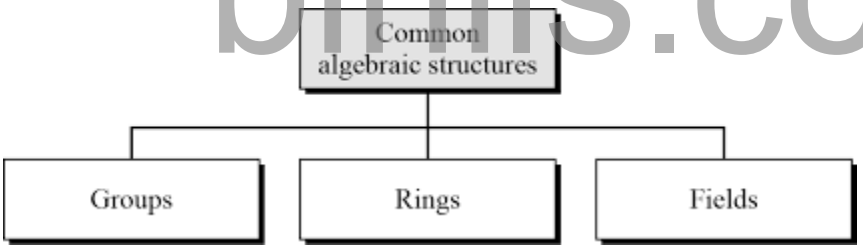
Catalog

MATHEMATICS OF SYMMETRIC KEY CRYPTOGRAPHY

- Cryptography is based on some specific areas of mathematics including number theory, linear algebra, and algebraic structures.
- Symmetric ciphers use symmetric algorithms to encrypt and decrypt data. These ciphers are used in symmetric key cryptography.
- A symmetric algorithm uses the same key to encrypt data as it does to decrypt data.

ALGEBRAIC STRUCTURES

- Algebra is about operations on sets.
- You have met many operations; for example:
  - addition and multiplication of numbers;
  - modular arithmetic;
  - addition and multiplication of polynomials;
  - addition and multiplication of matrices;
  - union and intersection of sets;
  - composition of permutations.



Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006

Sets

- A set is a well-defined collection of distinct objects, considered as an object in its own right.
- Two sets are equal if and only if they have the same members
  - That is, A = B if and only if$((x \in A) \Leftrightarrow (x \in B))$.
  - This means that, to prove that two sets are equal, you have to do two things:
    - (i) show that any element of A lies in B;
    - (ii) show that any element of B lies in A.

- (i) means that $A \subseteq B$ (that is, A is a subset of B), while
- (ii) means that $B \subseteq A$.
- So we can re-write our rule:
  - $A \subseteq B$ if and only if $((x \in A) \Rightarrow (x \in B))$,
  - $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.
- From two sets A and B we can build new ones:
  - union: $A \cup B = \{x : x \in A \text{ or } x \in B\}$;
  - intersection: $A \cap B = \{x : x \in A \text{ and } x \in B$;
  - difference: $A \backslash B = \{x : x \in A \text{ and } x \notin B\}$;
  - symmetric difference: $A \Theta B = (A \backslash B) \cup (B \backslash A)$.

Functions

- A function f from A to B is, informally, a "black box" such that, if we input an element $a \in A$, then an element $f(a) \in B$ is output.
- More formally, a function is a set of ordered pairs (that is, a subset of the cartesian product $A \times B$) such that, for any $a \in A$, there is a unique $b \in B$ such that $(a,b) \in f$; we write $b = f(a)$ instead of $(a,b) \in f$.
- The sets A and B are called the domain and codomain of f ; its image consists of the set
- $\{b \in B : b = f(a)$ for some $a \in A\}$, a subset of the codomain.
- A function f is surjective (or onto) if, for every $b \in B$, there is some $a \in A$ such that $b = f(a)$ (that is, the image is the whole codomain);
- injective (or one-to-one) if $a1 \neq a2$ implies $f(a1) \neq f(a2)$ (two different elements of A cannot have the same image);
- bijective if it is both injective and surjective.

Operations

- An operation is a special kind of function.
- An n-ary operation on a set A is a function f from $A^n = A \times \cdots \times A$ to A.
- That is, given any $a1,...,an \in A$, there is a unique element $b = f(a1,...,an) \in A$ obtained by applying the operation to these elements.
- The most important cases are $n = 1$ and $n = 2$; we usually say unary for "1- ary", and binary for "2-ary". We have already seen that many binary operations (addition, multiplication, composition) occur in algebra

Example

- Addition, multiplication, and subtraction are binary operations on R, defined by
  - f(a,b) = a+b (addition),
  - f(a,b) = ab (multiplication),
  - f(a,b) = a−b (subtraction).
- Taking the negative is a unary operation: f(a) = −a

