Conventional Encryption / Secret Key Cryptography

Stream and Block Ciphers

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the auto keyed Vigenère cipher and the Vernam cipher. A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. Using some of the modes of operation explained in Chapter 6, a block cipher can be used to achieve the same effect as a stream cipher.

Stream V/s Block Ciphers

Advantages of Stream Ciphers

- Speed of transformation. Because each symbol is encrypted without regard for any other plaintext symbols, each symbol can be encrypted as soon as it is read. Thus, the time to encrypt a symbol depends only on the encryption algorithm itself, not on the time it takes to receive more plaintext.

- Low error propagation. Because each symbol is separately encoded, an error in the encryption process affects only that character.

Disadvantages of Stream Ciphers

- Low diffusion. Each symbol is separately enciphered. Therefore, all the information of that symbol is contained in one symbol of the ciphertext.

- Susceptibility to malicious insertions and modifications. Because each symbol is separately enciphered, an active interceptor who has broken the code can splice together pieces of previous messages and transmit a spurious new message that may look authentic.

Advantages of Block Ciphers

- High diffusion. Information from the plain-text is diffused into several ciphertext symbols. One ciphertext block may depend on several plaintext letters.

- Immunity to insertion of symbols. Because blocks of symbols are enciphered, it is impossible to insert a single symbol into one block. The length of the block would then be incorrect, and the decipherment would quickly reveal the insertion.

Disadvantages of Block Ciphers

- Slowness of encryption. The person or machine using a block cipher must wait until an entire block of plaintext symbols has been received before starting the encryption process.

- Error propagation. An error will affect the transformation of all other characters in the same block.

The Feistel Cipher

Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2k possible transformations, rather than the 2n! transformations available with the ideal block cipher.

Diffusion and Confusion

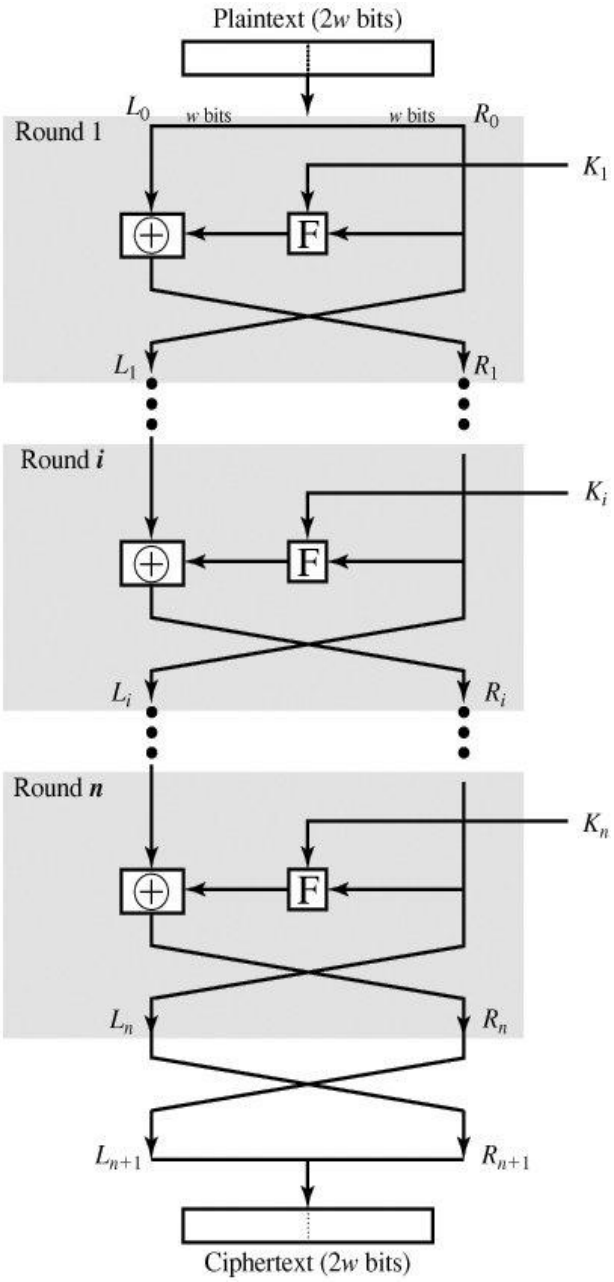
The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system. Shannon's concern was to thwart cryptanalysis based on statistical analysis. The reasoning is as follows. Assume the attacker has some knowledge of the statistical characteristics of the plaintext. For example, in a human-readable message in some language, the frequency distribution of the various letters may be known. Or there may be words or phrases likely to appear in the message (probable words). If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, or part of the key, or at least a set of keys likely to contain the exact key. In what Shannon refers to as a strongly ideal cipher, all statistics of the ciphertext are independent of the particular key used.

Confusion means that each binary digit (bit) of the ciphertext should depend on several parts of the key, obscuring the connections between the two.The property of confusion hides the relationship between the ciphertext and the key.

Diffusion means that if we change a single bit of the plaintext, then (statistically) half of the bits in the ciphertext should change, and similarly, if we change one bit of the ciphertext, then approximately one half of the plaintext bits should change.

Feistel Cipher Structure

Figure 3.2 depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. The plaintext block is divided into two halves, L0 and R0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs Li-1 and Ri-1, derived from the previous round, as well as a subkey Ki, derived from the overall K. In general, the subkeys Ki are different from K and from each other.



All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey K_i. Following this substitution, a **permutation** is performed that consists of the interchange of the two halves of the data.[4] This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

Block size, Key size, Number of rounds, Subkey generation algorithm, Round function, Fast software encryption/decryption, Ease of analysis.

