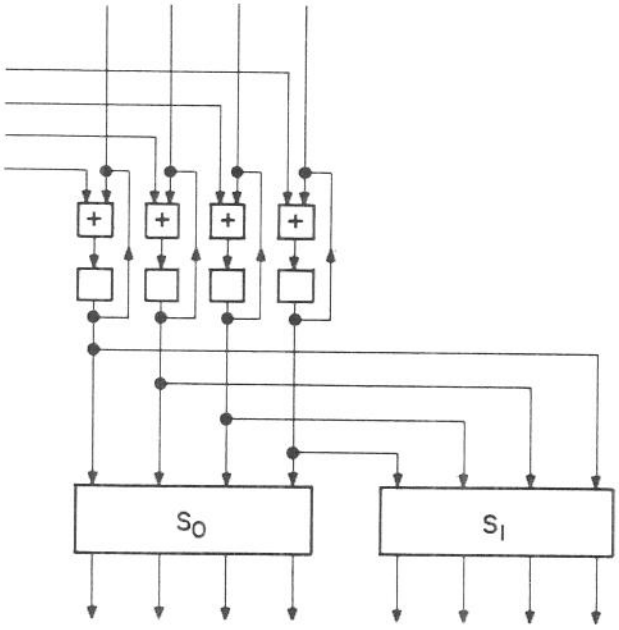


Figure 9

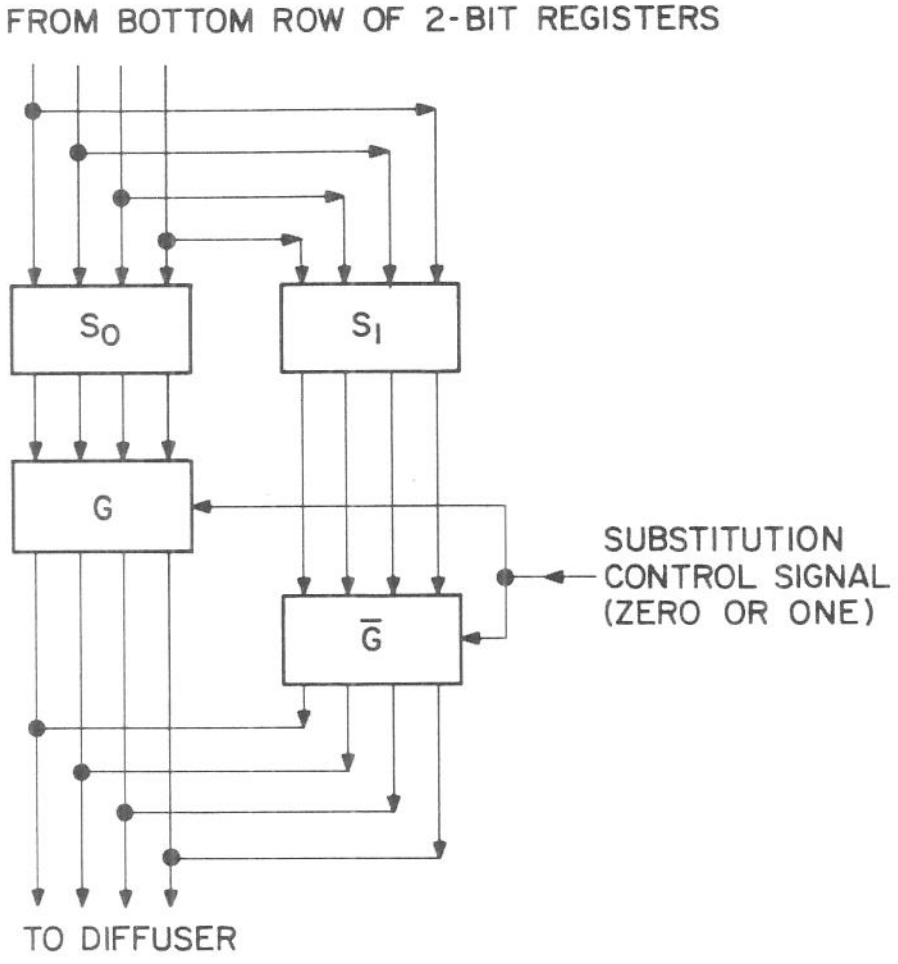


Figure 10

of this step see again Fig. 10. This S_0 - S_1 selection operation is, of course, simultaneously performed on all 16 S_0 - S_1 pairs, controlled by the respective 16 substitution control wires connecting key register positions and gating doublets G - $\overline{G}$. The result is that the S_0 - S_1 gating pattern will be irregular and will moreover be different at every round as we shall demonstrate. The purpose of S_0 - S_1 gating pattern key is to break up the regularity and consequent possibility of weaknesses which an iterative product might entail. The totality of outputs from the confuser comprises now 16 x 4 = 64 binary signals, appearing on the 64 outputs of the Confuser device in parallel. These 64 binary digits are now ready, again in parallel, to be operated on by the Diffuser device

This device, as indicated in the complete diagram, consists merely of 64 input terminals and 64 output terminals. In between these terminals we arrange a permutation, which will be specified later. It would, however, become a part of the CDI-Module and thus also be known to any outsider desiring this knowledge. We merely point out that the permutation in the diffuser device will simply rearrange the input digits to the device into a different order, i. e. it will not change the number of zeros or ones in the digit block, only their positions. In contrast, the devices S can potentially change any four digit input into any of the possible 16 outputs, depending on the internal point permutation selected for the S devices.

We are now ready to perform the Interruptor operation of the CDI-Module. The device effecting this consists merely of 64 devices for addition mod 2. The 64 inputs to the addition mod 2 devices arrive in parallel from the 64 output terminals of the diffuser device. A second set of signals is

derived in parallel from the 64 wires connected to 64 positions in the key registers. These 64 binary digits are then added Mod 2 to the contents of their respective individual units of the top row of the 64 2-bit registers.

To complete the first round of the encryption operation one more step is required. In the diagram illustrating the overall arrangement of the Data Protection system we have indicated that the 64 individual 2-bit registers can be commanded to shift as individuals. This is accomplished by the 64 wires connected to their 64 places in the key registers. As the signals (zero or one) on these 64 shift command wires are interrogated by the timing circuitry (not illustrated), the 64 registers may or