Notation

- we often write binary operations, not in functional notation, but in either of two different ways:
  - infix notation, where we put a symbol for the binary operation between the two elements that are its input, for example a + b, a − b, a · b, a ∗ b, a ◦ b, a • b; or
  - juxtaposition, where we simply put the two inputs next to each other, as ab (this is most usually done for multiplication).
- There are various properties that a binary relation may or may not have. Here are two. We say that the binary operation ◦ on A is
  - commutative if a ◦ b = b ◦ a for all a,b ∈ A;
  - associative if (a ◦ b) ◦ c = a ◦ (b ◦ c) for all a,b,c ∈ A.
  - For example, addition on R is commutative and associative; multiplication of 2×2 matrices is associative but not commutative; and subtraction is neither.

Relations

- A binary relation R on A is a subset of A × A. If (a,b) ∈ R, we say that a and b are related, otherwise they are not related, by R.
- As with operations, we often use infix notation, for example a < b, a ≤ b, a = b, a ∼= b, a ∼ b.
- But note the difference:
- + is an operation, so a+b is a member of A;
- < is a relation, so a < b is an assertion which is either true or false.
- Example Let A = {1,2,3}.
- Then the relation < on A consists of the pairs

  {(1,2),(1,3),(2,3)},
- while the relation ≤ consists of the pairs

{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)}.

- Also like operations, there are various laws or properties that a relation may have.
- We say that the binary operation R on A is
  - reflexive if (a,a) ∈ R for all a ∈ A;
  - irreflexive if (a,a) ∈/ R for all a ∈ A;
  - symmetric if (a,b) ∈ R implies (b,a) ∈ R;
  - antisymmetric if(a,b) and (b,a) are never both in R except possibly if a = b;
  - transitive if (a,b) ∈ R and (b,c ∈ R imply (a,c) ∈ R.
- For example, < is irreflexive, antisymmetric and transitive, while ≤ is reflexive, antisymmetric and transitive.

Equivalence relations and partitions

- A binary relation R on A is an equivalence relation if it is reflexive, symmetric and transitive.
- A partition P of A is a collection of subsets of A having the properties
- (a) every set in P is non-empty;
- (b) for every element a ∈ A, there is a unique set X ∈ P such that a ∈ X.
- The second condition says that the sets in P cover A without overlapping.

Algebraic Structure

- A non empty set S is called an algebraic structure w.r.t binary operation (*) if it follows following axioms:
- Closure:(a*b) belongs to S for all a,b ∈ S.
- Ex : S = {1,-1} is algebraic structure under *
- As 1*1 = 1, 1*-1 = -1, -1*-1 = 1 all results belongs to S.
- But above is not algebraic structure under + as 1+(-1) = 0 not belongs to S.

Semi Group

- A non-empty set S, (S,*) is called a semigroup if it follows the following axiom:
- Closure:(a*b) belongs to S for all a,b ∈ S.
- Associativity: a*(b*c) = (a*b)*c ∀ a,b,c belongs to S.
- Note: A semi group is always an algebraic structure.

- Ex : (Set of integers, +), and (Matrix ,*) are examples of semigroup.

Monoid

- A non-empty set S, (S,*) is called a monoid if it follows the following axiom:
- Closure:(a*b) belongs to S for all a,b ∈ S.
- Associativity: a*(b*c) = (a*b)*c ∀ a,b,c belongs to S.
- Identity Element: There exists e ∈ S such that a*e = e*a = a ∀ a ∈ S
- Note: A monoid is always a semi-group and algebraic structure.
- Ex : (Set of integers,*) is Monoid as 1 is an integer which is also identity element . (Set of natural numbers, +) is not Monoid as there doesn't exist any identity element. But this is Semigroup.
- But (Set of whole numbers, +) is Monoid with 0 as identity element.

