APPENDIX A

FIPS Publication 46

FIPS PUB 46

FEDERAL INFORMATION
PROCESSING STANDARDS PUBLICATION
1977 JANUARY 15

U.S. DEPARTMENT OF COMMERCE / National Bureau of Standards

DATA ENCRYPTION STANDARD

CATEGORY: ADP OPERATIONS
SUBCATEGORY: COMPUTER SECURITY

U.S. DEPARTMENT OF COMMERCE • Elliot L. Richardson, *Secretary*

Edward O. Vetter, *Under Secretary*

Dr. Betsy Ancker-Johnson, *Assistant Secretary for Science and Technology*

NATIONAL BUREAU OF STANDARDS • Ernest Ambler, *Acting Director*

Foreword

The Federal Information Processing Standards Publication Series of the National Bureau of Standards is the official publication relating to standards adopted and promulgated under the provisions of Public Law 89-306 (Brooks Bill) and under Part 6 of Title 15, Code of Federal Regulations. These legislative and executive mandates have given the Secretary of Commerce important responsibilities for improving the utilization and management of computers and automatic data processing systems in the Federal Government. To carry out the Secretary's responsibilities, the NBS, through its Institute for Computer Sciences and Technology, provides leadership, technical guidance, and coordination of government efforts in the development of technical guidelines and standards in these areas.

The series is used to announce Federal Information Processing Standards, and to provide standards information of general interest and an index of relevant standards publications and specifications. Publications that announce adoption of standards provide the necessary policy, administrative, and guidance information for effective standards implementation and use. The technical specifications of the standard are usually attached to the publication, otherwise a reference source is cited.

Comments covering Federal Information Processing Standards and Publications are welcomed, and should be addressed to the Associate Director for ADP Standards, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D.C. 20234. Such comments will be either considered by NBS or forwarded to the responsible activity as appropriate.

ERNEST AMBLER, *Acting Director*

Abstract

The selective application of technological and related procedural safeguards is an important responsibility of every Federal organization in providing adequate security to its ADP systems. This publication provides a standard to be used by Federal organizations when these organizations specify that cryptographic protection is to be used for sensitive or valuable computer data. Protection of computer data during transmission between electronic components or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by that data. The standard specifies an encryption algorithm which is to be implemented in an electronic device for use in Federal ADP systems and networks. The algorithm uniquely defines the mathematical steps required to transform computer data into a cryptographic cipher. It also specifies the steps required to transform the cipher back to its original form. A device performing this algorithm may be used in many applications areas where cryptographic data protection is needed. Within the context of a total security program comprising physical security procedures, good information management practices and computer system/network access controls, the Data Encryption Standard is being made available for use by Federal agencies.

Key Words: ADP security; computer security; encryption; Federal Information Processing Standard.

Nat. Bur. Stand. (U.S.), Fed. Info. Process. Stand. Publ. (FIPS PUB) 46, 17 pages (1977)
CODEN: FIPPAT

For sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161

Federal Information Processing Standards Publication 46

1977 January 15

ANNOUNCING THE

DATA ENCRYPTION STANDARD

•Federal Information Processing Standards are issued by the National Bureau of Standards pursuant to the Federal Property and Administrative Services Act of 1949, as amended, Public Law 89-306 (79 Stat 1127), Executive Order 11717 (38 FR 12315, dated May 11, 1973), and Part 6 of Title 15 Code of Federal Regulations (CFR).

Name of Standard: Data Encryption Standard (DES).

Category of Standard: Operations, Computer Security.

Explanation: The Data Encryption Standard (DES) specifies an algorithm to be implemented in electronic hardware devices and used for the cryptographic protection of computer data. This publication provides a complete description of a mathematical algorithm for encrypting (enciphering) and decrypting (deciphering) binary coded information. Encrypting data converts it to an unintelligible form called cipher. Decrypting cipher converts the data back to its original form. The algorithm described in this standard specifies both enciphering and deciphering operations which are based on a binary number called a key. The key consists of 64 binary digits ("0"s or "1"s) of which 56 bits are used directly by the algorithm and 8 bits are used for error detection.

Binary coded data may be cryptographically protected using the DES algorithm in conjunction with a key. The key is generated in such a way that each of the 56 bits used directly by the algorithm are random and the 8 error detecting bits are set to make the parity of each 8-bit byte of the key odd, i.e., there is an odd number of "1"s in each 8-bit byte. Each member of a group of authorized users of encrypted computer data must have the key that was used to encipher the data in order to use it. This key, held by each member in common, is used to decipher the data received in cipher form from other members of the group. The encryption algorithm specified in this standard is commonly known among those using the standard. The unique key chosen for use in a particular application makes the results of encrypting data using the algorithm unique. Selection of a different key causes the cipher that is produced for any given set of inputs to be different. The cryptographic security of the data depends on the security provided for the key used to encipher and decipher the data.

Data can be recovered from cipher only by using exactly the same key used to encipher it. Unauthorized recipients of the cipher who know the algorithm but do not have the correct key cannot derive the original data algorithmically. However, anyone who does have the key and the algorithm can easily decipher the cipher and obtain the original data. A standard algorithm based on a secure key thus provides a basis for exchanging encrypted computer data by issuing the key used to encipher it to those authorized to have the data. Additional FIPS guidelines for implementing and using the DES are being developed and will be published by NBS.

Approving Authority: Secretary of Commerce.

Maintenance Agency: Institute for Computer Sciences and Technology, National Bureau of Standards.

Applicability: This standard will be used by Federal departments and agencies for the cryptographic protection of computer data when the following conditions apply:

1. An authorized official or manager responsible for data security or the security of any computer system decides that cryptographic protection is required; and

2. The data is not classified according to the National Security Act of 1947, as amended, or the Atomic Energy Act of 1954, as amended.

However, Federal agencies or departments which use cryptographic devices for protecting data classified according to either of these acts can use those devices for protecting unclassified data in lieu of the standard.

In addition, this standard may be adopted and used by non-Federal Government organizations. Such use is encouraged when it provides the desired security for commercial and private organizations.

Data that is considered sensitive by the responsible authority, data that has a high value, or data that represents a high value should be cryptographically protected if it is vulnerable to unauthorized disclosure or undetected modification during transmission or while in storage. A risk analysis should be performed under the direction of a responsible authority to determine potential threats. FIPS PUB 31 (Guidelines for Automatic Data Processing Physical Security and Risk Management) and FIPS PUB 41 (Computer Security Guidelines for Implementing the Privacy Act of 1974) provide guidance for making such an analysis. The costs of providing cryptographic protection using this standard as well as alternative methods of providing this protection and their respective costs should be projected. A responsible authority then should make a decision, based on these analyses, whether or not to use cryptographic protection and this standard.

Applications: Data encryption (cryptography) may be utilized in various applications and in various environments. The specific utilization of encryption and the implementation of the DES will be based on many factors particular to the computer system and its associated components. In general, cryptography is used to protect data while it is being communicated between two points or while it is stored in a medium vulnerable to physical theft. Communication security provides protection to data by enciphering it at the transmitting point and deciphering it at the receiving point. File security provides protection to data by enciphering it when it is recorded on a storage medium and deciphering it when it is read back from the storage medium. In the first case, the key must be available at the transmitter and receiver simultaneously during communication. In the second case, the key must be maintained and accessible for the duration of the storage period.

Hardware Implementation: The algorithm specified in this standard is to be implemented in computer or related data communication devices using hardware (not software) technology. The specific implementation may depend on several factors such as the application, the environment, the technology used, etc. Implementations which comply with this standard include Large Scale Integration (LSI) "chips" in individual electronic packages, devices built from Medium Scale Integration (MSI) electronic components, or other electronic devices dedicated to performing the operations of the algorithm. Micro-processors using Read Only Memory (ROM) or micro-programmed devices using microcode for hardware level control instructions are examples of the latter. Hardware implementations of the algorithm which are tested and validated by NBS will be considered as complying with the standard. Procedures for testing and validating equipment for conformance with this standard are available from the Systems and Software Division, National Bureau of Standards, Washington, D.C. 20234. Software implementations in general purpose computers are not in compliance with this standard. Information regarding devices which have been tested and validated will be made available to all FIPS points of contact.

Export Control: Cryptographic devices and technical data regarding them are subject to Federal Government export controls as specified in Title 22, Code of Federal Regulations, Parts 121 through 128. Cryptographic devices implementing this standard and technical data regarding them must comply with these Federal regulations.

Patents: Crytographic devices implementing this standard may be covered by U.S. and foreign patents issued to the International Business Machines Corporation. However, IBM has granted nonexclusive, royalty-free licenses under the patents to make, use and sell apparatus which complies with the standard. The terms, conditions and scope of the licenses are set out in notices published in the May 13, 1975 and August 31, 1976 issues of the Official Gazette of the United States Patent and Trademark Office (934 O. G. 452 and 949 O. G. 1717).

Alternative Modes of Using the DES: The "Guidelines for Implementing and Using the Data Encryption Standard" describe two different modes for using the algorithm described in this standard. Blocks of data containing 64 bits may be directly entered into the device where 64-bit cipher blocks are generated under control of the key. This is called the electronic code book mode. Alternatively, the device may be used as a binary stream generator to produce statistically random binary bits which are then combined with the clear (unencrypted) data (1-64 bits) using an "exclusive-or" logic operation. In order to assure that the enciphering device and the deciphering device are synchronized, their inputs are always set to the previous 64 bits of cipher that were transmitted or received. This second mode of using the encryption algorithm is called the cipher feedback (CFB) mode. The electronic codebook mode generates blocks of 64 cipher bits. The cipher feedback mode generates cipher having the same number of bits as the plain text. Each block of cipher is independent of all others when the electronic codebook mode is used while each byte (group of bits) of cipher depends on the previous 64 cipher bits when the cipher feedback mode is used. The modes of operation briefly described here are further explained in the FIPS "Guidelines for Implementing and Using the Data Encryption Standard."

Implementation of this standard: This standard becomes effective six months after the publication date of this FIPS PUB. It applies to all Federal ADP systems and associated telecommunications networks under development as well as to installed systems when it is determined that cryptographic protection is required. Each Federal department or agency will issue internal directives for the use of this standard by their operating units based on their data security requirement determinations.

NBS will provide assistance to Federal organizations by developing and issuing additional technical guidelines on computer security and by providing technical assistance in using data encryption. A data encryption testbed has been established within NBS for use in providing this technical assistance. The National Security Agency assists Federal departments and agencies in communications security and in determining specific security requirements. Instructions and regulations for procuring data processing equipment utilizing this standard will be provided by the General Services Administration.

Specifications: Federal Information Processing Standard (FIPS 46) Data Encryption Standard (DES) (affixed).

Cross Index:

a. FIPS PUB 31, "Guidelines to ADP Physical Security and Risk Management"

b. FIPS PUB 39, "Glossary for Computer Systems Security"

c. FIPS PUB 41, "Computer Security Guidelines for Implementing the Privacy Act of 1974"

d. FIPS PUB—, "Guidelines for Implementing and Using the Data Encryption Standard" (to be published)

e. Other FIPS and Federal Standards are applicable to the implementation and use of this standard. In particular, the American Standard Code for Information Interchange (FIPS PUB 1)

and other related data storage media or data communications standards should be used in conjunction with this standard. A list of currently approved FIPS may be obtained from the Office of ADP Standards Management, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D.C. 20234.

Qualifications: The cryptographic algorithm specified in this standard transforms a 64-bit binary value into a unique 64-bit binary value based on a 56-bit variable. If the complete 64-bit input is used (i.e., none of the input bits should be predetermined from block to block) and if the 56-bit variable is randomly chosen, no technique other than trying all possible keys using known input and output for the DES will guarantee finding the chosen key. As there are over 70,000,000,000,000,000 (seventy quadrillion) possible keys of 56 bits, the feasibility of deriving a particular key in this way is extremely unlikely in typical threat environments. Moreover, if the key is changed frequently, the risk of this event is greatly diminished. However, users should be aware that it is theoretically possible to derive the key in fewer trials (with a correspondingly lower probability of success depending on the number of keys tried) and should be cautioned to change the key as often as practical. Users must change the key and provide it a high level of protection in order to minimize the potential risks of its unauthorized computation or acquisition. The feasibility of computing the correct key may change with advances in technology. A more complete description of the strength of this algorithm against various threats will be contained in the Guidelines for Implementing and Using the DES.

When correctly implemented and properly used, this standard will provide a high level of cryptographic protection to computer data. NBS, supported by the technical assistance of Government agencies responsible for communication security, has determined that the algorithm specified in this standard will provide a high level of protection for a time period beyond the normal life cycle of its associated ADP equipment. The protection provided by this algorithm against potential new threats will be reviewed within five years to assess its adequacy. In addition, both the standard and possible threats reducing the security provided through the use of this standard will undergo continual review by NBS and other cognizant Federal organizations. The new technology available at that time will be evaluated to determine its impact on the standard. In addition, the awareness of any breakthrough in technology or any mathematical weakness of the algorithm will cause NBS to reevaluate this standard and provide necessary revisions.

Comments: Comments and suggestions regarding this standard and its use are welcomed and should be addressed to the Associate Director for ADP Standards, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D.C. 20234.

Waiver Procedure: The head of a Federal agency may waive the provisions of this FIPS PUB after the conditions and justifications for the waiver have been coordinated with the National Bureau of Standards. A waiver is necessary if cryptographic devices performing an algorithm other than that which is specified in this standard are to be used by a Federal agency for data subject to cryptographic protection under this standard. No waiver is necessary if classified communications security equipment is to be used. Software implementations of this algorithm for operational use in general purpose computer systems do not comply with this standard and each such implementation must also receive a waiver. Implementation of the algorithm in software for testing or evaluation does not require waiver approval. Implementation of other special purpose cryptographic algorithms in software for limited use within a computer system (e.g., encrypting password files) or implementations of cryptographic algorithms in software which were being utilized in computer systems before the effective date of this standard do not require a waiver. However, these limited uses should be converted to the use of this standard when the system or equipment involved is upgraded or redesigned to include general cryptographic protection of computer data. Letters describing the nature of and reasons for the waiver should be addressed to the Associate Director for ADP Standards as previously noted.

Sixty days should be allowed for review and response by NBS. The waiver shall not be approved until a response from NBS is received; however, the final decision for granting the waiver is the responsibility of the head of the particular agency involved.

Where to Obtain Copies of the Standard:

Copies of this publication are for sale by the National Technical Information Service, U. S. Department of Commerce, 5285 Port Royal Road, Springfield, Virginia 22161. Order by FIPS PUB number and title. Prices are published by NTIS in current catalogs and other issuances. Payment may be made by check, money order, deposit account or charged to a credit card accepted by NTIS.

Federal Information Processing Standards Publication 46

1977 January 15

SPECIFICATIONS FOR THE

DATA ENCRYPTION STANDARD

The Data Encryption Standard (DES) shall consist of the following Data Encryption Algorithm to be implemented in special purpose electronic devices. These devices shall be designed in such a way that they may be used in a computer system or network to provide cryptographic protection to binary coded data. The method of implementation will depend on the application and environment. The devices shall be implemented in such a way that they may be tested and validated as accurately performing the transformations specified in the following algorithm.

DATA ENCRYPTION ALGORITHM

Introduction

The algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64-bit key. Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be enciphered is subjected to an initial permutation IP, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation IP^{-1}. The key-dependent computation can be simply defined in terms of a function f, called the cipher function, and a function KS, called the key schedule. A description of the computation is given first, along with details as to how the algorithm is used for encipherment. Next, the use of the algorithm for decipherment is described. Finally, a definition of the cipher function f is given in terms of primitive functions which are called the selection functions S_i and the permutation function P. S_i, P and KS of the algorithm are contained in the Appendix.

The following notation is convenient: Given two blocks L and R of bits, LR denotes the block consisting of the bits of L followed by the bits of R. Since concatenation is associative $B_1B_2 \ldots B_8$, for example, denotes the block consisting of the bits of B_1 followed by the bits of $B_2 \ldots$ followed by the bits of B_8.

Enciphering

A sketch of the enciphering computation is given in figure 1.

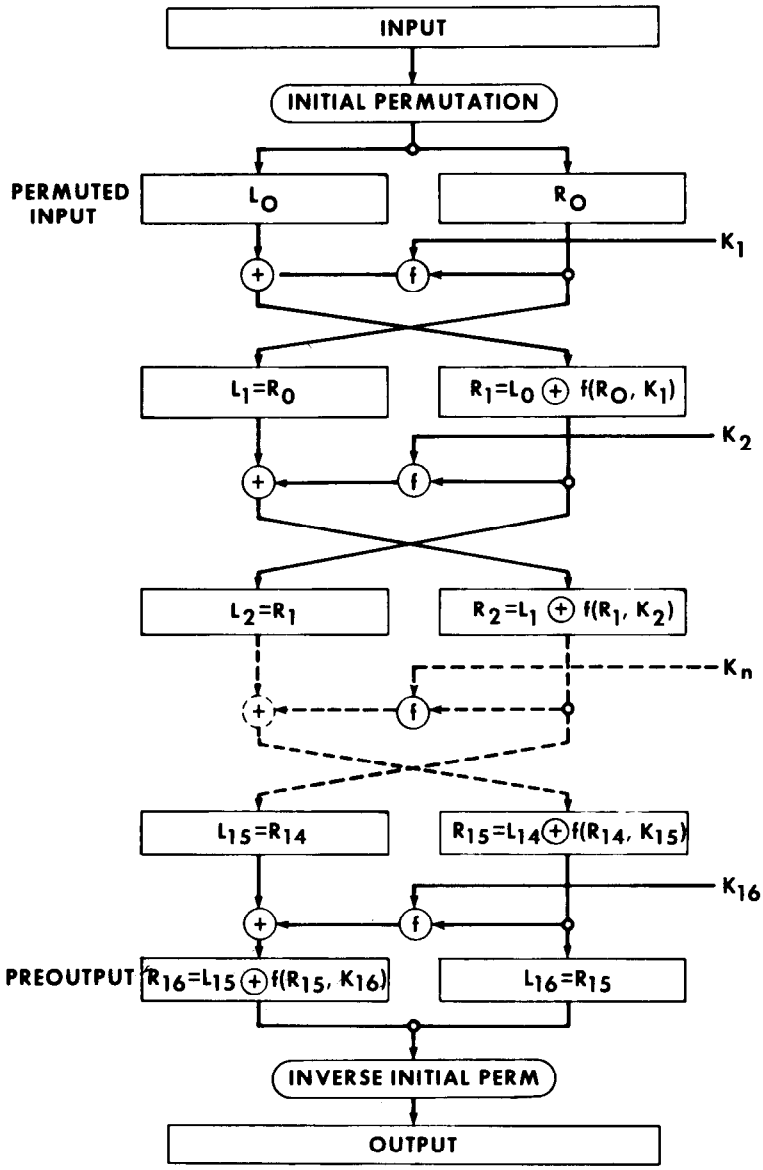


FIGURE 1. *Enciphering computation.*

The 64 bits of the input block to be enciphered are first subjected to the following permutation, called the initial permutation *IP*:

IP

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

That is the permuted input has bit 58 of the input as its first bit, bit 50 as its second bit, and so on with bit 7 as its last bit. The permuted input block is then the input to a complex key-dependent computation described below. The output of that computation, called the preoutput, is then subjected to the following permutation which is the inverse of the initial permutation:

IP^{-1}

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

That is, the output of the algorithm has bit 40 of the preoutput block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preoutput block is the last bit of the output.

The computation which uses the permuted input block as its input to produce the preoutput block consists, but for a final interchange of blocks, of 16 iterations of a calculation that is described below in terms of the cipher function f which operates on two blocks, one of 32 bits and one of 48 bits, and produces a block of 32 bits.

Let the 64 bits of the input block to an iteration consist of a 32 bit block L followed by a 32 bit block R. Using the notation defined in the introduction, the input block is then LR.

Let K be a block of 48 bits chosen from the 64-bit key. Then the output $L'R'$ of an iteration with input LR is defined by:

(1)
$$L' = R$$
$$R' = L \oplus f(R,K)$$

where $\oplus$ denotes bit-by-bit addition modulo 2.

As remarked before, the input of the first iteration of the calculation is the permuted input block. If $L'R'$ is the output of the 16th iteration then $R'L'$ is the preoutput block. At each iteration a different block K of key bits is chosen from the 64-bit key designated by KEY.

With more notation we can describe the iterations of the computation in more detail. Let KS be a function which takes an integer n in the range from 1 to 16 and a 64-bit block KEY as input and yields as output a 48-bit block K_n which is a permuted selection of bits from KEY. That is

$$K_n = KS(n, KEY) \tag{2}$$

with K_n determined by the bits in 48 distinct bit positions of KEY. KS is called the key schedule because the block K used in the n'th iteration of (1) is the block K_n determined by (2).

As before, let the permuted input block be LR. Finally, let L_0 and R_0 be respectively L and R and let L_n and R_n be respectively L' and R' of (1) when L and R are respectively L_{n-1} and R_{n-1}, and K is K_n; that is, when n is in the range from 1 to 16,

$$\begin{aligned} L_n &= R_{n-1} \\ R_n &= L_{n-1} \oplus f(R_{n-1}, K_n) \end{aligned} \tag{3}$$

The preoutput block is then $R_{16}L_{16}$.

The key schedule KS of the algorithm is described in detail in the Appendix. The key schedule produces the 16 K_n which are required for the algorithm.

Deciphering

The permutation IP^{-1} applied to the preoutput block is the inverse of the initial permutation IP applied to the input. Further, from (1) it follows that:

$$\begin{aligned} R &= L' \\ L &= R' \oplus f(L', K) \end{aligned} \tag{4}$$

Consequently, to ***decipher*** it is only necessary to apply the ***very same algorithm to an enciphered message block,*** taking care that at each iteration of the computation ***the same block of key bits*** K ***is used*** during decipherment as was used during the encipherment of the block. Using the notation of the previous section, this can be expressed by the equations:

$$\begin{aligned} R_{n-1} &= L_n \\ L_{n-1} &= R_n \oplus f(L_n, K_n) \end{aligned} \tag{5}$$

where now $R_{16}L_{16}$ is the permuted input block for the deciphering calculation and L_0R_0 is the preoutput block. That is, for the decipherment calculation with $R_{16}L_{16}$ as the permuted input, K_{16} is used in the first iteration, K_{15} in the second, and so on, with K_1 used in the 16th iteration.

The Cipher Function f

A sketch of the calculation of $f(R, K)$ is given in figure 2.