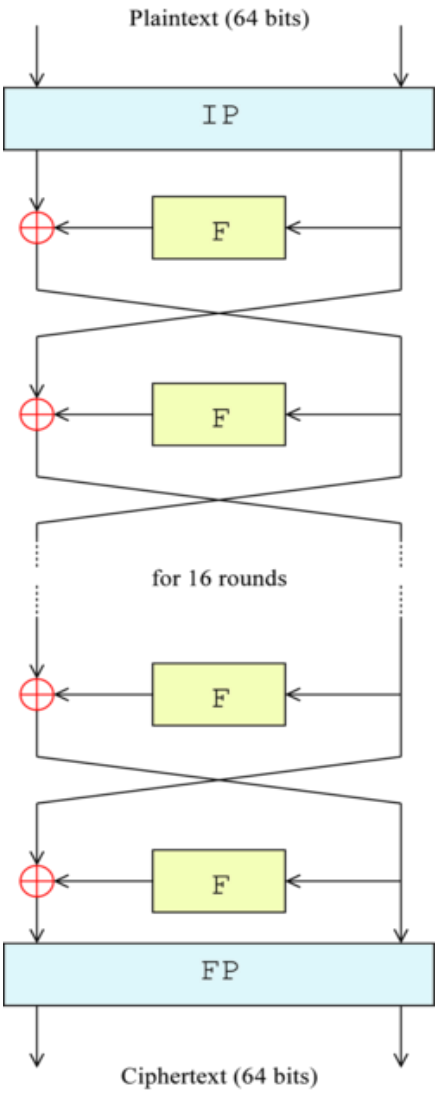
Data Encryption Standard (DES)

The *Data Encryption Standard (DES)* is a cipher (a method for encrypting information) selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976, and which has subsequently enjoyed widespread use internationally. The algorithm was initially controversial, with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny, and motivated the modern understanding of block ciphers and their cryptanalysis. DES

is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; in January, 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes.

Description: DES is the block cipher - an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits, and it is usually quoted as such.

Structure: The algorithm's overall structure is shown in Figure below there are 16 identical stages of processing, termed rounds. There is also an initial and final permutation, termed IP and FP (see appendix for IP and FP scheme), which are inverses (IP "undoes" the action of FP, and vice versa). IP and FP have almost no cryptographic significance, but were apparently included in order to facilitate loading blocks in and out of mid-1970s hardware, as well as to make DES run slower in software.
Before the main rounds, the block is divided into two 32-bit halves and processed alternately; this criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes - the only difference is that the subkeys are applied in the reverse order when decrypting. The rest of the algorithm is identical. This greatly simplifies implementation, particularly in hardware, as there is no need for separate encryption and decryption algorithms.



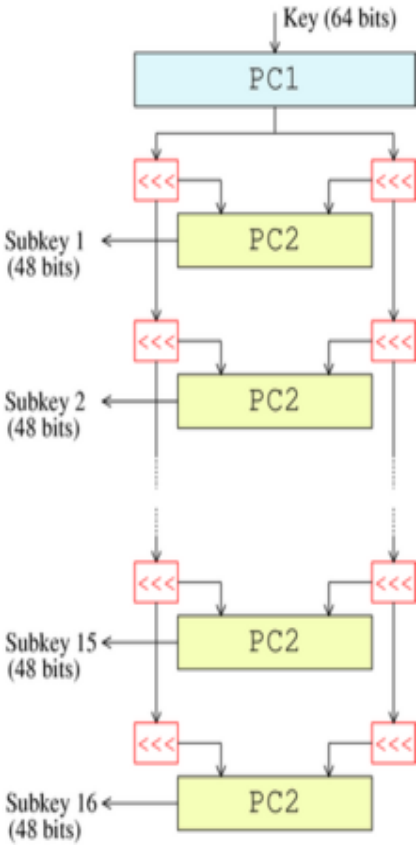
The ⊕ symbol denotes the exclusive-OR (XOR) operation. The F-function scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes.

The Feistel (F) function (Mangler Function): The F-function, depicted in Figure below, operates on half a block (32 bits) at a time and consists of four stages:
Expansion: the 32-bit half-block is expanded to 48 bits using the expansion permutation (see appendix for expansion permutation), denoted E in the diagram, by duplicating some of the bits.
Key Mixing: the result is combined with a subkey using an XOR operation. Sixteen 48-bit subkeys - one for each round - are derived from the main key using the key schedule described below.
Substitution: after mixing in the subkey, the block is divided into eight 6-bit pieces before processing by the S-boxes, or substitution boxes. Each of the eight S-boxes replaces its six input bits with four output bits according to a non-linear transformation, provided in the form of a lookup table (see appendix for table). The S-boxes provide the core of

the security of DES - without them, the cipher would be linear, and trivially breakable.
Permutation: finally, the 32 outputs from the S-boxes are rearranged according to a fixed permutation, the P-box (see appendix for permutation P).

The alternation of substitution from the S-boxes, and permutation of bits from the P-box and E-expansion provides so-called "confusion and diffusion" respectively, a concept identified by Claude Shannon in the 1940s as a necessary condition for a secure yet practical cipher.

Key Schedule: Figure below illustrates the key schedule for encryption - the algorithm which generates the subkeys. Initially, 56 bits of the key are selected from the initial 64 by Permuted Choice 1, PC-1(see appendix for PC1) - the remaining eight bits are either discarded or used as parity check bits. The 56 bits are then divided into two 28-bit halves; each half is thereafter treated separately. In successive rounds, both halves are rotated left by one or two bits specified for each round (see appendix for key rotation schedule), and then 48 subkey bits are selected by Permuted Choice 2, PC-2(see appendix for PC2) - 24 bits from the left half, and 24 from the right. The rotations (denoted by "<<<" in the diagram) mean that a different set of bits is used in each subkey; each bit is used in approximately 14 out of the 16 subkeys.

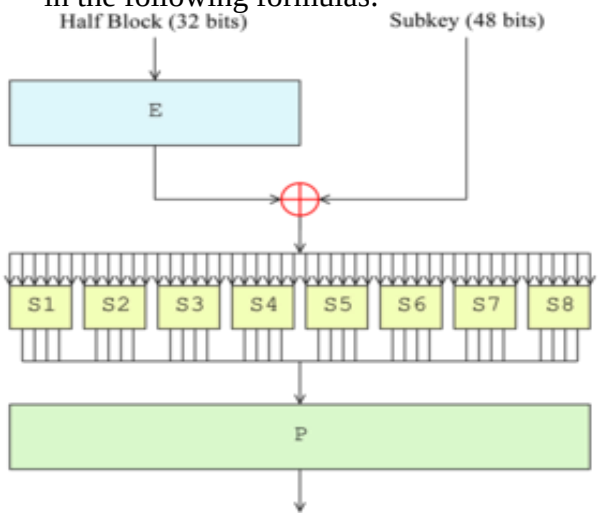


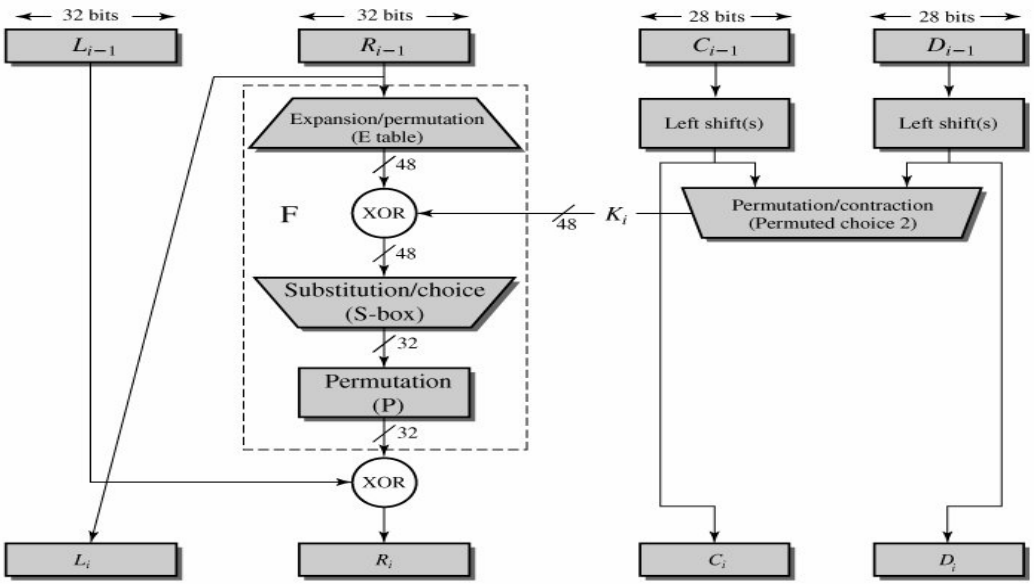
The key schedule for decryption is similar - the subkeys are in reverse order compared to encryption. Apart from that change, the process is the same as for encryption.

Weak and Semi-Weak Keys: There are sixteen DES keys that are not suggested for use. However the probability of getting such keys is very small as given by $16/2^{56}$. The sixteen weak keys are keys with all ones, all zeroes, alternating zeroes and ones, and alternating ones and zeroes for two 28 bits parts of the key generated when PC-1 is used.

Details of Single Round

Figure shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:





Note: see the table used in this algorithm in Text Book.

Security and cryptanalysis of DES:

Although more information has been published on the cryptanalysis of DES than any other block cipher, the most practical attack to date is still a brute force approach. Various minor cryptanalytic properties are known, and three theoretical attacks are possible which, while having a theoretical complexity less than a brute force attack, require an unrealistic amount of known or chosen plaintext to carry out, and are not a concern in practice. In spite of all the criticism and weaknesses of DES, there is no known example of anyone actually suffering monetary losses because of DES security limitations.

Brute force attack: For any cipher, the most basic way of attack is brute force - trying every possible key. The length of the key gives the number of possible keys, and hence the feasibility of this approach. For DES, questions were raised about the adequacy of its key size early on, even before it was adopted as a standard, and it was the small key size, rather than theoretical cryptanalysis, which dictated a need for a replacement algorithm. It is known that the NSA encouraged, if not persuaded, IBM to reduce the key size from 128 to 64 bits, and from there to 56 bits; this is often taken as an indication that the NSA thought it would be able to break keys of this length even in the mid-1970s.

2. Advanced Encryption Standard (AES)

The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits. The AES specification uses the same three key size alternatives but limits the block length to 128 bits. A number of AES parameters depend on the key length

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

AES Encryption Decryption Technique

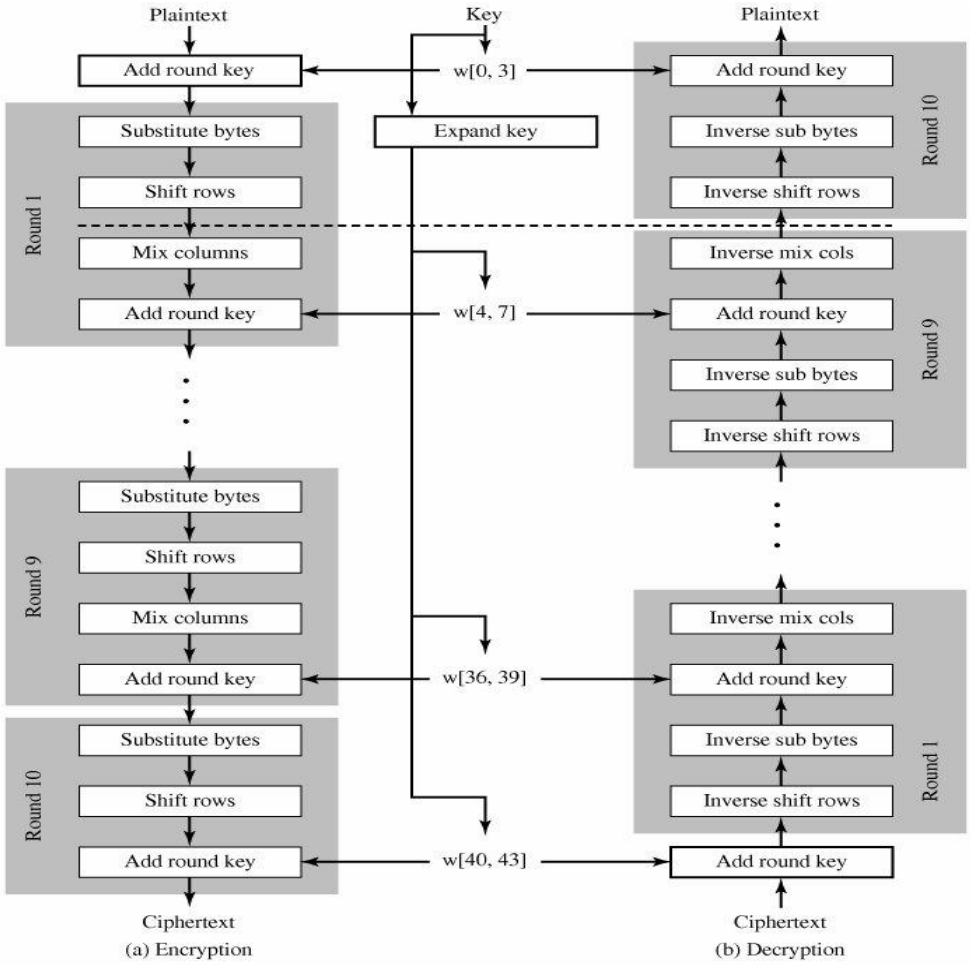
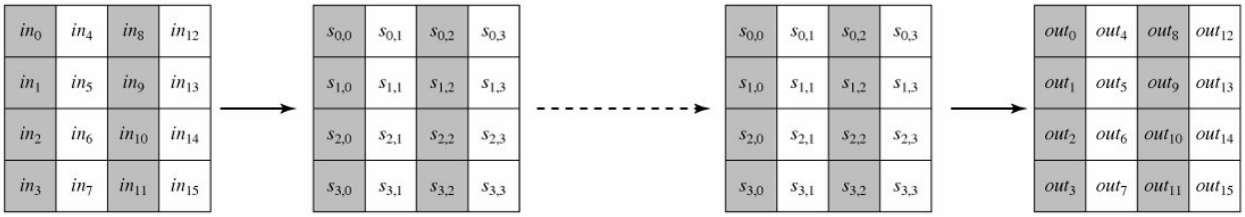


Figure 1 : The encryption/decryption Rounds on AES

This figure shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix.

AES data structures: the following matrixes are used in AES encryption and Decryptions



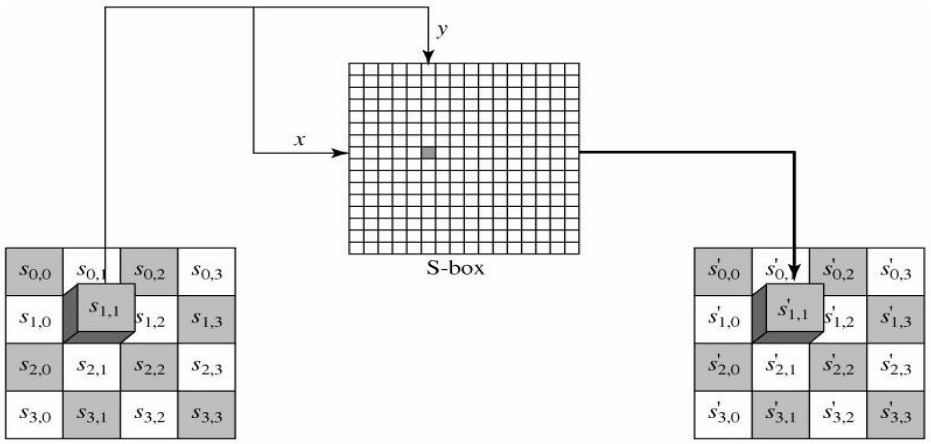
(a) Input, state array, and output

(b) Key and expanded key

The structure of AES is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages as shown in figure 1 above.

The four stages are:
Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block
ShiftRows: A simple permutation
MixColumns: A substitution that makes use of arithmetic over GF(2^8)
AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key

a) Substitute Bytes Transformation
The forward substitute byte transformation, called SubBytes, is a simple table lookup (Figure 5.4a). AES defines a 16 x 16 matrix of byte values, called an S-box (see table on Text Book) that contains a permutation of all possible 256 8-bit values. Each individual byte of State is mapped into a new byte in the following way:
The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2A}. Accordingly, the value {95} is mapped into the value {2A}.

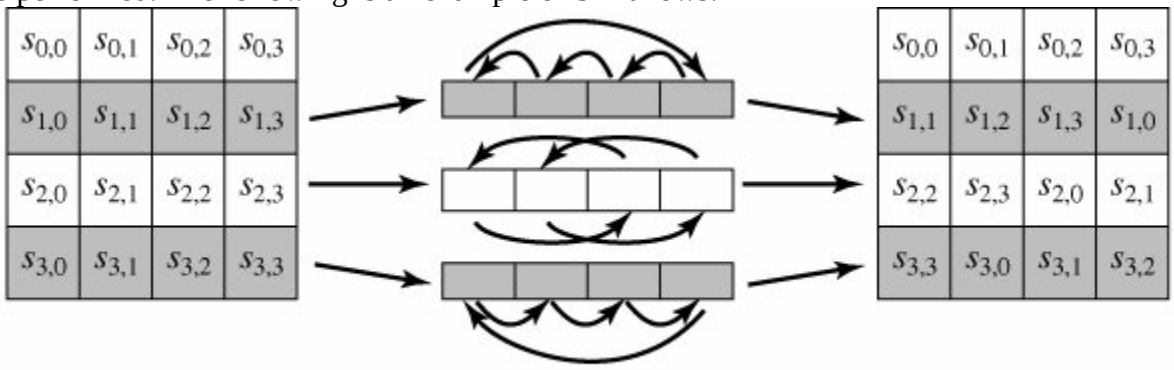


(a) Substitute byte transformation

The inverse substitute byte transformation, called InvSubBytes, makes use of the inverse S-box shown in (Text book). For example, that the input {2A} produces the output {95} and the input {95} to the S-box produces {2A}.

b) ShiftRows Transformation

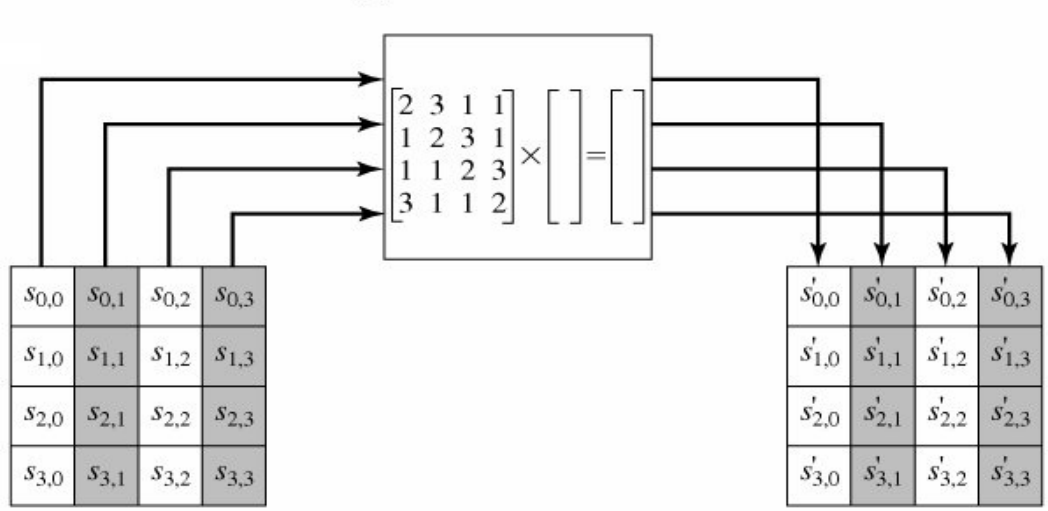
The forward shift row transformation, called ShiftRows, is depicted in Figure below. The first row of State is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3-byte circular left shift is performed. The following is an example of ShiftRows:



(a) Shift row transformation

c) MixColumns Transformation

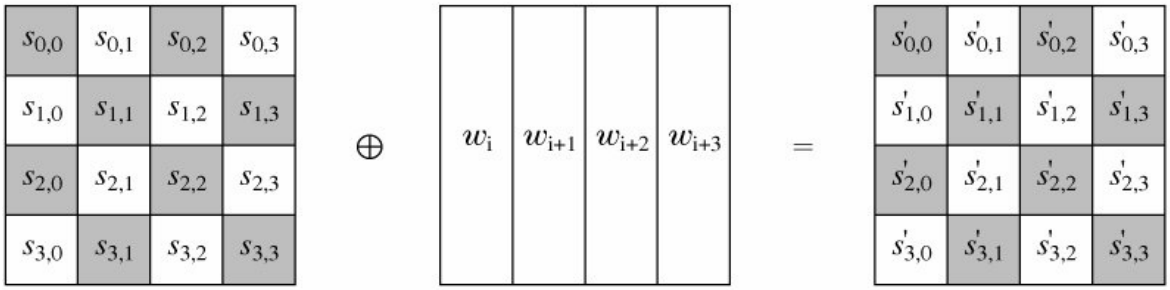
The forward mix column transformation, called MixColumns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column.



(b) Mix column transformation

d) AddRoundKey Transformation

In the forward add round key transformation, called AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key. As shown in Figure below, the operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key; it can also be viewed as a byte-level operation.



(b) Add Round Key Transformation

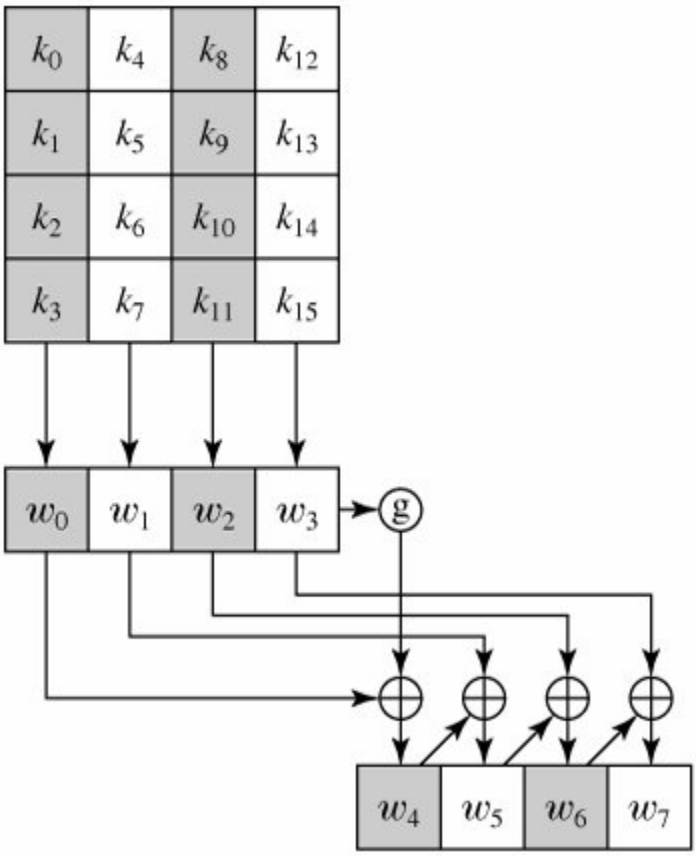
AES Key Expansion Algorithm

The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The following pseudocode describes the expansion:

```
KeyExpansion (byte key[16], word w[44])
{
   word temp
   for (i = 0; i < 4; i++)
            w[i] = (key[4*i], key[4*i+1],key[4*i+2],key[4*i+3]);
   for (i = 4; i < 44; i++)
   {
    temp = w[i  1];
```

```
    if (i mod 4 = 0)
          temp = SubWord (RotWord (temp))ExOR Rcon[i/4];
    w[i] = w[i4]
   }
}
```

The key expansion technique is shown in figure below



Where g is a special function consist of following sub functions

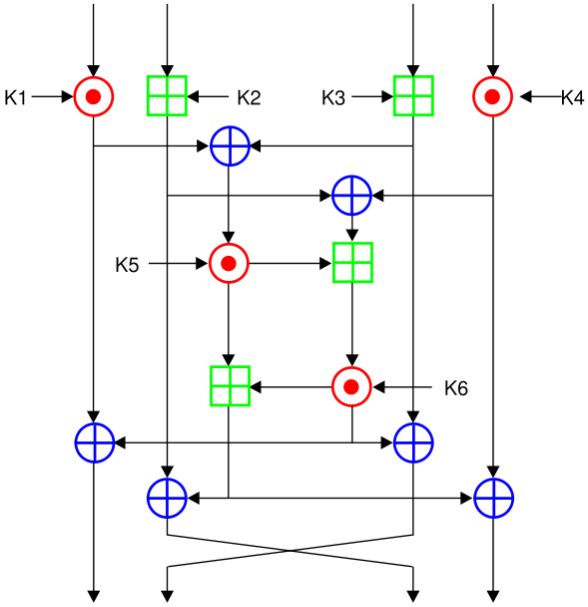
1. RotWord performs a one-byte circular left shift on a word. This means that an input word [b0, b1, b2, b3] is transformed into [b1, b2, b3, b0].
2. SubWord performs a byte substitution on each byte of its input word, using the S-box.
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j].

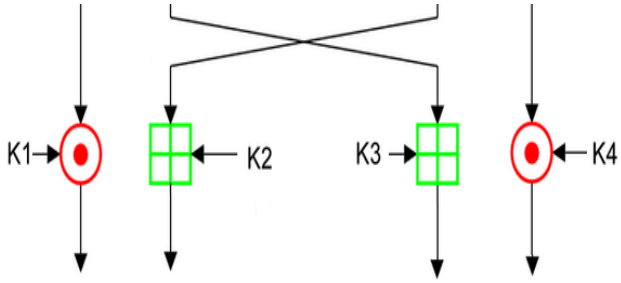
The round constant is a word in which the three rightmost bytes are always 0.(See the text book for more details)

3. International Data Encryption Algorithm (IDEA)

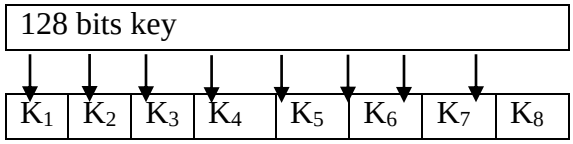
In cryptography, the IDEA is a block cipher designed by Xuejia Lai and James Massey and was first described in 1991. The algorithm was intended as a replacement for the Data Encryption Standard.

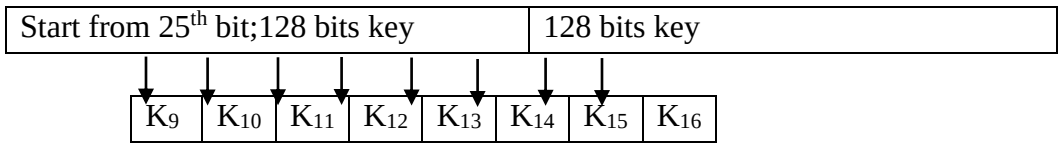
Operation: IDEA operates on 64-bit blocks using a 128-bit key, and consists of a series of eight identical rounds (see figure below) and an output round (the half-round, see figure below). The processes for encryption and decryption are similar. IDEA derives much of its security by interleaving operations from different groups - modular addition (addition mod 2^{16}, denoted by$\boxplus$), modular multiplication (multiplication mod $2^{16}+1$ denoted by $\odot$, where the all-zero word (0x0000) is interpreted as 2^{16}),, and bitwise eXclusive OR (XOR) (denoted by $\oplus$), - which are algebraically "incompatible" in some sense. The blow figures in left shows a round (1-8) and in right shows final "half round.



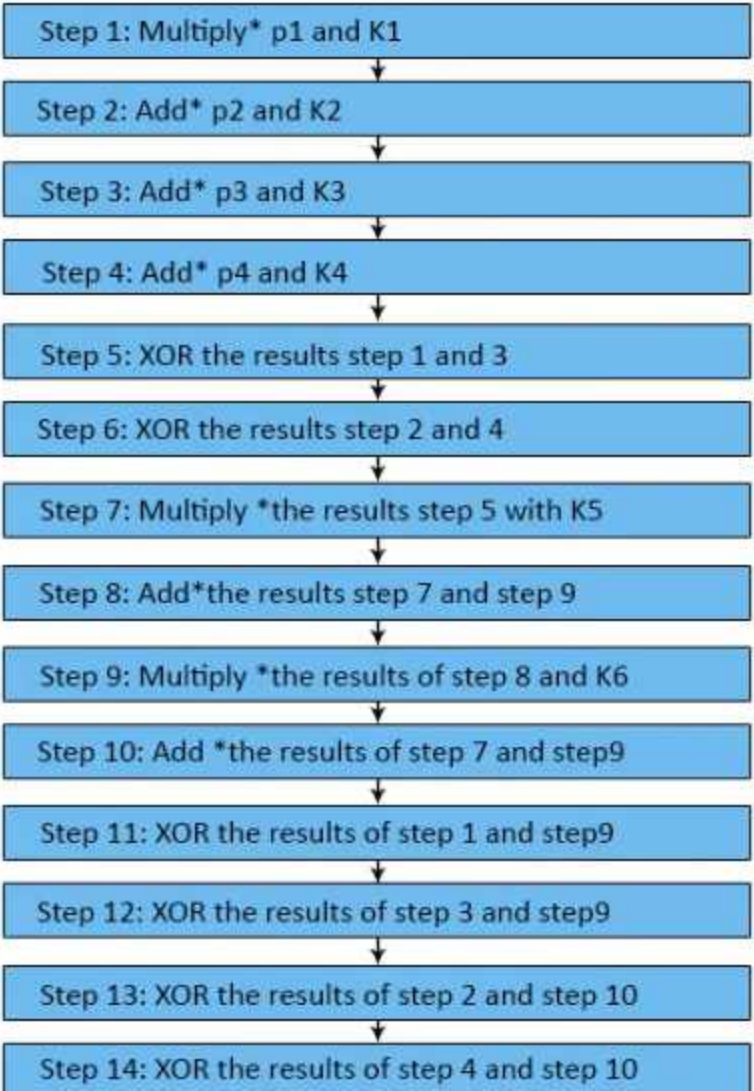


Key schedule: The 128 bit key is used to generate 52 sub keys of length 16 bits. 6 keys are used for each round and the remaining 4 keys are used in final half round. The figure below shows the key generation mechanisms where we start getting the subkeys from the starting position of the 128 bits key, so in first round we get 8 subkeys. In each of the round of sub key generation 128 bits keys undergoes a 25 bit position right to generate next set of keys i.e. start with bit position 0, 25, 50 and so on until all 52 keys are derived.





Round Operations: It has been mentioned above that IDEA uses 8 full rounds and 1 half round. We now break the 8 full round and make it 16 rounds such that there are total 17 rounds where 9 odd rounds (1, 3, …., 17) are identical and 8 even rounds (2, 4, ….,16) are identical. Each odd round takes 4 subkeys and each even round takes 2 subkeys.


Step 1: Multiply* p1 and K1
Step 2: Add* p2 and K2
Step 3: Add* p3 and K3
Step 4: Add* p4 and K4
Step 5: XOR the results step 1 and 3
Step 6: XOR the results step 2 and 4
Step 7: Multiply *the results step 5 with K5
Step 8: Add*the results step 7 and step 9
Step 9: Multiply *the results of step 8 and K6
Step 10: Add *the results of step 7 and step9
Step 11: XOR the results of step 1 and step9
Step 12: XOR the results of step 3 and step9
Step 13: XOR the results of step 2 and step 10
Step 14: XOR the results of step 4 and step 10

Odd Round: Odd round is simple and uses four keys where first and fourth keys are uses multiplication mod $2^{16}+1$ operation with first and fourth 16 bits fragment of 64 bits input block respectively. The second and third subkeys are subjected to addition mod 2^{16} with second and third bits fragment of 64 bits input block respectively. The output so obtained from operations with first and fourth input sub-block will be first and fourth input for next round and output from operations with second and third input sub-block will be reversed i.e. becomes third and second input for next round. If X_a, X_b, X_c, and X_d are input sub-blocks and K_a, K_b, K_c, and K_d are subkeys then we can write algebraic expression for odd round as:

$X_a = X_a \odot K_a$; $X_c = X_c \boxplus K_c$; $X_b = X_b \boxplus K_b$; $X_d = X_d \odot K_d$; (see figure above)

Even Round: Even round is bit complicated then the odd round. There are four input sub-blocks (X_a, X_b, X_c, and X_d) from the previous round and two subkeys (K_e and K_f). In even rounds first and second input sub-blocks are subjected to XOR operation to get single 16 bits output (say, Y_{in}) and third and fourth input sub-blocks are subjected to XOR operation to get single 16 bits output (say, Z_{in}). Y_{in} and Z_{in} are fed to mangler function along with K_e, and K_f to get outputs Y_{out} and Z_{out}, where Y_{out} is XOR'ed with X_a and X_b, to get first two input sub-blocks for next round and Z_{out} is XOR'ed with X_c and X_d, to get last two input sub-blocks for next round. Algebraic expressions for even round can be written as:

$Y_{in} = X_a \oplus X_b$; $Z_{in} = X_c \oplus X_d$; $Y_{out} = ((K_e \odot Y_{in}) \boxplus Z_{in}) \odot K_f$; $Z_{out} = (K_e \odot Y_{in}) \boxplus Y_{out}$;

$X_a = X_a \oplus Y_{out}$; $X_b = X_b \oplus Y_{out}$; $X_c = X_b \oplus Z_{out}$; $X_d = X_d \oplus Z_{out}$; (see figure above)

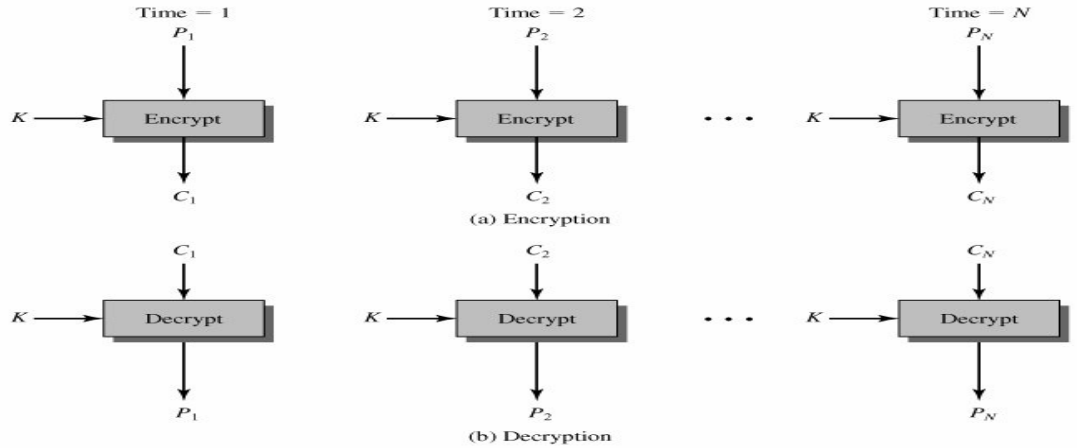
Security: The designers analyzed IDEA to measure its strength against differential cryptanalysis and concluded that it is immune under certain assumptions. No successful linear or algebraic weaknesses have been reported. Some classes of weak keys have been found but these are of little concern in practice, being so rare as to be unnecessary to avoid explicitly. As of 2004, the best attack which applies to all keys can break IDEA reduced to 5 rounds (the full IDEA cipher uses 8.5 rounds).

Modes of Operations

A mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream. The four modes are intended to cover virtually all the possible applications of encryption for which a block cipher could be used.

1. Electronic Codebook Mode

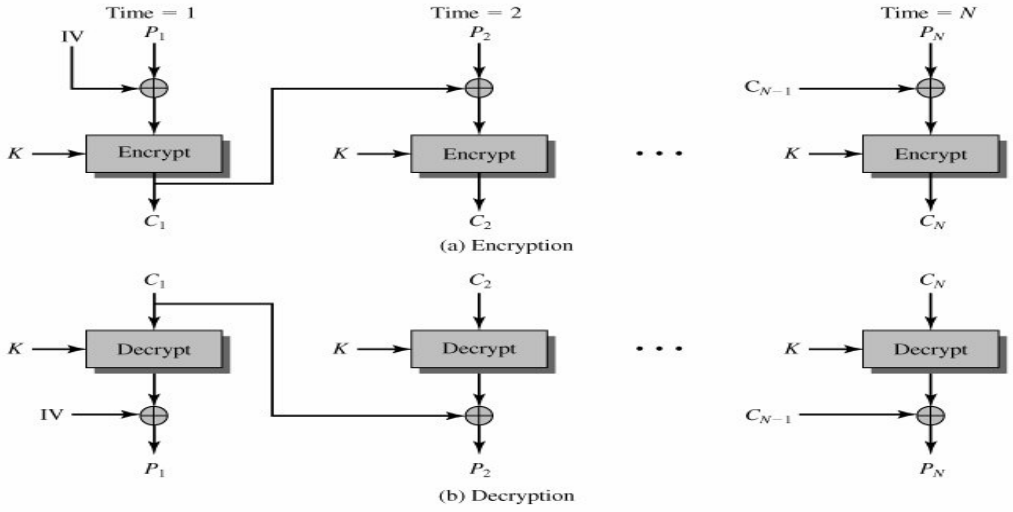
The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext. Therefore, we can imagine a gigantic codebook in which there is an entry for every possible b-bit plaintext pattern showing its corresponding ciphertext.



The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES key securely, ECB is the appropriate mode to use. The most significant characteristic of ECB is that the same b-bit block of plaintext, if it appears more than once in the message, always produces the same ciphertext. For lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities.

2. Cipher Block Chaining Mode

To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher block chaining (CBC) mode. In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks.



because of the chaining mechanism of CBC, it is an appropriate mode for encrypting messages of length greater than b bits.

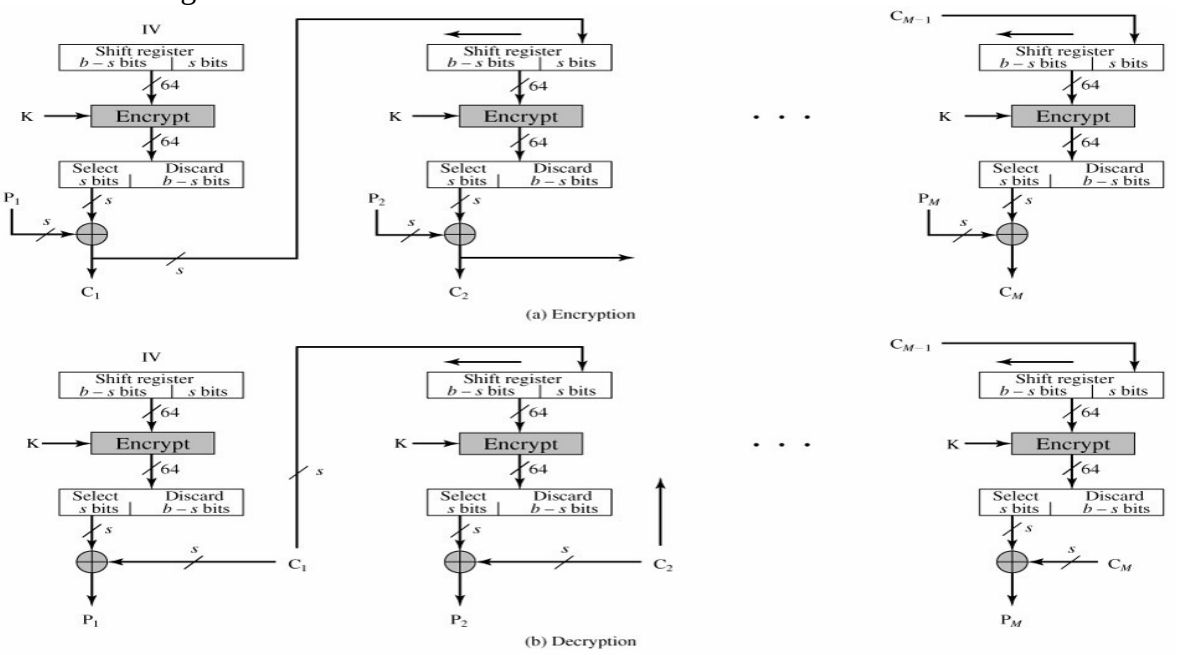
3. Cipher Feedback Mode

The DES scheme is essentially a block cipher technique that uses b-bit blocks. However, it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the

output feedback mode. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

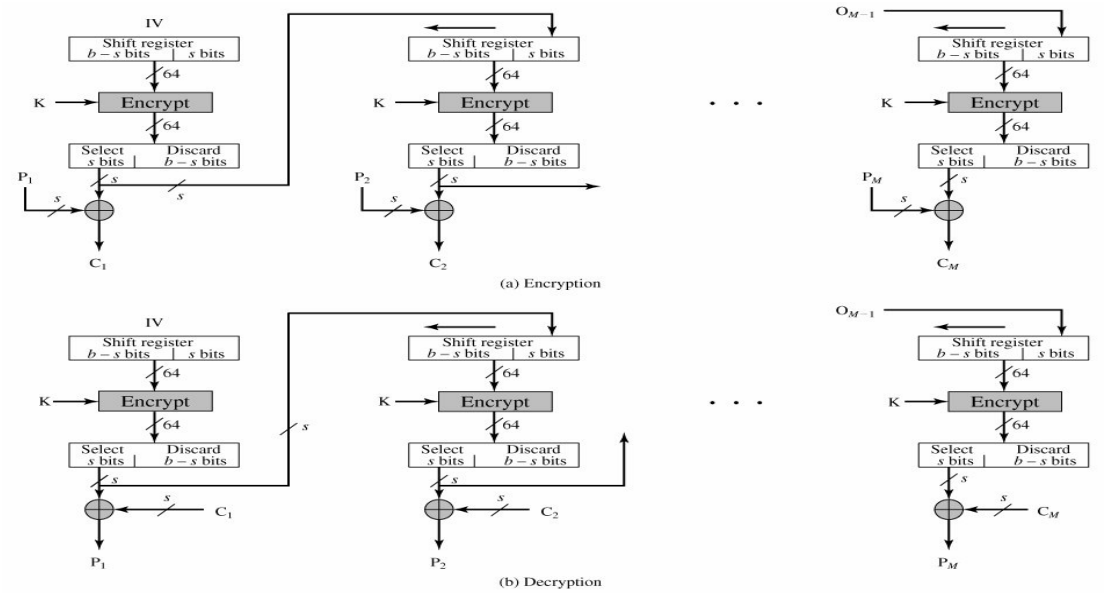
One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a cipher text output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted.

In the figure, it is assumed that the unit of transmission is s bits; a common value is s = 8. As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. In this case, rather than units of b bits, the plaintext is divided into segments of s bits.



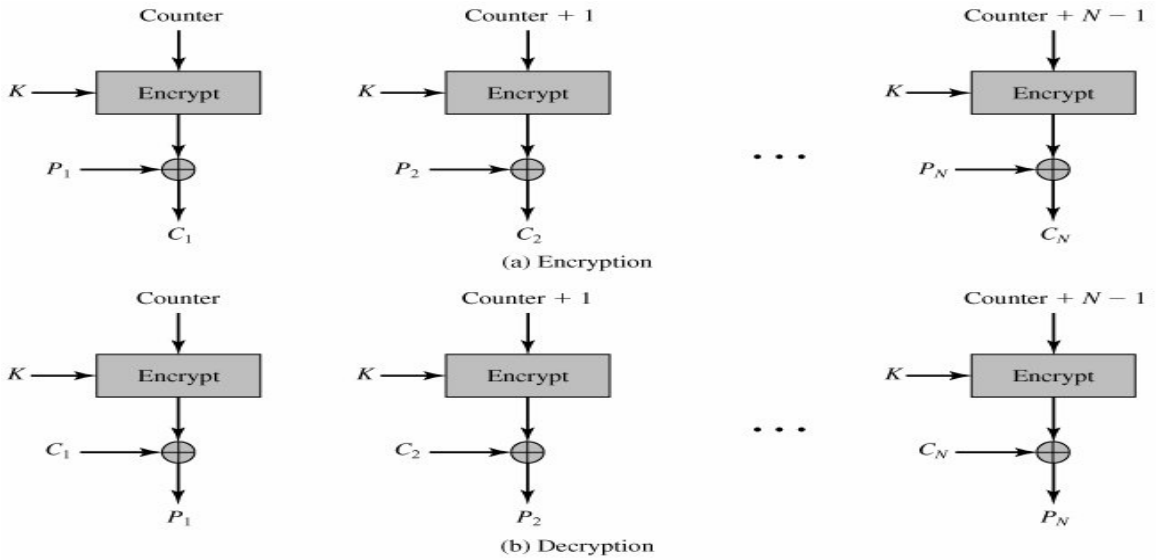
4. Output Feedback Mode

The output feedback (OFB) mode is similar in structure to that of CFB, as illustrated in figure. As can be seen, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register.



5. Counter mode

Figure depicts the CTR mode. A counter, equal to the plaintext block size is used. The only requirement stated in SP 800-38A is that the counter value must be different for each plaintext block that is encrypted. Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2^b where b is the block size). For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block.



Old questions

1. How many rounds are used in AES and what does the number of rounds depend on?
2. What are the steps that go into the construction of the 16*16 S-box lookup table for AES algorithm?
3. What are the characteristics of a stream cipher?