Group

- A non-empty set G, (G,*) is called a group if it follows the following axiom:
- Closure:(a*b) belongs to G for all a,b ∈ G.
- Associativity: a*(b*c) = (a*b)*c ∀ a,b,c belongs to G.
- Identity Element:There exists e ∈ G such that a*e = e*a = a ∀ a ∈ G
- Inverses:∀ a ∈ G there exists a^{-1} ∈ G such that $a*a^{-1} = a^{-1}*a = e$

Note:

- A group is always a monoid, semigroup, and algebraic structure.
- (Z,+) and Matrix multiplication is example of group.

Abelian Group or Commutative group

- A non-empty set S, (S,*) is called a Abelian group if it follows the following axiom:
- Closure:(a*b) belongs to S for all a,b ∈ S.
- Associativity: a*(b*c) = (a*b)*c ∀ a,b,c belongs to S.
- Identity Element:There exists e ∈ S such that a*e = e*a = a ∀ a ∈ S
- Inverses:∀ a ∈ S there exists a^{-1} ∈ S such that $a*a^{-1} = a^{-1}*a = e$
- Commutative: a*b = b*a for all a,b ∈ S
- Note : (Z,+) is a example of Abelian Group but Matrix multiplication is not abelian group as it is not commutative.

MODULAR ARITHMETIC

The Modulus

- If a is an integer and n is a positive integer, we define a mod n to be the remainder when a is divided by n . The integer n is called the modulus. Thus, for any integer a , we can rewrite Equation a=qn+r as follows:

$$a = qn + r \qquad 0 \le r < n;\ q = \lfloor a/n \rfloor$$

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

- Example: $11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$
- Two integers a and b are said to be congruent modulo n, if (a mod n)=(b mod n).
- This is written as a≡b (mod n) $73 \equiv 4 \pmod{23}; \qquad 21 \equiv -9 \pmod{10}$
- Note that if a ≡0(mod n), then n/a

Properties of Congruence

1. $a \equiv b \pmod{n}$ if $n|(a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

Modular Arithmetic Operations

- Modular arithmetic exhibits the following properties:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

- Example:

$11 \bmod 8 = 3; 15 \bmod 8 = 7$
$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$
$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$
$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$
$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$
$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$
$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

Congruent numbers

- Integers that leave the same remainder when divided by the modulus m are somehow similar, however, not identical. Such numbers are called "congruent" .

- For instance, 1 and 13 and 25 and 37 are congruent mod 12 since they all leave the same remainder when divided by 12.
- We write this as $1 \equiv 13 \equiv 25 \equiv 37 \bmod 12$. However, they are not congruent mod 13. Why not? Yield a different remainder when divided by 13.
- Find 5 numbers that are congruent to

1) 7 mod 5	2,12,17,-3,-10
2) 7 mod 25	32,57,82,-18,-43
3) 17 mod 25.	42,67,92,-8,-33

Euclid's algorithm

- The Euclidean algorithm, or Euclid's algorithm, is an efficient method for computing the greatest common divisor (GCD) of two integers (numbers), the largest number that divides them both without a remainder.
- The Euclidean algorithm can be based on the following theorem: For any nonnegative integer a and any positive integer b ,

  gcd(a,b) = gcd(b, a mod b)
- Example $\gcd(55, 22) = \gcd(22, 55 \bmod 22) = \gcd(22, 11) = 11$

The Algorithm

- The Euclidean Algorithm for finding GCD(A,B) is as follows:
- If A = 0 then GCD(A,B)=B, since the GCD(0,B)=B, and we can stop.
- If B = 0 then GCD(A,B)=A, since the GCD(A,0)=A, and we can stop.
- Write A in quotient remainder form (A = B·Q + R)
- Find GCD(B,R) using the Euclidean Algorithm since GCD(A,B) = GCD(B,R)

$$\gcd(18, 12) = \gcd(12, 6) = \gcd(6, 0) = 6$$
$$\gcd(11, 10) = \gcd(10, 1) = \gcd(1, 0) = 1$$

Example:

- Find the GCD of 270 and 192

- A=270, B=192
  - A ≠0
  - B ≠0

- Use long division to find that 270/192 = 1 with a remainder of 78. We can write this as: 270 = 192 * 1 +78

