CLASSICAL CRYPTOGRAPHY COURSE
BY LANAKI

30 JANUARY 1997
Revision 0

LECTURE 23

DATA ENCRYPTION STANDARD (DES)

COURSE NOTES

DES represents the technological completion point of our classical cryptography course. I originally designed this course to meet ACA member needs and to cover the various types of cryptographic systems that might be found in the ACA bimonthly publication "The Cryptogram." I believe that I have met and exceeded that goal.

DES is outside of the ACA purview. DES is also a marvelous example of putting many of the principles of our course into practice. For those interested, Schneier presents a detailed discussion of more advanced cryptosystems. He builds upon the DES theory discussed in this Lecture. [SCH2] Also, reference [NIST90] is a state-of-the-art discussion of modern cryptosystems and is highly recommended.

Lecture 24 will be devoted to special topics and will present additional cryptograms for study and solution. I will update and restructure my Volume II references and resources file. It along with solutions to Lecture 24's special topics problems will constitute my final Lecture 25. As with every Tae Kwon Do form, there is a birth and death of each movement; each movement having its time, beauty, place and purpose. So too, I have tried to create a course that has birth and death in each Lecture and terminates with the appropriate material. My gift is complete, only the packing is left.

Those students interested in course participation certificates please advise me by e-mail, so I have an idea how many to order.

Volume II of our textbook should be out of the editing mode and into publication by Aegean Park Press in February, 1977 (I think). I will make an announcement as soon as it is available. Those interested in signed copies please advise by private E-mail, and I will maintain a small inventory for that purpose. (On a personal note, I consider the signing of Volume I at our convention quite an honor and I thank you' ll for the recognition. Also thanks again to Dave Kennedy of NCSA and MEROKE for their excellent book reviews.)

SUMMARY

Since its official introduction in 1977 by the National Bureau of Standards, perhaps no other encryption standard has been dissected by researchers and customers as closely as the Data Encryption Standard (DES). DES brings together much of the work that we have studied in the past 22 lectures and adds some interesting new principles such as Involution, Work Factor, Data Dependency and Iterated Cryptosystems. There is a wealth of resources describing DES. Many of the earlier texts espoused that DES was unbreakable. Computer technology has changed the machinery and that assumption. The brilliant researchers, Biham and Shamir in their "Differential Cryptanalysis," proved that the DES was by itself not unbreakable and that the system could be broken in its weaker forms on a personal computer. [DIFF]

In 1997, NIST requested public comment on the AES or Advanced Encryption Standard to replace their old workhorse DES. It is amazing (and a testament to its founders) that DES in its various forms has lasted for so many years as a viable cryptographic package.

RESOURCES

There are several good references describing DES, its variants and cryptanalytical attacks against it. [MEYE], [KONH], [DENN], [KATZ], [STIN], [RHEE], [BIHA], [KOBL], [SCHN], [SEBE], [WALS], [RYSK], [SCH2] and [HOFF]. I will draw from most of these texts to describe DES and its demise.

INTRODUCTION

In my opinion, DES represents the turning point from classical cryptography into modern day cryptography. Actually, DES is nothing more than a complex combination of the substitution and transposition systems we have previously studied. We need to define some new concepts to understand why the DES combination of simpler systems makes for such a strong cryptosystem in situ.

Due to the invention of computer systems and the introduction of nation-wide computer networks, the twentieth century has drastically changed the range of data protection issues. As we move into the twenty-first century, our concerns for data protection center around protection of communication channels which not only connect terminals to computers but create communication networks among the host computers. Due to the natural attributes of any channel, we have a communication medium that is accessible to eavesdroppers so physical security is useless. Cryptosystems are used to enforce protection in communication channels.

BACKGROUND THEORY

1. INVERTIBLE TRANSFORMATIONS

Encryption is the key (the Australian's call it "primitive") operation at the disposal of cryptography. It is a special computation that operates on messages, converting them into representation that is meaningless for all parties other than the intended receiver. Transformations effected on messages are so intricate that it is beyond the means of the interloper to undo the process. Almost without exception all modern cryptosystems rely upon the difficulty of reversing the encryption transformation as a basis for a secure communication.

The encryption algorithm is chosen from a family of invertible transformations known as the general system, or cryptosystem. The parameter that selects the particular transformation from the family is called the enciphering key. The cryptosystem may take any of several forms, say a set of instructions, a piece of hardware or a program, one of which is selected by the enciphering key.

Public key schemes are based on a class of functions known as one-way trapdoor functions, derived from a class of computationally difficult problems termed NP (non-deterministic polynomial) problems. Private secret key schemes, in contrast, rely on a series of complex substitution and transposition operations, called involution, which are very hard to analyze mathematically. [STIN]

Formally, a cryptosystem is a single parameter family of invertible transformations,

Ek ; K n K

where K is the keyspace, which is of finite length, with elements K1,K2...Kn. If M is the message space and C is the cryptogram or ciphertext space, then the system must have the following properties:

o enciphering algorithm

Ek: M -> C

for any fixed encryption key K n K, is an invertible transformation of the message space into the cryptogram space, i.e. Ek(M) = C, where M n M and C n C;

o there is an inverse algorithm Ek^{-1} = Dk called the decryption algorithm

Dk: C -> M, such that Dk(C) = Dk[Ek(M)] = M ;

o the keys uniquely define the enciphered message,

Ek1(M) .ne. (not =) Ek2(M) if K1 .ne. K2

Modern Cryptography deals with the design and analysis of systems that provide secure communications or resist cryptanalysis. A system is said to be compromised via cryptanalysis if it is possible to recover the original message, plaintext, from the ciphertext, without knowledge of the key used in the encryption algorithm. Cryptanalysis draws from such disciplines as probability theory, number theory, statistics and algebra, topology, chaos theory, matrix theory, non linear calculus, algorithms, communication language, and redundant properties of the language being investigated.

Cryptanalysis is a system identification problem and the goal of cryptography is to build systems that are hard to identify. An ideal system is one that has a flat distribution for all statistical properties of the cipher, implying that the redundant qualities of the natural language have been obscured.

In 1948, Shannon characterized the two main methods of uniformly distributing the redundant characteristics of a natural language. [SHAN] First through diffusion , which spreads the correlations and dependencies of the messages over substrings as long as feasible so as to maximize the unicity distance (see section item 2). The second approach is confusion, where the functional dependencies of the related variables are mad as complex as possible so as to increase the time needed to analyze the system. DES takes maximum advantage of both of these approaches.

The 'noisy channel' problem is analogous to the problem of secure communication in cryptography - the noise corresponding to the enciphering transformation and the received message as the ciphertext. The role of the sender, though, is to make the recovery of the original message as difficult as possible, if not impossible. Cryptographers seek to devise encryption techniques that produce ciphertext that cannot be distinguished from purely random bit strings by the opponent. [SEBE]

```
             Figure 21-1
            Noisy Channel

                noise
                  |
                  |
      M ----------+------------> M'
    sender                  receiver
```

George Simmon's points out that the statistical communication channel of the coding/decoding model has been replaced by a game theoretic channel; nature has been replaced by an intelligent opponent. Game theory is a mathematical theory (and a political and relationship [I'm ok - you're not] theory) that deals with the general features of competitive situations, placing particular emphasis on the decision-making process of adversaries. [SIMM]

It is not sufficient that a cryptosystem be able to thwart cryptanalysis alone. It should frustrate any and all aims of unauthorized parties attempting to subvert the integrity of a supposedly secure channel.

2. OPPONENT ATTACKS AGAINST A CRYPTOSYSTEM

Seberry identifies the typical aims of an opponent as follows:

a. to determine the content of the message M;
b. to alter message M to M' and have M' accepted by the receiver as a message from transmitter of M;
c. to initiate communications to a receiver and have the interloper posing as an authorized transmitter. This
   is also called 'spoofing'.

Traditionally, the first of these aims, known as the privacy problem, has been our interest for about 22 lectures. Electronic communications has acquired a more ubiquitous presence in public and private spheres. The latter two aims have become more important in systems design. Foiling these aims are known as the problem of authentication and the problem of dispute. [SEBE]

3. DENNING MODEL

A cipher is considered breakable if it is possible to determine the plaintext or key from the ciphertext, or to determine the key from plaintext-ciphertext pairs. Dr. Denning defines four basic methods of attack to determine the adequacy of a prospective cryptosystem: ciphertext-only, known-plaintext, chosen-plaintext, and chosen-ciphertext.

Under a ciphertext-only attack, a cryptanalyst must determine the key solely from the intercepted ciphertext, through the method of encryption, the plaintext language, the subject matter of the ciphertext, and certain probable words may be known.

Under a known plaintext attack, a cryptanalyst knows some plaintext-ciphertext pairs. Knowledge of probable words allows a close approximation to the known plaintext attack. Encrypted programs are particularly vulnerable because of the appearance of keywords - e.g. begin, end, var, procedure, if, then. Even if the positions of these words are not known, reasonable guesses may be made.

Under a chosen plaintext attack, a cryptanalyst is able to acquire the ciphertext corresponding to selected plaintext. This is the most favorable condition for the cryptanalyst. A database system may be particularly vulnerable to this type of attack if users can insert elements into the database, and then observe the changes in the stored ciphertext. Bayer and Metzger call this the planted record problem. [BAYE]

Public-key systems have introduced a fourth kind of attack: a chosen-ciphertext attack. Although the plaintext is not likely intelligible, the key may be deduced.

A cipher is unconditionally secure if, no matter how much ciphertext is available or intercepted, there is not enough information in the ciphertext to determine the plaintext uniquely. With one exception, the one- time pad, all ciphers are breakable given unlimited resources, so we are more interested in ciphers that are computationally infeasible to break. A cipher is computationally secure or strong if it cannot be broken by systematic analysis with available resources.

4. THREATS TO DATA STORED IN COMPUTER SYSTEMS

Information transmitted over electronic lines is vulnerable to passive wiretapping, which threatens secrecy, and to active wiretapping, which threatens authenticity. Passive wiretapping (eavesdropping) refers to the interception of messages, usually without detection. Active wiretapping (tampering) refers to deliberate modifications made to the message stream. Encryption protects against message modification and injection of false messages by making it infeasible for the opponent to create ciphertext that deciphers into meaningful plaintext. Note that whereas it can be used to detect message modification, it can not prevent it.

Encryption does not protect against replay , because an opponent could simply replay the previous ciphertext. Protocols requiring acknowledgments normally prevent against intentional message deletion. Data in computer systems is subject to similar threats. Threats to secrecy include: browsing, leakage and inference. Browsing refers to searching through main memory or storage for information and confidential programs. If access controls are not employed, ciphertext searching for identical information pairs may be effective. Leakage refers to the transmission of data to unauthorized persons by processes with legitimate access to the data. Inference refers to the deduction of confidential data about a particular individual by correlating released statistics about groups of individuals.

Threats to authenticity include tampering and accidental destruction. Tampering with data in computer systems is analogous to active wiretapping on communication channels. Accidental destruction refers to the unintentional overwriting or deletion of data. Norton Utilities has been a great help in this area.

Computer systems are also vulnerable to another problem: masquerading or spoofing. If an intruder can gain access to a system under another users account, he has access to all the information within that users domain. Digital signatures provide a means to authenticate users and processes.

5. INFORMATION THEORY - SHANNON'S CONCEPTS

Security is directly related to the difficulty associated with the inverting encryption transformation(s) of a cryptosystem. The protection afforded by the encryption procedure can be evaluated by the uncertainty facing an opponent in determining the permissible keys used. Shannon [SHAN] characterized a system that has perfect security with the following property: if an opponent knows E (the encryption transformation) and has an arbitrary amount of cipher, he/she is still left with a choice between all messages from the message space when attempting to recover the corresponding plaintext for some ciphertext.

Let $P_c(M)$ be the probability that a message M was sent given that C was received, with $C = E(M)$.

Perfect security is defined as:

$$Pc(M) = P(M)$$

where P(M) is the probability that message M will occur. Let Pm(C) be the probability of receiving ciphertext C given that M was sent. Then Pm(C) is the sum of the probabilities P(K) of the keys that encipher M as C:

$$Pm(C) = \overset{K}{\underset{K,Ek(M) = C}{d}} \qquad P(K)$$

where the bold K means across the space of keyspace K. Usually there will only be one key K that satisfies

$$Ek(M) = C$$

A necessary and sufficient condition for perfect secrecy is that for every C,

$$Pm(C) = P(C)$$

This means that the probability of receiving ciphertext C is independent of encrypting it with plaintext M. Perfect secrecy can only be assured if the length of the key is as long as the message sent, and the cardinality of the key space is the same as that of the message space. These conditions ensure that the uncertainty of the key and cipher are maintained and maximized. Ciphers that could not be shown to have perfect secrecy but did not disclose sufficient information to allow the key to be determined, Shannon called ideally secret. By not revealing more information than the unicity distance, these systems were effectively unbreakable.

The opponent is faced with at least as much uncertainty with respect to the message as he is with the key. The only system that fits this definition is the one time pad. The key used is a non-repeating stream of random bits, and is discarded after each transmission. A separate key is used for each transmission as two ciphertexts encrypted with the same key could be correlated. Being in possession of C adds no information to the task of recovering M = Dk(C). Systems based on Shannon's equivocation are unconditionally secure, meaning the system will resist cryptanalysis even in the presence of infinite computing power. The security of the system is derived from statistical uncertainty. If Hc(K), the entropy of the key, never approach zero for any message length, then the cipher is considered unconditionally secure.

Shannon assumed in devising his perfect ciphers that opponents has access to unlimited computing power. It is far from unreasonable though, to believe that any single opponent, or cartel of opponents, except NSA, is in possession of inexhaustible computing resources. Such security measures as warranted by Shannon would appear excessive, for what they are guarding against is not a tangible threat. Modern cryptosystems look beyond uncertainty and unicity distances to establish a basis of security and, in particular, the work factor, the ratio of the complexity of cryptanalyzing a system to decryption, is taken as a strong indication of a system's security. Security can be cited in terms of the number of person/computer years needed to break the system. The subtle distinction can be drawn between perfect secrecy and cryptosecrecy, the first being asymptotically defined while the latter appeals to the concept of intractability. There does really exist a general method to prove a cryptosystem is cryptosecure. Designers have come to rely upon certification by cryptanalysts, who with considerable zest attempt to compromise the system using ad hoc and heuristic measures, as an indication of a system's security. History has repeatedly shown that systems purported to be unbreakable by their inventors were demonstrated to be far less secure than thought after being scrutinized by cryptanalysts.

We have described the four basic attacks on a cryptosystem. The systems security does not depend on the concealment of its encryption transformation or algorithm. Kerchkhoff's principle provides that the algorithm is available for all to examine and study. When E is revealed, a very difficult or inefficient method is also revealed to compute the inverse of E. Given the ciphertext C, the cryptanalyst can examine the message space exhaustively until M is found such that E(M) = C. This method is also called brute force. Whenever a key of finite length is employed, it can always be compromised by direct search methods. The success of such an attack depends upon the work factor associated with the cipher, i.e. the minimal number of computations needed to invert the system. It should be noted that the unicity distance indicates the number of characters needed to determine the key, but it makes no comment on the complexity of this task. A system can disclose more ciphertext than its unicity distance considers safe but still may remain cryptosecure.