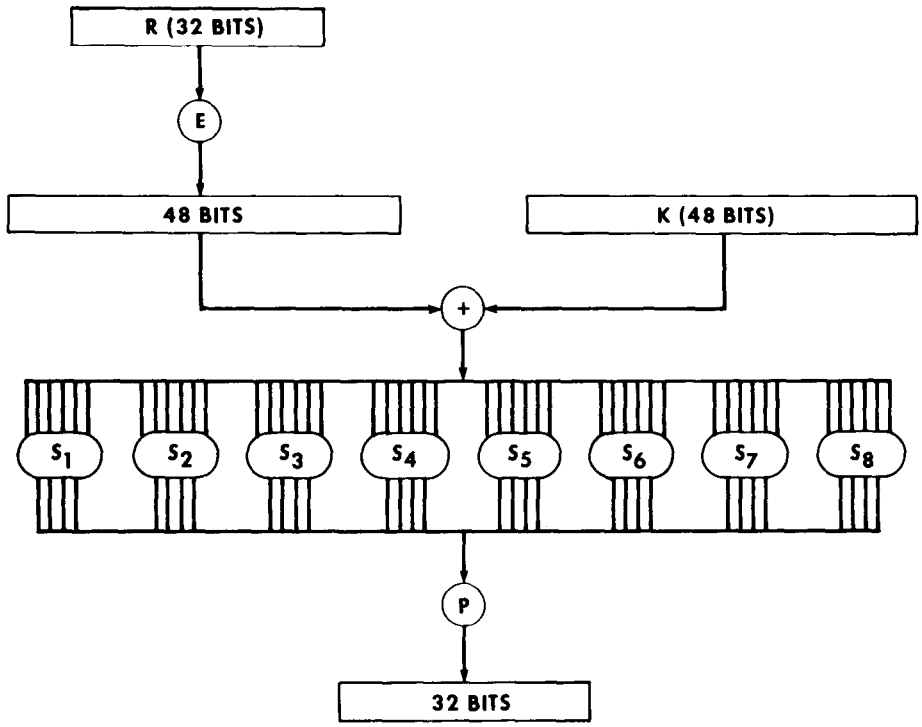


FIGURE 2. *Calculation of* f (R, K).

Let E denote a function which takes a block of 32 bits as input and yields a block of 48 bits as output. Let E be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

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 first three bits of $E(R)$ are the bits in positions 32, 1 and 2 of R while the last 2 bits of $E(R)$ are the bits in positions 32 and 1.

Each of the unique selection functions $S_1, S_2, \ldots, S_8$, takes a 6-bit block as input and yields a 4-bit block as output and is illustrated by using a table containing the recommended S_1:

S_1

Column Number

Row No.	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

If S_1 is the function defined in this table and B is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of B represent in base 2 a number in the range 0 to 3. Let that number be i. The middle 4 bits of B represent in base 2 a number in the range 0 to 15. Let that number be j. Look up in the table the number in the i'th row and j'th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of S_1 for the input B. For example, for input 011011 the row is 01, that is row 1, and the column is determined by 1101, that is column 13. In row 1 column 13 appears 5 so that the output is 0101. Selection functions $S_1, S_2, \ldots, S_8$ of the algorithm appear in the Appendix.

The permutation function P yields a 32-bit output from a 32-bit input by permuting the bits of the input block. Such a function is defined by 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

The output $P(L)$ for the function P defined by this table is obtained from the input L by taking the 16th bit of L as the first bit of $P(L)$, the 7th bit as the second bit of $P(L)$, and so on until the 25th bit of L is taken as the 32nd bit of $P(L)$. The permutation function P of the algorithm is repeated in the Appendix..

Now let $S_1, \ldots, S_8$ be eight distinct selection functions, let P be the permutation function and let E be the function defined above.

To define $f(R, K)$ we first define $B_1, \ldots, B_8$ to be blocks of 6 bits each for which

(6) $$B_1 B_2 \ldots B_8 = K \oplus E(R)$$

The block $f(R, K)$ is then defined to be

(7) $$P(S_1(B_1)S_2(B_2) \ldots S_8(B_8))$$

Thus $K \oplus E(R)$ is first divided into the 8 blocks as indicated in (6). Then each B_i is taken as an input to S_i and the 8 blocks $S_1(B_1)$, $S_2(B_2)$, . . ., $S_8(B_8)$ of 4 bits each are consolidated into a single block of 32 bits which forms the input to P. The output (7) is then the output of the function f for the inputs R and K.

APPENDIX

PRIMITIVE FUNCTIONS FOR THE DATA ENCRYPTION ALGORITHM

The choice of the primitive functions KS, S_1, . . ., S_8 and P is critical to the strength of an encipherment resulting from the algorithm. Specified below is the recommended set of functions, describing S_1, . . ., S_8 and P in the same way they are described in the algorithm. For the interpretation of the tables describing these functions, see the discussion in the body of the algorithm.

The primitive functions S_1, . . ., S_8 are:

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

The primitive function P is:

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

Recall that K_n for $1 \leq n \leq 16$, is the block of 48 bits in (2) of the algorithm. Hence, to describe KS, it is sufficient to describe the calculation of K_n from KEY for $n = 1, 2, \ldots, 16$. That calculation is illustrated in figure 3. To complete the definition of KS it is therefore sufficient to describe the two permuted choices, as well as the schedule of left shifts. One bit in each 8-bit byte of the KEY may be utilized for error detection in key generation, distribution and storage. Bits 8, 16, . . ., 64 are for use in assuring that each byte is of odd parity.

Permuted choice 1 is determined by the following table:

$PC-1$

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

The table has been divided into two parts, with the first part determining how the bits of C_0 are chosen, and the second part determining how the bits of D_0 are chosen. The bits of KEY are numbered 1 through 64. The bits of C_0 are respectively bits 57, 49, 41, . . ., 44 and 36 of KEY, with the bits of D_0 being bits 63, 55, 47, . . ., 12 and 4 of KEY.

With C_0 and D_0 defined, we now define how the blocks C_n and D_n are obtained from the blocks C_{n-1} and D_{n-1}, respectively, for $n = 1, 2, \ldots, 16$. That is accomplished by adhering to the following schedule of left shifts of the individual blocks:

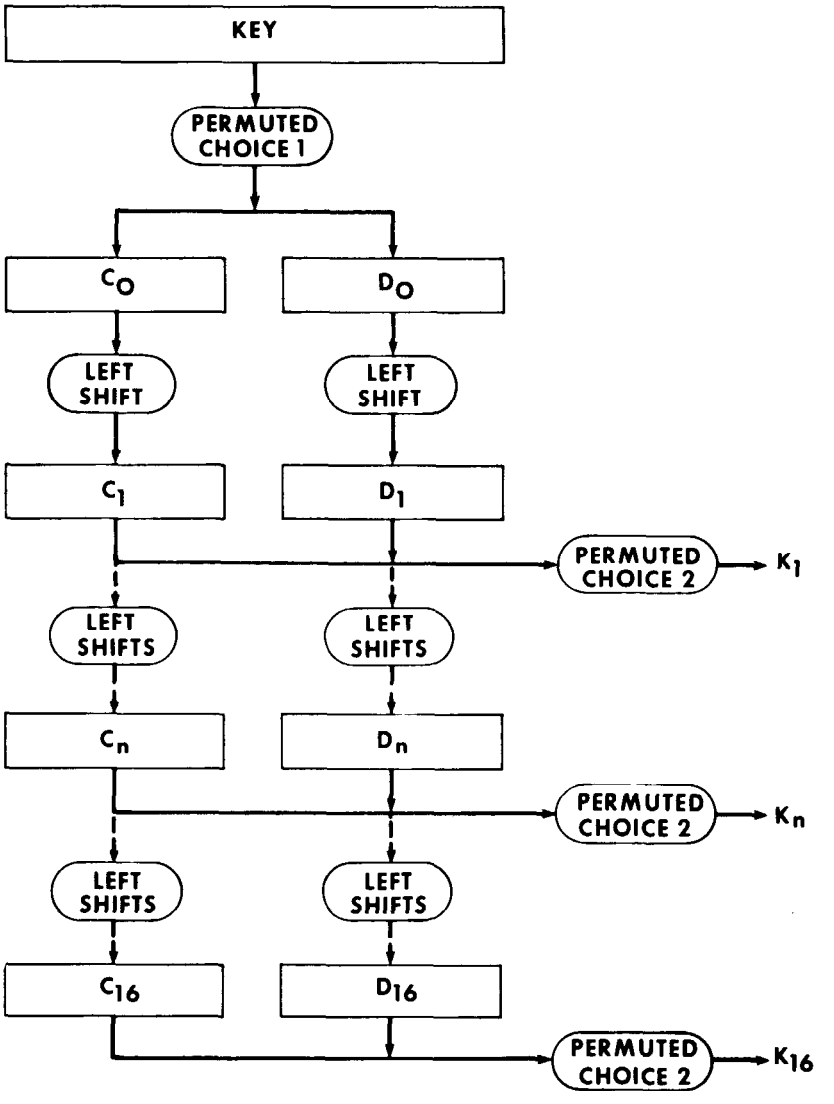


FIGURE 3. *Key schedule calculation.*

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

For example, C_3 and D_3 are obtained from C_2 and D_2, respectively, by two left shifts, and C_{16} and D_{16} are obtained from C_{15} and D_{15}, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3, . . ., 28, 1.

Permuted choice 2 is determined 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

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on with the 47th bit the 29th, and the 48th bit the 32nd.

APPENDIX B

Further Computations of Interest

TIME-MEMORY TRADE-OFF

In certain cases, the number of trials needed in a cryptographic attack can be significantly reduced by making use of a table of precomputed values. In effect, the attack is made more efficient by trading off computational time for memory. The technique is referred to as a *time-memory trade-off*. A method of searching key space using a time-memory trade-off is described by Hellman [1]. (Storage requirements may be quite large depending on the size of the precomputed table.)

Let R represent the total number of combinations associated with a particular cryptographic parameter under attack. For example, an attack against CE(M) is possible by precomputing the CE of $R = 2^m$ messages, where m is the number of bits in CE(M). If R different CEs are computed and the messages and CEs are stored in a table, an opponent can replace any intercepted message with another one by finding a message in the table with the same CE-value. Although only one table look-up is necessary, the attack requires R CE-values and messages to be stored.

The attack can be made more efficient by finding a better trade-off between table size and computation time. To illustrate this, let r1 be the number of precomputed values and r2 the number of intercepted values. Note that the attack succeeds if there is at least one match between the two sets; otherwise, the attack fails. Let p be the probability of success and $q = 1 - p$ the probability of failure. Assuming statistical independence, one obtains

$$q = (1 - r1/R)^{r2}$$

Taking the logarithm of each side of the equation and replacing $\ln(1 - r1/R)$ by its series expansion, one obtains

$$\begin{aligned} \ln q &= r2 \ln (1 - r1/R) \\ &= r2[-(r1/R) + (1/2)(r1/R)^2 - (1/3)(r1/R)^3 \ldots] \end{aligned}$$

for $r1/R \neq 1$, where ln stands for the natural logarithm. Thus q becomes

$$q = e^{(-r2\,r1/R)[\,1 - (1/2)(r1/R)^2 \ldots\,]}$$

The probability of success is thus

$$p = 1 - q = 1 - e^{(-r2r1/R)[1 - (1/2)(r1/R)\ldots]}$$

Since, in most cases, $r1/R \ll 1$, the probability of success is approximated by

$$p \simeq 1 - e^{(-r2r1/R)}$$

Moreover, since r1 is a measure of storage (as well as of preattack computation time) and r2 is a measure of computation time, a trade-off can be made between r1 and r2. For example, if $r1 = r2 = R^{1/2}$, there is a probability of success of $p = 1 - (1/e) = 0.63$.

BIRTHDAY PARADOX

A closely related, but different example is the celebrated problem of repeated birthdays, or *birthday paradox* [2]. Suppose there are r people gathered together. What is the probability q that no two persons have the same birthday. To answer this question, assume that each person can have his birthday on any one of the 365 days in the year and that each day of the year is equally likely to be the person's birthday. Each group of r people can then be characterized by r numbers between 1 and 365, where each number refers to a specific birth date. There are thus 365^r such sets of numbers. To assure that no two birthdays (no two numbers in each set of r numbers) are equal, any one of the 365 days can be chosen for the first person, any one of the remaining 364 can be chosen for the second person, and so on. Thus, for the rth person, there are $365 - r + 1$ days that can be chosen. Altogether there are $(365)(364)\ldots(365 - r + 1)$ different ways in which r birthdays can be selected. Thus, q is evaluated as follows:

$$q = (365)(364)\ldots(365 - (r - 1))/365^r$$

The probability that two or more of the r people have the same birthday is given by $p = 1 - q$. Table B-1 gives the values for p and $1 - p$ for different values of r. From the list, one sees that p is about 0.5 for $r = 23$. Thus, if 23 people are gathered together, there is about a 50-50 chance that some two of them have the same birthday, which is a much smaller number than most people would guess.

In general, if one can choose among R (instead of 365) outcomes, the probability for q becomes

$$\begin{aligned} q &= R(R - 1)(R - 2)\ldots(R - (r - 1)/R^r \\ &= (1 - 1/R)(1 - 2/R)\ldots(1 - (r - 1)/R) \end{aligned}$$

Using again the logarithmic transformation, one obtains

$$\ln q = \sum_{i=1}^{r-1} \ln(1 - i/R)$$

r	q	p
4	0.984	0.016
8	0.926	0.074
12	0.833	0.167
16	0.716	0.284
20	0.589	0.411
22	0.524	0.476
23	0.493	0.507
24	0.462	0.538
28	0.346	0.654
32	0.247	0.753
40	0.109	0.891
48	0.039	0.961
56	0.012	0.988
64	0.003	0.997

r denotes number of people and p denotes the probability that two or more of the r people have the same birthday.

Table B-1. Probabilities for the Birthday Problem

If $(r - 1)/R \ll 1$, then ln q can be approximated by using only the first term in the series expansion of ln, which results in

$$\ln q = -\sum_{i=1}^{r-1} i/R$$
$$= -(r-1)(r/2R)$$

For $r \gg 1$, this reduces further to

$$q = e^{-r^2/2R}$$

Thus, p is given by

$$p = 1 - q = 1 - e^{-r^2/2R}$$

In that case, the probability of at least one match (p) is about 1/2 when $e^{r^2/2R} = 2$ or $r^2 = 2R(\ln 2)$. For the case where R = 365, one obtains r = 22.5, which is in good agreement with the actual result (Table B-1).

REFERENCES

1. Hellman, M. E., "A Cryptanalytic Time-Memory Trade-Off," *IEEE Transactions on*

Information Theory, **IT-26**, No. 4, 401–406 (1980).

2. Parzen, E., *Modern Probability Theory and Its Applications,* Wiley, New York, 1960.

APPENDIX C

Plastic Card Encoding Practices and Standards

GENERAL PHYSICAL CHARACTERISTICS

A strip of magnetic material applied to the back of a plastic card has the capacity to handle multiple bands of encoded data. Track 1 is encoded as the uppermost band followed by Track 2 and then Track 3. Original specifications for the magnetic material allowed for Tracks 1 and 2 only, which are read only tracks. The additional Track 3 provides a capability for read or write or both.

TRACK 1

The International Airlines Transportation Association promulgated the development of Track 1 as the official track for airline use and, in fact, even defined the data and encoding format(s) for the [American National Standards Institute (ANSI)] standard. Its reason for developing this standard was to allow for use of customer-operated ticket dispensing machines to alleviate the congestion at airport ticket counters [1].

Today, other parties besides the airlines are interested in Track 1 because it is the only encoded track that permits the encoding of the cardholder's name. With this alphanumeric capacity, the cardholder's name can be printed on an EFT terminal receipt inexpensively; otherwise, the name would have to be sent from the computer which, most likely, will be more time consuming and costly [1].

There are 26 formats for Track 1, which are designated by format codes A through Z. The Track 1 format corresponding to format code B is shown below (proposed revised format [2]).

Field Name	Length (characters)
Start sentinel	1
Format code = "B"	1 (alpha only)
Primary Account Number	Up to 19
Separator (SEP)	1

Field Name	Length (characters)
Country Code	3
Name	2 to 26
Surname	
Surname separator = "/"	
First name or initial	
Space (when required)	
Middle Name or Initial	
Period (when followed by title)	
Title (when used)	
SEP	1
Expiration Date or SEP	4 or 1
Discretionary Data	(the balance up to maximum record length)
End sentinel	1
Longitudinal Redundancy Check (LRC)	1
MAXIMUM TOTAL	79

Format code A is reserved for proprietary use by the card issuer. Format codes C through M are reserved for ANSI use in connection with other data formats of Track 1. Format codes N through Z are available for use by individual card issuers.

TRACK 2

The American Bankers Association (ABA) led the development of Track 2 on behalf of the two credit card companies (Interbank and Visa) and their members. The intent was to have a standardized plastic card which could be used at point-of-sale (POS) terminals to obtain authorization for credit card transactions [1].

Today, in the financial industry, Track 2 is the most widely used encoding method for plastic cards. It has a strong following because most EFT terminals are connected (on-line) directly to a computer that accesses the cardholder's account data files. Also, it is the preferred choice of the ABA and is the only track recognized and supported by Visa and MasterCard in their debit/credit programs [1].

The format of Track 2 is shown below (proposed revised format [2]).

Field Name	Length (characters)
Start sentinel	1
Primary Account Number	Up to 19
Separator (SEP)	1

Field Name	Length (characters)
Country Code	3
Expiration date or SEP	4 or 1
Discretionary Data	(the balance up to maximum record length)
End Sentinel	1
Longitudinal redundancy check (LRC)	1
MAXIMUM TOTAL	40

Although Track 2 is widely accepted, there is a serious potential concern about it because of its limited encoding capacity—only 40 positions. The argument supporting the current capacity stresses that all the necessary information to authorize a transaction is at the data center thereby eliminating the need to encode extraneous data. On the other hand, those suggesting that capacity be increased feel that greater capacity would allow certain transactions to be approved directly at the terminal, or, at least, minimize the data sent between terminal and computer for each transaction. Those who hold this view are investigating the alternatives of using Tracks 1 and 3 with their on-line terminals in order to take advantage of the increased capacity [1].

TRACK 3

Track 3 was developed for use in off-line EFT terminals but was designed to be compatible with the other current plastic card standards. Thus, Track 3 is compatible with the ANSI standard for embossing plastic cards (ANSI X4.13-1979) and the ANSI standard for physical characteristics of magnetic stripes (ANSI X4.16-1973). More recently, financial institutions have started to consider its use in on-line systems because of its greater data storage capacity [1].

The format of Track 3 follows [3]:

Field Name	Usage[1]	Status[2]	Length (Characters)
Start sentinel	M	S	1
Format code	M	S	2
Primary account number (PAN)	M	S	19
Separator (SEP)	M	S	1
Country code or SEP	M	S	3 or 1
Currency	M	S	3
Currency exponent	M	S	1
Amount authorized per cycle period	M	S	4
Amount remaining this cycle	M	D	4
Cycle begin	M	D	4

Field Name	Usage[1]	Status[2]	Length (Characters)
Cycle length	M	S	2
Retry count	M	D	1
PIN control parameters or SEP	M	S	6 or 1
Interchange control	M	S	1
Type of account and service restriction (PAN)	M	S	2
Type of account and service restriction (SAN-1)	M	S	2
Type of account and service restriction (SAN-2)	M	S	2
Expiration date or SEP	M	S	4 or 1
Card sequence number	M	S	1
Card security number or SEP	M	D	9 or 1
First subsidiary account number (SAN-1)	O	S	variable[3]
SEP	M	S	1
Second subsidiary account number (SAN-2)	O	S	variable
SEP	M	S	1
Relay marker	M	S	1
Crypto check digits or SEP	M	D	6 or 1
Discretionary data	O	D	variable
End sentinel	M	S	1
Longitudinal redundancy check (LRC)	M	D	1
MAXIMUM TOTAL			107

[1] "M" indicates that usage is mandatory; "O" that it is optional.

[2] Dynamic fields (denoted by "D") shall be updated as appropriate by interchange partners. Static fields (denoted by "S") shall be updated by the card issuer only.

[3] There is no maximum length for a variable length field, except that the total number of characters in track 3 must not exceed 107.

Details of the specific data elements encoded on Track 3 are contained in reference 3.

REFERENCES

1. Thomas, O. T. "Funds Transfer Research Department Working Paper #4," United States League of Savings Associations, Chicago, IL (June 1980).
2. Proposed American National Standard X4.16, *Magnetic Stripe Encoding for Financial Transaction Cards,* American National Standards Institute, X4, New York (Draft, October 1980).
3. American National Standard X9.1-1980, *Magnetic Stripe Data Content for Track 3,* American National Standards Institute, X9, New York, 1980.

APPENDIX D

Some Cryptographic Concepts and Methods of Attack

FURTHER DISCUSSION OF AUTHENTICATION PARAMETERS

One Way Functions

As stated in Chapter 11, a one-way function is defined as follows:

> *A function f is a* one-way function *if, for any argument x in the domain of f, it is easy to compute the corresponding value y = f(x); yet for almost all y in the range of f, it is computationally infeasible, given a value of y and knowledge of f, to calculate any x whatsoever with the property that f(x) = y. It is important to note that a function is defined which is not invertible from a computational point of view, but whose non-invertibility is entirely different from that normally encountered in mathematics. A function f is normally called "noninvertible" when the inverse of a point y is not unique, i.e., there exist distinct points x1 and x2 such that f(x1) = y = f(x2). This is not the sort of inversion difficulty that is required here. Rather, it must be overwhelmingly difficult, given a value y and knowledge of f, to calculate any x whatsoever with the property that f(x) = y [1].*

The intent of this section is to demonstrate that the design of one-way functions is not as straightforward as might be expected at the onset. It is particularly true if constraints are introduced. This is shown by discussing the case where a growth path is provided from a system using an authentication parameter which is not one-way (i.e., PIN ⊕ ID) to one which is one-way [i.e., PIN ⊕ ID ⊕ f(KP, PIN, ID)].

To discuss details, let it be assumed that personal verification is achieved with the aid of authentication parameters stored in a verification table together with the corresponding user ID (Table D-1). (The APs are generated at the entry point and compared with the appropriate AP of reference at the issuer as discussed in Chapter 11). Also, let AP be a one-way function f of KP, PIN, and ID, i.e.,

$$AP = f(KP, PIN, ID)$$

Since AP is one-way, a secure implementation must allow that AP be public. (An opponent could get AP information, for example, if he could read the verification table.)

ID1, AP1 of Reference
ID2, AP2 of Reference
⋮ ⋮ ⋮ ⋮
IDn, APn of Reference

Table D-1. Verification Table

The condition that KP and PIN must not be derivable from AP and ID is usually easily satisfied by treating KP and PIN as keys. For example, $AP = E_{KP \oplus PIN}(ID)$ satisfies such a condition. But a more stringent condition is that it must not be possible to derive a set of KPs and PINs such that the same AP is generated with the appropriate ID (assumed to be known). In other words, it must not be computationally feasible to find equivalent values KP* and PIN* (Figure D-1) such that

$$AP = f(KP^*, PIN^*, ID)$$

This condition is also satisfied with $AP = E_{KP \oplus PIN}(ID)$, since the evaluation of $KP^* \oplus PIN^* = KP \oplus PIN$ for a given ID, AP pair is equivalent to the effort of key exhaustion. (ID can be considered plaintext and AP the corresponding ciphertext for which the key must be found.)