- Find GCD(192,78), since GCD(270,192)=GCD(192,78)
  - A=192, B=78
  - A ≠0
  - B ≠0
  - Use long division to find that 192/78 = 2 with a remainder of 36. We can write this as: 192 = 78 * 2 + 36
- Find GCD(78,36), since GCD(192,78)=GCD(78,36)
  - A=78, B=36
  - A ≠0
  - B ≠0
  - Use long division to find that 78/36 = 2 with a remainder of 6. We can write this as: 78 = 36 * 2 + 6
- Find GCD(36,6), since GCD(78,36)=GCD(36,6)
  - A=36, B=6
  - A ≠0
  - B ≠0
  - Use long division to find that 36/6 = 6 with a remainder of 0. We can write this as: 36 = 6 * 6 + 0
- Find GCD(6,0), since GCD(36,6)=GCD(6,0)
  - A=6, B=0
  - A ≠0
  - B =0, GCD(6,0)=6
- So we have shown:
- GCD(270,192) = GCD(192,78) = GCD(78,36) = GCD(36,6) = GCD(6,0) = 6
- GCD(270,192) = 6

Properties

- GCD(A,0) = A
- GCD(0,B) = B

- If A = B·Q + R and B≠0 then GCD(A,B) = GCD(B,R) where Q is an integer, R is an integer between 0 and B-1

Congruence

- If n is a positive integer, we say the integers a and b are congruent modulo n, and write a ≡ b (mod n), if they have the same remainder on division by n.
- Example:

{…,−6,1,8,15,…} are all congruent modulo 7 because their remainders on division by 7 equal 1. {…,−4,4,12,20,…} are all congruent modulo 8 since their remainders on division by 8 equal 4.

Properties

1. a≡a for any a;
2. a≡b implies b≡a;
3. a≡b and b≡c implies a≡c;
4. a≡0 iff n|a;
5. a≡b and c≡d implies a+c≡b+d;
6. a≡b and c≡d implies a−c≡b−d;
7. a≡b and c≡d implies ac≡bd;

Congruent Matrices

Two square matrices A and B are called congruent if there exists a nonsingular matrix P such that

$$B = P^T A P,$$

where P^T is the transpose.

Groups, rings, and fields

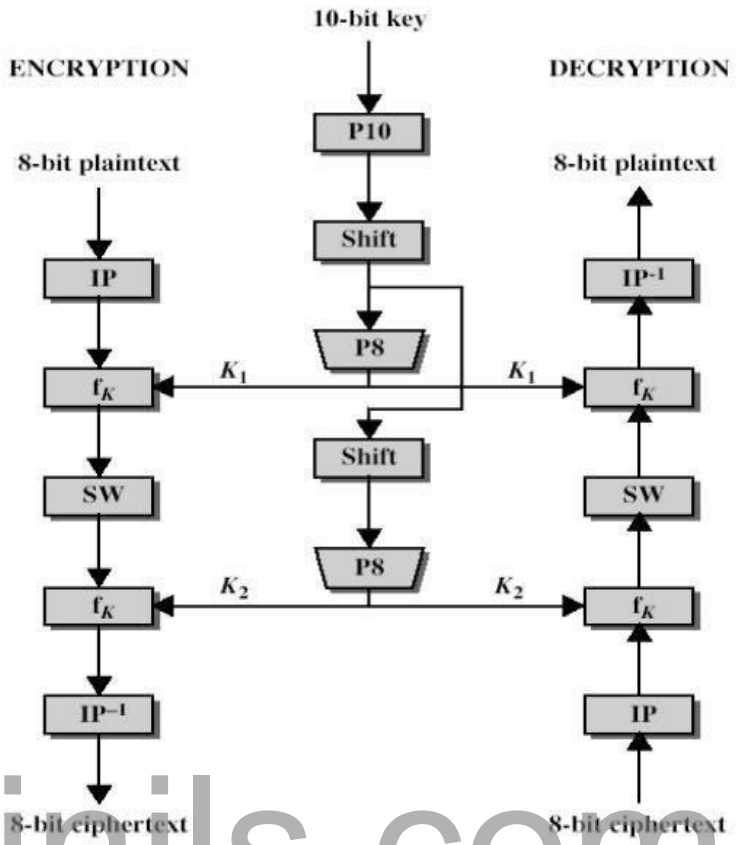
- Groups, rings, and fields are the fundamental elements of a branch of mathematics known as abstract algebra, or modern algebra.
- In abstract algebra, we are concerned with sets on whose elements we can operate algebraically; that is, we can combine two elements of the set, perhaps in several ways, to obtain a third element of the set.
- These operations are subject to specific rules, which define the nature of the set.
- By convention, the notation for the two principal classes of operations on set elements is usually the same as the notation for addition and multiplication on ordinary numbers