A system is considered computationally secure if the task of inverting E is computationally infeasible or intractable. You might recognize this as similar to the properties of NP (Non deterministic polynomial) problems.

6. ENTROPY AND EQUIVOCATION

Information theory measures the amount of information in a message by the average number of bits (binary digits in 0, and 1's for a computer) needed to encode all possible messages in an optimal encoding. For example the Sex field in a personnel database, contains only one bit of information because a 0 can represent a Male and a 1 can represent a Female. We could spell the words out , take up more space, but not yield more information. In computer systems, programs and text files are usually encoded in 8-bit ASCII codes, regardless of the amount of information in them. Furthermore, text files can be compressed by about 40% without losing any information.

The amount of information in a message is formally measured by the entropy of the message. The entropy is a function of the probability distribution over the set of all possible messages.

Let X1, ..., Xn be n possible messages occurring with probabilities of p(X1),......p(Xn), where:

$$\sum_{i=1}^{n} p(X_i) = 1$$

The entropy of a given message is defined by the weighted average:

$$H(x) = - \sum_{i=1}^{n} p(X_i) \log_2 p(X_i)$$

If we write this sum over all messages X:

$$H(x) = - \sum_{X} p(X) \log_2 \left[\frac{1}{p(X)}\right]$$

In the example above, with the p(male) = p(female) = 1/2

$$H(X) = 1/2 \log_2 (2) + 1/2 \log_2 (2) = 1/2 + 1/2 = 1$$

which confirms our observation that only 1 bit of information is required in the sex field of the database.

Intuitively, each term log2 (1/p(X)) represents the number of bits needed to encode message X in an optimal encoding - that is it minimizes the expected number of bits transmitted over a channel. The weighted average H(X) gives the expected number of bits in the optimally encoded message.

Because 1/p(X) decreases as p(X) increases, an optimal encoding uses short codes for frequently occurring messages at the expense of using longer ones for infrequent messages. Morse code applies this principle. The most frequent letters use the shortest codes.

The entropy of a message H(M), also measures its uncertainty, in that it indicates the number of bits of information that must be required to recover a message distorted by a noisy channel or concealed through ciphers. The uncertainty of a message cannot exceed log2n bits, where n is the possible number of messages.

The rate of language for messages of length k is defined as:

$$r = H(X)/k$$

which denotes the average number of bits of information in each character. For English, when k is large, r has been estimated to lie between 1.0 bits/letter and 1.5 bits/letter. The absolute rate of a language is the maximum number of bits of information that could be encoded in each character assuming all combinations of characters are equally likely.

If there are K letters in the language, then the absolute rate is given by

$$R = \log2K$$

which is the maximum entropy of the individual characters. For English, this is 4.7 bits/letter. The actual rate of English (3.2 bits/letter) is much less as it is highly redundant, like all natural languages. Redundancy stems from the underlying structure of a language, in particular certain letter and combinations of letter occur frequently, while others have a negligible likelyhood of occurring (e.g. the letters E, T, A, I, N and O occur very frequently, as do digrams TH and EN, while Z and X are infrequent). The redundancy of a language with rate r is defined as D = R - r. When r =1 and R = 4.7, the ratio D/R shows that English is about 79% redundant!

We note that the more redundant a language is, the stronger the statistical relations between the letters in a sequence. On the other hand, if a language has no redundancy then occurrences of subsequent letters are statistically independent.

We can easily calculate the entropy of a single letter H1(M). Also the entropy H2(M) of two-letter words can be found relatively easily. Unfortunately, the amount of calculation for Hn(M) grows exponentially as a function of n. The practical redundancy of a language is expressed as:

$$rl = \underset{n \to l}{\text{limit}} \quad Hn(M)/n$$

l = infinity

Equivocation, defined as the conditional entropy of message M given that ciphertext C has occurred, is:

$$Hc(M) = \underset{C}{d}\, P(C)\ \underset{M}{d}\, Pc\ (M)\ \log2 \left[\frac{1}{Pc(M)} \right]$$

where Pc(M) is the conditional probability of message M given ciphertext C has occurred. Shannon measured the secrecy of a cipher with respect to its key equivocation, Hc(K); for ciphertext C and key K, it may be interpreted as the degree of uncertainty in K given C, and expressed as;

$$Hc(K) = \underset{C}{d}\, P(C)\ \underset{K}{d}\, Pc\ (K)\ \log2 \left[\frac{1}{Pc(K)} \right]$$

where Pc(K) is the probability of K given C. If Hc(K) is zero then there is no uncertainty in the cipher, making it unbreakable.

The unicity distance of a cipher is defined as the minimum message length that forces Hc(K) to approximate zero. So, the unicity distance of a cipher is the amount of ciphertext needed to uniquely determine the key. Intuitively, as the length of the ciphertext increases, the equivocation of the cipher decreases.

Seberry presents an interesting discussion of the equivocation of a simple cryptographic system. [SEBE] The results show that the calculation of equivocation become more complex as the number of messages and keys grow. She shows that the unicity distance of a cipher may be calculated or estimated, but, unfortunately, we may not be able to use this knowledge to break the cipher. Based on unicity, she divides all ciphers into two classes:

o the class of ciphers whose unicity distances exist and are finite;

o the class of ciphers whose unicity distances are infinite. Ciphers of this class are unbreakable (so-called ideal ciphers).

Shannon defined the unicity distance of a cipher in order to be able to get some quantitative measure of:

1. the security of the cipher (if the unicity distance of a code is small then the cipher is insecure); and

2. an indication of the amount of ciphertext, N needed to break the cipher.

It is given by:

$$N \; w \; \frac{H(K)}{D}$$

where D is the redundancy of the language (3.2 bits per letter for English) and H(K) is the information content of the key.

7. SYMMETRIC ALGORITHMS - PRODUCT CIPHER

A product cipher E is the composition of t functions (ciphers) F1,...,Ft where each Fi may be a substitution or a transposition. Rotor machines are product ciphers, where Fi is implemented by rotor Ri, 1s i s t. See Lectures 21 and 22.

The famous ENIGMA (Lecture 9) machine used by Germany, Japan, and their allies were of the multiple rotor type. A variation, the Hagelin machine was used extensively by diplomatic posts for many years. [KAHN]

These machines use symmetric algorithms - the same secret key must be known to both the sender and receiver.

8. MIXING TRANSFORMATIONS

Shannon proposed composing different kinds of functions to create "mixing transformations", which randomly distribute the meaningful messages uniformly over the set of all possible cipher text messages. [SHAN]

Mixing transformations could be created, for example, by applying a transposition followed by an alternating sequence of substitutions and simple linear operations. An algorithm (formal set of mathematical procedures or steps to accomplish a goal) embodying this approach was known as LUCIFER and was designed by IBM in the early '70's. LUCIFER used a transformation that alternately applied substitutions and transpositions. Figure 23-1 shows how the principle works with some small blocks (in practice much longer blocks are used.) Figure 23-1 gives a minute illustration of how substitutions and then permutation may be used to encipher using involutions only. The first three letters are substituted by removing one to the right in the alphabet and the second three letters are moved two to the right.

This can be deciphered by reversing the order of the operations and applying the inverse of each substitution and permutation.