An example of a weak one-way function is

$$AP = E_{KP \oplus PIN}(ID \oplus PIN)$$

because a KP* and PIN* can be found such that

$$E_{KP^* \oplus PIN^*}(ID \oplus PIN^*) = E_{KP \oplus PIN}(ID \oplus PIN)$$

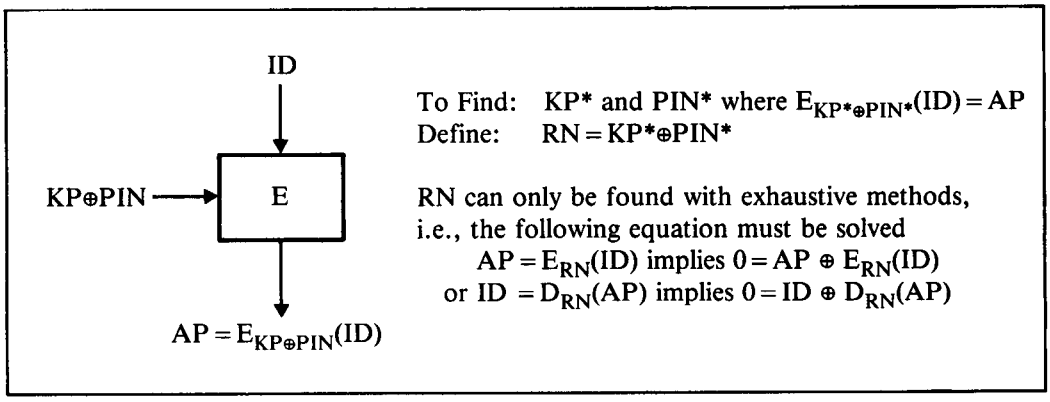


Figure D-1. Example of a Strong One-Way Function

To show this, let RN = KP* ⊕ PIN* be an arbitrarily selected quantity. The equation for AP can thus be rewritten as

$$AP = E_{RN}(ID \oplus PIN^*)$$

from which it follows that

$$PIN^* = ID \oplus D_{RN}(AP)$$

(It is assumed that the implementation allows up to 16-digit PINs even though 4-digit PINs may be used most frequently.) Furthermore, from the definition of RN, one concludes that

$$KP^* = RN \oplus PIN^* = ID \oplus RN \oplus D_{RN}(AP)$$

Thus, the equivalent values are only an easily computed function of the given ID, AP pair and an arbitrary quantity RN. The reason for the weakness is that changing the argument from ID in $AP = E_{KP \oplus PIN}(ID)$ to ID ⊕ PIN introduces an additional degree of freedom which allows the evaluation of equivalent KPs and PINs.

In Figure D-2 it is shown that $AP = E_{KP \oplus PIN}(ID \oplus KP)$ and $AP = KP \oplus E_{KP \oplus PIN}(ID)$ are not one-way functions either.

Attack Using Repeated Trials

The weak examples discussed so far have allowed the direct evaluation of equivalent parameters. But even if this is not the case, a candidate for a one-way function must be tested under the assumption that an opponent has a large set of (ID, AP) values, since he may, for example, obtain access to a verification table storing these values (Table D-2). Let the number of (ID, AP) pairs available to an opponent be equal to n and let the total number of possible AP values be N (i.e., 2^{64} if AP is represented by one block of the DES output).

If m trials are performed to obtain KP* and PIN*, the probability, q, of not generating a correct table entry is thus

$$q = \left(1 - \frac{n}{N}\right)^m$$

Taking the natural logarithm (ln) on both sides yields

$$\ln q = m \ln\left(1 - \frac{n}{N}\right)$$

Using the first two terms in a series expansion of $\ln\left(1 - \frac{n}{N}\right)$ results in

$$\ln q \simeq -m(n/N) + m(n^2/2N^2)$$

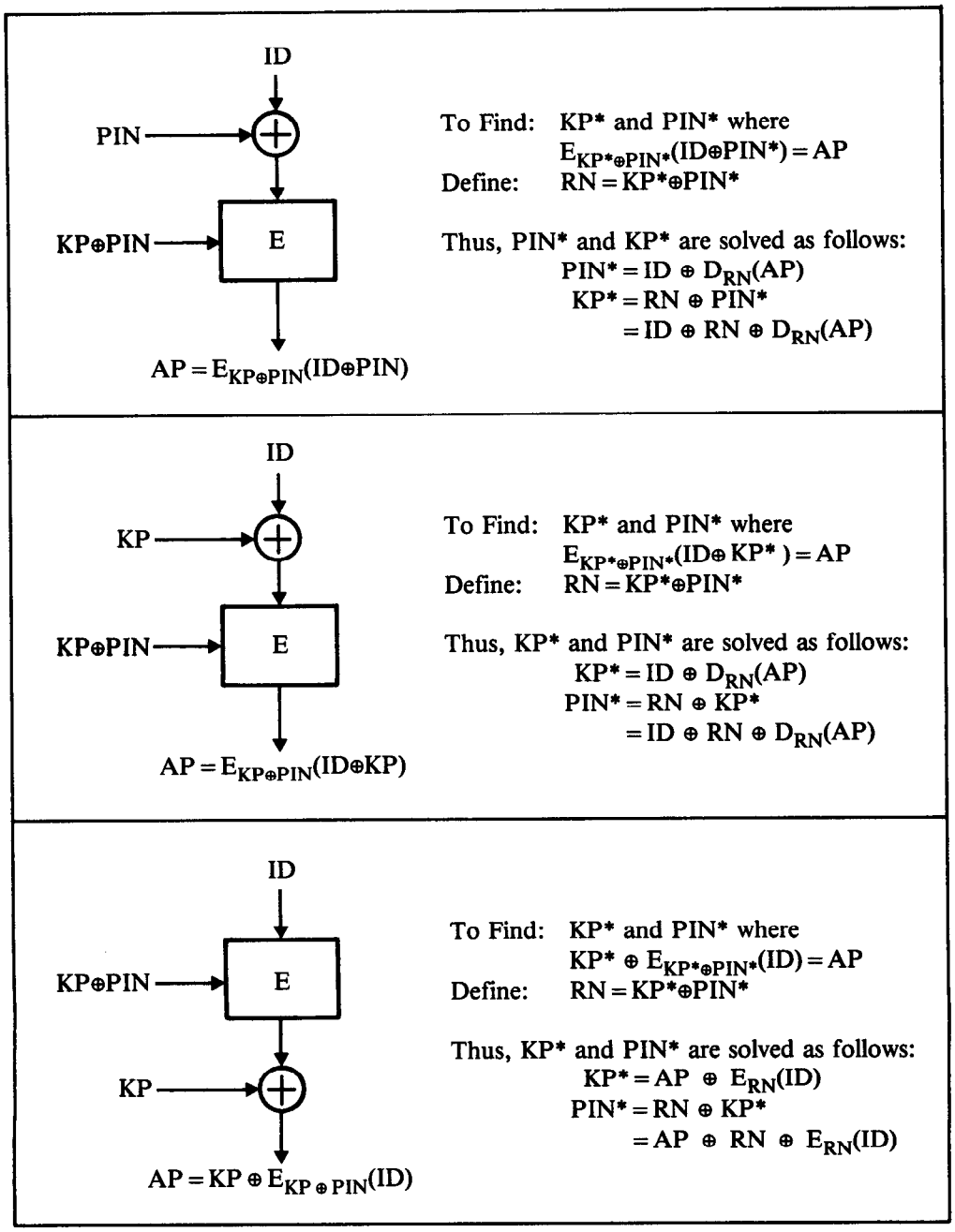


Figure D-2. Examples of Functions that are Not One-Way

If $n^2/2N^2 \ll n/N$ or $n/2N \ll 1$ (which is an easily satisfied condition), then the second term can be neglected and one obtains

$$\ln q = -m(n/N)$$
$$q = e^{-mn/N}$$

The probability, p, of finding at least one correct table value is thus $p = 1 - q$, or

$$p = 1 - e^{-mnN}; \qquad n/2N \ll 1$$

Values of p as a function of mn/N are given in Table D-2.

For $mn/2N \ll 1$, the probability p can be expressed as

$$p = mn/N; \qquad mn/2N \ll 1$$

If m and n are given, p is determined entirely by the number of bits ($\log_2(N)$) in the output (or range) of the one-way function. The minimum tolerable number of test combinations for AP is therefore determined by (1) how many trials (m) can be economically performed by an opponent and (2) how much information (n) the opponent has available. Since this information can be thought of as a dictionary, the attack falls into the category of dictionary attacks (discussed in Chapter 2) and will be labeled as such.

mn/N	$e^{-mn/N}$	$p = 1 - e^{-mn/N}$
16	0.000^+	1.000^-
8	0.000^+	1.000^-
4	0.018	0.982
3	0.050	0.950
2	0.135	0.865
1.5	0.223	0.777
1.0	0.368	0.632
0.5	0.607	0.393
0.25	0.779	0.221
0.20	0.819	0.181
0.10	0.905	0.095
0.05	0.951	0.049
0.01	0.990	0.010

Legend:

N = total number of possible AP values
n = number of AP, ID pairs available to an opponent
m = number of exhaustion trials to obtain equivalent values for KP and PIN

Table D-2. Values of p as a Function of mn/N

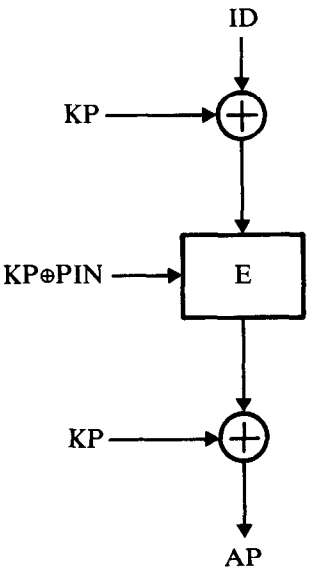


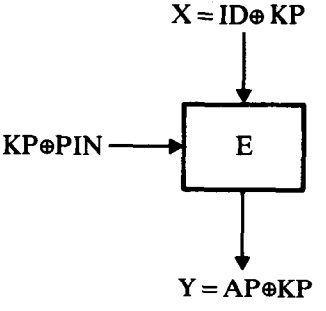
Figure D-3. Example of One-Way Function Which Yields to a Dictionary Attack

A weak one-way function which yields to the dictionary attack is AP = KP ⊕ $E_{KP \oplus PIN}$(ID ⊕ KP). AP can be thought of as the ciphertext corresponding to a triply encrypted ID. Two encryption steps are defined by modulo 2 addition and the third by the DES (Figure D-3).

An equivalent way of looking at it is to consider X = (ID ⊕ KP) to be the plaintext and Y = AP ⊕ KP to be the corresponding ciphertext (Figure D-4). The weakness which can be exploited is that X ⊕ Y = AP ⊕ ID, since KP is canceled out in the operation. But knowledge of a set of valid (ID, AP) pairs also implies knowledge of a valid set of (AP ⊕ ID) pairs.

To solve for

$$AP = KP^* \oplus E_{KP^* \oplus PIN^*}(ID \oplus KP^*)$$



Note that X⊕Y = AP⊕ID.

Figure D-4. Equivalent Representation of the One Way Function in Figure D-3.

select two arbitrary quantities, RN1 and RN2, and define them to be

$$RN1 = ID \oplus KP^*$$
$$RN2 = KP^* \oplus PIN^*$$

According to Figure D-4, RN1 can be considered equivalent plaintext, X*, and $E_{RN2}(RN1)$ can be considered equivalent ciphertext, Y*. The attack of finding equivalent values for RNi and KP succeeds when X* ⊕ Y* is an element in the set of known AP ⊕ ID values.

Thus RN1 is encrypted with RN2 and the result is added modulo 2 to RN1. If this quantity is in the set of known (AP ⊕ ID) values, the attack has succeeded, since in that case, it can be concluded that

$$RN1 \oplus E_{RN2}(RN1) = ID \oplus AP$$

i.e., plaintext added modulo 2 to ciphertext equals an element in the given table. But, by definition

$$RN1 = ID \oplus KP^*$$

Thus

$$KP^* = AP \oplus E_{RN2}(RN1)$$

and

$$PIN^* = RN2 \oplus KP^* = AP \oplus RN2 \oplus E_{RN2}(RN1)$$

If there is no match with a given table entry, a different RN1 and RN2 are selected and the process continues (RN2 could actually be fixed and RN1 could be variable, or vice versa). By making N sufficiently large the attack can be blocked (Table D-2).

To further illustrate the problems associated with designing one-way functions, consider a design in which a weak authentication parameter (AP = ID ⊕ PIN) is to be made into a strong one (which is one-way) at some future time by simply Exclusive-ORing ID ⊕ PIN with an additional quantity. To solve the problem, let

$$AP = (ID \oplus PIN) \oplus f(KP, PIN, ID)$$

where AP and f(KP, PIN, ID) must be strong one-way functions of KP, PIN, and ID. Again, it must be assumed that KP and PIN cannot be deduced from ID and AP. Furthermore, it must not be possible to find equivalent values such that

$$AP = (ID \oplus PIN^*) \oplus f(KP^*, PIN^*, ID)$$

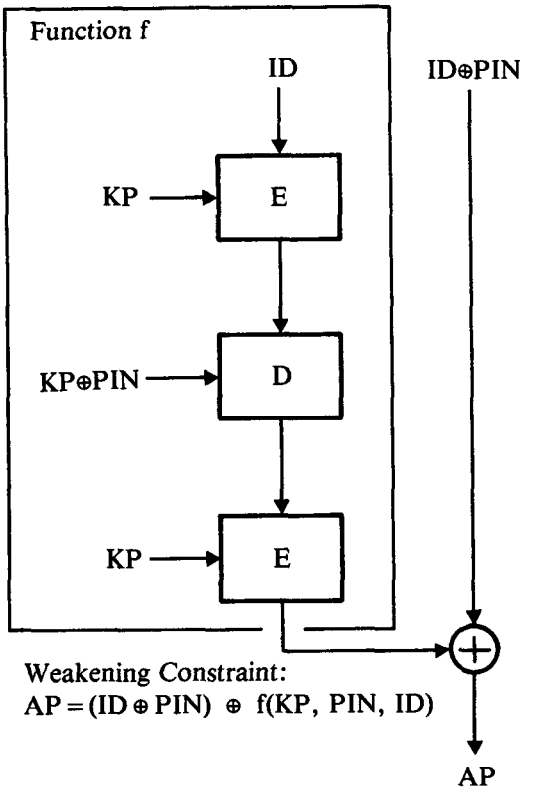


Figure D-5. Example of a One-Way Function Even with a Weakening Constraint

The problem at hand is more difficult than the one previously discussed, since the presence of (ID ⊕ PIN) makes it more difficult to design a strong one-way function. For example, let

$$f(KP, PIN, ID) = E_{KP \oplus PIN}(ID)$$

which was shown to be strong if used alone to define AP. In this case, AP is now defined as

$$AP = (ID \oplus PIN) \oplus E_{KP \oplus PIN}(ID)$$

Again, consider whether it is possible to solve for a KP* and PIN* such that

$$AP = (ID \oplus PIN^*) \oplus E_{KP^* \oplus PIN^*}(ID)$$

Defining, as before,

$$RN = KP^* \oplus PIN^*$$

one obtains

$$PIN^* = AP \oplus ID \oplus E_{RN}(ID)$$
$$KP^* = PIN^* \oplus RN = AP \oplus ID \oplus RN \oplus E_{RN}(ID)$$

Thus, choosing an appropriate ID, AP pair, together with an arbitrary quantity, allows the evaluation of PIN* and KP* in one trial. Therefore, the suggested AP is extremely weak. The lesson to be learned is as follows.

Coupling a strong one-way function with additional information may result in a weak one-way function.

To make sure that no exploitable degrees of freedom are introduced, a function f is defined that uses triple encryption (Figure D-5). The function defining AP is thought to be a strong one-way function of KP, PIN, and ID.

FURTHER DISCUSSION OF AUTHENTICATION PARAMETERS AND PERSONAL AUTHENTICATION CODES

Implementation Examples

To analyze the authentication parameter AP = ID ⊕ PIN, let two cases be assumed.

1. Personal verification is done with the aid of a verification table which stores the following quantities

$$\begin{array}{l} ID1\ E_{KA}(ID1 \oplus PIN1) \text{ of reference} \\ ID2\ E_{KA}(ID2 \oplus PIN2) \text{ of reference} \\ \vdots \\ IDn\ E_{KA}(IDn \oplus PINn) \text{ of reference} \end{array}$$

   where ⊕ indicates modulo 2 addition and KA is a system key. In this approach, ID and ID ⊕ PIN are routed through the network securely, and ID ⊕ PIN is translated into E_{KA}(ID ⊕ PIN) at the issuer. If this quantity is identical to the stored reference, the user is accepted. Otherwise, he is rejected.

2. Instead of storing E_{KA}(ID ⊕ PIN) in a verification table, it is stored on the bank card as a Personal Authentication Code (PAC). The authentication key, KA, is stored at the node where authentication takes place (e.g., the issuer, the switch, or at any other designated node). In this approach, ID, PIN ⊕ ID, as well as PAC are routed throughout the network.

At the authenticating node, the received ID ⊕ PIN is encrypted, as before, with the stored authentication key, KA. The result is compared with the received PAC. If both quantities are identical, the user is accepted; otherwise, the user is rejected.

Attack Against a 16-Digit PIN

Consider the following attack against implementation method 2 above. An opponent opens an account at a bank. The assigned ID and PIN, and the calculated value of ID ⊕ PIN, are thus known quantities. The opponent also knows PAC = E_{KA} (ID ⊕ PIN) since it is stored on the bank card.

If he wants to attack account number ID*, he only has to use an equivalent PIN defined PIN*, which can be evaluated as follows.

$$\text{PIN}^* = (\text{ID} \oplus \text{PIN}) \oplus \text{ID}^* \tag{D-1}$$

The appropriate PAC* is

$$\text{PAC}^* = E_{KA}(\text{ID}^* \oplus \text{PIN}^*) = E_{KA}(\text{ID} \oplus \text{PIN}) = \text{PAC} \tag{D-2}$$

which is identical to his own PAC, since by definition (Equation D-1)

$$\text{PIN}^* \oplus \text{ID}^* = \text{ID} \oplus \text{PIN}$$

Even with method 1, the system is vulnerable since an opponent knows how to change the reference in the verification table (which could be done temporarily during the attack). He only has to change the PAC of the user under attack to his own PAC according to Equation D-2.

Attack Against a 12-Digit PIN

To show an attack against a shorter PIN (i.e., 12 digits), the PIN and ID blocks are defined using the proposed ANSI method (see PIN Block Construction and Account Block Construction, Appendix E). If the PIN block and the account number block are added together, modulo 2, the result is as follows.

$$\begin{aligned} \text{AP} = {} & \text{C, N, P1, P2, A1} \oplus \text{P3, (A2} \oplus \text{P4)} \oplus \text{F,} \\ & \ldots, (\text{A10} \oplus \text{P12}) \oplus \text{F, A11} \oplus \text{F, A12} \oplus \text{F} \end{aligned} \tag{D-3}$$

The nature of the attack is to make the input to the algorithm equal to a known value. In that case, the resulting output is also known. In other words, the personal authentication code arrived at by encrypting the authentication parameter is predictable.

From Equation D-3 it follows that for a 12-digit PIN all quantities except the last two (i.e., A11 ⊕ F and A12 ⊕ F) are constant or can be controlled by an opponent. The control field C, is fixed and determined by the character

set. The PIN length field, N, is determined by the institution and hence is also fixed. (It is assumed that every user of the institution has a 12-digit PIN.) The next two fields (P1, P2) can be controlled by an opponent who knows what PIN information he has. The next ten quantities, A1 ⊕ P3 through (A10 ⊕ P12) ⊕ F, can be made equal to a predetermined value by requiring that

$$AN \oplus P = AN^* \oplus P^*$$
$$P^* = AN \oplus AN^* \oplus P; \quad N = 1, 2, \ldots, 10$$

where AN and P are the parameters associated with the account the opponent opened. AN* is the nth digit of the account number to be attacked, and P* is the (n + 2)nd PIN digit the opponent must use to attack the system.

Thus the only two quantities not constant or under the control of the opponent are A11 ⊕ F and A12 ⊕ F. Any account number to be attacked must therefore have A11 and A12 as the least significant digits. Since they represent only a total of 100 combinations, 1% of all accounts can be attacked. However, the probability of success can be increased if the opponent opens several accounts. If he opens ten accounts, roughly 10% of all accounts can be attacked. If he opens a few hundred accounts, then all or nearly all the accounts can be attacked.

Proposals for Authentication Parameters and Personal Authentication Codes

A recommended solution to the general method of attack against PIN given above is to redefine the authentication parameter AP as

$$AP = E_{CI \circledast PIN}(ID) \tag{D-4}$$

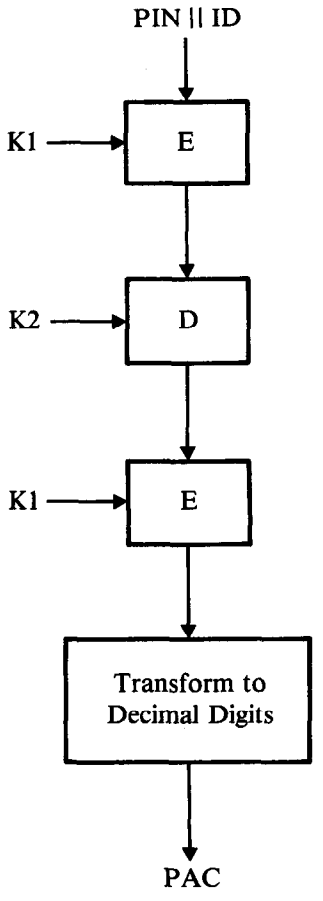
where CI represents information stored on a user's card. This would be implemented in addition to the authentication parameter recommended by the American Bankers Association (ABA):

$$AP = ID||PIN \tag{D-5}$$

where ID||PIN represents the concatenation of ID and PIN. (Since this is a preliminary proposal, the information is subject to change.)

If CI is public, both authentication parameters (Equations D-4 and D-5) have equal cryptographic security. In either case, AP is routed through the network in enciphered form. If CI represents secret card information (e.g., a personal key), then the first method is stronger since AP is, in that case, a one-way function of CI and PIN. Since the system must be able to determine which authentication parameter is used, a bit is stored on the card to indicate this.

A preliminary ABA proposal for the computation of a PAC is shown in Figure D-6. To generate n PIN digits, the set of 16 hexadecimal digits in the triply encrypted PIN||ID are scanned for digits that fall in the range 0 to 9



Note: || denotes concatenation.

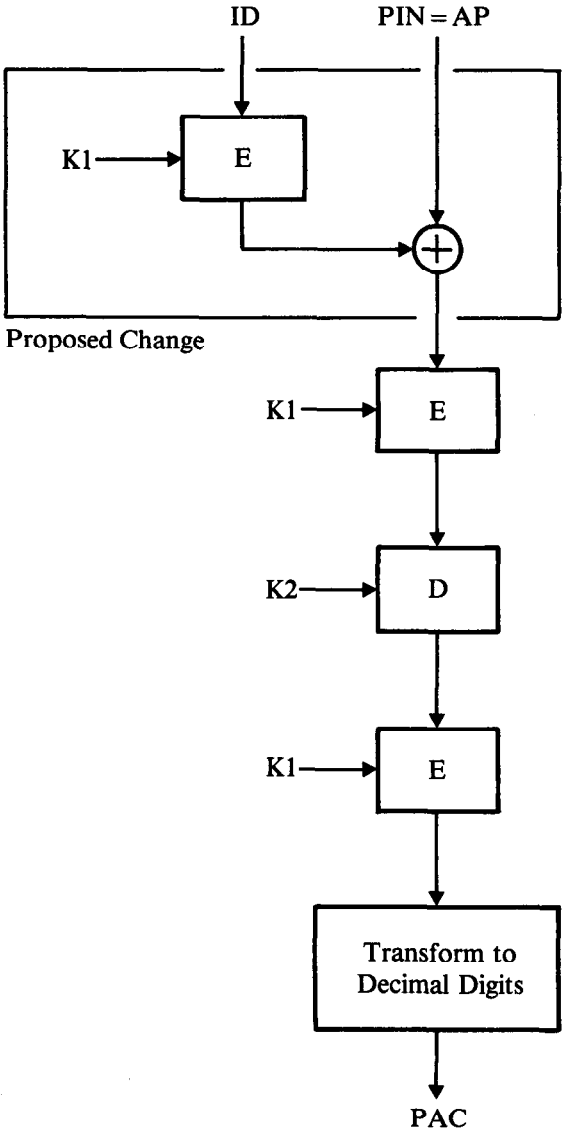
Figure D-6. Preliminary ABA Proposal for Personal Authentication Code

(i.e. decimal digits). If the number of decimal digits found is equal to or larger than n, the first n digits are defined the PIN. If the number is less than n, say r, then n-r additional decimal digits are generated from the first n-r hexadecimal digits in the range A to F by subtracting 10 from their values. The scheme has two disadvantages.

1. The transformation is biased towards certain digits (0 through 5), which reduces the effective number of PIN combinations.
2. Only a limited number of PIN digits are used (e.g., up to four).

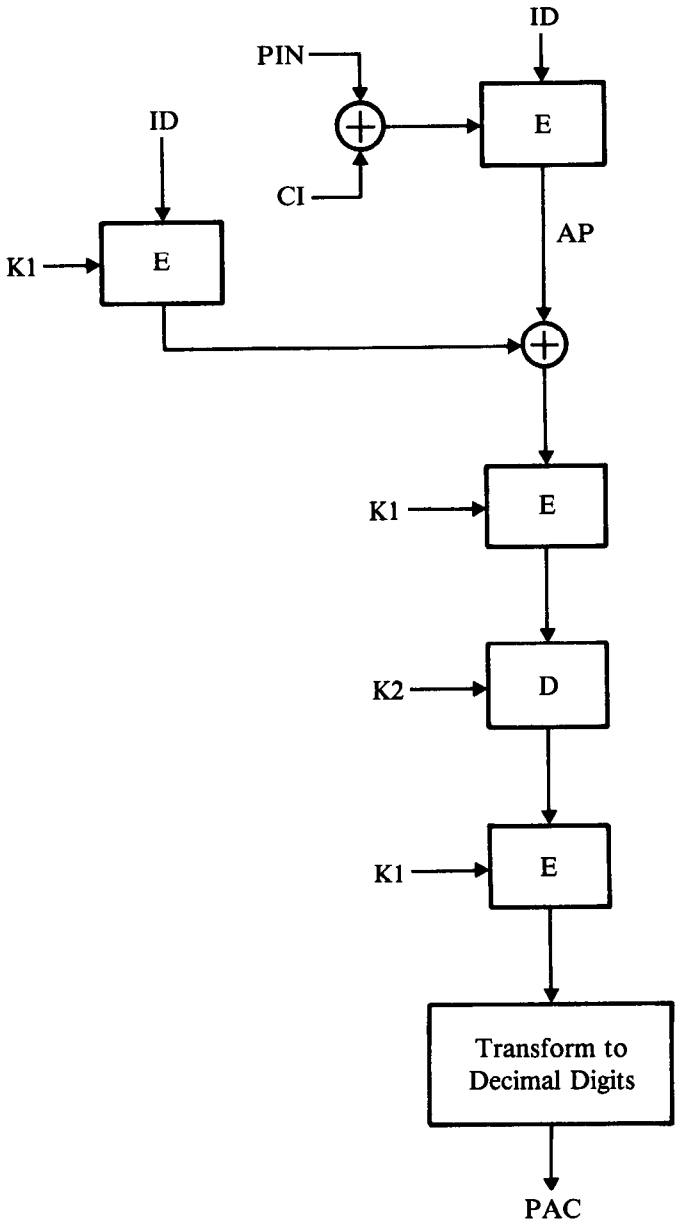
To eliminate the first disadvantage, a different transformation could be devised. For example, instead of generating 64 bits (16 hexadecimal digits) with one triple encryption step, 128 bits (32 hexadecimal digits) could be generated using two triple encryption steps. In that case, the probability of obtaining decimal digits directly is much higher (i.e., there are fewer instances in which it is necessary to transform hexadecimal digits A through F to decimal digits). The second disadvantage could be overcome by allocating

64 bits for both the ID and PIN and couple both by introducing another encryption under K1 as shown in Figure D-7. In a more secure approach, PIN is replaced by a one-way function of CI and ID as shown in Figure D-8. If the additional encryption of ID with CI is not acceptable, ID could be coupled to AP as shown in Figure D-9. The entry of ID at both points [labeled (a) and (b) in Figure D-9] is necessary. If only one entry were used, the approach would be weak for the case where K1 = K2.



Note: The proposed change allows the use of up to 16 PIN digits and, at the same time, achieves coupling with ID.

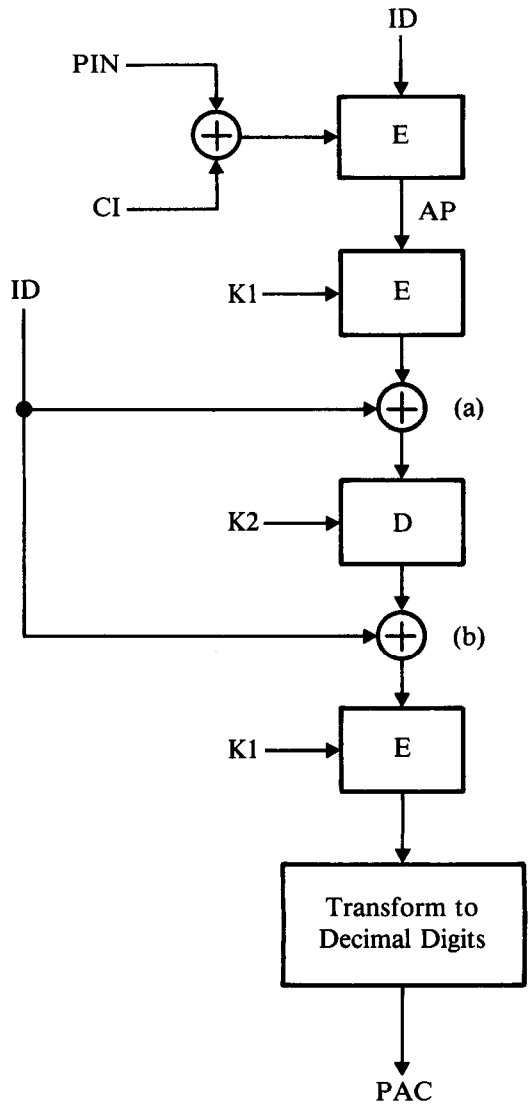
Figure D-7. Modified Preliminary ABA Scheme



Legend: AP = Authentication Parameter
CI = Card Information
ID = Used Identifier
PIN = Personal Identification Number
PAC = Personal Authentication Code

Note: During initialization at the issuer, an AP is calculated for each given ID, PIN, and CI. ID and AP are also used to calculate PAC, which is stored on the bank card. During personal verification, AP is calculated at the entry point from the entered values of ID, PIN, and CI. The entered ID and AP, and the calculated PAC are then sent to the authenticator.

Figure D-8. Generation of Authentication Parameter and Personal Authentication Code—Method 1



Note: During initialization at the issuer, an AP is calculated for each given ID, PIN, and CI. ID and AP are also used to calculate PAC, which is stored on the bank card. During personal verification, AP is calculated at the entry point from the entered values of ID, PIN, and CI. The entered ID and AP, and the calculated PAC are then sent to the authenticator.

Figure D-9. Generation of Authentication Parameter and Personal Authentication Code—Method 2

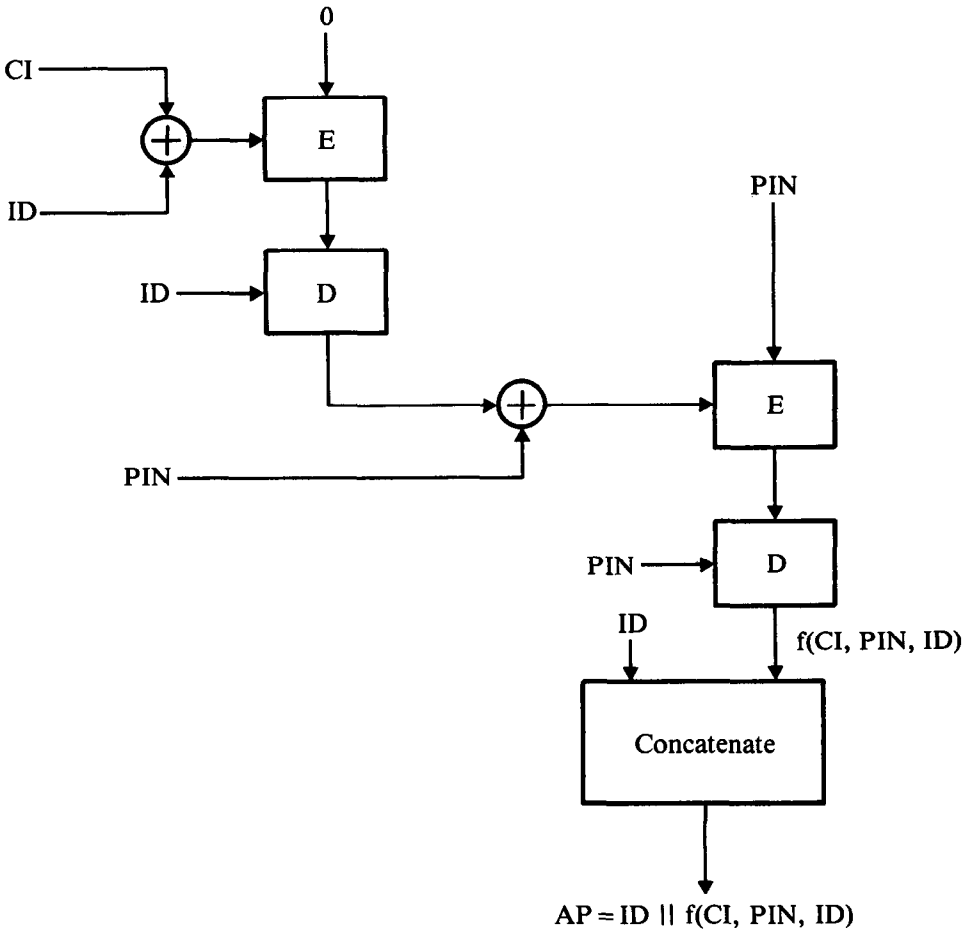


Figure D-10. Computation of AP with Migration Properties.

The AP value defined in Figure D-10 has the advantage that it provides a migration path to a very secure approach incorporating secret card information. Thus when CI ≠ 0, AP = ID||f(ID, PIN, CI), where f(ID, PIN, CI) is a one-way function and CI is secret card information. In the degenerate case (where CI = 0), AP = ID||PIN (i.e., the output of the function is equivalent to that defined by the ABA proposal, Figure D-6.

The Advantage of an AP that Depends on ID

The advantage of using ID as well as PIN in generating an authentication parameter is shown by the analysis of three different methods, where AP is a function of

1. PIN only, e.g., AP = PIN
2. PIN and ID, e.g., AP = $E_{PIN}(ID)$
3. PIN, KP, and ID, e.g., AP = $E_{KP \oplus PIN}(ID)$

Furthermore, let m be the number of different PIN combinations and n the number of different user IDs. To simplify the discussion, the assumption $n \leqslant m$ shall be made such that each user has a unique PIN. (Although, in general, this assumption will not hold, it is nevertheless useful to analyze this simple case to demonstrate the usefulness of coupling ID and PIN.) In that case, there are then m PIN choices for user 1, $m - 1$ PIN choices for user 2, and so on. In general, there are $m - (i - 1)$ PIN choices for the ith user.

Assume also that $PAC = E_{KA}(AP)$ is stored in a verification table and that it is possible to obtain $E_{KA}(AP^*)$ for a trial value of AP*. (A good key management scheme would prevent this.) If $E_{KA}(E_{PIN^*}(ID))$ can be related to each table entry $E_{KA}(E_{PIN}(ID))$, there are at most m (on the average about m/2) trials required to obtain the correct PIN for a given ID. On the other hand, to obtain all PIN/ID correspondences, the maximum number of trials needed would be:

$$\begin{aligned} S &= m + (m-1) + \cdots + [m-(n-1)] \\ &= mn - [1 + 2 + \cdots + (n-1)] = mn - [n(n-1)/2] \\ &= n[m-(n-1)/2]; \qquad n < m \end{aligned}$$

whereas on the average only about half of that number (S/2) are required.

To simplify the computation, assume that there are as many users as there are PIN combinations ($m = n$). Then the number of trials an opponent needs to get all PIN/ID combinations is equal to the number of trials when $n = m - 1$ (because the last (nth) PIN is automatically determined if all the others are known). Using the above equation for S (with $n = m - 1$) yields

$$\begin{aligned} S &= (m-1)[m-(m-2)/2] \\ &= (m-1)(m/2+1); \qquad n = m \end{aligned}$$

It should be pointed out that the number of trials is drastically reduced if PAC is only a function of the PIN and not of the ID. For example, let $PAC = E_{KA}(PIN)$. Then $E_{KA}(PIN^*)$ can be related uniquely to $E_{KA}(PIN)$ and thus a proper ID can be evaluated with certainty for each trial of PIN (provided that the PIN is currently assigned to some user). Hence, the advantage of

Goal of Attack	Maximum Number of Trials Needed to Exhaust the Combinations of PIN, i.e., to Obtain the Correct PIN/ID Relationship		
	AP = PIN	$AP = E_{PIN}(ID)$	$AP = E_{KP \oplus PIN}(ID)$
Arbitrary (ID, PIN)	1	m	more than 10^{17}
Particular (ID, PIN)	m	m	more than 10^{17}
All (ID, PIN)s	m	$(m-1)(m/2+1) \simeq (m^2/2) + (m/2) - 1$	more than 10^{17}

Table D-3. Maximum Number of Trials Needed to Exhaust the Combinations of PIN

coupling the ID together with the PIN is clearly demonstrated for the case where n = m. The results are summarized in Table D-3. The same advantage can be gained for the case where n > m, although the computations are omitted from the discussion.

INCREASING EXHAUSTIVE ATTACK WORK FACTOR BY IMPLEMENTATION METHODS

Multiple Encryption and Block Chaining

Usually the main emphasis is placed on the strength of the algorithm. An equally important factor on overall strength, however, is how the algorithm is implemented. For example, the work factor for exhaustive attacks can be significantly increased by using the DES with multiple encryption. A particularly attractive approach is one which allows compatibility between a basic single encryption scheme and a more secure multiple encryption scheme. This can be achieved by defining the cryptographic operations

$$Y = E_{K1} D_{K2} E_{K1}(X)$$
$$X = D_{K1} E_{K2} D_{K1}(Y)$$

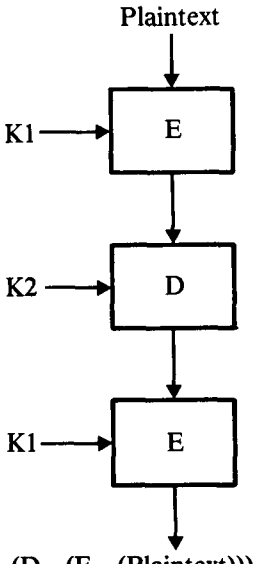
where K1 and K2 are independently chosen (56-bit) keys [2] as shown in Figure D-11. This operation has the property that for the case where K1 = K2 = K, the relationships $Y = E_K(X)$ and $X = D_K(Y)$ hold.

Thus a high security multiple encryption implementation using K1 and K2 could communicate with a less secure single encryption scheme just by setting K1 equal to K2. Although, technically speaking, the method is not quite as strong as a method using three different keys, it is much stronger than double encryption with two different keys as analyzed below.

If arbitrary plaintext and corresponding ciphertext are known, (single encryption is assumed), the key can be determined by enciphering the plaintext with one key after the other until the correct ciphertext is produced. This requires, on the average, 2^{55} trials for the DES. Such an attack is, in theory, always possible regardless of the implementation. This is so because it must be assumed that some encrypted data will at some time become public.

A significant decrease in the exhaustive work factor is possible if plaintext (selected by an opponent) and corresponding ciphertext can be obtained (called a selected plaintext attack). One way to defend against the selected plaintext attack is to introduce "noise" into the encryption procedures such that the system itself does not encrypt the plaintext selected by the opponent. Block chaining achieves this via an initializing vector, as discussed in Chapter 2. *As a consequence, all attacks described below will fail when block chaining is used (a recommended mode of DES operation).* This represents an additional powerful argument in favor of chaining methods.

To determine the consequence of a weak implementation which allows a selected plaintext attack, some cryptanalytic techniques are discussed em-



$E_{K1}(D_{K2}(E_{K1}(\text{Plaintext}))) = \text{Ciphertext}$

Note: $E_{K1}(D_{K2}(E_{K1}(\text{Plaintext}))) = E_{K1}(\text{Plaintext})$ when $K1 = K2$.

Figure D-11. Multiple Encryption Process with Migration Properties Using Two Independent Keys

phasizing the possible tradeoffs between them in terms of the storage requirements and time requirements needed to conduct the analysis. To show how such an attack, if it were possible, reduces the exhaustion work factor, let it be assumed that $E_K(P) = C$ (where C represents ciphertext) is available to an opponent who can select the plaintext, P.

The attacks discussed below are mostly of academic interest. As a practical matter, it appears that a viable business case cannot be made for development of a special purpose computer capable of key exhaustion, and storage of large tables, on the order of 2^{56} entries, is technically infeasible.

Reduction of Exhaustion Work Factor for Selected Plaintext Attack

Time-Memory Tradeoff: Approach 1

The time needed to attack a key after ciphertext C for selected plaintext P has been obtained (i.e., $C = E_K(P)$) can be minimized if a table of all plaintext and matching ciphertext is precomputed for each possible key (k1, k2, . . . , kn), where $n = 2^{56}$ as shown in Table D-4. All ciphertext values are then sorted and properly stored. Once this initial task is completed (which may require years or even hundreds of years depending upon the capability of the opponent), the attack against any key requires now a lengthy table lookup but no exhaustive encryption procedure since that was done beforehand. In this case the attack time against any key is drastically reduced at the expense of precomputing, sorting, and storing a table with 2^{56} entries.

k1	$E_{k1}(P) = C1$
k2	$E_{k2}(P) = C2$
⋮	⋮ ⋮
kn	$E_{kn}(P) = Cn$

Table D-4. Precomputed Ciphertext Values for a Selected Plaintext, P, Using a Single Encryption Step

This attack is the most straightforward one and it trades maximum storage space for minimum time needed to attack any key once the appropriate cryptogram (i.e., $E_K(P)$) has been obtained. A variation of this procedure suggested by Professor Martin Hellman of Stanford University, at the 1977 National Computer Conference held in Anaheim, California, is possible by reducing storage requirements at the expense of computation time.

Time-Memory Tradeoff: Approach 2

Instead of storing $Ci = E_{Ki}(P)$ in a table together with Ki, a string of encryptions, say t, is performed for a selected starter key, ki, as shown in Figure D-12. Let there be T tables of m double word entries and let a key (randomly or systematically) selected out of the N possible keys ($N = 2^{56}$) be stored in the first word of each of the possible Tm table entries. (Note that t is related to computation time and m is related to memory size.)

Each key stored in the first word of each table entry is used to calculate a corresponding ciphertext (using t encryption processes as indicated in Figure D-12), which is stored in the second word of the table entry. Let the transformation shown in Figure D-12 be different for each of the T tables for reasons explained below. In that case, mt encryption steps are needed to create each table containing starter keys k1,1 through km,1 and corresponding ciphertext C1,t through Cm,t where each ciphertext is arrived at by encrypting a selected plaintext, P, a number of times, t, in the way shown in Figure D-12. After that the ciphertexts (C) are sorted for each of the T tables. This completes the process of precomputation.

The attack against an unknown key, K, proceeds as follows: If the obtained $Y1 = E_K(P)$ is equal to a table entry, $Cr,t = E_{kr,t}(P)$, then it can be concluded that K is equal to kr,t.[1] Since the seed key in that case is equal to kr,1 the unknown key, equal to kr,t, can be obtained by performing t − 1 encryption steps starting with kr,1 according to Figure D-12. (A different transform from 64 to 56 bits is used depending on which one of the T tables Cr,t is found in.) If $Y1 = E_K(P)$ is not found in any of the tables, Y1 (which is 64 bits long) is transformed with the T different transforms to create Y1trans1, Y1trans2, . . . , Y1transT (which are 56 bits long).

[1] To avoid unnecessary complexity, a third subscript to indicate a particular table is not introduced. It should be understood that the starter keys for each table are different (as well as the transformations).

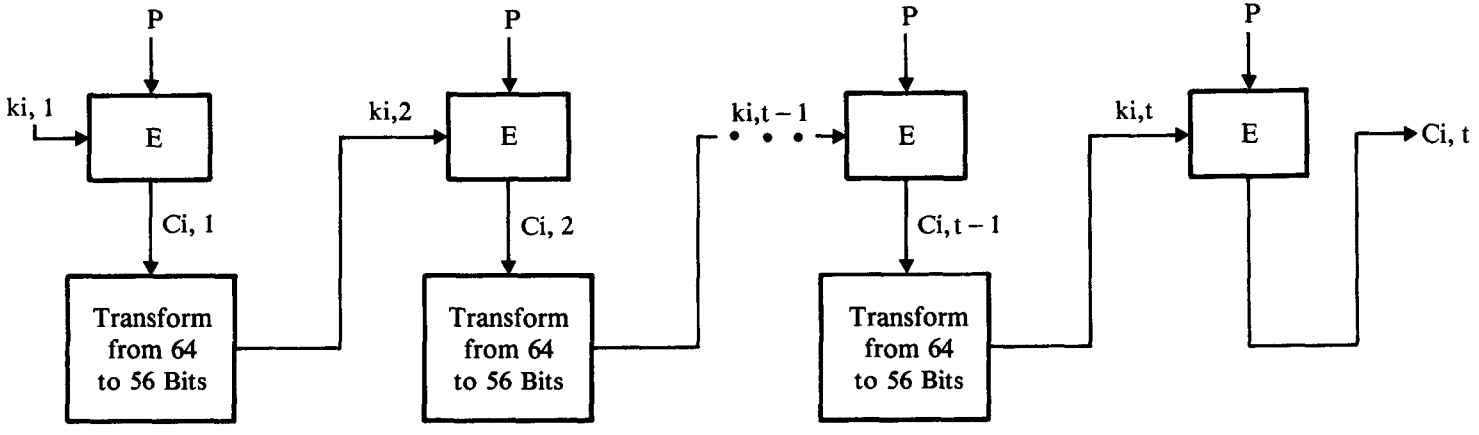


Figure D-12. Precomputed Ciphertext for a Selected Plaintext P Using a Sequence of Encryption Steps

P is next encrypted with Y1transl through Y1transT. If $E_{Y1trans1}(P)$ is equal to the value Cs,t in Table 1, then it can be concluded that ks,t − 1 is the correct key and that ks,1 is the corresponding seed key to generate ks,t − 1. To generate the correct key, ks,t − 1, requires, in this case, t − 2 encryption steps starting with ks,1 in addition to the T encryptions with Y1trans1 through Y1transT and the one to generate $E_K(P)$. If $E_{Y1trans1}(P)$ is not in table 1, a test is then made to determine if $E_{Y1trans2}(P)$ is in table 2; if $E_{Y1trans2}(P)$ is not in table 2, a test is then made to determine if $E_{Y1trans3}(P)$ is in table 3; and so on. If no match is found, then $E_{Y1trans1}(P)$ through $E_{Y1transT}(P)$ are transformed under their respective 64-bit to 56-bit transformations to produce Y2trans1, Y2trans2, . . . , Y2transT and these values are used as keys to encrypt P to produce $E_{Y2trans1}(P)$, $E_{Y2trans2}(P)$, . . . , $E_{Y2transT}(P)$. Again, $E_{Y2trans1}(P)$ is checked against table 1, $E_{Y2trans2}(P)$ is checked against table 2, and so forth. If no match is found, the entire procedure is repeated until the t-1 encryptions are exhausted (i.e., the procedure ends with $E_{Yt\text{-}1trans1}(P)$, $E_{Yt\text{-}1trans2}(P)$, . . . , $E_{Yt\text{-}1transT}(P)$). In any case, if the correct key is kr,s, then there are s − 1 encryption steps performed starting with kr,1. In addition, there is the initial step to create $E_K(P)$ (which is performed on the system storing the secret key K) plus (t − s)T steps (maximum) to pinpoint the correct row (s in this case) in the precomputed T tables. Hence altogether there are (s − 1) + 1 + (t − s)T = s + (t − s)T encryption steps required.

The parameters mT (number of starter keys) and t (number of encryptions to arrive at C from the appropriate starter key) must be selected such that the key to be attacked can, for all practical purposes, be obtained with high probability. Since the total number of keys which come into play in the table generation process is equal to mtT, the attack will always succeed if all N possible keys (i.e., 2^{56} for DES) are present. To evaluate the probability, po(mt,N), that none of the N keys are missing in the mtT possible locations, the following approach can be taken.

From a probabilistic point of view the different mtT slots in the table generation process can be considered balls which are placed randomly into N cells. If a cell is not selected it means that the key corresponding to that cell cannot be obtained with this method. The probability, po(mtT,N), that all N cells are selected (i.e., that none of them is empty), is evaluated in reference 3 as follows.

$$po(mtT,N) = \sum_{i=0}^{N} (-1)^i \binom{N}{i} [1 - (i/N)^{mtT}] \qquad \text{(D-6)}$$

where

$$\binom{N}{i} = \frac{N!}{i!\,(N-i)!}$$

The direct numerical evaluation of the equation is practical only for the case

of relatively small N and mtT. For larger values of interest, a good approximation can be found [3] if

$$\Delta = Ne^{-mtT/N} \tag{D-7}$$

remains bounded. Since this is the case here (as shown below) one can use

$$po(mtT,N) = e^{-\Delta} \tag{D-8}$$

Solving for mtT/N yields

$$mtT/N = \ln(N) - \ln\{\ln[1/po(mtT,N)]\} = \ln\{N/\ln[1/po(mtT,N)]\} \tag{D-9}$$

Choosing different values for N and po(mtT,N), the required ratio mtT/N can be evaluated. The results are shown in Table D-5. Note that $N = 2^{56}$, 2^{112}, and 2^{168} can be related to the situation where a time-memory tradeoff is used to attack single encryption, double encryption, and triple encryption, respectively. It is, however, shown below that double encryption can be attacked more efficiently with another exhaustive technique. For triple encryption, on the other hand, the work factor is so high that no viable exhaustion method is available.

It must be realized that Table D-5 covers the (unrealistic) case where it is possible to recover any key. In the practical situation, one is content with recovering a certain percentage of keys. Hence, in that case, it is not necessary to have all keys represented in the matrix of mtT entries. Let it therefore be assumed, that on the average, a certain fraction of the keys cannot be evaluated. In that case the ratio mtT/N can be smaller than the ones quoted in Table D-5. For example, the probability pr(mtT,N) that exactly r keys out of the total N keys are missing in the matrix of mtT entries can be evaluated as [3]

$$pr(mtT,N) = e^{-\Delta}\Delta^r/t!$$
$$\Delta = Ne^{-mtT/N}$$

This represents the Poisson distribution. The expected value of the number of keys that cannot be recovered is thus Δ with a variance also equal to Δ.

The expected (average) value of the fraction of keys which cannot be recovered is thus

$$E(r/N) = E(r)/N = \Delta/N = e^{-(mtT/N)} \tag{D-10}$$

and the variance is

$$Var\,(r/N) = Var\,(r)/N^2 = e^{-mtT/N}/N \tag{D-11}$$

As long as $E(r/N)/[Var(r/N)]^{0.5} \gg 1$ (i.e., $[Ne^{-mtT/N}]^{0.5} \gg 1$), the distribu-

	N		
po(mtT,N)	2^{56} ln(N) = 38.816	2^{112} ln(N) = 77.633	2^{168} ln(N) = 116.449
1/N	35.2	73.3	111.7
10^{-11}	35.5	74.3	113.1
10^{-10}	35.6	74.4	113.2
0.000001	36.2	74.9	113.7
0.00001	36.4	75.0	113.8
0.0001	36.6	75.4	114.2
0.001	36.9	75.7	114.5
0.01	37.3	76.1	114.9
0.1	38.0	76.8	115.6
0.25	38.5	77.3	116.1
0.5	39.2	78.0	116.8
0.75	40.1	78.9	117.7
0.9	41.1	79.9	118.7
0.99	43.4	82.2	121.0
0.999	45.7	84.5	123.3
0.9999	48.0	86.8	125.7
0.99999	50.3	89.1	128.0
0.999999	52.6	91.4	130.3
$1-10^{-10}$	61.8	93.7	132.6
$1-10^{-11}$	64.1	96.0	134.9
1-(1/N)	77.6	155.3	232.9

Legend:

mtT: Number of keys generated in the table generation procedure
N: Total number of possible key combinations
$mtT/N = \ln\{N/\ln[1/po(mtT,N)]\}$

Table D-5. Ratios mtT/N for Different Probabilities, po(mtT,N), of Having All Keys Present in the Precalculation Process

tion of r (Poisson) does not have to be considered. In that case the expected value alone (which is only a function of the ratio mtT/N) determines mtT/N.

Thus for a given expected value E(r/N) the quantity mtT/N can be evaluated as follows.

$$mtT/N = \ln[1/E(r/N)] \qquad \text{(D-12)}$$

Some numerical results are given in Table D-6.

The difference between the results shown in Table D-5 and Table D-6 is as follows. In Table D-5 a value of mtT/N can be obtained as a function of the probability that all keys are represented in the precomputed matrix requiring mtT computations. In Table D-6 a value of mtT/N can be obtained as a function of the average fraction of nonrecoverable keys. Choosing, for

E(r/N)	mtT/N
Fraction of Keys Which, on the Average, Cannot Be Obtained	Total Number of Precalculations Divided by Total Number of Keys
10^{-11}	25.3
10^{-10}	23.0
0.000001	13.8
0.00001	11.5
0.0001	9.2
0.001	6.9
0.01	4.6
0.1	2.3
0.25	1.4
0.3679	1.0

Legend:

mtT: Number of keys generated in the table generation procedure
N: Total number of possible key combinations
$mtT/N = \ln[1/E(r/N)]$

Table D-6. Ratio mtT/N for Different Average Fractions of Nonrecoverable Keys E(r/N)

example, mtT/N = 11.5, practically guarantees that 99.999% of the keys can be obtained (Table D-6).

The underlying assumption made so far is that all keys occur at random. To justify the assumption that keys are generated randomly requires that certain conditions for m and t are met to prevent (with high probability) the following situation from happening. Assume that a key is duplicated in two rows of one of the T tables (e.g., ki,r = kj,s). In that case this accidental equivalence of two row entries leads to an equivalence of the rest of the two rows.

To evaluate the probability of such an event, let pi represent the probability that all keys in row i are different from each other and are different also from the previously generated keys (i − 1)t. Thus

$$pi = \prod_{j=1}^{t} [N - (i-1)t - (j-1)]/N$$

Since the last term (j = t) represents the smallest factor, (N − it + 1)/N, which in turn is larger than [1 − (it/N)], a lower bound for pi can be found as follows.

$$pi > [1 - it/N)]^t$$

Taking the natural logarithm and approximating $\ln[1 - (it/N)]$ with $(-it/N)$ results in

$$pi > e^{-(it^2/N)}; \quad it/N \ll 1 \tag{D-13}$$

Since pi is close to one, the number of different keys in each of the T tables can be made close to mt by requiring that $mt^2 \ll N$. Furthermore, by choosing a different 64-bit to 56-bit transformation for each of the T tables it can also be assumed that the key generation processes in the different tables are statistically independent.

If the transformation selects 56 bits out of the 64 bits, there are actually (64) (63) (62) $\cdots$ (10) (9) = 64!/8! such transformations. With such a scheme accidental equivalence of two row entries in the different T matrices does not lead to an equivalence of the rest of the two rows. Thus choosing individual transformations for each table entry justifies the assumption of random generation of keys.

Now consider the problem from a different point of view. In principle, the relationship between mt and N is arbitrary. However, the ratio mt/N represents the average number of balls per cell (i.e., the average number of keys in the table generation process per possible key). If this ratio is excessively large, then there will probably be no empty cells (i.e., the attack will always work). In this case po(mt,N) is near unity. On the other hand, if mt/N tends to zero, then practically all cells must be empty (i.e., the attack will most likely not succeed). In this case po(mt,N) is near zero.

An approximation for this latter case can be derived as follows. Assuming that $mt \ll N$, one can also assume that all mt keys in the table generation process are different. The probability of finding the correct key is then mt/N. To increase the probability of success one could generate T tables. In that case the probability, q, of not finding the correct key is $q = [1 - (mt/N)]^T$ provided that all keys in the T tables are different. (The justification for treating the key generation process in the different tables as independent is that each of them uses a different transformation.)

Taking the logarithm and using the fact that $mt^2 \ll N$ one arrives at

$$\ln(q) = T \ln(1 - (mt/N)) = -mtT/N$$
$$q = e^{-mtT/N} \tag{D-14}$$

The probability of finding the correct key is $p = 1 - q$ and hence

$$p = 1 - e^{-mtT/N}; \quad mt \ll N$$

This is the same result obtained in Equation D-12. Solving for mtT/N one obtains

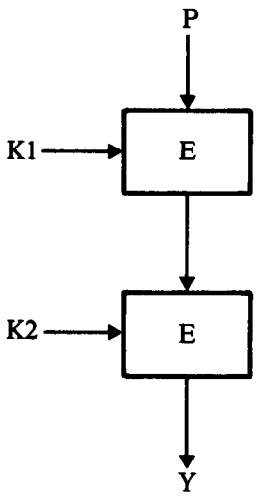
$$mtT/N = \ln(1/q)$$

The Meet-in-the-Middle Attack Against Double Encryption

Method Using an Off-Line Attack

Let $Y = E_{K2}E_{K1}(X)$ represent the double encryption scheme to be analyzed where K1 and K2 are independently chosen as shown in Figure D-13. Assuming, as before, that selected plaintext, P, can be used, a table of ciphertext $E_{ki}(P)$ is constructed for all possible keys, after which the ciphertext is sorted (see also Table D-4). To evaluate the unknown keys, the given Y (i.e., P enciphered under the unknown keys K1 and K2) is decrypted with a trial key, $D_{K2trial}(Y)$, and the value is located in the aforementioned table. The corresponding key from the selected table entry is the trial key for K1 (i.e., K1trial). It is clear that $E_{K2trial}E_{K1trial}(P) = Y$ by design. To check if K1trial is indeed equal to K1 and K2trial is indeed equal to K2, some additional pairs of plaintext and corresponding ciphertext (in the order of ten) must be used. This can be done by encrypting additional plaintext (arbitrary in this case) with the unknown keys. If these additional pairs can also be generated with K1trial and K2trial the keys are accepted as correct. Otherwise, the procedure is repeated.

This attack requires precomputation and sorting of 2^{56} ciphertexts (as in Table D-4) as was the case in the attack against a single encryption scheme. In addition, about 2^{55} trial values, $D_{K2trial}(Y)$, must be generated which are used to determine a candidate for K1 (i.e., K1trial) before the correct keys are determined. To evaluate the actual work factor also requires taking into account the sorting procedure and table lookup time to find the match of



Legend:
P = Plaintext
$Y = E_{K2}(E_{K1}(P))$ = Ciphertext

Figure D-13. Double Encryption Process

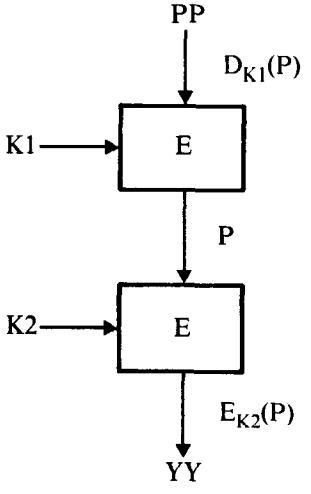
$E_{K1trial}(P)$ with $D_{K2trial}(Y)$. But, importantly, it is possible to do all this on a system that is under the control of the opponent (off-line attack).

Method Using Combinations of Off-Line and On-Line Attacks

A reduction of the exhaustion time (after ciphertext for selected plaintext is obtained) to attack a key is possible at the expense of additional precomputations. These precomputations, however, must be done on the system that contains the key or keys to be attacked (on-line attack).

Assume that an input (PP for plaintext to be doubly encrypted) is used in a double encryption scheme to yield a known bit pattern P as an intermediate result as suggested by Figure D-14. Let YY be the ciphertext obtained using PP and a method of double encryption. Since PP and YY are corresponding pairs of plaintext and ciphertext for double encryption, there is only a certain set of keys (K1 and K2) which satisfy the condition that the intermediate result is equal to a selected value, P.

To evaluate the set of these keys, a relationship between K1 and PP and between K2 and YY must be established. To do this two tables are created as shown in Table D-7. Each of the quantities PPi [Table D-7 (a)] are doubly encrypted with the (unknown) keys K1 and K2, which are to be evaluated. But since unknown keys are involved, this set (in contrast to the ones in Table D-7) must be created on the system that contains K1 and K2. In other words, an on-line attack is necessary to generate the needed quantities enumerated in Table D-8.



Legend:
PP = Chosen Plaintext
$YY = E_{K2}(E_{K1}(PP))$ = Corresponding Ciphertext

Figure D-14. Generation of Selected Plaintext Set (PP) for On-Linc Attack on Double Encryption

(a)		(b)	
k1	$D_{k1}(P) = PP1$	k1	$E_{k1}(P) = C1$
k2	$D_{k2}(P) = PP2$	k2	$E_{k2}(P) = C2$
⋮	⋮ ⋮	⋮	⋮ ⋮
kN	$D_{kN}(P) = PPN$	kN	$E_{kN}(P) = CN$

Note: The entries in the table can be obtained on a system different from the one in which the keys to be attacked are installed.

Table D-7. Relationship Between Plaintext (PP), Ciphertext (C), and Keys (k1, . . . , kN) for a Given Selected Intermediate Result (P)

Table D-8 can now be used to find a trial key K1trial determined by PP [with the aid of Table D-7 (a)] and a trial key K2trial determined by YY = C [with the aid of Table D-7 (b)] for each PP/YY pair in Table D-8. In each case the relationship $E_{K1trial}(PP) = P$, $D_{K2trial}(YY) = P$, and $E_{K1trial}E_{K2trial}(PP) = YY$ holds. To check for the correct keys (i.e., if K1trial = K1 and K2trial = K2), additional plaintext and corresponding ciphertext values must be used as discussed above.

Since Table D-8 contains a large set of PP/YY pairs, these table entries could also be used to do the actual checking. To summarize, the attack takes place as follows.

1. All N (2^{56}) keys are exhausted to generate a table [Table D-7 (a)] whose entries are {ki, $D_{ki}(P) = PPi$; i = 1, 2, . . . , N}. Since in the on-line attack these PP values are used as selected plaintext against the double encryption method, the table does not have to be sorted. (Note that one PP value after the other can be used as originally generated.)
2. All N (2^{56}) keys are exhausted to generate a table [Table D-7 (b)] whose entries are {ki, $E_{ki}(P) = Ci$; i = 1, 2, . . . , N}. The C values must

PP1	$E_{K2}E_{K1}(PP1) = YY1$
PP2	$E_{K2}E_{K1}(PP2) = YY2$
⋮	⋮ ⋮ ⋮
PPN	$E_{K2}E_{K1}(PPN) = YYN$

Note: The entries in the table can only be obtained on the system which allows operations with the unknown keys (K1 and K2) to be performed.

Table D-8. Relationship Between Selected Plaintext Set (PP) and Corresponding Ciphertext Set (YY) for Double Encryption

be sorted to locate the proper entry satisfying the condition YY = C. in step 5 below.

3. All N (2^{56}) PP values created in step 1 are now used as selected plaintext in the double encryption scheme to be attacked. The resulting ciphertext values are recorded serially in a table where entries are {PPi, $E_{k2} E_{k1}$ (PPi) = YYi; i = 1, 2, . . . , N}. Since K1 and K2 are the actual keys, the entries in this table can only be created on the system which allows operations with K1 and K2 to be performed (on-line attack).
4. From Table D-7 (a) the key associated with PP1 is identified with K1trial.
5. From Table D-7 (b) the key associated with YY1 = Cj is identified with K2trial provided that such a match occurs.
6. A sufficient number of PP/YY pairs are selected from Table D-8. (e.g., ten) to check if K1trial = K1 and K2trial = K2. This is done by doubly encrypting each of the selected PP values with K1trial and K2trial and checking for equality with the corresponding YY values. Only if all tests are positive are the keys K1trial and K2trial accepted as being correct. Otherwise, Steps 4, 5, and 6 are repeated using PP2 and YY2 next, thence PP3 and YY3, and so on; in which case, at each iteration different keys for K1trial and K2trial are selected (steps 4 and 5) and tested (step 6).

The major disadvantage of this method is the requirement of an on-line attack to create Table D-8 (step 3). As a matter of fact, it can be stated that such an attack is not a viable one since it will take an enormous amount of time to create such a table. A special purpose computer cannot be used, since the attack must take place on the system wherein the keys to be attacked have been installed. Furthermore, this activity must continually be conducted in secrecy if it is to be of any value.

The reason for this discussion is to highlight a concept. If precomputation is performed on a system under the control of an opponent (which allows the use of special purpose computers), then there exists at least the possibility of a meaningful time-memory tradeoff. On the other hand, if precomputation must also involve the use of the system to be attacked to any significant degree, then such an attack is really only of academic value and thus not of much practical interest.

Thus the most viable attack against double encryption is the one discussed above using an off-line attack. This involves creating a table of 2^{56} sorted entries and about 2^{55} encryption steps using a selected plaintext/ciphertext pair.

Attack Against Triple Encryption With Three Independent Keys

To attack the triple encryption scheme of Figure D-15 using a combination of on-line and off-line attacks, the basic ideas discussed to attack a double encryption scheme (Figure D-14) also apply. The fundamental concept re-

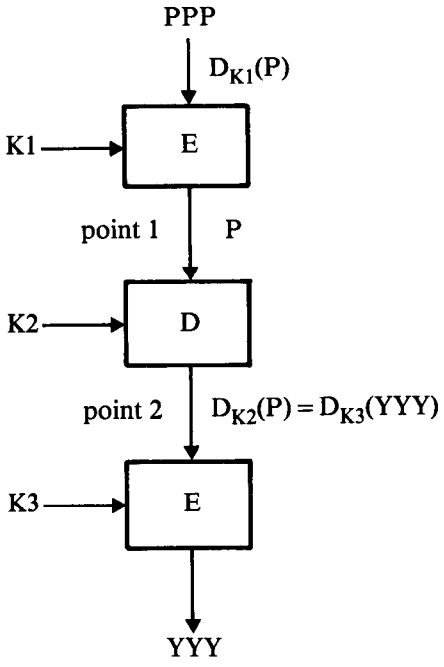


Figure D-15. Generation of Selected Plaintext Set (PPP) for On-Line Attack on Triple Encryption

quires that a preselected intermediate value P exist, as shown in Figure D-15. Going backwards, this allows the generation of a set of preselected plaintext (PPP) for any possible key (K1). This is accomplished by creating the entries in Table D-9 (see also Table D-7).

The same table can be used to determine the result at point 2 in Figure D-15 for any selected trial key K2trial. Thus only one table is needed here in contrast to the attack on double encryption (Table D-7). This is because the double encryption method of Figure D-14 employs the E-E operation whereas the triple encryption method of Figure D-15 uses the E-D-E operation. (This is the security price paid, although a small one, for the migration property.)

k1	$D_{k1}(P) = PPP1$
k2	$D_{k2}(P) = PPP2$
⋮	⋮ ⋮
kN	$D_{kN}(P) = PPPN$

Note: The entries in the table can be obtained on a system different from the one in which the keys to be obtained are installed.

Table D-9. Relationship Between Plaintext (PPP) and Keys (k1, k2, . . . , kN) for a Given Selected Intermediate Plaintext (P)

PPP1	$E_{K3}D_{K2}E_{K1}(PPP1)$	= YYY1
PPP2	$E_{K3}D_{K2}E_{K1}(PPP2)$	= YYY2
⋮	⋮	⋮
PPPN	$E_{K3}D_{K2}E_{K1}(PPPN)$	= YYYN

Note: The entries in the table can only be obtained on the system to be attacked (i.e., on the system where encryptions/decryptions under the unknown keys K1, K2, and K3 can be performed). This requirement puts the attack into the class of attacks that are of academic interest only (i.e., the attack is not viable).

Table D-10. Relationship Between Selected Plaintext Set (PPP) and Corresponding Ciphertext Set (YYY) for Triple Encryption

Each of the PPPi quantities in Table D-9 are next used as input data and triply encrypted with the (unknown) keys K1, K2, and K3. But since the unknown keys are involved, this set (in contrast to Table D-9) must be created on the system that contains K1, K2, K3. In other words, an on-line attack is necessary to generate the needed quantities enumerated in Table D-10.

This attack falls into the category described in Reference 4. Since the groundwork to understand this attack has already been described in the previous section, only a summary of the attack is given here.

1. All N (2^{56}) keys are exhausted to generate a table (Table D-9) whose entries are {ki, $D_{ki}(P)$ = PPPi; i = 1, 2, . . . , N}. Since in the on-line attack these PPP values are used as selected plaintext against the triple encryption method, the table does not have to be sorted. This establishes the set {PPPi; i = 1, 2, . . . , N} to be used in the selected plaintext on-line attack. Since the second operation is a decrypt operation in the multiple encryption scheme, the same table can be used to predict the output at point 2 (Figure D-15). This requires Table D-9 to be sorted.
2. All N (2^{56}) PPP values created in step 1 are now used as selected plaintext in the triple encryption scheme to be attacked. The resulting ciphertext values are recorded serially in a table whose entries are {PPPi, $E_{K3}D_{K2}E_{K1}(PPPi)$ = YYYi; i = 1, 2, . . . , N}. Since K1, K2, and K3 are the actual keys, the entries in this table can only be created on the system which allows operations with K1, K2, K3 to be performed (on-line attack).
3. A value for K3trial is selected and used to decrypt YYY1 to create $D_{K3trial}(YYY1)$. Using this value with the table created in step 1 (Table D-9), a match is sought and if found identifies the corresponding key as K2trial. If no match is found, another K3trial is selected

and the procedure repeated. The possible values of K3trial are thus exhausted until a value is found such that $D_{K3trial}(YYY1)$ matches some PPP value in Table D-9. If a match is found, the procedure continues with steps 4 and 5. If steps 4 and 5 fail to recover K1, K2, and K3, then step 3 is continued. That is, values of K3trial are again selected and tested until all N (2^{56}) trial values have been exhausted. At this point, the trial ciphertext YYY1 is discarded and step 3 is repeated using ciphertext YYY2 (i.e., the 2^{56} K3trial keys are again tested, if necessary, using YYY2 as the trial ciphertext). If YYY2 fails to recover the keys K1, K2, and K3, then YYY3 is used, and so on. Eventually a value of YYY, say YYYi, will be tested that allows K1, K2, and K3 to be recovered. This will occur, on the average, after 2^{55} YYY values have been tested. Note that up to this point 2^{56} K3trial keys are tested for each trial YYY value.

4. From Table D-9, the key associated with PPPi (PPP1 if YYY1 is the ciphertext selected at step 3, PPP2 if YYY2 is selected at step 3, and so on) is identified with K1trial.
5. A sufficient number of PPP/YYY pairs are selected from Table D-10 (e.g., ten) to check if K1trial = K1, K2trial = K2, and K3trial = K3. This is done by triply encrypting each of the selected PPP values with K1trial, K2trial, and K3trial. By checking for equality with the corresponding YYY values, a decision can be made to accept or reject the trial keys. Only if all tests are positive are the keys accepted as being correct. Otherwise, step 3 is continued until another match is found, whereupon steps 4 and 5 are again repeated.

Therefore the attack against three independent keys requires:

1. Generating and sorting a table of 2^{56} entries via an off-line attack (i.e., Table D-9).
2. Generation of 2^{56} related plaintext and corresponding ciphertext values via an on-line attack (i.e., Table D-10).
3. Generation of 2^{56} candidates for K3 for about 2^{55} PPP/YYY pairs, which must take place after the on-line attack is completed.
4. Test for correctness of trial keys via selected elements in Table D-10.

Thus the attack takes in the order of 2^{56} words of memory and 2^{112} operations. The described attack is presented only to illustrate exhaustion of E-D-E using three independent keys. Other more advantageous tradeoffs between memory space and computation time are very likely possible.

Attack Against Triple Encryption with Two Independent Keys

In this case K1 = K3, so that a selected PPP also determines K1 and K3. Thus trial keys for K3 do not have to be generated. Hence the 2^{56} trials to arrive at the correct K3 can be eliminated, and the attack reduces to:

1. Generate and sort a table of 2^{56} entries (i.e., Table D-9).
2. Generate 2^{56} selected plaintext and corresponding ciphertext values in an on-line attack (i.e., Table D-10).
3. Select PPP1/YYY1. The value of PPP1 is used with Table D-9 to determine k1, which is the value for K1trial. YYY1 is deciphered under K1trial and a check is made to see if $D_{K1trial}(YYY1)$ matches some value of PPP (say PPPj) in Table D-9. If a match occurs, then the value of PPPj determines a key, kj, where kj becomes the trial key K2trial. K1trial and K2trial are then tested via selected elements in Table D-10 (e.g., 10) created in step 2 (i.e., the plaintext PPP is triply encrypted using K1trial and K2trial and the result is compared for equality with the corresponding ciphertext YYY). If there is ciphertext agreement, then K1trial and K2trial are accepted as K1 and K2. Otherwise, reject K1trial and K2trial and repeat step 3 using PPP2/YYY2. If PPP2/YYY2 fails to recover K1 and K2, then step 3 is repeated using PPP3/YYY3, and so on. Eventually (after about 2^{55} trials) a PPPi/YYYi will be found that recovers K1 and K2.

Thus the attack takes in the order of 2^{56} words of memory and 2^{55} operations.

REFERENCES

1. Diffie, W. and Hellman, M. E., "New Directions in Cryptography," *IEEE Transactions on Information Theory,* **IT-22**, No. 6, 644–654 (1976).
2. Matyas, S. M. and Meyer, C. H., *Cryptographic System Using Multiple Encipherment,* Filed US Patent Office (June 1980).
3. Feller, W., *An Introduction to Probability Theory and Its Applications,* Third Edition, Wiley, New York, 1968.
4. Hellman, M. E. and Merkle, R. C., "On the Security of Multiple Encryption," *Communications of the ACM,* **24**, No. 7, 465–467 (July 1981).

APPENDIX E

Cryptographic PIN Security—Proposed ANSI Method[1]

STORAGE OF PINS

When stored, the PIN must always be encrypted. Encryption for PIN storage may be reversible or irreversible.[2] The encryption must conform to the following requirements:

1. The Data Encryption Algorithm (DEA) is used with a 56-bit secret cryptographic key composed of 56 random bits.[3]
2. PIN encryption (reversible or irreversible) must incorporate the account number (or some other card or account related data), or a portion thereof, in such a way that the verification process would provide detection of substitution of one stored value for another stored value.

When derived PINs are calculated for use in verification, any temporary or transient computer storage area used in the calculation must be cleared immediately after use.

TRANSMISSION OF PINS

Whenver the PIN is electronically transmitted, it must be provided with a high level of protection. This protection may be provided through physical means whenever the transmission medium (wire, cable, fiber-optics, etc.) or network nodes can be physically protected. Whenever the communica-

[1] The material in this appendix is based on an ANSI (American National Standards Institute) Draft Standard for PIN Identification Number Management and Security [1], which is subject to change.

[2] *Reversible encryption* is defined as a cryptographic transformation of plaintext to ciphertext such that the ciphertext can be converted back to the original plaintext. *Irreversible encryption* is defined as a cryptographic transformation of plaintext to ciphertext such that the ciphertext cannot be converted back to the original plaintext by other than exhaustive procedures.

[3] The DEA is the ANSI equivalent of the DES.

tion medium and/or network nodes cannot be physically protected (e.g., common carrier facilities, circuit switched facilities, EDP facilities), the PIN is cryptographically protected using the DEA. Additional requirements to assure that the PIN has not been replaced or modified during transmission will be set forth in another ANSI document.

In an interchange environment, the minimum level of protection provided to the PIN anywhere in the path of transmission is to be equal to the highest level required by any financial institution whose liability for a financial transaction can be affected by the misuse of the PIN.

Reversible PIN Encryption

For the purpose of security of the PIN between financial institutions, DEA shall be used in a reversible encryption mode as specified by the technique described in this section. The technique specifies the number, position, and function of bits within a 64-bit block used as input to the DEA operating in the Electronic Code Book (ECB) mode (64 bits in, 64 bits out). The 64-bit output of the DEA is transmitted (or stored in the case of file protection) in its entirety.

Cleartext PIN Block Format

The PIN is assumed to consist of 4–12 decimal digits which the customer enters, with the first digit entered referred to as digit 1 and the last referred to as digit N. The cleartext PIN format specified must be used in interchange and may be used in the terminal to acquirer segment of the network. Bit positions are specified based on numbering the bits in the DEA input block, i.e., 1 to 64 from left to right.

The cleartext PIN and customer's account number are constructed as follows:

PIN Block Construction

Bit	1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	64
	C	N	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F	

C - Control field. Currently, the only defined value is 0000 (binary) which designates the block as containing 4 to 12 four-bit decimal PIN digits representing user-entered characters. Further use of different values of this field could accommodate different character sets.

N - PIN length entered field. Four-bit binary number with permissible values of 0100 (=4) to 1100 (=12).

P - PIN digit. Four-bit field having possible values from 0000 to 1001 (the binary representation of the decimal numbers 0 through 9).

F - Fill digit. A hexadecimal digit "F" (1111 binary) which must not depend upon PIN value.

P/F - PIN digit or fill digit, as determined by PIN length entered field (N).

Account Number Block Construction

Bit	1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	64
	0	0	0	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	

A - The first 12 digits of the account number where A12 is the least significant digit (right-most) and A1 is the most significant digit (left-most). Account number check digits are excluded.

0 - A decimal pad digit-zero. The first four-bit fields of the account number block are always padded with this value.

Next the PIN and account number blocks are Exclusive-ORed and the result is transferred to the DEA input register. Any network or interchange node having access to the cleartext PIN block (i.e., during a decryption/reencryption or during PIN verification) should reject transactions having any of the following:

1. Initially a PIN format field of value other than 0.
2. A PIN length field of value less than 4 or greater than 12.
3. Any PIN digit from P1 through PN of value greater than 9 (binary 1001).

This serves as a reasonableness check on PIN encryption/decryption processing but does not serve as any indication of validity of the customer-entered PIN.

Ciphertext PIN Format

The formatted cleartext PIN block, defined above, is then encrypted using the DEA in Electronic Codebook Mode (ECB). The 64-bit output cipher block is called the reversibly encrypted PIN (or encrypted PIN). The encrypted PIN is transmitted between financial institutions as a 64-bit entity in bit-oriented communications, as eight 8-bit bytes in 8-bit transparent character oriented communications, as sixteen characters representing hexadecimal digits (0, 1, . . . , 9, A, . . . , F), or by any other data representation agreed upon by sender and receiver.

The order of transmission is from left to right. For example, in 8-bit transparent communications the first byte transmitted will contain bits 1

through 8 of the DEA output block; the eighth byte will contain bits 57 through 64 of the DEA output block.

Received Ciphertext PIN

The receiver collects the 64-bit encrypted PIN as transmitted and enters it into the decrypting DEA device which must contain the same cryptographic key that was used to encrypt it. The decrypted PIN must then be physically protected while it is being verified or while it is being reencrypted using a different cryptographic key.

A PIN that does not verify must not be accepted. The entire transaction must be suspect until a PIN is received from the acquiring source that does verify in its entirety.

REFERENCES

1. *American National Standard for Personal Identification Number Mangement and Security, Draft Standard*, American National Standards Institute, Technical Committee X9.A3, Revision 5 (November 5, 1980).

APPENDIX F

Analysis of the Number of Meaningful Messages in a Redundant Language

A language is *redundant* whenever for some value N it can be shown that not all possible sequences of N characters occur with equal probability. That English is a redundant language is easily demonstrated by examining the probabilities of individual letters: p(A) = .080, p(B) = .015, . . . , p(Z) = .001 (based on a count of individual letters in a large sample of English text, see Table 12-1). Similar nonuniform distributions are observed for digrams, trigrams, and longer phrases. For example, the phrase "Hit th_ ball" (where _ denotes a missing letter) is clearly understood to mean "Hit the ball," which demonstrates that the various choices for the missing letter (and the resulting phrases upon substitution) are not equally likely.

If the 26-character English alphabet (no blanks) is reduced to a 2-character alphabet consisting of a vowel marker ("v") and a consonant marker ("c"), where vowels = {A, E, I, O, U}, the resulting language is also redundant. Table F-1 contains vowel and consonant N-grams for English ($N \leqslant 5$).

An approximation for the number of meaningful messages of N characters can be obtained by using a discrete Markov process to simulate the creation of text. A discrete Markov process consists of a finite number of states, q1, q2, . . . , qn, and a set of transition probabilities $\{p_i(j)\}$, where $p_i(j)$ represents the probability that the system will go to state qj given that it is in the state qi. If each state transition produces an output symbol, the Markov process can be treated as an information source that produces a stream of output characters.

For an nth order Markov process, the probability that a given character will be the next one depends on the previous n output characters, but not on characters preceeding those. Conforming to this model, a Jth order Markov approximation for message probability is given by

$$p(a_1 a_2 \ldots a_N) \simeq p(a_1 a_2 \ldots a_J) \prod_{i=J+1}^{N} p_{B_i}(a_i) \qquad \text{(F-1)}$$

where

$p(a_1 a_2 \ldots a_N)$ is the probability of message $a_1 a_2 \ldots a_N$,

5-gram		4-gram		3-gram		2-gram		1-gram	
VVVVV	7								
VVVVC	165	VVVV	172						
VVVCV	2476								
VVVCC	5417	VVVC	7893	VVV	8065				
VVCVV	8500								
VVCVC	65543	VVCV	74043						
VVCCV	74499								
VVCCC	57585	VVCC	132084	VVC	206127	VV	214192		
VCVVV	3634								
VCVVC	89792	VCVV	93426						
VCVCV	200227								
VCVCC	260828	VCVC	461055	VCV	554481				
VCCVV	65403								
VCCVC	387921	VCCV	453314						
VCCCV	220012								
VCCCC	82956	VCCC	302968	VCC	756292	VC	1310773	V	1524965
CVVVV	167								
CVVVC	7729	CVVV	7896						
CVVCV	71580								
CVVCC	126698	CVVC	198278	CVV	206174				
CVCVV	84927								
CVCVC	395481	CVCV	480408						
CVCCV	378868								
CVCCC	245362	CVCC	624230	CVC	1104638	CV	1310812		
CCVVV	4266								
CCVVC	108434	CCVV	112700						
CCVCV	280141								
CCVCC	363438	CCVC	643579	CCV	756279				
CCCVV	47306								
CCCVC	255599	CCCV	302905						
CCCCV	82943								
CCCCC	22096	CCCC	105039	CCC	407944	CC	1164223	C	2475035

Based on a sample of 8000 excerpts of 504 letters taken from the Brown University Corpus of Present-Day American English [1]. Vowels = { A, E, I, O, U }.

Table F-1. Vowel-Consonant N-gram Frequencies in 4 Million Characters of English Text ($N \leqslant 5$)

$p_{B_i}(a_i)$ is the conditional probability that character a_i follows block B_i, and

$B_i = a_{i-J}a_{i-J+1} \ldots a_{i-1}$ is a block of J characters.

The symbol $\simeq$ denotes approximately equal to.

In a zero-order approximation, output characters are independent. Hence, for a zero-order approximation, message probability can be expressed by

$$p(a_1 a_2 \ldots a_N) \simeq \prod_{i=1}^{N} p(a_i) \qquad \text{(F-2)}$$

where $p(a_i)$ is the probability that character a_i appears next. For example, with the values for p(a) given in Table 12-1, the probability for the word CIPHER is approximated by

$$\begin{aligned} p(\text{CIPHER}) &\simeq p(C)p(I)p(P)p(H)p(E)p(R) \\ &\simeq (.031)(.073)(.020)(.055)(.125)(.061) \\ &\simeq 1.90 \times 10^{-8} \end{aligned}$$

Suppose that the word CIPHER is reduced to its corresponding vowel-consonant pattern, cvccvc. A first-order approximation for the probability of cvccvc is given by

$$p(\text{cvccvc}) \simeq p(c)p_c(v)p_v(c)p_c(c)p_c(v)p_v(c)$$

Using the N-gram frequencies in Table F-1, it follows that

$$\begin{aligned} p(c) &\simeq 2{,}475{,}035/4{,}000{,}000 = .619 \\ p_c(v) &\simeq 1{,}310{,}812/2{,}475{,}035 = .530 \\ p_v(c) &\simeq 1{,}310{,}773/1{,}524{,}965 = .860 \\ p_c(c) &\simeq 1{,}164{,}223/2{,}475{,}035 = .470 \end{aligned}$$

and therefore p(cvccvc) is computed as

$$\begin{aligned} p(\text{cvccvc}) &\simeq p(c)p_c(v)p_v(c)p_c(c)p_c(v)p_v(c) \\ &\simeq (.619)(.530)(.860)(.470)(.530)(.860) = .060 \end{aligned}$$

An approximation for the number of meaningful messages in message space $\underline{X}$ can be obtained using the zero-order approximation of message probability given by Equation F-2. When message length (N) is very large, each message will contain about Np1 occurrences of the first character, Np2 occurrences of the second character, and so on. Hence for very large N, most messages will have roughly the same probability, p, i.e.,

$$p \simeq p1^{Np1}p2^{Np2} \ldots pn^{Npn} \qquad \text{(F-3)}$$

where n is the number of different characters.

If s is the number of different sequences with probability p, then s is approximated by

$$\begin{aligned} s &\simeq 1/p \\ &\simeq p1^{-Np1}p2^{-Np2} \ldots pn^{-Npn} \\ &\simeq \prod_{i=1}^{n} 1/pi^{Npi} \end{aligned}$$

$$\simeq 2^{\left[\log_2 \prod_{i=1}^{n} p_i^{-Np_i}\right]}$$

$$\simeq 2^{\left[-N \sum_{i=1}^{n} p_i \log_2 p_i\right]}$$

$$\simeq 2^{NG_1}$$

where

$$G_1 = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{F-4}$$

is called the *entropy per character* for the message source.

Using the values for p(a) in Table 12-1, the value for G_1 is computed as 4.17. Moreover, using this value for G_1 allows the zero-order approximation of s, given by Equation F-4, to be written as

$$s = 2^{N4.17}$$

When a higher-order approximation for message probability is used (see Equation F-1), it is possible to obtain a correspondingly higher-order approximation for s. Before this result can be derived, a few terms must be defined.

Let U be a discrete probability space in which the elementary events {u1}, {u2}, . . . , {un} have probabilities p(u1), p(u2), . . . , p(un). The *entropy* of U is defined as

$$H(U) = -\sum_{u} p(u) \log_2 p(u) \tag{F-5}$$

(The notation, $\sum_{u}$, means that the summation is over all elements u in the set U.)

H is an information theoretic measure of uncertainty which can vary between the limits 0 and $\log_2 n$, where n is the total number of elements in the probability space U. It can be shown that H(U) = 0 if and only if there is a single u in U such that p(u) = 1. Consequently, when there is a single u in U such that p(u) = 1, so that no uncertainty exists over which event in U will occur, the measure H(U) has the value zero. But H(U) = 0 is interpreted to mean that there is no uncertainty over which event in U will occur. Moreover, it can be shown that H(U) = $\log_2 n$ if and only if each of the n elements in U is equally probable. Consequently, when there is maximum uncertainty over which event in U will occur, the measure H(U) is maximized.

When the entropy measure H is applied to the message (plaintext), transformation (key), and cryptogram (ciphertext) spaces, the following interpretation is obtained. H(X) and H(K) represent the uncertainty over which message and key were used during encipherment; H(Y) represents the uncertainty over which cryptogram was produced.

Let G_N represent the *entropy per character of blocks of N characters.* G_N is expressed as

$$G_N = -\frac{1}{N}\sum_{B} p(B)\log_2 p(B) \tag{F-6}$$

where

B is a block of N characters, and

p(B) is the probability that B is produced by the message source.

Let the union of U and V be a joint discrete probability space. The conditional entropy of U given v, an element of V, is defined as

$$H(U|v) = -\sum_{u} p(u|v)\log_2 p(u|v) \tag{F-7}$$

where p(u|v) is the conditional probability of u given v. The *average conditional entropy of* U *given* V, or the *equivocation of* U *given* V is defined as

$$H(U|V) = \sum_{v} p(v)H(U|v) \tag{F-8}$$

Conditional entropy and equivocation have the following meanings when applied to enciphering systems. $H(\underline{X}|y)$ measures the uncertainty regarding which message in $\underline{X}$ was enciphered to produce cryptogram y. $H(\underline{X}|\underline{Y})$, on the other hand, measures this same uncertainty, except that it is averaged over all possible cryptograms. $H(\underline{X}|\underline{Y}) = 0$ may be interpreted to mean that regardless of the particular cryptogram y there is no uncertainty regarding which message produced it. Moreover, $H(\underline{X}|\underline{Y}) = 0$ implies that $H(\underline{X}|y) = 0$ for each cryptogram in $\underline{Y}$.

The conditional entropy of the message source is a measure of the uncertainty regarding the nature of the Nth character given that the previous N − 1 characters are known. As N increases, this measure accounts for more and more of the interdependencies between characters.

Let

$$F_N = -\sum_{B,\,a} p(B, a)\log_2 p_B(a) \tag{F-9}$$

where

B is a block of N − 1 characters,

a is a single character following B,

p(B, a) is the probability of N-gram (B, a), and

$p_B(a)$ is the conditional probability that character a follows block B and is equal to p(B, a)/p(B).

From Equations F-6 and F-9, it follows that

$$F_N = NG_N - (N-1)G_{N-1} \tag{F-10}$$

G_N and F_N are two different measures which allow the entropy per character of a message source to be evaluated. It can be shown that both G_N and F_N are monotonically decreasing with N and bounded below by 0. Moreover, according to [2, Theorem 5], the limit of G_N, as N approaches infinity (∞), exists and is equal to R, called the *rate of the language,*

$$\lim_{N \to \infty} G_N = R \tag{F-11}$$

and, moreover, according to [2, Theorem 6]

$$F_N \leqslant G_N \tag{F-12a}$$

and

$$\lim_{N \to \infty} F_N = R \tag{F-12b}$$

From the asymptotic equipartition property [2, Appendix 3], it can be shown that when a Jth-order approximation to message probability is used and $N \gg J$, then most sequences produced by the message source will have the same probability of occurrence, p, and the number of different sequences, s, with probability p is approximated by

$$s = 1/p = 2^{NF_{J+1}} \tag{F-13}$$

(The notation $\gg$ means much greater than.) According to the asymptotic equipartition property, for all practical purposes, the possible messages of N characters can be divided into two groups: one group of high and fairly uniform probability, the second group of negligibly small total probability. The high probability group consists of those messages that are intelligible or meaningful. It contains approximately $2^{NF_{J+1}}$ sequences. The low probability group contains those messages that are meaningless.

Actually, the Equation F-13 demonstrates that s is a function of message length, N, and the order of the approximation of message probability, J. This relationship can be expressed notationally by defining $s_{N,J}$ as

$$s_{N,J} = 2^{NF_{J+1}} \tag{F-14}$$

The following interpretation of $s_{N,J}$ can now be made with respect to cryptanalysis. The situation in which F_{J+1} is used to compute s (i.e., when a Jth-order approximation of message probability is employed) is comparable to the situation where the analyst has no more than (J + 1)-grams, or (J + 1)-order statistics, available to attack the system. Consequently, if digrams are

used to attack an enciphering system, then the number of meaningful messages that the analyst must cope with ought to be 2^{NF_2}. The cryptographer or designer of a cryptographic system, on the other hand, is interested in knowing what is the minimum number of meaningful messages that the analyst must always cope with. This number applies, of course, when the analyst is able to employ high-order statistics to attack the system. From Equations F-12b and F-14, it follows that

$$\lim_{J \to \infty} s_{N,J} = 2^{NR}; N \geqslant J \tag{F-15}$$

where

$$s_{N,0} \geqslant s_{N,1} \geqslant \cdots \geqslant s_{N,J} \geqslant s_{N,J+1} \geqslant \cdots \geqslant 2^{NR}$$

for all $N > 0$. It follows from this discussion, then, that meaningful messages are actually those which the decision procedure admits as being meaningful, whatever the sequences may be. The disparity between what a computer procedure admits as meaningful when, for example, digram statistics are used, and what the human recognizes as meaningful, may be substantial.

If $\underline{X}$ is the set of all N-letter messages, it follows from Equation F-5 that

$$H(\underline{X}) = -\sum_{x} p(x)\log_2 p(x)$$

where p(x) is the probability of message x (see Equation F-5). But, when the analyst uses (J + 1)-gram statistics to attack the system, p(x) must be computed via Equation F-1, and it follows that

$$p(x) = p(a_1 a_2 \ldots a_N)$$

$$\simeq p(a_1 a_2 \ldots a_J) \prod_{i=J+1}^{N} p_{B_i}(a_i)$$

(see Equation F-1).

Hence, when Equation F-1 is used together with Equation F-5, it follows that

$$H(\underline{X}) \simeq NF_{J+1}; N \gg J \tag{F-16}$$

and so, from Equation F-14, it follows that

$$s_{N,J} \simeq 2^{H(\underline{X})}; N \gg J \tag{F-17}$$

The important aspect of this result is that $H(\underline{X})$ depends on the order of the Markov approximation for message probability.

For small values of N ($N \leqslant 3$), G_N and F_N can be computed directly from Equations F-6 and F-9 using N-gram frequencies (see Table F-2 for com-

Alphabet Type	F_0	F_1	F_2	F_3
	(Bits per Character)			
26-letter[1]	4.70	4.17	3.62	3.22
26-letter[2]	4.70	4.14	3.56	3.30
27-letter[3]	4.76	4.03	3.32	3.10

[1]Computed from a sample of 1 million N-grams.
[2]Obtained from Reference 3.
[3]Space (blank) is included as an additional letter.
F_0 is defined as $\log_2$(Alphabet Size).

Table F-2. Computed Values of F_N for English ($N \leqslant 3$)

puted values of F_N). However, N-gram frequencies could not be used to evaluate R, as illustrated by the following example. Suppose that R is equal to 1 bit per character (R = 1.0 bpc). This means that, on the average, 100 bits would be required, to represent a block of 100 characters, and there would be roughly 2^{100} meaningful English sequences of 100 characters (2^{100} is approximately equal to

1,000,000,000,000,000,000,000,000,000,000,000

or 10^{30}). Since most of these (10^{30}) sequences have never even been written down, it is impossible to measure their relative frequencies or estimate their respective probabilities.

In order to determine the value or R, one must be able to evaluate F_n for $N > 3$. One such way, described in reference 3, is based on the fact that anyone who can read and write a language possesses an enormous built-in knowledge of the statistics of that language. This can be demonstrated by measuring a person's ability to predict the Nth character in a message after seeing the preceeding N − 1 characters. Table F-2 gives upper and lower bounds on F_N that were obtained in this manner from an actual experiment involving a human subject [3,4]. In this experiment, a 27-letter English alphabet was employed (26 letters plus blank). Since the blank is almost completely redundant when sequences of one or more words are involved, the values of F_N in the 27-letter case will be 4.5/5.5 of F_N for the 26-letter alphabet when N is reasonably large [3]. Thus, F_N for a 26-letter alphabet can be obtained from F_N for a 27-letter alphabet via the relation

$$F_n(26 \text{ letters}) = (5.5/4.5)F_N(27 \text{ letters}) \qquad \text{(F-18)}$$

According to Shannon [3] (Table F-3), the value of R for a 27 letter English alphabet is approximately 1.0 bits per character. From Equation F-18, it can be seen that the value of R for a 26 letter English alphabet thus becomes about 1.22 bits per character.

1	2		3	
N (Characters)	F_N Upper Bound (Bits per Character)	F_N Lower Bound	F_N Upper Bound (Bits per Character)	F_N Lower Bound
1	4.03	3.19	4.72	3.95
2	3.42	2.50	3.81	3.19
3	3.0	2.1	3.34	2.44
4	2.6	1.7		
5	2.7	1.7	2.9	2.2
6	2.2	1.3		
7	2.8	1.8		
8	1.8	1.0		
9	1.9	1.0	2.5	1.9
10	2.1	1.0		
11	2.2	1.3		
12	2.3	1.3		
13	2.1	1.2		
14	1.7	0.9		
15	2.1	1.2		
17			2.4	1.3
33			1.9	1.1
65			2.0	1.2
100	1.3	0.6		
129			1.9	1.1

[1]Based on a 27 letter English alphabet (includes blank).
[2]Obtained from Reference 3.
[3]Obtained from Reference 4.

Table F-3. Bounds on F_N for a 27-Letter English Alphabet (Human Prediction Experiment in which the Preceding N-1 Characters were Known)

When the 26-letter English alphabet is reduced to a 2-character alphabet consisting of vowels and consonants, Equations F-6 and F-9 can be used to compute values of G_N and F_N for larger values of N than can be computed when a 26-letter alphabet is considered. Table F-4 contains computed values of G_N and F_N for $N \leqslant 10$ that were obtained from an N-gram analysis using 4 million characters of English text. It can be seen that F_N rapidly approaches a value of R = .80 bits per character.

A vowel-consonant alphabet also permits the accuracy of the approximation for $S_{N,J}$ (given by Equation F-14) to be evaluated in situations where N and J are small. Let J = 1 (use a first-order approximation of message probability) and let N take on values 2, 3, . . . , 15. For each value of N, compute

N (Characters)	G_N (Bits per Character)	F_N
0	1.000	1.000
1	.959	.959
2	.900	.840
3	.871	.812
4	.855	.807
5	.845	.806
6	.838	.805
7	.834	.805
8	.830	.805
9	.827	.804
10	.825	.804

G_N is computed from 4 million English N-grams using Equation F-6.
F_N is computed from the relation $F_N = N(G_N) - (N\text{-}1)(G_{N-1})$.

Table F-4. Computed Values of G_N and F_N for English Vowel-and-Consonant N-grams

N	2^N	$2^{N(F_2)}$	$p(2^{N(F_2)}$ most probable)
2	4	3.2	0.96
3	8	5.7	0.93
4	16	10.3	0.93
5	32	18.4	0.92
6	64	33.0	0.91
7	128	59.0	0.90
8	256	110	0.90
9	512	189	0.90
10	1024	339	0.89
11	2048	607	0.89
12	4096	1086	0.88
13	8192	1945	0.88
14	16384	3482	0.87
15	32768	6434	0.87

$p(2^{N(F_{J+1})}$ most probable) represents the sum of the probabilities of the $2^{N(F_{J+1})}$ most probable messages out of the 2^N total messages, the probability of which is approximated by Equation F-1 under the condition that $J = 1$.
Vowels = { A, E, I, O, U }.

Table F-5. Accuracy of Approximation ($s_{N,J} = 2^{N(F_{J+1})}$) Using a Vowel-Consonant English Alphabet.

the probability for each of the 2^N possible N character messages using Equation F-1. This list of 2^N messages is then sorted in descending sequence according to the computed value of message probability (most probable message to least probable message). Using this sorted list, obtain the sum of the probabilities for the first $2^{NF_J + 1}$ messages. This value is denoted by $p[2^{NF_J + 1}$ most probable$]$. The amount that $p[2^{NF_J + 1}$ most probable$]$ differs from 1 is a measure of the accuracy of the approximation. Table F-5 contains the results of this analysis.

REFERENCES

1. Francis, W., *A Standard Sample of Present-Day Edited American English for Use with Digital Computers,* Linguistics Department, Brown University, Providence, RI, 1964.
2. Shannon, C. E., "A Mathematical Theory of Communication," *Bell System Technical Journal,* 27, Part I 479–523, Part II 623–656 (1948).
3. Shannon, C. E., "Predictions of entropy in printed English," *Bell System Technical Journal,* **30**, 50–64 (1951).
4. Burton, N. G., and Licklider, J. C. R., "Long-Range Constraints in The Statistical Structure of Printed English," *American Journal of Psychology,* **68**, 650–653 (1955).

APPENDIX G

Unicity Distance Computations

In the following discussion of unicity distance computations, it is assumed that only ciphertext is available for analysis. It is further assumed that the reader is familiar with the information measures and unicity distance results given in An Expansion of Shannon's Approach Using Information Theory, Chapter 12.

TRANSPOSITION

Let $x = a_1 a_2 \ldots a_N$ be a message of N characters, which has been segmented into j blocks of T characters:

$$x = B_1 B_2 \ldots B_j$$

where

$$Bi = a_1(i), a_2(i), \ldots, a_T(i)$$

In a transposition cipher, encipherment is performed by rearranging the characters in each block of x according to a permutation function f:

$$f(x) = f(B_1), f(B_2), \ldots, f(B_j)$$
$$= C_1, C_2, \ldots, C_j = y$$

Decipherment is performed using the corresponding inverse permutation function f^{-1}:

$$f^{-1}(y) = f^{-1}(C_1), f^{-1}(C_2), \ldots, f^{-1}(C_j) = x$$

For example, suppose

$$f = \begin{array}{|cccccccc|} \hline 7 & 4 & 5 & 1 & 8 & 3 & 2 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \end{array}$$

is a permutation of the integers 1, 2, . . . , 8, which is interpreted to mean that the character in position 7 is written in position 1, the character in position 4 is written in position 2, and so on. Then encipherment of the message x = "data encryption standard" with spaces removed and appropriate fill characters added, results in cryptogram

$$\begin{aligned} y &= f(x) \\ &= f(\text{dataencr}), f(\text{yptionst}), f(\text{andardxx}) \\ &= \text{caedrtan} \quad \text{sioyttpn} \quad \text{xaraxdnd} \end{aligned}$$

In this case, decipherment would be carried out with the inverse permutation

$$f^{-1} = \begin{array}{|cccccccc|} \hline 4 & 7 & 6 & 2 & 3 & 8 & 1 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \end{array}$$

Since there are T characters in each block, the number of possible permutation functions is (T)(T − 1) . . . (1) = T!.[1] Effectively, encipherment cancels or breaks down the usual 2-gram, 3-gram, and so on, language statistics between adjacent and neighboring characters, and spreads them over the entire block. Except for interblock dependencies, which can be ignored for moderate and large values of T, only the frequencies of individual characters remain undisturbed or unaltered.

The unicity distance for a transposition cipher is the value of N for which

$$H(\underline{K}) - H(\underline{Y}) + H(\underline{X}) = 0$$

where

$$\begin{aligned} H(\underline{K}) &= \log_2(T!) \\ H(\underline{Y}) &= NF'_{J+1} \\ H(\underline{X}) &= NF_{J+1} \end{aligned}$$

(see Equation 12-8). F'_{J+1} is a measure of the entropy per character of the cryptogram space, $\underline{Y}$ (see Equation F-9). Substituting and solving for N(ud), one obtains:

$$ud = \log_2(T!)/(F'_{J+1} - F_{J+1}) \qquad \text{(G-1)}$$

Consider a transposition cipher on English in which 1-gram language

[1] Actually, if one omits the arrangement in which the block is unchanged, there are T! − 1 permutation functions. However, for moderate values of T, the difference between T! and T! − 1 is negligible.

statistics are used to attack the cipher. In that case, J = 0 and the entropy per character in the cryptogram space is

$$F'_1 = -\sum_{i=1}^{26} p(b_i)\log_2 p(b_i)$$

where $b_1, b_2, \ldots, b_{26}$ denotes the cipher alphabet, $a_1, a_2, \ldots, a_{26}$ denotes the plain alphabet, and $b_1 = a_1$, $b_2 = a_2$, $\ldots$ $b_{26} = a_{26}$. Since there are about 8.0% As, 1.5% Bs, 3.1% Cs, and so on, in the messages being enciphered, there will also be about 8.0% As, 1.5% Bs, 3.1% Cs, and so on, in the produced cryptograms. In that case, $p(b_i) = p(a_i)$, $F'_1 = F_1 = 4.17$, and ud equals infinity. This shows that 1-grams cannot be used to break a transposition cipher.

For large T, the usual 2-gram, 3-gram, and so on, language statistics in the message space are no longer present in the cryptogram space. This implies that F'_2, F'_3, and so on, are about equal to F'_1 (4.17). Thus Equation G-1 can be written as

$$ud = \log_2(T!)/(4.17 - F_{J+1}); \quad \text{for large T} \qquad \text{(G-2)}$$

In situations where J and T are of comparable magnitudes, values for F'_2 and F'_3 can be evaluated as follows. A large sample of English text is enciphered with each of the T! permutation functions. The resulting T! cryptograms are combined to form a single sample, which can then be used to determine the frequency of each (J + 1)-gram. The (J + 1)-grams are then used with Equation F-9 to compute F'_{J+1}.

Unicity distances for transposition on English (26 character alphabet) are given in Table G-1. These values were computed from Equation G-2 using T and J as variables.

J	F_{J+1}	T (Characters per Block)				
		10	20	50	100	1000
0	4.17	∞	∞	∞	∞	∞
1	3.62	40	111	390	954	15502
2	3.22		65	226	553	9473
6	2.81			150	386	6269
14	2.02				244	3966
99	1.22					2890

∞ denotes infinity. For each value of J, (J + 1)-grams are presumed to be used to attack the cipher. F_1, F_2, and F_3 were obtained from Table F-2. F_7, F_{15}, and F_{100} were computed from Equation F-18 using an average of the upper and lower bounds on F_N for a 27-character alphabet (see column 1, Table F-3). Unicity distance was computed using Equation G-2.

Table G-1. Unicity Distance in Characters for Transposition on English (26 Letter Alphabet)

SIMPLE SUBSTITUTION

In Chapter 12, the unicity distance for simple substitution on English was computed with a random cipher in which the approximation $\{H(\underline{Y}) \simeq \log_2 r = N4.70\}$ was made. A more accurate value of $H(\underline{Y})$, derived below, shows that for small N, the approximation $\{H(\underline{Y}) \simeq \log_2 r\}$ is fairly good.

A cipher is *pure* if the keys in $\underline{K}$ are equally likely and, for every k1, k2, and k3, in $\underline{K}$, there is a k4 such that

$$E_{k1}(D_{k2}(E_{k3}(x))) = E_{k4}(x) \qquad \text{for all } x \text{ in } \underline{X}$$

(see reference 1). For example, simple substitution is a pure cipher if its keys are equally likely. In a pure cipher, the messages and cryptograms can be divided into sets of residue classes $C_1, C_2, \ldots, C_s$, and $C'_1, C'_2, \ldots, C'_s$, respectively, such that

1. Each message and cryptogram is an element of one and only one residue class.
2. Enciphering any message in C_i with any key produces a cryptogram in C'_i. Deciphering any cryptogram in C'_i with any key leads to a message in C_i.
3. The number of messages in C_i, say t_i, is equal to the number of cryptograms in C'_i and is a divisor of n the number of keys.
4. Each message in C_i can be enciphered into each cryptogram in C'_i by exactly n/t_i different keys. The same is true for decipherment.

In a simple substitution cipher, the residue class corresponding to a given cryptogram y is the set of all cryptograms that are obtained from y via the operation $E_{ki}(D_{kj}(y))$, where ki and kj vary over each key in $\underline{K}$. Thus all the cryptograms in a residue class have the same pattern of repeated letters. For example, if cryptogram abaabc is in residue class C'_i, then so are the cryptograms babbax, abaabd, jqjjqt, mammae, and pippin. Therefore, each y in C'_1 is equally likely and

$$p(y) = p(C_i)/t_i; \qquad \text{for each } y \text{ in } C'_i \tag{G-3}$$

where $p(C_i)$ is the probability of residue class C_i and t_i is the number of different cryptograms in C'_i. Hence, $H(\underline{Y})$ is computed, as described in reference 1, by

$$\begin{aligned} H(\underline{Y}) &= -\sum_i t_i(p(C_i)/t_i)\log_2(p(C_i)/t_i) \\ &= -\sum_i p(C_i)\log_2(p(C_i)/t_i) \end{aligned} \tag{G-4}$$

Values of $H(\underline{Y})/N$ ($N = 1, 2, \ldots, 8$) were computed for simple substitution on English using Equation G-4 (Table G-2). Values for $p(C_i)$ were obtained

Ciphertext Length N	Number of Residue Classes	H(Y)/N N-gram Sample Size 50,000	100,000	500,000	1,000,000
1	1	—	—	—	4.700
2	2	—	—	—	4.700
3	5	—	—	—	4.699
4	15	4.695	4.696	4.696	—
5	52	4.690	4.692	4.692	—
6	203	4.684	4.686	4.686	—
7	877	4.677	4.679	4.680	—
8	4140	4.666	4.670	4.673	—

$H(\underline{Y})/N = 4.70$ for the Random Cipher.

Table G-2. Computed Values of $H(\underline{Y})/N$ for Simple Substitution on English (26 letter alphabet)

from samples of English text by counting the number of N-grams in each residue class C_i and dividing the result by the sample size. For values of N greater than 8, the number of residue classes rapidly becomes large. In that case, it is not possible to obtain accurate estimates for $p(C_i)$ unless very large sample sizes are used.

The approximation $\{H(\underline{Y}) \simeq \log_2 r = N4.70\}$ is quite good for small values of N. This could account for the close agreement between computed unicity distance and the observed number of characters needed to break simple substitution on English when high-order language statistics are used in the cryptanalysis, since the observed value of N = 25 is still reasonably small. However, for large values of N, the approximation $\{H(\underline{Y}) \simeq \log_2 r = N4.70\}$ no longer holds. This could account for the disparity between computed unicity distance and the observed number of characters needed to break

J	F_{J+1}	Expected Number of Different Letters	Number of Keys	Unicity Distance (ud)	Approximate Value of N to Break the Cipher (Observed Values)
0	4.17	26	26!	167.0	Several Thousand (Fig. 12-5)
1	3.62	24	26!/2!	81.0	About 500 (Fig. 12-5)
2	3.22	23	26!/3!	58.0	About 100 (Ref. 2)
14	2.02	14	26!/12!	22.2	About 25 (Ref. 3)

Unicity distance is computed from Equation 11-8. It is assumed that $H(\underline{Y}) \simeq N4.70$.
For an explanation of J and F_{J+1}, see Table G-1.

Table G-3. Unicity Distance in Characters for Simple Substitution on English (16 Letter Alphabet)

simple substitution on English when low-order language statistics are used in the cryptanalysis, since the observed values of N are much larger (Table G-3).

When unicity distance is computed with Equation 12-8 using $F_{J+1} = 4.17$ (J = 1), a value of ud = 167 characters is obtained. However, observed results indicate that several thousand characters of ciphertext are needed to break a simple substitution cipher when only single letter probabilities are used (see Figure 12-5).

HOMOPHONIC SUBSTITUTION

In a *homophonic substitution* cipher (sometimes called substitution with variants, substitution with multiple substitutes, or multiple substitution), each character in the plain alphabet $a_1, a_2, \ldots, a_t$, has a corresponding set of unique cipher characters, or substitutes, $S_1, S_2, \ldots, S_t$, such that

1. The cipher characters in each set, S_i, are different from those in any other set, S_j.
2. L_i denotes the number of characters in S_i ($L_i = |S_i|$).
3. L denotes the total number of characters in the cipher alphabet ($L = L_1 + L_2 + \ldots + L_t$).

Simple substitution is therefore just a special case of homophonic substitution in which $L_1 = L_2 = \ldots = L_t = 1$. In all other cases, $\log_2 L$ (the bits needed to represent a character in the cipher alphabet) is greater than $\log_2 t$ (the bits needed to represent a character in the plain alphabet), thus indicating that homophonic substitution is an expansion cipher.

Encipherment is accomplished by replacing each character in the message with one of its allowed substitutes: plain character a_1 is replaced by an element in S_1, a_2 is replaced by an element in S_2, and so on. (It is assumed that substitutes are selected randomly, i.e., the probability of selecting any particular element in S_i is $1/L_i$.) Decipherment is the reverse of this process: cipher characters in S_1 are replaced by a_1, cipher characters in S_2 are replaced by a_2, and so on.

Consider an example in which the substitutes for each character in the plain alphabet are as follows:

e: c u 7	r: 1 9	m: 2	v: h
t: e r w	h: b	f: s	k: o
a: p 0	l: k	p: q	x: a
o: 3 5	d: d	g: x	j: i
i: f 8	c: g	w: t	q: z
n: j v	u: 6	y: m	z: 4
s: n #		b: y	

The message "data encryption standard," with spaces removed, can be en-

ciphered in 1 X 2 X 3 X . . . X 2 X 1 = 331,776 ways. One character can be substituted for d, two characters can be substituted for a, three characters for "t," and so on, as shown below:

plaintext:	d	a	t	a	e	n	c	r	y	p	t	i	o	n	s	t	a	n	d	a	r	d
1st choice:	d	p	e	p	c	j	g	l	m	q	e	f	3	j	n	e	p	j	d	p	l	d
2nd choice:		0	r	0	u	v		9			r	8	5	v	#	r	0	v		0	9	
3rd choice:			w		7						w					w						

In a homophonic substitution cipher, encipherment of message xj with key ki defines a set of candidate cryptograms, Yij:

$$E_{ki}(xj) = Yij$$

where, as part of the encipherment process, the communicant selects or generates (usually randomly and on a character-by-character basis) one of the cryptograms in Yij. However, only one message is recovered upon decipherment:

$$D_{ki}(yij) = xj; \quad \text{for each } yij \text{ in } Yij$$

Since ki determines a set of cryptograms instead of one cryptogram, homophonic substitution does not satisfy our definition for a cipher (see Chapter 12, A Cipher with Message and Key Probabilities). The problem is easily avoided if one assumes that the enciphering algorithm E has a fixed rule for deciding which of the substitutes should be used to encipher each plaintext letter. For example, substitutes could be selected on a rotating basis, on the basis of the plaintext letter's position in the message, on the basis of the message's context (surrounding letters), and so forth. However, this has not been done here, since it would introduce an additional degree of complexity that can be avoided in the present discussion.

Since each ki defines a set of cryptograms, the assumptions leading to Equation 12-8 are not satisfied (i.e., Equation 12-8 cannot be used to calculate unicity distance). Hence, a new equation for computing unicity distance is derived.

From the general relation

$$H(U, V, W) = H(U|V, W) + H(V, W)$$

(see Equation 12-7i) it follows, with an appropriate change of variables, that

$$H(\underline{X}, \underline{K}, \underline{Y}) = H(\underline{X}|\underline{K}, \underline{Y}) + H(\underline{K}, \underline{Y})$$

and

$$H(\underline{Y}, \underline{K}, \underline{X}) = H(\underline{Y}|\underline{K}, \underline{X}) + H(\underline{K}, \underline{X})$$

Hence, it follows that

$$H(\underline{K}, \underline{Y}) - H(\underline{K}, \underline{X}) = H(\underline{Y}|\underline{K}, \underline{X}) - H(\underline{X}|\underline{K}, \underline{Y})$$

In a homophonic substitution cipher, since $x = D_k(y)$ (i.e., a knowledge of k and y permits x to be recovered), it follows that

$$H(\underline{X}|\underline{K}, \underline{Y}) = 0$$

and consequently that

$$H(\underline{K}, \underline{Y}) = H(\underline{Y}|\underline{K}, \underline{X}) + H(\underline{K}, \underline{X})$$

(Note that $H(\underline{Y}|\underline{K}, \underline{X}) > 0$, since $E_k(x)$ defines a set instead of only one element.) But, by Equation 12-7h, $H(\underline{K}, \underline{Y})$ can be rewritten as

$$H(\underline{K}, \underline{Y}) = H(\underline{K}|\underline{Y}) + H(\underline{Y})$$

Moreover, since messages and keys are selected independently, it follows from Equations 12-7h and 12-7j that

$$H(\underline{K}, \underline{X}) = H(\underline{K}) + H(\underline{X})$$

A general equation for $H(\underline{K}|\underline{Y})$ is thus obtained:

$$H(\underline{K}|\underline{Y}) = H(\underline{K}) - H(\underline{Y}) + H(\underline{Y}|\underline{K}, \underline{X}) + H(\underline{X})$$

The unicity distance of a homophonic substitution cipher in which only ciphertext is available for analysis is the value of N (N = cryptogram length in characters) for which

$$H(\underline{K}) - H(\underline{Y}) + H(\underline{Y}|\underline{K}, \underline{X}) + H(\underline{X}) = 0 \qquad \text{(G-5)}$$

provided that such an N exists. Except for the extra term, $H(\underline{Y}|\underline{K}, \underline{X})$, Equation G-5 is the same as Equation 12-8.

How many keys there are in a homophonic substitution cipher depends on whether the information is requested by the communicant who uses the cipher or the opponent who attacks it. The opponent's idea of how many keys there are in the cipher can be quite different from that of the communicant. For example, the number of substitutes per character in the plain alphabet $(L_1, L_2, \ldots, L_t)$ and the number of characters in the cipher alphabet (L), are apt to be secret parameters of the cipher system. They would be known to the communicant but not to the opponent. An intercepted cryptogram would not always reveal the entire cipher alphabet. Thus, if L′ denotes the number of different characters in the intercepted cryptogram, the analyst would not know whether $L' < L$ or $L' = L$. In that case, the opponent would have to approximate n, the number of keys, without knowing $L_1, L_2, \ldots, L_t$ or L.

Since any character in the cipher alphabet can be assigned to any character in the plain alphabet, an upper bound on n is obtained as follows:

$$n \leqslant t^T \tag{G-6}$$

In the worst case, where the analyst has no knowledge about the structure of the key space, t^T could be used for n. This would provide an upper bound on unicity distance.

However, if the analyst has a prior knowledge of $L_1, L_2, \ldots, L_t$, the number of keys can be obtained as follows:

$$\begin{aligned} n &= \binom{L}{L_1}\binom{L-L_1}{L_2}\cdots\binom{L-L_1-L_2\ldots-L_{t-1}}{L_t} \\ &= L!/[(L_1!)(L_2!)\ldots(L_t!)] \end{aligned} \tag{G-7}$$

In those cases where the analyst does not know $L_1, L_2, \ldots, L_t$, but has some knowledge of the key space, it may still be possible to approximate n using Equation G-7. For example, if the analyst knows or suspects that each plain character has an equal number of substitutes ($L_1 = L_2 = \ldots = L_t$), then this value can be approximated by L'/t, the observed number of characters in the cipher alphabet divided by the number of characters in the plain alphabet. If the analyst knows or suspects that the number of substitutes per character is proportional to the probability of that character appearing in the plain alphabet, $L_i = (pi)(L)$, then L_i' can be approximated by $(pi)(L')$ (rounding up or down to the nearest whole number, as appropriate).[2]

Theorem: In a homophonic substitution cipher

$$H(\underline{Y}|\underline{K}, \underline{X}) = N \sum_{i=1}^{t} p(a_i)\log_2 L_i; \qquad N \geqslant 1 \tag{G-8}$$

where $a_1, a_2, \ldots, a_t$ are the characters in the plain alphabet of $\underline{X}$.

Proof: The proof follows directly from Equation 12-7e if the relationship

$$p(b_{i1}b_{i2}\ldots b_{iN}) = p(a_{i1}a_{i2}\ldots a_{iN})(1/L_{i1})(1/L_{i2})\ldots(1/L_{iN})$$

is used, where $b_{i1}b_{i2}\ldots b_{iN}$ is an arbitrary cryptogram in $\underline{Y}$ (over the alphabet $b_1, b_2, \ldots, b_L$), $a_{i1}a_{i2}\ldots a_{iN}$ is the corresponding message in $\underline{X}$ (over the alphabet $a_1, a_2, \ldots, a_t$), and b_{i1} is one of the L_{i1} substitutes for a_{i1}, b_{i2} is one of the L_{i2} substitutes for a_{i2}, and so on.

[2] A better approximation of n could be obtained if L_i' were allowed to assume values in an interval about $(pi)(L')$ (e.g., the interval obtained using a method of confidence limits for some chosen level of confidence).

Corollary: From the above theorem, it follows that

1. If $L_1 = L_2 = \ldots = L_t$, then $H(\underline{Y}|\underline{K}, \underline{X}) = N(\log_2 L - \log_2 t)$
2. If $L_i \simeq p(a_i)(L)$ for $i = 1, 2, \ldots, t$, then $H(\underline{Y}|\underline{K}, \underline{X}) = N(\log_2 L - F_1)$

where F_1 is a measure of the entropy of $\underline{X}$.

If the approximation $N\log_2 L$ is used for $H(\underline{Y})$ and the values

$$H(\underline{K}) = \log_2 n$$

$$H(\underline{Y}) = N\log_2 L$$

$$H(\underline{Y}|\underline{K}, \underline{X}) = N \sum_{i=1}^{t} p(a_i)\log_2 L_i$$

$$H(\underline{X}) = NF_{J+1}$$

are substituted into Equation G-5, one obtains (solving for N):

$$N = \log_2 n/[\log_2 L - F_{J+1} - \sum_{i=1}^{t} p(a_i)\log_2 L_i] \qquad \text{(G-9)}$$

From the above corollary, when $L_1 = L_2 = \ldots = L_t$ Equation G-9 reduces to

$$N = \log_2 n/(\log_2 t - F_{J+1}) \qquad \text{(G-10)}$$

For English, $\log_2 t = \log_2 26 = 4.70$. Hence Equation G-10 shows that the analyst can attack the cipher using 1-grams ($J = 0$). This confirms one's intuition that $L_1 = L_2 = \ldots = L_t$ should not cause 1-gram, 2-gram, and so on, language statistics to be destroyed during encipherment. However, from the corollary, when $L_i \simeq p(a_i)(L)$, Equation G-9 reduces to

$$N = \log_2 n/(F_1 - F_{J+1}) \qquad \text{(G-11)}$$

For English, the value of F_1 is 4.17. Hence Equation G-11 shows that the analyst can no longer attack the cipher using 1-grams ($J = 0$). Again, this confirms one's intuition that 1-gram language statistics are destroyed during encipherment, since L_i is proportional to pi.

Two notable examples of homophonic substitution are the Beale Ciphers [4] and the Zodiac Murder Ciphers [5]. The Beale Ciphers consist of three numeric cryptograms, denoted B1, B2, B3, allegedly constructed by one Thomas Jefferson Beale in the year 1820. The purpose of these cryptograms was to describe the location, contents and respective heirs of a treasure in gold, silver, and jewels. B2, which describes the contents of the treasure, was broken several decades later, after Beale's key to B2 was discovered. This key

Cryptogram	N (Length in Characters)	L′ (Cryptogram Alphabet Size)	Log_2n (n = Number of keys)	Unicity Distance in Characters				
				F_2 3.62	F_3 3.22	F_6 2.81	F_{15} 2.02	F_{100} 1.22
B1	520	279	1094.7	1990	1152	805	509	371
B2	763	179	685.5	1246	722	504	319	232
B3	618	263	1029.0	1871	1083	757	479	349
Z1	408	46	151.5	275	159	111	70	51
Z2	340	63	217.6	396	229	160	101	74

"B" denotes Beale Ciphers; "Z" denotes Zodiac Murder Ciphers.
B2 and Z1 have been solved; B1, B3, and Z2 have not been solved.
Unicity distance is computed using Equation G-11.

Table G-4. Computed Unicity Distances for the Beale Ciphers and the Zodiac Murder Ciphers

was constructed by sequentially numbering each word in the Declaration of Independence ("When in the course of . . .") and assigning each number to the corresponding initial letter of each word (1 = w, 2 = i, 3 = t, 4 = c, 5 = o, etc.). The key to B2 did not, however, provide the solution to B1 or B3. Subsequently, amateur and professional cryptanalysts have expended untold amounts of time and money attempting to decipher the remaining cryptograms. No solution is known to exist.

The Zodiac Murder Ciphers, denoted Z1 and Z2, were constructed by the so-called Zodiac Killer who haunted the San Francisco Bay area in the late 1960s. Z1 was broken by an amateur cryptanalyst shortly after it was published by the news media. No solution to Z2 is known to exist.

Will cryptanalysts, or even amateur cryptogram buffs ever be successful in decoding these yet unbroken cryptograms? Some insight into this question can be gained if the unicity distances for these cryptograms are computed (Table G-4).

In each case, the number of keys (n) was calculated from L′ (the number of different characters in the cryptogram) using the following procedure: Let p1, p2, . . . , p26 represent the values of p(a) in Table 12-1, which have been sorted into descending sequence (i.e., p1 ⩾ p2 ⩾ . . . etc.). The values of $L'_1, L'_2, \ldots, L'_{26}$ were calculated as follows:

$$L'_i = q_i - q_{i-1}; \quad i = 1, 2, \ldots, 26$$

where

$$q_0 = 0$$

$$q_i = L' \sum_{j=1}^{i} pj; \quad i = 1, 2, \ldots, 26$$

and the values of q_i are rounded off to the nearest whole number.

For example, the values of $L'_1, L'_2, \ldots, L'_{26}$ for cryptogram B1 in Table G-4 were obtained from L′ = 279 as shown in Table G-5:

i	Character (i)	p(i)	$\sum_{j=1}^{i} p(j)$	$L' \sum_{j=1}^{i} p(j)$	L'_i
1	E	.1251	.1251	34.9	35
2	T	.0925	.2176	60.7	26
3	A	.0804	.2980	83.1	22
4	O	.0760	.3740	104.3	21
5	I	.0726	.4466	124.6	21
	Remainder of Computation is not Shown				

Table G-5. Computation of $L'_1, L'_2, \ldots, L'_{26}$ for L′ = 279

The procedure guarantees that $L'_1 + L'_2 + \ldots + L'_{26} = L'$.

There are two interesting results obtained as a consequence of these computations. Z1, which was broken by an amateur analyst, is well beyond the unicity distance, even when only digram statistics are used to attack the cipher. On the other hand, even when 7-gram language statistics are employed, B1 and B3 are still below the unicity distance. Moreover, B1 and B3 remain unsolved, even though they have been subjected to repeated cryptanalysis for more than 100 years. Thus, theoretical results (and the conclusions naturally inferred from these results) agree with observed results.

REFERENCES

1. Shannon, C. E., "Communication Theory of Secrecy Systems," *Bell System Technical Journal,* **28**, 656–715 (1949).
2. Matyas, S. M., *A Computer Oriented Cryptanalytic Solution for Multiple Substitution Enciphering Systems,* Doctoral Thesis, University of Iowa (1974).
3. Friedman, W. F., "Cryptology," *Encyclopedia Britannica,* p. 848 (1973).
4. Ward, J. B., *The Beale Papers,* Virginia Book and Job Print, Lynchburg, Virginia, 1885.
5. "Vallejo Mass Murder," *San Francisco Examiner and Chronicle,* August 3, 1969, Section A, p. 9; August 10, 1969, Section A, p. 26.

Other Publications of Interest

6. Deavours, C. A., "Unicity Points in Cryptanalysis," *Cryptologia,* **1**, No. 1, 46-68 (1977). (1977).
7. Lu, S. C. and Lee, L. N., "Message Redundancy Reduction by Multiple Substitution," *COMSAT Technical Review* **9**, No. 1, 37–47 (Spring 1979).
8. Tanaka, H. and Kaneku, S., "Data Compression Approach to Cryptography," *Proceedings 1979 Carnahan Conference on Crime Countermeasures,* University of Kentucky, Lexington (May 16–18, 1979).
9. Reeds, J. "Entropy Calculations and Particular Methods of Cryptanalysis," *Cryptologia,* **1**, No. 3, 235-254 (1977).

APPENDIX H

Derivation of p(u) and p(SM)

In this appendix, emphasis is placed on analysis of the random cipher with the objective of obtaining the correct message for an intercepted cryptogram, but not necessarily the correct key. Primarily, this analysis is carried out to provide the reader with additional insight into the problems of cryptanalysis.

In the definitions given below, y represents an intercepted cryptogram (i.e., y is an element of the set $\underline{Y}'$). Recall that

1. M is the random variable defined as the number of keys that will decipher an intercepted cryptogram into a meaningful message.
2. M′ is the random variable defined as the number of keys, except for the key originally used to produce the given cryptogram, that will decipher an intercepted cryptogram into a meaningful message.
3. U is the random variable defined as the number of different meaningful messages produced when an intercepted cryptogram is deciphered with all possible keys.

Now let the mutually exclusive events U1 and U2 be defined as follows:

1. Event U1 occurs if at least one incorrect key deciphers the intercepted cryptogram into the correct message when an intercepted cryptogram is deciphered with all possible keys.
2. Event U2 occurs if no incorrect key deciphers the intercepted cryptogram into the correct message when an intercepted cryptogram is deciphered with all possible keys.

(See Figures H-1 and H-2.)

It follows from the definition of U, U1, and U2 that

$$p(U = u) = p(U = u, U1) + p(U = u, U2)$$

(Note that event U1 or U2 can occur, but not both.) For the case where $u = 1$ occurs in conjunction with U1, it is implied that all m′ ($m' \geqslant 1$) incorrect keys

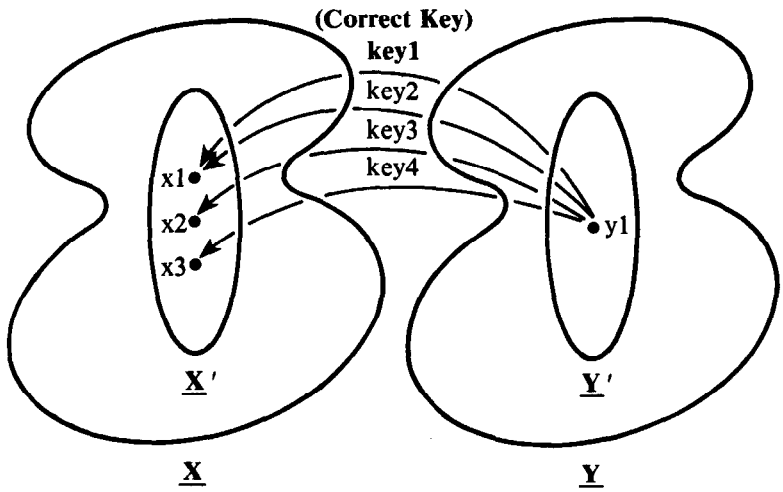


Figure H-1. Example in which U = 3 and Event U1 Occurs

must decipher the intercepted cryptogram into the correct message, and so

$$p(U = 1, U1) = \sum_{m'=1}^{n-1} \left(\frac{1}{s}\right)^{m'} p(m')$$

On the other hand, u = 1 in conjunction with U2 implies that there cannot be any incorrect keys that decipher the intercepted cryptogram into a meaningful message, and so

$$p(U = 1, U2) = p(m' = 0)$$

Consequently,

$$p(U = 1) = p(m' = 0) + \sum_{m'=1}^{n-1} \left(\frac{1}{s}\right)^{m'} p(m') \qquad \text{(H-1a)}$$

and so, from Equation 12-3b, it follows that

$$\begin{aligned} p(U = 1) &= e^{-\lambda'} + \sum_{m'=1}^{n-1} e^{-\lambda'} \frac{(\lambda'/s)^{m'}}{(m')!} \\ &= e^{-\lambda'} e^{\lambda'/s} \qquad \text{for } s/r \ll 1 \end{aligned} \qquad \text{(H-1b)}$$

For the case where u ⩾ 2, conditional probabilities can be used to obtain

$$p(U = u) = \sum_{m'=u}^{n-1} p(U = u, U1|m')p(m') + \sum_{m'=u-1}^{n-1} p(U = u, U2|m')p(m')$$

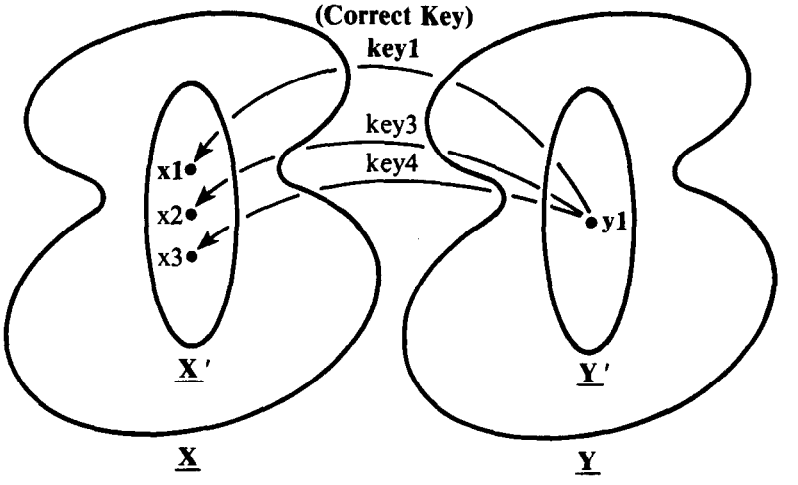


Figure H-2. Example in which U = 3 and Event U2 Occurs

Consider the situation where m′ balls are randomly placed into u cells. The probability that each cell will contain at least one ball is given by

$$po(m', u) = \sum_{v=0}^{u} (-1)^v \binom{u}{v} \left(1 - \frac{v}{u}\right)^{m'}$$

(see reference 1). Because m′ decipherments produce only messages within the set of u specific messages, it follows for a random cipher that po(m′, u) represents the probability that these m′ decipherments will lead to each of the u specific messages at least once. A special case of interest is m′ = u, which results in

$$po(u, u) = u!/u^u \qquad \text{(H-2)}$$

Consider the situation in which m′ decipherments result in any of s possible messages. If u specific messages are designated out of s possible messages, then the probability that the m′ decipherments will lead only to the designated u messages, and that these m′ decipherments will lead to each of the u messages, at least once, is given by

$$(u/s)^{m'} po(m', u)$$

Since for U1 to occur the correct message must always be included, there are $\binom{s-1}{u-1}$ different ways to select u messages out of s possibilities. Hence it follows that

$$p(U = u, U1 | m') = \binom{s-1}{u-1} \left(\frac{u}{s}\right)^{m'} po(m', u)$$

Using similar reasoning, it follows that

$$p(U = u, U2|m') = \binom{s-1}{u-1}\left(\frac{u-1}{s}\right)^{m'} po(m', u-1)$$

and so,

$$p(U = u) = \binom{s-1}{u-1}\left[\sum_{m'=u}^{n-1}\left(\frac{u}{s}\right)^{m'} po(m', u)p(m')\right.$$
$$\left. + \sum_{m'=u-1}^{n-1}\left(\frac{u-1}{s}\right)^{m'} po(m', u-1)p(m')\right]; \qquad s \geqslant u \geqslant 2 \tag{H-3}$$

In order for the process of cryptanalysis to be successful, the value of u should be much less than s. In that case, the following approximation for $\binom{s-1}{u-1}$ is obtained:

$$\binom{s-1}{u-1} = \frac{s^{u-1}}{(u-1)!}\prod_{i=1}^{u-1}\left(1 - \frac{i}{s}\right)$$
$$\simeq \frac{s^{u-1}}{(u-1)!}\, e^{-u(u-1)/2s}; \qquad u \ll s \tag{H-4}$$

Using Equations H-2 and H-4 together with the change of variable $m' = u + i$ allows Equation H-3 to be rewritten as

$$p(U = u) = p(m' = u-1)e^{-u(u-1)/2s}$$
$$\times \left[1 + \sum_{i=0}^{n-1-u}\left(\frac{u}{s}\right)^{i+1}\frac{po(u+i, u)}{po(u, u)}\frac{p(m' = u+i)}{p(m' = u-1)}\right. \tag{H-5}$$
$$\left. + \sum_{i=0}^{n-1-u}\left(\frac{u-1}{s}\right)^{i+1}\frac{po(u+1, u-1)p(m' = u+i)}{po(u-1, u-1)p(m' = u-1)}\right]$$

for $u \geqslant 2$.

Equation 12-3a, which is the accurate expression for p(m′), can now be used in Equation H-5. In the present situation, where it may be assumed that $s/r \ll 1$, the Poisson approximation to the binomial given by Equation 12-3b is used for p(m′). Hence it follows that

$$p(m' = u+i)/p(m' = u-1) = (\lambda')^{i+1}(u-1)!/(u+i)!$$

which can be substituted into Equation H-5 to obtain

$$p(U = u) = p(m' = u - 1)e^{-u(u-1)/2s} \times \left[1 + \sum_{i=0}^{n-1-u} \left(\frac{\lambda' u}{s}\right)^{i+1} \frac{(u-1)!}{(u+i)!} \frac{po(u+i, u)}{po(u, u)} + \sum_{i=0}^{n-1-u} \left[\frac{\lambda'(u-1)}{s}\right]^{i+1} \frac{(u-1)!}{(u+i)!} \frac{po(u+i, u-1)}{po(u-1, u-1)}\right] \tag{H-6}$$

for $u \geqslant 2$.

Neglecting terms of order $(1/s)^2$ and higher, and using

$$\frac{po(u, u-1)}{po(u-1, u-1)} = \frac{u}{2}$$

whose derivation is left to the reader as an exercise, Equation H-6 can be written as

$$p(U = u) \simeq p(m' = u - 1)[1 + \lambda'/s + (u-1)(\lambda' - 1)/2s - (u-1)^2/2s] \tag{H-7}$$

Neglecting terms of order $(1/s)^2$ and higher in the accurate equation for $p(U = 1)$, Equation H-1a, it follows that

$$p(U = 1) \simeq e^{\lambda'}(1 + \lambda'/s) = p(m' = 0)(1 + \lambda'/s)$$

Hence, it follows that Equation H-7 is valid for $u \geqslant 1$. Recognizing that the first three moments of the Poisson distribution are λ', $(\lambda')^2 + \lambda'$, and $(\lambda')^3 + 3(\lambda')^2 + \lambda'$, respectively, and using Equation 12-3b for $p(m')$, results in the expected value

$$\begin{aligned} E(U) &\simeq \sum_{u=1}^{s} u\, p(U = u) \\ &\simeq (\lambda' + 1)[1 - (\lambda'/2s)(\lambda' + (2/\lambda') + 1)] \\ &\simeq E(M)[1 - (\lambda'/2s)(\lambda' + (2/\lambda') + 1)] \end{aligned} \tag{H-8}$$

From Equation H-8, it can thus be seen that $E(U) \leqslant E(M)$, as expected.

Recall, from Equation 12-2, that $p(SM)$ is the probability of solving for the correct message. Hence it follows from Equation H-7 that

$$\begin{aligned} p(SM) &\simeq \sum_{u=1}^{s} (1/u)p(U = u) \\ &\simeq (1/\lambda)(1 - e^{-\lambda})(1 + (\lambda'/2s)) \\ &\simeq p(SK)(1 + (\lambda'/2s)) \end{aligned} \tag{H-9}$$

From Equation H-9, it can be seen that $p(SM) \geqslant p(SK)$ as expected. However, the factor by which p(SM) is larger than p(SK) is $1 + \lambda'/2s$. At the unicity point for the random cipher, where $\lambda = 1$, it is observed that $1 + \lambda'/2s \simeq 1$. Consequently, when $\lambda \leqslant 1$, it is the case that $p(SM) \simeq p(SK)$. Therefore, Equation H-9 gives the reader additional insight into cryptanalysis, but does not provide an equation for the computation of p(SM) that would be used in practice.

REFERENCE

1. Feller, W., *An Introduction to Probability Theory and its Applications,* 3rd ed., Wiley, New York, 1968.

Index