SIMPLIFIED DATA ENCRYPTION STANDARD (S-DES)

Simplified DES, developed by Professor Edward Schaefer of Santa Clara University, is an educational rather than a secure encryption algorithm. It has similar properties and structure to DES with much smaller parameters.

OVERVIEW

- The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output.
- The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.
- The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output. The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.



Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006

- The encryption algorithm involves five functions: an initial permutation (IP); a complex function labeled f_K, which involves both permutation and substitution operations and depends on a key input; a simple permutation function that switches (SW) the two halves of the data; the function f_K again; and finally a permutation function that is the inverse of the initial permutation (IP^{-1}).
- The use of multiple stages of permutation and substitution results in a more complex algorithm, which increases the difficulty of cryptanalysis.
- The function f_K takes as input not only the data passing through the encryption algorithm, but also an 8-bit key. The algorithm could have been designed to work with a 16-bit key, consisting of two 8-bit subkeys, one used for each occurrence of f_K. Alternatively, a single 8-bit key could have been used, with the same key used twice in the algorithm.
- A compromise is to use a 10-bit key from which two 8-bit subkeys are generated. In this case, the key is first subjected to a permutation (P10). Then a shift operation is

performed. The output of the shift operation then passes through a permutation function that produces an 8-bit output (P8) for the first subkey (K_1). The output of the shift operation also feeds into another shift and another instance of P8 to produce the second subkey (K_2).

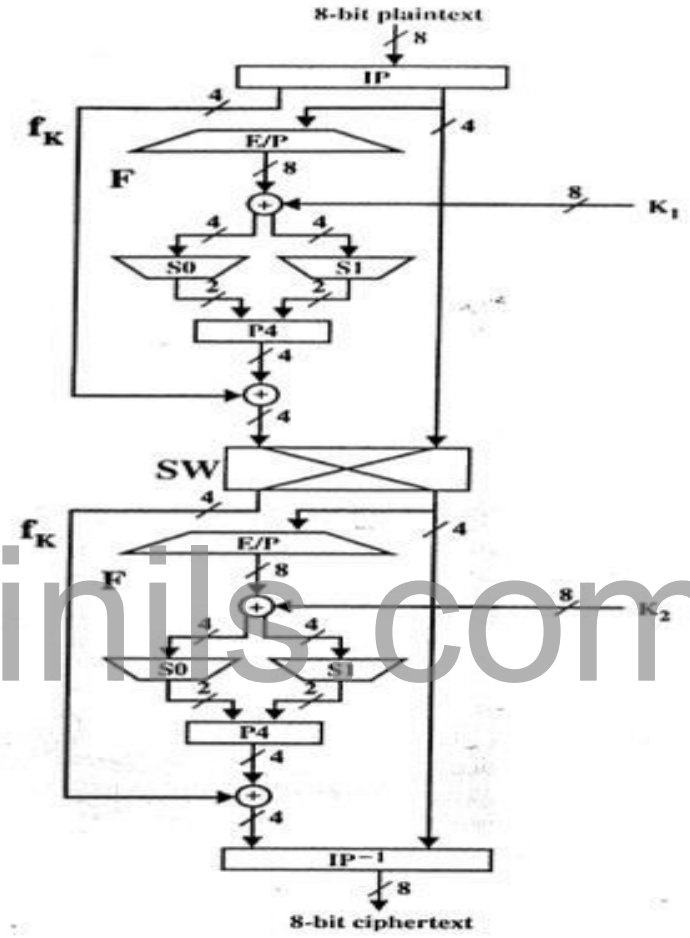


Figure 3.3 Simplified DES Scheme Encryption Detail.

Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006

- We can concisely express the encryption algorithm as a composition1 of functions:
  - IP-1 ○ fK2 ○ SW ○ fK1 ○ IP
- which can also be written as:
  - ciphertext = IP-1(fK2 (SW (fK1 (IP(plaintext)))))
- where K1 = P8(Shift(P10(key)))
  - K2 = P8(Shift(Shift(P10(key))))

- Decryption is essentially the reverse of encryption:
  - plaintext = IP-1(fK1 (SW (fK2 (IP(ciphertext)))))

KEY GENERATION

- S-DES depends on the use of a 10-bit key shared between sender and receiver.
- From this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption algorithm

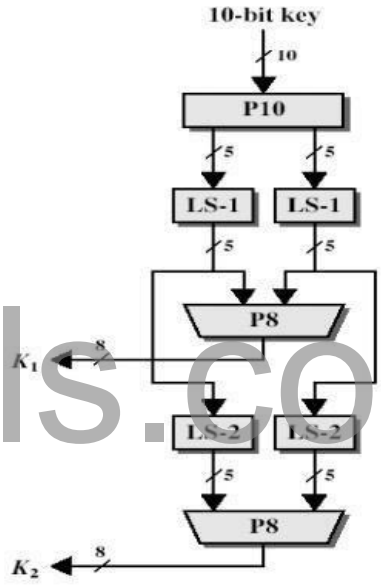


Figure: key generation for S-DES

Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006

- First, permute the key in the following fashion. Let the 10-bit key be designated as (k1, k2, k3, k4, k5, k6, k7, k8, k9, k10). Then the permutation P10 is defined as:
- P10(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5, k2, k7, k4, k10, k1, k9, k8, k6)
- P10 can be concisely defined by the display:

P10									
3	5	2	7	4	10	1	9	8	6

- This table is read from left to right; each position in the table gives the identity of the input bit that produces the output bit in that position.
- So the first output bit is bit 3 of the input; the second output bit is bit 5 of the input, and so on. For example, the key (1010000010) is permuted to (1000001100).
- Next, perform a circular left shift (LS-1), or rotation, separately on the first five bits and the second five bits. In our example, the result is (00001 11000). Next we apply P8, which picks out and permutes 8 of the 10 bits according to the following rule:

P8							
6	3	7	4	8	5	10	9

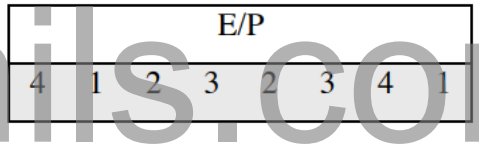
- The result is subkey 1 (K1).
- In our example, this yields (10100100) We then go back to the pair of 5-bit strings produced by the two LS-1 functions and perform a circular left shift of 2 bit positions on each string. In our example, the value (00001 11000) becomes (00100 00011). Finally, P8 is applied again to produce K2. In our example, the result is (01000011).

S-DES ENCRYPTION

- encryption involves the sequential application of five functions.
- Initial and Final Permutations
- The input to the algorithm is an 8-bit block of plaintext, which we first permute using the IP function:
  - IP 2 6 3 1 4 8 5 7
- This retains all 8 bits of the plaintext but mixes them up.
- At the end of the algorithm, the inverse permutation is used:
  - IP–1 4 1 3 5 7 2 8 6
- It is easy to show by example that the second permutation is indeed the reverse of the first; that is, IP–1(IP(X)) = X.