Figure 23-1
Involution Example

	A	B	C	D	E	F
S1	B	C	D	F	G	H
P1	H	F	G	C	D	B
S2	I	G	H	E	F	D
P2	G	I	E	H	D	F

9. ITERATED CRYPTOSYSTEMS

We define an Iterated Cryptosystem as part of a family of cryptographically strong functions based on iterating a weaker function n times. Each iteration is called a round and the cryptosystem is called a n-round cryptosystem. The round-function is a function of output of the previous round and a sub-key which is a key dependent value calculated via a key-scheduling algorithm. The round-function is usually based on lookup tables (also known as S Boxes), bit permutations, arithmetic operations and the exclusive-or operation (usually denoted in most texts by a circle with a plus sign in it - in my ASCII lecture I will use an alt-241 character (q) enclosed in parentheses to mean the exclusive-or operation.)

LUCIFER was introduced in section 8. The round-function of LUCIFER has a combination of non-linear S boxes and a bit permutation. The input bits are divided into groups of four consecutive bits. Each group is translated by a reversible S box giving a four bit result. The output bits of all the S boxes are permuted in order to mix them when they become input to the following round. In LUCIFER only two fixed S boxes (S0 and S1) were chosen. Each S box can be used at any S box location and the choice is key dependent. For a block size of 128 bits and a 16 round cryptosystem there are 512 S box entries for which 512 key bits are needed (for the eight round variants 256 key bits are needed). A key expansion algorithm that repeats each key bit four times reduces the key size to 128 bits. Decryption is accomplished by running the data backwards using the inverse of each S box.

10. DATA ENCRYPTION STANDARD (DES)

The Data Encryption Standard (DES) is an improved version of LUCIFER. DES is not, as my HAZMAT friend suggests " a synthetic estrogen, diethylstilbestrol used as a growth stimulant in food animals. Residues in meat are thought to be carcinogenic."

DES is based on concepts described in Sections 1-9. It was developed by IBM, scrutinized by NSA, and adopted by the U.S. National Bureau of Standards (NBS) in 1977. For a time it was the de-facto world encryption standard.

The Data Encryption Standard (DES) is a mathematical algorithm used for the cryptographic protection of computer data. The algorithm is designed for use with binary-coded data and uses a 64-bit key to encipher 64 bits of information. The 64-bit key is of prime importance since a unique key results in the cryptographic generation of a unique set of 64-bits of cipher text from 64 bits of plain text. Since the algorithm is known to the general public, the cryptographic security of the DES is dependent on the security used to protect the key. Encrypted information can be transformed into the original plain text through a reversal of the algorithm process using the same key that was employed for encryption.

The Data Encryption Algorithm (DEA) was designed so that 56 bits of the 64 bit key are used for the encryption process and the remaining 8 bits are used only for parity error-detecting bits. The key is divided into eight 8-bit bytes (8 bits = 1 byte). In an 8-bit byte, 7 bits are used by the algorithm and the eight bit can be used to maintain odd parity. From a complete 64-bit block of plain text enciphered with a 56-bit key.

11. OVERVIEW OF THE DEA

DEA incorporates the following steps to encipher a 64-bit message (block of data) using a 64-bit key:

1. A transposition operation, referred to as the initial permutation (IP). This transposition does not use the 64-bit key and operates solely on the 64 data bits.

2. A complex key-dependent product transformation that uses block ciphering to increase the number of substitution and reordering patterns.

3. A final transposition operation, referred to as the inverse initial permutation (IP-1), which is an actual reversal of the transformation performed in the first step.

```
                    Figure 23-2
            Overview of Enciphering process

   Plaintext       [  64 data bits ]      Input

                           |
                           |
                           |
              ZDDDDDDDDDDDDDDDDDDDDDDDDDD?
              3 Initial Permutation, IP3
              @DDDDDDDDDDDDDDDDDDDDDDDDDDY
   Standard                |
   Data                    |
   Encryption              |
   Algorithm       ZDDDDDDDDDDDDDDDDDDD?
                   3     Product      3
   DEA             3  Transformation  3
                   @DDDDDDDDDDDDDDDDDDDY
                           |
                           |
                           |
                  ZDDDDDDDDDDDDDDDDDDDDD?
                  3  Inverse Initial  3
                  3    Permutation    3
                  3      IP -1        3
                  @DDDDDDDDDDDDDDDDDDDDDY
                           |
                           |
                           |

    Ciphertext     [  64 data bits ]    Output
```

The three major steps for DEA are shown in Figure 23-2. The IP and IP-1, are simple bit transpositions; the product transformation is fairly complex. Product transformations are successive applications of substitution and transposition ciphers. Large blocks of data are transformed as a unit, providing the advantage of increasing the number of substitution and reordering patterns. This is also called block ciphering.

In the product ciphering step of DEA, the block cipher substitutions are under the control of a cipher key while transpositions are performed according to a fixed sequence. Figure 23-3 depicts one iteration of the product transformation, which includes the following operations:

1. The 64-bit block of plaintext is divided into two 32-bit blocks, denoted by Li and Ri for the left and right halves, respectively.

2. The rightmost 32 bits of the input block become the leftmost 32 bits of the output block.

3. The rightmost 32 bits of the input block, Ri-1, goes through a selection process yielding 48-bit data block. This is a fixed selection and is it not key dependent. We call this an expansion permutation.

4. The 64-bit key is used to generate a 48-bit subkey Kn, where 1 s n s 16. Each Ki is unique and corresponds to the ith iteration of the product transformation.

5. The 48-bit subkey is added (modulo 2) to the output of step 3 yielding a 48-bit result. This is also called XORed (q).

6. The 48-bit output of step 5 is divided into eight 6-bit groups, that are each subjected to a unique substitution cipher that yields eight 4-bit groups, that are concatenated to form a 32-bit output.

7. The 32-bit output of step 6 is permuted by simple transposition to produce a 32-bit block.

8. The 32-bit output of step 7 is added modulo 2 to the left-most 32 bits of the input block, denoted Li-1, yielding Ri, which is the rightmost 32 bits of the 64-bit output block.

Steps 1 - 8 are repeated 16 times; this constitutes the major part of the product transformation. The last step is a block transformation (i.e. exchange) of the left and right halves of the output of the last iteration. The deciphering process is the exact reversal of the encipherment process, in reverse order, K16 to K1.

```
                    Figure 23-3
               One Iteration of DEA
 ZDDDDDDDDDDDDDDDDD?    ZDDDDDDDDDDDDDDDDD?
 3    Li-1        3     3    Ri-1        3
 @DDDDDDDDDDDDDDDDDY    @DDDDDDDDDDDDDDDDDY
 1,2,3 ..3       32      1,2,3 ...3     32
         3                        3
         3                        3
       ZDD?        ZDD?           3
       3 q3<-------3 f3<--------  o
       @DBY        @DDY           3
         3  ZDDDDDDDDDDDDDDDDDDDDDY
         3- 3---------------------o
  ZDDDDDDDDDDDDDDDDD?    ZDDDDDDDDDADDDDDDDDDDD?
  3                3     3                     3
  3    Li = Ri-1   3     3 Ri=Li-1(q)f(Ri-1,K1)3
  @DDDDDDDDDDDDDDDDDY    @DDDDDDDDDDDDDDDDDDDDDY
  1,2,3 ..3      32       1,2,3 ... 3       32
          3                         3
          ~                         ~
```

12. COMPONENTS OF THE DATA ENCRYPTION ALGORITHM

Lets dissect the algorithm:

There are 6 components that make up the DEA:

1. The key schedule calculations, which generate 16 subkeys,

2. The XOR or modulo-2 addition [we use alt241 to represent this (q)],

3. The cipher function, which comprises the main operations in the product transformation,

4. The block transposition that yields the "preoutput block" which serves as input to the inverse initial permutation,

5. The initial permutation described as a selection table,

6. The inverse initial permutation described as a selection table.

13. KEY SCHEDULE CALCULATIONS

The key schedule calculations generate 16 subkeys, referred to as Kn, required for enciphering and deciphering processes. Each Kn is 48 bits long and is derived through the use of permutation, selection, and shifting operations.

The bits are numbered 1 - 64, going from left to right. Parity bits are numbered 8,16,24,32,40,48,56, and 64
leaving the following bits for key schedule computations:

```
1  through 7
9  through 15
17 through 23
25 through 31
33 through 39
49 through 55
57 through 63
```

The key schedule calculations are executed as follows:

1. The non-parity bits in the key go through a permutation operation yielding two 28 bit blocks
   denoted by C0 and D0. This is the starting point for computing the subkeys.

2. C0 and D0 are circularly left shifted one place yielding C1 and D1

3. Selected bits from C1 and D1 are tapped off yielding subkey K1.

4. C1 and D1 are circularly left shifted one place yielding C2 and D2

5. Selected bits from C2 and D2 are tapped off yielding subkey K2.

6. The process continues for subkeys K3 through K16. Each Ci and Di is obtained from the preceding value
   after a prescribed number of circular left shifts.

The key schedule calculations are summarized in Figure 23-4. Each subkey, denoted by Ki, is obtained through a selection operation from Ci and Di. Ci and Di are obtained from Ci-1 and Di-1, respectively, through prescribed shift operations.

```
               Figure 23-4
          Key Schedule Calculations

               [64-bit Key]
                     |
                     |
            (permuted choice 1)
       ZDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD?
  ZDDDDADDDD?                  ZDDDDADDDD?
  3   C0    3                  3   D0    3
  @DDDDDDDDDY                  @DDDDDDDDDY
       |                            |
  [circular left              [circular left
   shift 1 place]             shift 1 place]
       |                            |
  ZDDDDDDDDDD?                 ZDDDDDDDDDD?
  3   C1    3                  3   D1    3
  @DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |         choice 2)
  |                             |           @DD>K1
  |                             |
[circular left              [circular left
 shift 1 place]             shift 1 place]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C2    3                  3   D2    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |         choice 2)
  |                             |           @DD>K2
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |

ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C3    3                  3   D3    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |         choice 2)
  |                             |           @DD>K3
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C4    3                  3   D4    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |         choice 2)
  |                             |           @DD>K4
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C5    3                  3   D5    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2)
  |                             |            @DD>K5
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C6    3                  3   D6    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2)
  |                             |            @DD>K6
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C7    3                  3   D7    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2)
  |                             |            @DD>K7
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C8    3                  3   D8    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2)
  |                             |            @DD>K8
  |                             |
  |                             |
[circular left              [circular left
 shift 1 place ]            shift 1 place ]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C9    3                  3   D9    3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2)
  |                             |            @DD>K9
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C10   3                  3   D10   3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2 )
  |                             |            @DD>K10
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C11   3                  3   D11   3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2 )
  |                             |            @DD>K11
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C12   3                  3   D12   3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2 )
  |                             |            @DD>K12
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C13   3                  3   D13   3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2 )
  |                             |            @DD>K13
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C14   3                  3   D14   3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2 )
  |                             |            @DD>K14
  |                             |
  |                             |
[circular left              [circular left
 shift 2 places]            shift 2 places]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C15   3                  3   D15   3
@DDDDDDDDDY                  @DDDDDDDDDY
  |    |                        |  |
  |    -------------------------|-------->(permuted
  |                             |          choice 2 )
  |                             |            @DD>K15
  |                             |
  |                             |
[circular left              [circular left
 shift 1 place ]            shift 1 place ]
     |                            |
ZDDDDDDDDD?                  ZDDDDDDDDD?
3   C16   3                  3   D16   3
@DDDDDDDDDY                  @DDDDDDDDDY
       |                            |
       ---------------------------------->(permuted
                                           choice 2 )
                                             @DD>K16
```

Initially CO and DO are obtained from the 64-bit key through the use of permuted choice 1, which is summarized in Table 23-1.

```
                    Figure 23-5
                 Permuted Choice 1
                 to calculate C0 & D0

           [   cipher key (64-bits    ]
           1 2 3 ...   |         63 64

           |                        |
           |  leftmost   rightmost  |
           |     bits       bits    |

               (permuted choice 1)

           |                        |

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

           |                         |
           |                         |
[    C0 (28 bits)  ]        [  D0 (28 bits)       ]
1 2 3          27 28        1 2 3             27 28
```

The cipher key active bits used to determine C0 are 57,49...36 etc. Similarly, the bits of D0 are respectively bits 63,55..4 of the cipher key.

Permuted choice 2 is used to select a particular key Kn from the concatenation of Cn and Dn. Cn and Dn are 28 bits long so that CnDn combined has bits that run from 1-56.

```
                Figure 23-6
           ( Compression Permutation )

              Permuted Choice 2
            calculation of subkey Ki

                 <- 56 bits ->

      <- 28 bits ->    |   <- 28 bits ->
    --------------------------------------
    |      Ci          |        Di       |
    --------------------------------------
    1 2 3 ..                         55 56

            |                      |
            |----------o-----------|
                       |
                       |
             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

                       |
                       |

            [        Ki       ]
            1 2 3            48

                <-48 bits->
```

The number of circular left shifts for each iteration in the key schedule calculation are:

Iteration	# of circular 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

14. XOR - MODULAR- 2 ADDITION

A bit-by-bit addition modular 2 operation is used in many steps of DEA; We denote it by q and define it as follows:

```
q     0   0
 ---------
0   | 0   1
1   | 1   0
```

so for example:

```
  1 0 0 1 0 1 1 0
q 1 1 0 1 0 0 1 1
  ---------------
= 0 1 0 0 0 1 0 1
```

15. CIPHER FUNCTION

The cipher function comprises the main operations in the product transformation, f(A,Kn) where A is a string of 32 data bits representing Ri for encryption or Li for decryption, and Kn is a 48 bit subkey determined by the key schedule.

The cipher function combines the following operations:

1. A selection operation E that operates on the argument A of 32 bits and produces a 48 bit result.

2. A XOR addition which adds the result of the selection operation E and the 48 bit key Kn on a bit by bit basis yielding a 48 bit results.

3. A unique set of selection functions Si that converts the 48 bit result of step 2 to a set of 32 bits.

4. A permutation operation P that operates on the result of step 3 and produces a 32 bit result.

```
      Figure 23-7
     Cipher Function

  <-32 bits->
[     A     ]

      |

      E

      +
      +
  <-48 bits->                <-48 bits->
[     result   ]          [     Kn        ]

       3                          3
       3                          3
       @DDDDDDDDDDD q DDDDDDDDDDDY

                    |

                ZDDDDDDD?
                3  Si   3
                3 S Box 3
                @DDDDDDDY
                    |
      <-32 bits->         <-32 bits->
     [    A     ]        [  result   ]

                    |

                    P

                    |

               ( f(A, Kn )
              <- 32 bits ->
```

The selection function E, in Figure 23-8, yields a 48-bit result wherein the bits of the result are respectively 32,1,2,...1,.etc of the symbolic argument A which may represent Ri or Li depending on the function.

```
     Figure 23-8
     E Operation
 (Expansion Permutation)

  < -32 bits - >

  [     A     ]

        |
        |

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

        |
        |

[   E result     ]
  <- 48 bits ->
```

16. S BOXES

The unique set of selection functions Si take a 6-bit block as input and yield a 4 bit result. A selection function represented by a 4 X 16 matrix of numbers used in a prescribed manner.

Input to the unique S Boxes 1-8 (selection functions Si) is a 48-bit block, denoted symbolically as B1B2B3B4B5B6B7B8. Each Bi contains 8 bits. S1 is used for B1, etc. The result of the selection of Si with Bi as an argument Si(Bi) is computed:

1. The first and lasts bits of Bi represent a binary number in the range of 0 - 3 denoted m.

2. The middle four bits of Bi represent a binary number in the range of 0 - 15 denoted n.

3. Using zero-origin indexing, the number located in the mth row and nth column of the Si's matrix is selected as a four bit binary block.

4. The result of step 3 is the output of the selection function Si.

The output of a complete set of selection functions, is a bit string IS:

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)

each of the Si's are 4 bit outputs.

For example:

```
                Input to S1

                1 0 1 1 0 0
                ---
               =2

                  -------
                  = 6

         use row 2, column 6 of S1
```

Table 23-1 gives the matrices corresponding to the selection function S1 through S8.

Table 23-1

S BOX MATRICES

Matrices for the Selection functions S1 through S8

S1

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

S2

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

S3

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

S4

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

S5

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

S6

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

S7

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

S8

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 output of the set of eight selection functions, S1 through S8 is a string of 32 bits. This 32-bit output goes through a permutation operation P which yields a 32-bit result and completes the cipher function. P does not complete the algorithm, but only the cipher function denoted by f(A, Kn). This final permutation in the cipher function is given in Figure 23-9. The permutation operation P yields a 32-bit result wherein the bits of the result are 16, 7, ... etc of the 32-bit result of the set of selection functions.

```
              Figure 23-9
              P Operation
 Permutation Operation P of the Cipher Function

       Output of Selection Functions

            < -32 bits - >

            [     A      ]

                  |
                  |

             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

                  |
                  |

        [   Result of Cipher Function  ]
               <- 32 bits ->
```

17. PREOUTPUT BLOCK

The output of the last iteration in the product transformation goes through a block transformation yielding a 64-bit result called the preoutput block. See Figure 23-10. It is a simple exchange of R16 and L16. The bits of R16 are followed by the bits of L16 and constitutes a 64-bit block, with bits numbered from 1 - 64 from left to right.

```
                Figure 23-10
                Preoutput Block


   <- 32 bits ->              <- 32 bits ->
  [    L16     ]              [   R16    ]   Output
                                              of
         |........................|          Cipher
                                             Function
                     |
                     |
                     |
              (Block Transformation)

                     |
                     |
                                        Preoutput
         [    R16    |   L16    ]       block
                <-- 64 bits -->
```

18. IP

The initial permutation is the first step in the standard data encryption algorithm and is a key-independent permutation as shown in Figure 23-11. The output of the IP are respectively, bits 58, 50,...2 etc. of the plain text input to the block. The result of the IP is a 64-bit block. The leftmost 32 bits constitute L0 and the rightmost 32 bits constitute R0. L0 and R0 are the initial input blocks to the product transformation.

```
               Figure 23-11

                   IP

              <-- 64 bits -->
        [     Plain Text Input    ]
        1,2,3..              63,64

                    |
                    |


          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   IP
          57 49 41 33 25 17  9  1   matrix
          59 51 43 35 27 19 11  3
          61 53 45 37 29 21 13  5
          63 55 47 39 31 23 15  7

                    |
                    |


              <-- 64 bits -->             Result of
        [     Plain Text Input      ]       IP
        1,2,3..                63,64

        |                           |

    <- 32 bits ->            <- 32 bits ->
   [    L0      ]           [     R0     ]
   1,2,3.....   32          33,34.....   64


             >>     Input to cipher function
```

19. IP -1 INVERSE INITIAL PERMUTATION

The output of the product transformation is the preoutput block, which is subjected to a permutation which is the inverse of the IP. The IP-1 is shown in Figure 23-12. The output of IP-1, which is synonymously, the cipher text output of the algorithm, is bits 40, 8...to 25 of the preoutput block.

The 64-bit cipher text output of the DEA can be used as a string of data bits for transmission or storage, or may be converted back into BCD characters for further processing.

```
      Figure 23-12

            IP-1

       <-- 64 bits -->
[     Preoutput Block       ]
1,2,3.........         63,64

             |
             |


  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   IP-1
  36  4 44 12 52 20 60 28   matrix
  35  3 43 11 51 19 59 27
  34  2 42 10 50 18 58 26
  33  1 41  9 49 17 57 25

             |
             |


       <-- 64 bits -->             Result of
[     Cipher Text           ]       IP-1
1,2,3..                63,64
```

19. THE ENCIPHERING PROCESS

The enciphering process can be summarized symbolically. Given two blocks L and R and using the convention that LR denotes the block consisting of bits of L followed by bits of R, the initial permutation (IP) is specified as:

L0R0 <--- IP (<64-bit input block>)

Let KS denote the key schedule calculations, where the function KS yields a 48-bit subkey Kn for arguments n and KEY, where Key is a 64-bit cipher key, it follows that:

Kn <-- KS(n, KEY)

denotes the calculation of subkey Kn.

The 16 iterations in the product transformation that use the cipher function are then represented symbolically as:

Ln <-- Rn-1
Rn <-- Ln-1 (q) f(Rn-1, Kn)

(where: alt 241 = (q) is the symbol for XOR w/o the circle around it as published elsewhere. XOR denotes a bit-by-bit modulo-2 addition.)

and f is the cipher function. Ln and Rn are computed as n goes from 1 -16. The preoutput block is R16L16 and the result of the DEA is specified as:

<64-bit cipher text> <-- IP-1 (R16L16)

20. THE DECIPHERING PROCESS

The process of deciphering a 64-bit ciphertext message block involves the same algorithm as encipherment, as stated in FIPS publication 46 (Data Encryption Standard, U.S. Department of Commerce, National Bureau of Standards, FIPS publication 46, 1977 January 15, p.10):

.....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.

This is precisely the case because the IP and IP-1 are by definition inverses of each other.

Applying the notation given above, the result of the initial permutation (IP) is:

R16L16 <-- IP(<64-bit cipher text>)

where the expression takes the final block transformation into consideration. The 16 iterations in the product transformation are represented:

Rn-1 <-- Ln
Ln-1 <-- Rn (q) f(Ln,Kn)

where the Ln and Rn are computed as n goes from 16 - 1. The result of the decipherment is then specified:

<64-bit plain text> <-- IP-1 (L0R0)

21. DETAILED BIT-BY-BIT EXAMPLE

Both Katzan and Meyer present detailed (and I mean detailed - both encipherment and decipherment) examples of DES. [KATZ], [MEYE] All the references cited previously give examples of DES in varying levels of detail.

22. AVALANCHE CRITERIA

We have seen that in the E operation (expansion permutation) the right half of the data Ri is expanded from 32 bits to 48 bits. Because this operation changes the order of the bits as well as repeating certain bits, it is a true expansion permutation. This operation has two purposes: it makes the result the same size as the key for the XOR operation, and it provides a longer result that can be compressed during the substitution operation.

Neither of those is its main cryptographic purpose, though. By allowing one bit to affect two substitutions, the dependency of the output bits on the input bits spreads faster. This is called the avalanche criteria. Much of DES's design revolves around reaching as quickly as possible the condition of having every bit of the ciphertext depend on every bit of the plaintext and every bit of the key. Meyer and Matyas and Konheim discuss this principle in detail. [MEYE], [KONH] Meyer notes that statistical output dependency is reached after just five rounds of DES. [MEYE] Konheim's data suggests eight rounds are required to reach full output dependency of the data.

23. MODES OF DES

DES can be used for encryption in several officially defined modes. Some are more secure than others. ECB (electronic codebook) mode simply encrypts each 64-bit block of plaintext one after another under the same 56-bit DES key. In CBC (cipher block chaining) mode, each 64-bit plaintext block is XORed with the previous ciphertext block before being encrypted with the DES key. Thus the encryption of each block depends on previous blocks and the same 64-bit plaintext block can encrypt to different ciphertext depending on its context in the overall message. CBC mode helps protect against certain attacks, although not against exhaustive search or differential cryptanalysis. CFB (cipher feedback) mode allows one to use DES with block lengths less than 64 bits. Detailed descriptions of the various DES modes can be found in [SCH2]

In practice, CBC is the most widely used mode of DES, and is specified in several standards. For additional security, one could use triple encryption with CBC, but since single DES in CBC mode is usually considered secure enough, triple encryption is not often used.

24. HARDWARE AND SOFTWARE IMPLEMENTATIONS OF DES

As of this writing the recordholder for the fastest DES chip is a prototype developed at DEC. It supports ECB and CBC modes and is based on a GaAs gate array of 50,000 transistors. Data can be encrypted and decrypted at a rate of 1 gigabit per second, which translates to 16.8 million blocks per second. This is impressive!

VLSI's 6868 "gatekeeper chip" performs DES encryption in only 8 clock cycles or less, but does DES ECB triple encryption in 25 clock cycles, and the OCB or CBC triple-DES in 35 clock cycles.

A software implementation of DES on the IBM 3090 mainframe can perform 32,000 DES encryptions per second. Table 23-2 details some of the commercial DES chips and computer implementations of DES.

Table 23-2a
Commercial DES Chips

Manufacturer	Chip	Year	Clock	Data Rate
AMD	Am9518	1981	3 Mhz	1.3 MBytes/s
AMD	Am9568	1981	4 Mhz	1.5 MBytes/s
AMD	Amz8086	1982	4 MHz	1.7 MBytes/s
ATT	T7000A	1985	4 Mhz	1.9 MBytes/s
CE-Infosys	C003	1992	20 Mhz	12.5 MBytes/s
CE-Infosys	C003a	1994	30 Mhz	20.0 MBytes/s
Cryptech	12C102	1989	20 Mhz	2.8 MBytes/s
Newbridge	20C03a	1991	25 Mhz	3.6 MBytes/s
Newbridge	20C03w	1992	8 Mhz	0.6 MBytes/s
Newbridge	95C68-09	1993	33 Mhz	14.7 MBytes/s
Pijnenburg	PCC100	1993	25 Mhz	2.5 MBytes/s
Semaphore Communications	284	1993	40 Mhz	35.5 MBytes/s
VLSI Technology	VM007	1993	32 Mhz	200.0 MBytes/s
VLSI Technology	VM009	1993	33 Mhz	14.0 MBytes/s
Vlsi Technology	6868	1995	32 Mhz	64.0 MBytes/s
Western Digital	2001	1984	3 Mhz	0.2 MBytes/s

Table 23-2b
DES Speeds

Processor	Speed (MHz)	DES Blocks per second
8088	4.7	370
68000	7.6	900
80286	6	1,100
68020	16	3,500
68030	16	3,900
80386	25	5,000
68030	50	10,000
68040	25	16,000
68040	40	23,000
80486	66	43,000
Sun ELC		26,000
HyperSparc		32,000
RS6000-350		53,000
Sparc 10/52		84,000
DEC Alpha - 4000/610		154,000
HP9000/887	125	196,000

25. SECURITY OF DES

DES is a secret-key, symmetric cryptosystem: when used for communication, both sender and receiver must know the same secret key, which is used both to encrypt and decrypt the message. DES can also be used for single-user encryption, such as to store files on a hard disk in encrypted form. In a multi-user environment, secure key distribution may be difficult; public-key cryptography was invented to solve this problem. DES operates on 64-bit blocks with a 56-bit key. It was designed to be implemented in hardware, and its operation is relatively fast (previous section). It works well for bulk encryption, that is, for encrypting a large set of data.

NIST has recertified DES as an official U.S. government encryption standard every five years; DES was last recertified in 1993, by default. NIST has indicated, however, that it may not recertify DES in 1997. Since NIST has asked for public submissions/comment on development of the Advanced Encryption Standard (AEC), I suspect that DES is on its last hurrah.

DES has never been officially been ``broken'', despite the efforts of many researchers over many years. The obvious method of attack is brute-force exhaustive search of the key space; this takes 2^{55} steps on average. Early on it was suggested that some country could build a special-purpose computer capable of breaking DES by exhaustive search in a reasonable amount of time. Later, Hellman demonstrated a time-memory trade-off that allows improvement over exhaustive search if memory space is plentiful, after an exhaustive precomputation. These ideas fostered doubts about the security of DES. There were accusations that the NSA had intentionally weakened DES. Despite these suspicions, no feasible way to break DES faster than exhaustive search was discovered. in 1991, the cost of a specialized computer to perform exhaustive search was estimated by Wiener at one million dollars. In 1997 dollars and technology, this figure is closer to $100,000, a figure well within a small corporation's economics.

The first attack on DES that is better than exhaustive search was announced by Eli Biham and Adi Shamir using a new technique known as differential cryptanalysis. This attack requires encryption of 2^{47} chosen plaintexts, i.e., plaintexts chosen by the attacker. Although a theoretical breakthrough, this attack is not practical under normal circumstances because it requires the attacker to have easy access to the DES device in order to encrypt the chosen plaintexts. Another attack, known as linear cryptanalysis, does not require chosen plaintexts. Both of these attacks are described in Schneier's book. [SCH2], [RSA2]

The consensus is that DES, when used properly, is secure against all but the most powerful players. In fact, triple encryption DES may be secure against anyone at all. Biham and Shamir have stated that they consider DES secure. It is used extensively in a wide variety of cryptographic systems, and in fact, most implementations of public-key cryptography include DES at some level.

How secure is DES today? Tough question. If we consider just key length, a brute-force DES-cracking machine can find a key in an average of 3.5 hours in 1993. This figure has been reduced by an order of magnitude in 1996. [NICH]

Winn Scwartau writes that the NSA had built a massively parallel DES-cracking machine as early as the mid-1980's. [SCHW] Harris Corporation built a machine using the Cray Y-MP as a front-end. Supposedly, the problem is reduced by several orders of magnitude. Both contextual and statistical attacks can reduce the DES effective key size. Schneier reports as a "rumor" that NSA routinely cracks DES in 3 to 5 minutes, depending on the amount of preprocessing, at a cost of $50,000 per machine! NSA may also have giant databanks of plain and ciphertext to perform the statistical calculations on and then go out to an array of optical disks and retrieve the key. [SCH2]

26. RSA CHALLENGE

On Tuesday, 28 January, my machine had just started cracking when this message came over the net:

"EXPORTABLE CRYPTOGRAPHY TOTALLY INSECURE: CHALLENGE CIPHER BROKEN IMMEDIATELY "

January 28, 1997 - Ian Goldberg, a UC Berkeley graduate student, announced today that he had successfully cracked RSA Data Security Inc.'s 40-bit challenge cipher in just under 3.5 hours.

RSA challenged scientists to break their encryption technology, offering a $1000 award for breaking the weakest version of the code. Their offering was designed to stimulate research and practical experience with the security of today's codes.

The number of bits in a cipher is an indication of the maximum level of security the cipher can provide. Each additional bit doubles the potential security level of the cipher. A recent panel of experts recommended using 90-bit ciphers, and 128-bit ciphers are commonly used throughout the world, but US government regulations restrict exportable US products to a mere 40 bits.

Goldberg's announcement, which came just three and a half hours after RSA started their contest, provides very strong evidence that 40-bit ciphers are totally unsuitable for practical security. "This is the final proof of what we've known for years: 40-bit encryption technology is obsolete," Goldberg said.

The US export restrictions have limited the deployment of technology that could greatly strengthen security on the Internet, often affecting both foreign and domestic users. "We know how to build strong encryption; the government just won't let us deploy it. We need strong encryption to uphold privacy, maintain security, and support commerce on the Internet -- these export restrictions on cryptography must be lifted," Goldberg explained. Fittingly, when Goldberg finally unscrambled the challenge message, it read: "This is why you should use a longer key."

Goldberg used UC Berkeley's Network of Workstations (known as the NOW) to harness the computational resources of about 250 idle machines. This allowed him to test 100 billion possible "keys" per hour – analogous to safecracking by trying every possible combination at high speed. This amount of computing power is available with little overhead cost to students and employees at many large educational institutions and corporations.

Goldberg is a founding member of the ISAAC computer security research group at UC Berkeley. In the Fall of 1995, the ISAAC group made headlines by revealing a major security flaw in Netscape's web browser.

On 29 January 1997, Conrad Schlundt reported that another successful attack on RC5 (40 bit) had been made by Germano Caronni to the challenge at:

http://www.rsa.com/rsalabs/97challenge/

Quote:

"Some of you may remember my mail calling for CPU power Tuesday evening. Thank you for all donations. We broke the challenge, and got the key (F043F18131), and the message: 'This is why you should use a longer key.'

A group of persons having access to the student clusters of German and Swiss universities, joined by several researchers, assistants and other students in the last minute, collected voluntary access to > 1160 CPUs. 966 different machines were involved. We used these machines for one evening (local time) to attack the 40 bit challenge of RC5, and succeeded after about 4 hours.

The machines that joined in ranged from 486/33 PC's to Alphas and heavy multiprocessor machines such as 40 processor SUNs and SGIs. A significant fraction of the whole computing power came from Ultra Sparcs and Pentiums.

At 09:05 AM PST we finally gained access to the RSA DSI challenge page, and our search took us until 12:54 PM PST, about 4 hours.

Starting point in the key space was 00 00 00 00 00 and we had to progress to the MSB of 81, thus we actually searched half of the key space. 558098542641 keys were computed in 13798 seconds, this are about 40.4 million keys per second. An average machine has thus calculated 42000 keys per second. Now, this is not top performance, especially if you compare it to the performance the other known contestant (Ian Goldberg) could muster. [100 mio. keys per second, 3.5 hours to solution].

Due to a severe bug in our server (which I have to say is entirely my fault), our rate is much lower than the one possible. We had about 1000 machines at our best point, and 800 for the sustained attack.

But assume you just get a campus with 150 Ultra Sparc stations, where each can to 156250 keys per second. Any student can then crack that 40 bit key in 23812 seconds (on average), or otherwise said *in one night*. I can't help but repeat: 40 bit keys are *worse* than no encryption at all, as they give you a false feeling of security. "

27. PRACTICAL CONSIDERATIONS

When using DES, there are several practical considerations that can affect the security of the encrypted data. One should change DES keys frequently, in order to prevent attacks that require sustained data analysis. In a communications context, one must also find a secure way of communicating the DES key to both sender and receiver. Use of RSA or some other public-key technique for key management solves both these issues: a different DES key is generated for each session, and secure key management is provided by encrypting the DES key with the receiver's RSA public key. RSA, in this circumstance, can be regarded as a tool for improving the security of DES (or any other secret key cipher).

If one wishes to use DES to encrypt files stored on a hard disk, it is not feasible to frequently change the DES keys, as this would entail decrypting and then reencrypting all files upon each key change. Instead, one should have a master DES key with which one encrypts the list of DES keys used to encrypt the files;

one can then change the master key frequently without much effort.

A powerful technique for improving the security of DES is triple encryption, that is, encrypting each message block under three different DES keys in succession. Triple encryption is thought to be equivalent to doubling the key size of DES, to 112 bits, and should prevent decryption by an enemy capable of single-key exhaustive search. Of course, using triple-encryption takes three times as long as single-encryption DES.

It has been frequently asked whether DES encryption is closed under composition; i.e., is encrypting a plaintext under one DES key and then encrypting the result under another key always equivalent to a single encryption under a single key? Algebraically, is DES a roup? If so, then DES might be weaker than would otherwise be the case; for a more complete discussion. However, the answer is no, DES is not a group; this issue was settled only recently, after many years of speculation and circumstantial evidence. This result seems to imply that techniques such as triple encryption do in fact increase the security of DES. [SCHN], [RSA2]

28. EXPORT RESTRICTIONS

Export of DES, either in hardware or software, is strictly regulated by the U.S. State Department and the NSA. The government rarely approves export of DES, despite the fact that DES is widely available overseas; financial institutions and foreign subsidiaries of U.S. companies are exceptions.

29. WEAK KEYS

Because of the way the initial key is modified to get a subkey for each round of the algorithm, certain keys are weak keys. Since the original DES key is split in half, and each half is shifted independently, if all the bits are 0's or 1's, then the key used for any cycle is the same for all cycles: K1 = K2 = K3 =Ki =Kn. Additionally, some pairs of keys encrypt plaintext to the identical ciphertext. This is due to the key generation mechanism in DES. Instead of generating 16 different subkeys, these keys generate only two different subkeys. Each of these subkeys is used eight times in the algorithm. We label these semi-weak keys. A similar problem occurs when some keys only produce only four subkeys and used only four times in the algorithm. These are possibly weak keys. Before we get uptight and say that DES is condemned to algorithm Hell, the total number of possibly weak keys is 64 out of 72,057,594,037,927,936 possible keys. Selecting a random key, the odds of picking one out of the 64 are worse than winning the Texas lottery twice in a month.

Table 23-3
DES Weak Keys in Hex

Weak Key Value (with parity bits)	Actual Key
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

DES Semiweak Key Pairs

01FE 01FE 01FE 01FE	and	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	and	E01F E01F F10E F10E
01E0 01E0 01F1 01F1	and	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	and	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	and	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	and	FEE0 FEE0 FEF1 FEF1

30. LENGTH OF THE KEY

IBM's original submission to NBS had a 112-bit key. When DES became a standard, that was reduced to 56 bits. Hellman presented an argument in 1980 against small key size: by trading memory space for time, it would be possible to speed up the searching process. By computing 2**56 possible results of encrypting a single plaintext block under every possible key. Then to break an unknown key. all that would be required would be to insert the plaintext block into the encryption stream, recover the resulting ciphertext, and look the key up. By 1987 hips performing 512,000 encryptions per second were being developed, and a version capable of checking over a million keys per second was feasible. In 1993, Michael Wiener designed a $1 million machine that could complete a brute-force attack on DES in an average of 3.5 hours. In 1990 Biham and Shamir discovered differential cryptanalysis, a technique that puts to rest the question of key length.

31. NUMBER OF ITERATIONS

Why 16 rounds? After 5 rounds every ciphertext bit is a function of every plaintext bit and every key bit. After eight rounds the ciphertext is essentially a random function of every plaintext bit and key but.

Answer: because it could be broken. Biham and Shamir's differential cryptanalysis technique works well on every number of rounds fewer than 16, using a known plaintext attack. In each case, differential cryptanalysis was more efficient than a brute force attack.

32. DIFFERENTIAL CRYPTANALYSIS and LINEAR CRYPTANALYSIS

Schneier presents a detailed discussion of both differential cryptanalysis and linear cryptanalysis. [SCH2]. [BIHA] is the definitive reference on the subject of differential cryptanalysis. Both attacks on DES can be performed in less interations than a brute-force attack. The subject is fascinating and beyond the scope of this lecture.