may not perform their vertical shift, depending on the values which then obtain. It is also under consideration that the vertical shift operation might not be keyed at all, and that all 64 2-bit registers be shifted in unison by the fixed routine of the time circuitry. Although we stated earlier that the system utilizes three distinct cryptographic operations, this latter shift, if used, would comprise a fourth keyed operation. If utilized in keyed form, we shall refer to it as the MANGLER. With this description we have presented one complete round of the product encipherment operations. We are now ready to repeat the performance. To this end we must now turn our attention to the key registers, which we have so far only vaguely mentioned. On the right-hand side of the diagram for the data protection system we observe a bank of shift registers. For the actual device we would have sixteen 16-bit registers; the illustration shows only six eight-digit registers.

At the beginning of each crypto period, i.e. the time span during which a particular set of keys remains in use, i.e. remains unaltered, these registers are loaded with 256 randomly generated binary digits. These comprise "THE" key to this system and are the ingredients that in fact represents SECRECY! In an actual system these digits might be read into these registers via a magnetic or other convenient card reader, each individual participant of the Data Bank Community having his private key or keys as described in the section on the VAULT.

Before the second round of enciphering operations commences, all key registers will be cyclically shifted by one position, so that now the 64 plus 16 plus 64 wires conveying key instructions to the 64 two bit registers, 16 S_0 - S_1 pairs and the 64 Mod 2 adders, will present a different key pattern. This new pattern is the one which we will follow for the next CDI and Shift operating for the second round. Having terminated the second round, in the same manner but with a key pattern different from the first key pattern, we are now ready for the third round of operations. Again the sixteen key register will be cyclically shifted by one step resulting again in a different key pattern. With sixteen 16-bit registers 16 such rounds would be performed. At the end of this, the key registers would have cycled their key contents back into the original positions, ready for the next 16 rounds, of product encipherment, on a new set of 128 information bits. The number of 16 rounds has been chosen as a reasonable guess. A choice will be made later as to whether more (say 32) or fewer (say 8), or some other number will be preferred.

Two more aspects of the System must be described. Between the outputs of the key registers and the key instructions wired to the S_0 - S_1 and shift units, a permutation box called the "Key Effect Router" has been inserted. This box, just as the diffuser box, would be part of the system, and known to all. There are judicious ways of selecting this permutation, again irrelevant for this disclosure. The only solid requirement would be that the 16 connections to the S_0 - S_1 Gates are instructed each from different one of the 16 key registers.

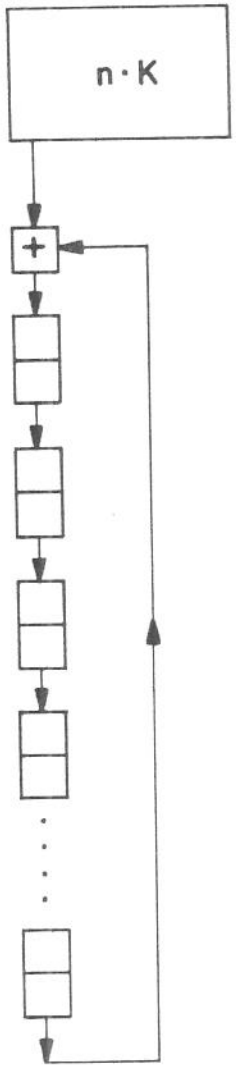
We shall in the next section describe a modification of this basic scheme, in which some of the pattern control is derived internally from some of the message bits, at various rounds of the product operations. This would constitute a form of internally variant key, assuring not only that at each product state the key pattern changes, throughput the product, but also from product to product, i.e. for different information blocks, the pattern of all product stages could potentially be varied in a nonsystematic manner, not accessible to those who do not possess the basic key.

There remains to be described one more operationally important feature of the Crypto System. It concerns the initial keying process. It is clear, that in any system which for its integrity depends on the secrecy of a key, it is desirable to provide some protection against possible unfavorable accidents, typical for humans, such as for instance the losing of keys! Moreover, even if a key is not lost, it could be stolen by brute or subtle force, or even voluntarily given away, due to mental breakdown or some other instability

in the person entrusted with such vital information. To prevent such a calamity, we propose to compartment the keying information as follows: (See Fig.11)

Here the Key reader Box is marked n. K. This symbolism is to indicate that "n" (i. e as many as desired) individual keys may be read into the key registers. This is done as follows: the first person entrusted with a key to this system inserts his key card into the reader and has it there recorded step by step by the shifting operation at the end of which the sixteen 16-bit groups recorded on his key card are now electronically stored in the key registers. We are now ready for another trustee, with a different key to insert his key card into the reader. The key reader device will now read his key bits, bit by bit into the key shift registers in such a manner that the second set of key bits is added bit by bit mod 2 to the old key bits as the new key is shifted into the register and the previous one simultaneously cycled through the box. At the end of the operation we have the sum Mod 2 of two keys in each individual register. (We illustrate the process for only one of the registers.) As many different trustees as desired can be used. All except one can defect, and the system will still have an inviolate key, as if no one had defected! The order of recording is, of course, irrelevant. We must consider, however, that all trustees must be available for insertion of their personal key to operate the box. An appropriate trade-off must be worked out by the practical user.

We should mention that the keys could, of course, also be read in by a typing device, say from memory, in case possibility of loss of key card



SUPERPOSITION OF KEYS

Figure 11

seems undesirable. The electronically stored key would vanish automatically, in case of attempts to physically attack the box. We should also mention, that even in a software system it would be desirable for these reasons to have a physically separate key-device. Finally, if "enough" keys are added Mod 2 in the above manner, even memorized English (or other language) words might become permissible.

In Fig.12 we present a modification of our basic scheme. It will be noted that the modification consists of a different use of S_0 - S_1 pairs. As before the gating pattern key operates on individual S_0 - S_1 pairs. But in contrast to the first scheme, in which the key routes the respective four outputs from the bottom row of the message register either through S_1 or S_0, the key bits operate now on neighboring sets of four digits. A zero key bit might thus route the first four digits coming from the left most four digits straight down through S_0 and the next four digits straight through S_1, while a one key bit might switch the first four digits over to S_1 and the second set over to S_0, causing both quadruplets to cross each other.

This method of breaking up the regularity of the product iteration effects a certain economy in the uses of S switches, since for a given message dimension it requires only half the number of these devices as compared to the basic definition. As a result only half as many pairs S_0 - S_1 have to be controlled, which in turn implies that care must be taken lest the gating pattern key becomes too small per round. To utilize 16 bits of key per round, as in the basic system, this economy version would require a message dimension of 256 digits.

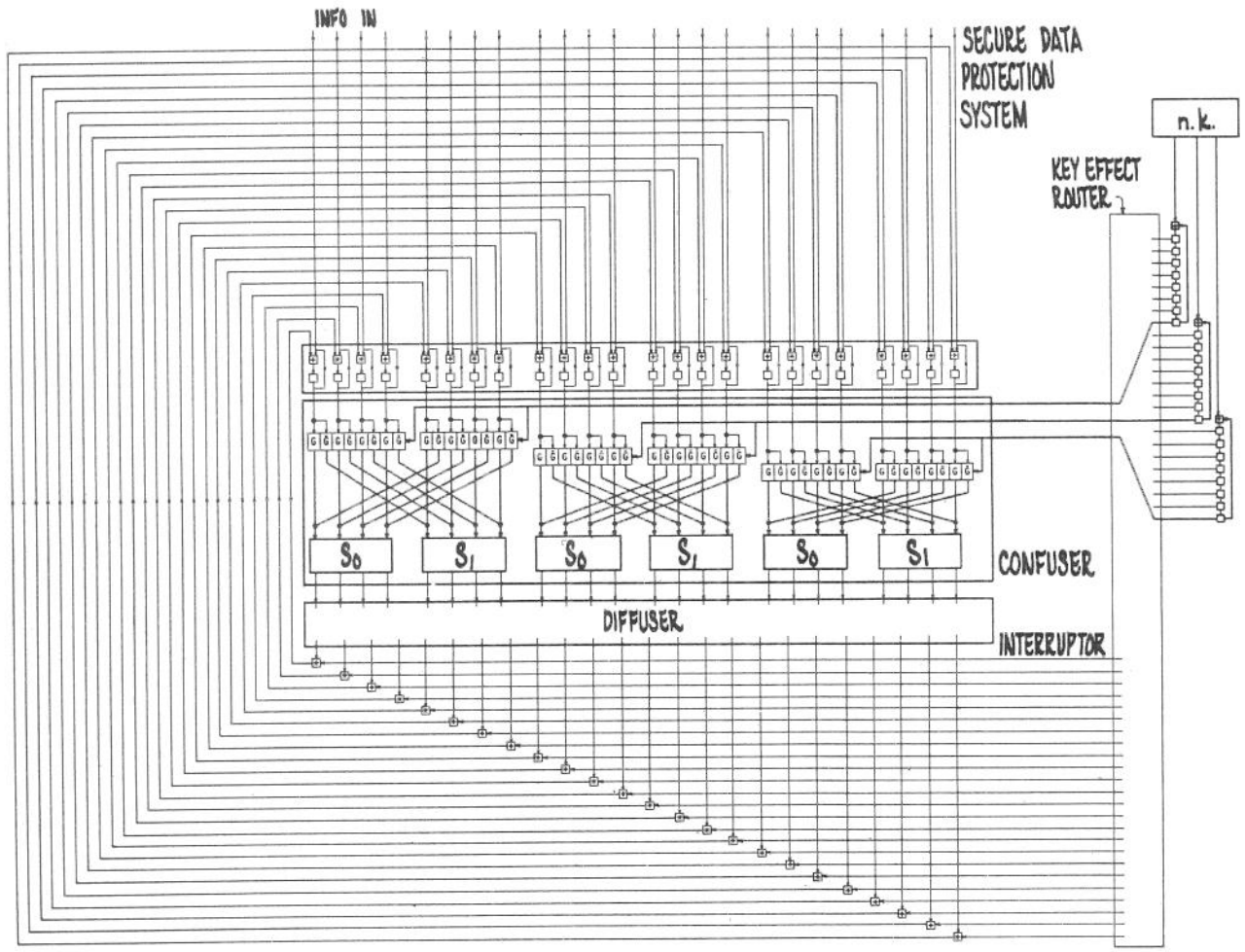


Figure 12

6. INTERACTION OF CRYPTOGRAPHIC DATA PROCESSING AND COMPUTATIONAL PROCESSING WITHIN VAULT

6.1 Introduction

Within the physical security of the Vault, three distinct types of data processing take place:

1) Purely cryptographic operations - i.e., encryption and decryption per se.

2) Data processing in which ordinary data processing and cryptographic operations interact (e.g. "verification"), and cryptographically protected filing operations.

3) Purely computational data processing which is essentially outside the province of the cryptographic designer. Relevant problems in this area exists in any computing center or data bank independent of the use of cryptographic techniques.

In this section we are dealing with operations of type 2.

Of great concern seems to be the fact that in a data bank using cryptographic techniques, along with a time-sharing computer, many different keys and the respective private information which they protected before decryption, appear now in the clear. It is hence not impossible that data, after processing, i.e., in their new clear text representation, might accidentally be re-enciphered in someone else's key and transmitted to that terminal, availing the wrong subscriber to someone else's confidences.

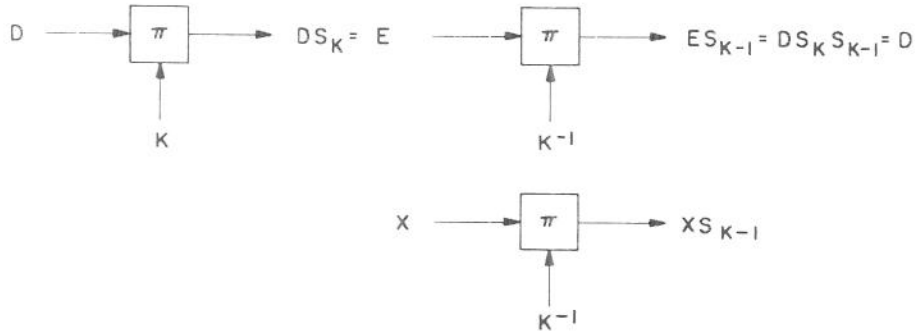
To prevent this, we here describe an automatic coding procedure, which would make this very unlikely to occur. It is called secured data processing. We should like to differentiate between secure data encryption to protect the

data while they are conveyed through a hostile environment (the communications channel) and the use of cryptographic techniques in a basically friendly environment (e.g., the vault) where cryptographically protected data processing interacts with ordinary coding processes to provide very efficient checks against both accidental and intended data corruptions! Verification has already been discussed. The present topic describes a method of properly decrypting (i.e., not just deciphering!!) encrypted data, so that they can be "securely" handled within the vault.

In Fig.13 we merely indicate on top of the page that any data D when encrypted with our System $S_{(K)}$ and the key K will be transformed into a cryptogram $(D)\,S_K$. Sending it through the box keyed with K^{-1}, will, of course, be deciphered back into the original data D.

In the second line of the top diagram we further observe that if in turn, we send an arbitrary digit block X through the deciphering box, it will now result in an encipherment of X! We shall use this observation to provide a secured checking procedure. For this purpose we bracket cryptograms $(D_3)\,S_A \oplus (D_2)\,S_A \oplus (D_1)\,S_A$ representing a data train which has just arrived at the vault for processing, at beginning and end with some address marker A belonging to this subscriber. Sending the encrypted data belonging to A through the cryptobox with inverse key K_A^{-1}, will result in the deciphered set of data $D_3 \oplus D_2 \oplus D_1$ bracketed by the enciphered address! Any human operator having access to these data, would, hence, not know to whom they belong. We assume that the brackets A were memorized before decryption

Secured Data Processing



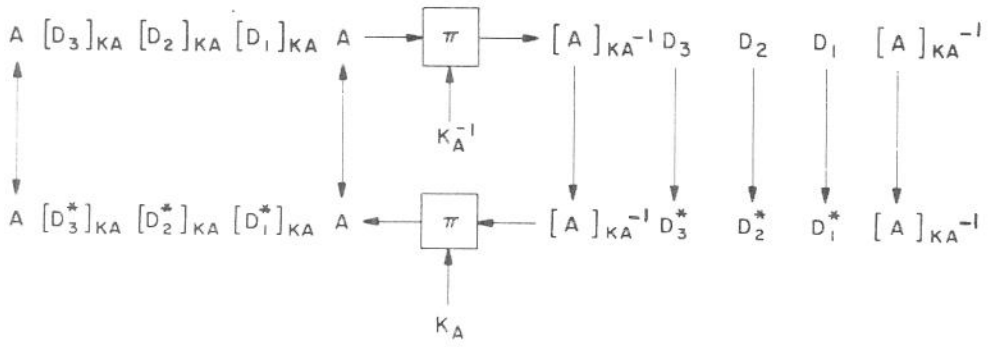


Figure 13

in a suitable device. In the meantime, data $D_1 \oplus D_2 \oplus D_3$ are processed yielding say $D_1^* \oplus D_2^* \oplus D_3^*$. These are now ready for re-encryption and subsequent filing or transmission to their owner.

For this purpose, we send them back through the cryptobox now keyed for encryption. The new data train is thus transformed into cryptograms, $(D_3^*)\, S_A \oplus (D_2^*)\, S_A \oplus (D_1^*)\, S_A$, with the brackets now reappearing into the clear! If the proper key, belonging to the rightful owner of these data was used, "A memorized" and "A just deciphered" will match and we can proceed to file or transmit the processed and re-encrypted data with a high assurance that they will not end up at the wrong address!

6.2 Multiple Level Encipherment for Secure Filing

Of special importance to a Data Bank is the problem of secure filing. In view of the great variety and individuality of the information likely to be present in a large data bank, we must observe that the inherent difficulties presented by such a data classification task are not the fault of any encipherment operation but rather an administrative problem typical of filing operations per se.

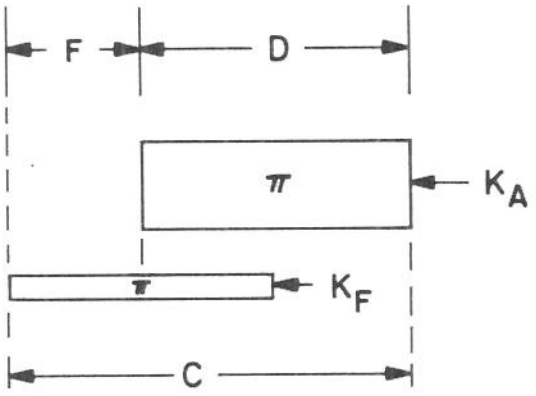
Cryptographic coding could, however, introduce a significant burden onto a filing system, if it were basically cumbersome, for instance if it were too time consuming to perform decipherment operations to check the content of the files for presence of any particular topic desired for special data processing.

Two procedures are here envisaged to make secure and rapid filing possible.

1) The use of indicator file tags,

2) The use of our block cipher in a special rapid mode using only two rounds in the cipher product.

We observe that the degree of confidence attached to the various types of information to be found in a data bank file will vary enormously and will have to be adjusted to a large extent to what is agreeable to the customer. We assume that all data to be stored in the file will receive a preliminary organization according to the level of confidence and privacy they require. The lowest level would presumably require no encipherment at all. All data would receive a file tag F, attached to the data. This file tag, possibly consisting of digital indicators, would simple denote whether a particular file topic is present in the cryptogram it is attached to. Thus, the first digit position in F might indicate whether or not the cryptogram contains tax information, the next digit medical information, etc. In general the indicator tag will not require the same level of security as the data cryptogram itself, since mere knowledge of the general nature of the information contained in the data cryptogram is not as sensitive as the detailed data themselves. However, in some cases even the file tag will require encryption. For these purposes we have devised the file mode of encryption. In this mode we use our crypto device not programmed to operate with the usual 8 to 16 rounds but rather in a special mode of only two rounds. This is symbolized in Fig. 14 by drawing the box for the filing mode (with filing key K_F) much thinner than the box operating in the normal mode.



MULTIPLE LEVEL ENCIPHERMENT
WITH RAPID ACCESS FILE-MODE

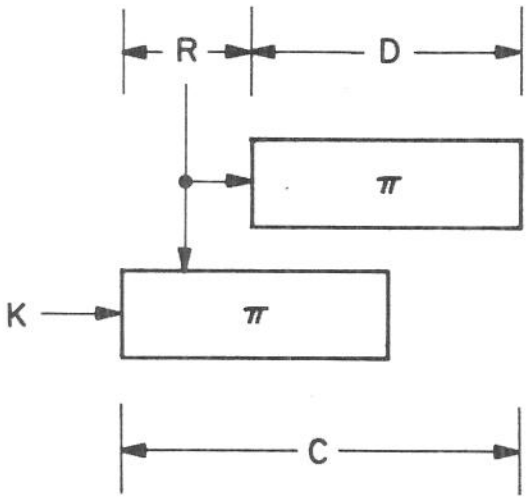
Figure 14

The rapid filing mode is not only much faster than the normal mode but indeed also not much slower than the reading of the file tag without any encryption. The basic justification making this special file mode acceptable is due to the fact that with our data encryption system already after only two rounds of the product operation every digit potentially becomes a very involved function of all the input digits Moreover, since the file tag never appears outside the vault, it is not exposed to the dangers that a cryptogram on a communications line may be subjected to. For the file cryptogram we can hence afford to take a somewhat more relaxed attitude with regard to encipherment than for the data cryptogram.

6.3 Substitutions with Variants

Although computer communications are frequently almost completely non-redundant, there are some commands that are highly stereotyped nevertheless. Lest this cause concern we introduce the notion of a substitution with variants. In its simplest form it consists merely of a cipher block in which the stereotyped data digits are followed by a group of random digits. Thus, if the data block is small, say five digits, and fixed, it would be followed by a sequence of 123 randomly selected digits. The encipherment of even a fixed message would thus look different every time.

In Fig.15 we illustrate a very effective method of producing variants, using the step code principle. Random digits R are here used to key the product system for the primary encipherment, which is then followed by the



SUBSTITUTION WITH VARIANT KEY

Figure 15

second step, enciphered with the fixed key K. The resulting cipher C will thus look different every time, even if D remains fixed. The system might be called a variant key cipher.

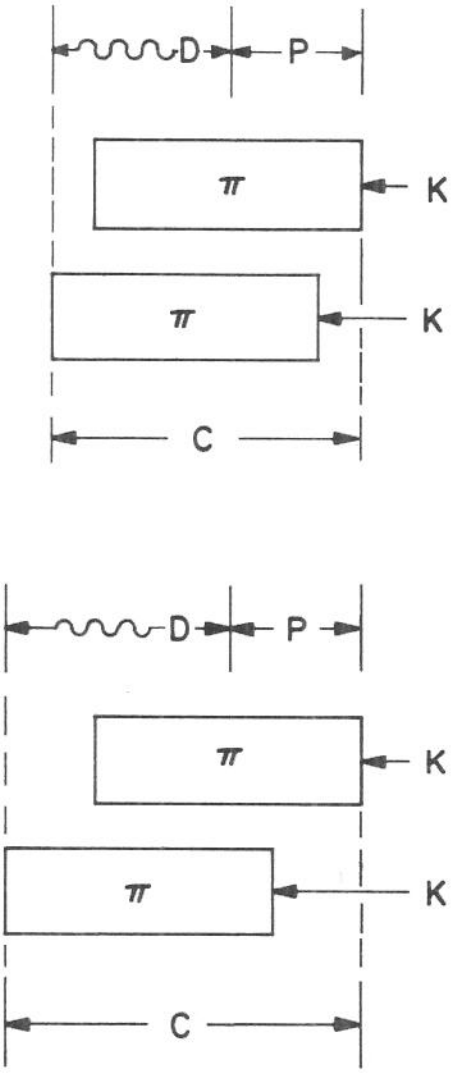
6.4 Harmonica Codes

Another useful application of the stepping principle are so-called Harmonica Codes, in these, as indicated in Fig. 16, the two steps of the code could be shifted with respect to each other by varying amounts, making it thus possible to encrypt data blocks of varying dimension, if so desired.

6.5 Key-List Generator*

We conclude this proposal with a suggestion for generating keys for the various subscribers to the Data Bank System Instead of storing the presumably sizeable key lists in the vault locations, keys would simply be generated by inserting the address code of the subscriber (say his name, or similar information) in the clear into the cryptographic box, keyed with the Corporate Key known to no one but the Company In this way the individual key could be generated and regenerated at will, with a minimum of storage burden to the CPU.

*Suggestion due to C. D. Cullum



HARMONICA CODES

Figure 16

7. ACKNOWLEDGEMENTS

The author wishes to thank the many colleagues at the Thomas J. Watson Research Center for their constant encouragement and inspiration. In particular, he is indebted to Bryant Tuckerman for many months of critical discussions and fruitful suggestions and J. L. Smith for his analyses on many practical aspects.