The Function fK

- The most complex component of S-DES is the function fK, which consists of a combination of permutation and substitution functions.
- The functions can be expressed as follows.
- Let L and R be the leftmost 4 bits and rightmost 4 bits of the 8-bit input to fK, and let F be a mapping (not necessarily one to one) from 4-bit strings to 4-bit strings.
- Then we let
  - f K(L, R) = (L ! F(R, SK), R)
- where SK is a subkey and ! is the bit-by-bit exclusive-OR function.
- For example, suppose the output of the IP stage is (10111101) and F(1101, SK) = (1110) for some key SK.
  - Then fK(10111101) = (01011101) because (1011) ! (1110) = (0101).
- We now describe the mapping F. The input is a 4-bit number (n1n2n3n4).
- The first operation is an expansion/permutation operation:



- For what follows, it is clearer to depict the result in this fashion:

$$\begin{array}{|cc|cc|} n_4 & n_1 & n_2 & n_3 \\ n_2 & n_3 & n_4 & n_1 \end{array}$$

- The 8-bit subkey K1 = (k11, k12, k13, k14, k15, k16, k17, k18) is added to this value using exclusive OR

$$\begin{array}{|cc|cc|} n_4 \oplus k_{11} & n_1 \oplus k_{12} & n_2 \oplus k_{13} & n_3 \oplus k_{14} \\ n_2 \oplus k_{15} & n_3 \oplus k_{16} & n_4 \oplus k_{17} & n_1 \oplus k_{18} \end{array}$$

- Let us rename these 8 bits:

$$\begin{array}{c|ccc|c} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \end{array}$$

- The first 4 bits (first row of the preceding matrix) are fed into the S-box S0 to produce a 2- bit output, and the remaining 4 bits (second row) are fed into S1 to produce another 2-bit output. These two boxes are defined as follows:

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

- The S-boxes operate as follows.
- The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box, and the second and third input bits specify a column of the Sbox. The entry in that row and column, in base 2, is the 2-bit output. For example, if (p0,0p0,3) = (00) and (p0,1p0,2) = (10), then the output is from row 0, column 2 of S0, which is 3, or (11) in binary. Similarly, (p1,0p1,3) and (p1,1p1,2) are used to index into a row and column of S1 to produce an additional 2 bits.
- Next, the 4 bits produced by S0 and S1 undergo a further permutation as follows

P4			
2	4	3	1

- The output of P4 is the output of the function F. The Switch Function The function fK only alters the leftmost 4 bits of the input. The switch function (SW) interchanges the left and right 4 bits so that the second instance of fK operates on a different 4 bits. In this second instance, the E/P, S0, S1, and P4 functions are the same. The key input is K2.

THE STRENGTH OF DES

- The strength of DES fall into two areas: key size and the nature of the algorithm.

The Use of 56-Bit Keys :

- With a key length of 56 bits, there are 2^{56} possible keys, which is $7.2*10^{16}$ approximately keys. Thus, on the face of it, a brute-force attack appears impractical.
- Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.
- As far back as 1977, Diffie and Hellman postulated that the technology existed to build a parallel machine with 1 million encryption devices, each of which could perform one encryption per microsecond. This would bring the average search time down to about 10 hours. The authors estimated that the cost would be about $20 million in 1977 dollars.
- DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose "DES cracker" machine that was built for less than $250,000.
- Hardware prices will continue to drop as speeds increase, making DES virtually worthless.
- There are a number of alternatives to DES, the most important of which are AES and triple DES.

The Nature of the DES Algorithm:

- Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm.
- The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes.

- This assertion is tantalizing, and over the years a number of regularities and unexpected behaviors of the S-boxes have been discovered
- Despite this, no one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes

Timing Attacks:

- A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts.
- A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
- HEVI99 reports on an approach that yields the Hamming weight (number of bits equal to one) of the secret key. This is a long way from knowing the actual key, but it is an intriguing first step.
- The authors conclude that DES appears to be fairly resistant to a successful timing attack but suggest some avenues to explore.
- Although this is an interesting line of attack, it so far appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES