Cryptography

Cryptography is the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents. ... When transmitting electronic data, the most common use of cryptography is to encrypt and decrypt email and other plain-text messages.. In data and telecommunications, cryptography is necessary when communicating over any Untrusted medium, which includes just about any network, particularly the Internet.

There are five primary functions of cryptography:

1. *Privacy/confidentiality:* Ensuring that no one can read the message except the intended receiver.
2. *Authentication:* The process of proving one's identity.
3. *Integrity:* Assuring the receiver that the received message has not been altered in any way from the original.
4. *Non-repudiation:* A mechanism to prove that the sender really sent this message.
5. *Key exchange:* The method by which crypto keys are shared between sender and receiver.

In cryptography, we start with the unencrypted data, referred to as *plaintext*. Plaintext is *encrypted* into *ciphertext*, which will in turn (usually) be *decrypted* back into usable plaintext. The encryption and decryption is based upon the type of cryptography scheme being employed and some form of key. For those who like formulas, this process is sometimes written as:

$C = E_k(P)$

$P = D_k(C)$

where **P** = plaintext, **C** = ciphertext, **E** = the encryption method,

D = the decryption method, and **k** = the key.

Given this, there are other functions that might be supported by crypto and other terms that one might hear:

- *Forward Secrecy* (aka *Perfect Forward Secrecy*): This feature protects past encrypted sessions from compromise even if the server holding the messages is compromised. This is accomplished by creating a different key for every session so that compromise of a single key does not threaten the entirely of the communications.
- *Perfect Security*: A system that is unbreakable and where the ciphertext conveys no information about the plaintext or the key. To achieve perfect security, the key has to be at least as long as the plaintext, making analysis and even brute-force attacks impossible. One-time pads are an example of such a system.

Finally, *cryptography* is most closely associated with the development and creation of the mathematical algorithms used to encrypt and decrypt messages, whereas *cryptanalysis* is the science of analyzing and breaking encryption schemes. *Cryptology* is the umbrella term referring to the broad study of secret writing, and encompasses both cryptography and cryptanalysis.

3. TYPES OF CRYPTOGRAPHIC ALGORITHMS

There are several ways of classifying cryptographic algorithms. For purposes of this paper, they will be categorized based on the number of

keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms that will be discussed are (Figure 1):

- *Secret Key Cryptography (SKC):* Uses a single key for both encryption and decryption; also called *symmetric encryption*. Primarily used for privacy and confidentiality.
- *Public Key Cryptography (PKC):* Uses one key for encryption and another for decryption; also called *asymmetric encryption*. Primarily used for authentication, non-repudiation, and key exchange.
- *Hash Functions:* Uses a mathematical transformation to irreversibly "encrypt" information, providing a digital fingerprint. Primarily used for message integrity.

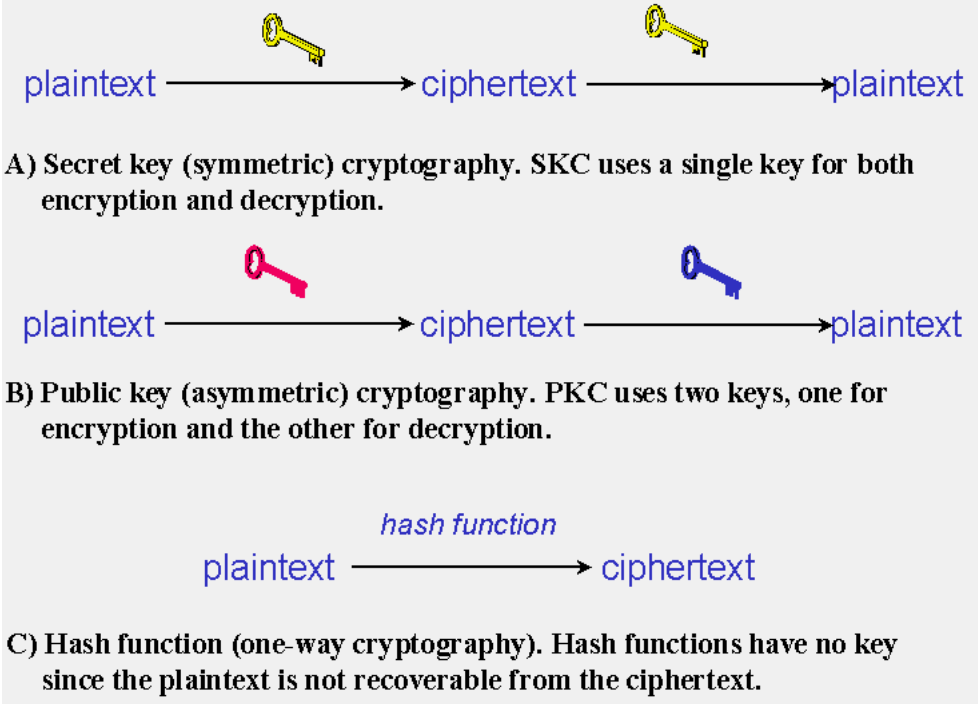


FIGURE 1: Three types of cryptography: secret key, public key,

and hash function.

Secret key cryptography methods employ a single key for both encryption and decryption. As shown in Figure 1A, the sender uses the key to encrypt the plaintext and sends the ciphertext to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called *symmetric encryption*.

With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key (more on that later in the discussion of public key cryptography).

Secret key cryptography schemes are generally categorized as being either *stream ciphers* or *block ciphers*.

Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing. A stream cipher is a symmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (keystream). In a stream cipher, each plaintext digit is encrypted one at a time with the corresponding digit of the keystream, to give a digit of the ciphertext stream. Stream ciphers encrypt data a single bit, or a single byte, at a time in a stream. Block ciphers encrypt data in a specific-sized block such as 64-bit or 128-bit blocks. Stream ciphers are more efficient than block ciphers when encrypting data in a continuous stream.

Classical Substitution Ciphers

• where letters of plaintext are replaced by other letters or by numbers or symbols

• or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

Monoalphabetic cipher is a **substitution cipher** in which for a given key, the **cipher** alphabet for each plain alphabet is fixed throughout the encryption process. For **example**, if 'A' is encrypted as 'D', for any number of occurrence in that plaintext, 'A' will always get encrypted to 'D'.

Monoalphabetic Cipher

rather than just shifting the alphabet

- could shuffle (jumble) the letters arbitrarily
- each plaintext letter maps to a different random ciphertext letter
- hence key is 26 letters long

Plain: abcdefghijklmnopqrstuvwxyz

Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext: ifwewishtoreplaceletters

Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA

Caesar Shift

One of the earliest forms of encryption techniques used by the Romans named Caesar Shift Cipher. The idea of Caesar shift Cipher was basically simple where the each character of the plain text is replaced with fixed predefined number of the character from the alphabet. Usually it is called shifting of the letter as each letter is replaced by

a letter further along in the alphabet. The decryption process was completely reverse. The encrypted text then needs to be replaced with same sifting of the letter.

Encryption: If we consider the following English Alphabet with their corresponding positions below:

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

A B C D E F G H I J K L M

13 14 15 16 17 18 19 20 21 22 23 24 25

N O P Q R S T U V W X Y Z

Now let's consider my name as plain text which is **PARVES**. So if we look at the position of the letter of each alphabet we get 15, 0, 17, 21, 4, 18 . Now if we have sift key of 3 then we get the number as 18, 3, 20, 24, 7, 21. So we get the name now as **SDUYHV**.

If by shifting key goes beyond the maximum numerical alphabet letters 26 we have to wrap around and start from 0. The mathematical explanation for such shifting can be illustrated by the following formula:

a ≡ b (mod m) means m is a divisor of a − b.

In our case m is the size of the key which is in our case the number of character set which is 26. Now we have to take each numeric number of the character of our plain text and add 3 with it. If that number is between 0 to 15 do nothing if not after doing modules whatever is the reminder will be the new numeric number of the character set.

So in our case the encryption is done by following steps

P −→ 15 −→ 15 + 3 ≡ 18 (mod 26) −→ S

A −→ 0 −→ 0 + 3 ≡ 3 (mod 26) −→ D

R −→ 17 −→ 17 + 3 ≡ 20 (mod 26) −→ U

V −→ 21 −→ 21 + 3 ≡ 24 (mod 26) −→ Y

E −→ 4 −→ 4 + 3 ≡ 7 (mod 26) −→ H

Decryption: If the take the decrypted text and sift it to the right 3 character we will get the original message. So in these encryption techniques the memorization of the key was not required as there was no pattern to it. Also since the key space of 26 English character even if the attacker did not know the key sift he/she can try all the combination and make sense out of the encrypted message by trying out all the 26 shifting possible. In the figure below we can understand how the Caesar Cipher works better:

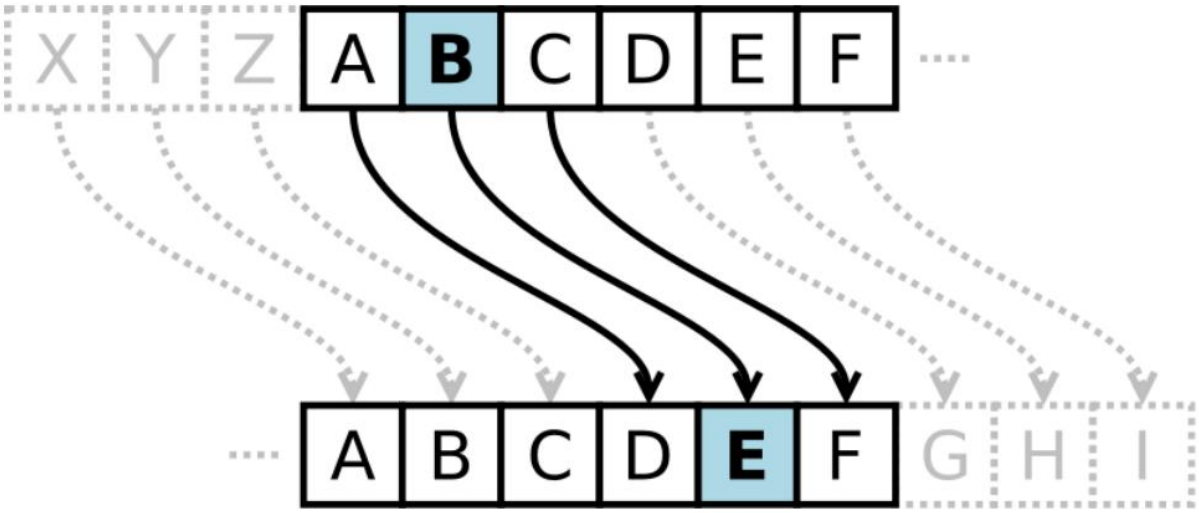
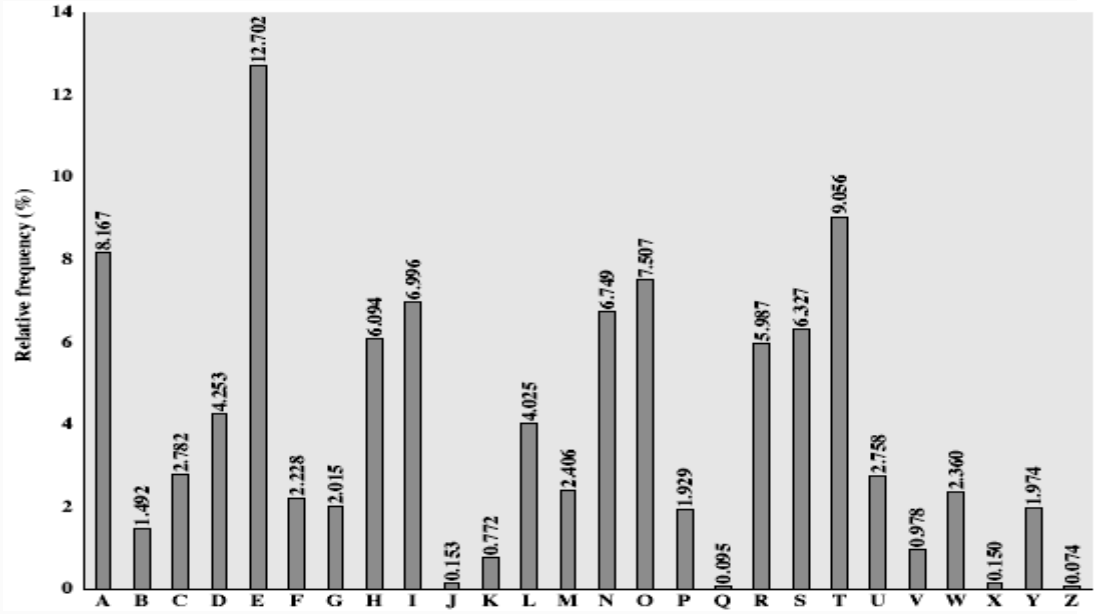


Fig-2: Caesar Cipher

Language Redundancy and Cryptanalysis

human languages are redundant
• eg "th lrd s m shphrd shll nt wnt"
• letters are not equally commonly used
• in English E is by far the most common letter
• followed by T,R,N,I,O,A,S
• other letters like Z,J,K,Q,X are fairly rare
• have tables of single, double & triple letter frequencies for various languages

English Letter Frequencies



Language Redundancy and Cryptanalysis

Polyalphabetic Ciphers

A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The Vigenère cipher is probably the best-known example of a polyalphabetic cipher, though it is a simplified special case.

The development of Polyalphabetic Substitution Ciphers was the cryptographers answer to Frequency Analysis. It is another approach to security is to use multiple cipher alphabets

- called **polyalphabetic substitution ciphers**
- makes cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- use a key to select which alphabet is used for each letter of the message
- use each alphabet in turn
- repeat from start after end of key is reached

Vigenère Cipher :- is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers, based on the letters of a keyword. It employs a form of polyalphabetic substitution.

- simplest polyalphabetic substitution cipher is the **Vigenère Cipher**
- effectively multiple caesar ciphers
- key is multiple letters long K = k1 k2 ... kd
- i^{th} letter specifies i^{th} alphabet to use
- use each alphabet in turn

- repeat from start after d letters in message
- decryption simply works in reverse

Example

- write the plaintext out
- write the keyword repeated above it
- use each key letter as a caesar cipher key
- encrypt the corresponding plaintext letter
- eg using keyword *deceptive*

key: deceptivedeceptivedeceptive

plaintext: wearediscoveredsaveyourself

ciphertext:ZICVTWQNGRZGVTWAVZHCQYGLMGJ

One Time Pad

In cryptography, the one-time pad (OTP) is an encryption technique that cannot be cracked, but requires the use of a one-time pre-shared key the same size as, or longer than, the message being sent. In this technique, a plaintext is paired with a random secret key (also referred to as a one-time pad).

One-time pad uses a random key throughout the message; hence the key need not be repeated. A key must be discarded after using for encrypting and decrypting a message. Each new message can have a new key of the message length.

- if a truly random key as long as the message is used, the cipher will be secure
- is unbreakable since ciphertext bears no statistical relationship to the plaintext
- since for **any plaintext** & **any ciphertext** there exists a key mapping one to other
- can only use the key **once** though
- have problem of safe distribution of key

Transposition Ciphers

In cryptography, a transposition cipher is a method of encryption by which the positions held by units of plaintext are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed.

transposition ciphers are not highly secure because they do not change the letters in the plaintext or even cover up frequencies, but they can be built upon to make more secure methods of encryption..

- now consider classical **transposition** or **permutation** ciphers
- these hide the message by rearranging the letter order
- without altering the actual letters used

- can recognise these since have the same frequency distribution as the original text

Row Transposition Ciphers

- a more complex scheme
- write letters of message out in rows over a specified number of columns
- then reorder the columns according to some key before reading off the rows

Key: 4 3 1 2 5 6 7

Plaintext: a t t a c k p

o s t p o n e

d u n t i l t

w o a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Cryptography

Product Cipher

- In cryptography, a product cipher combines two or more transformations in a manner intending that the resulting cipher is more secure than the individual
- ciphers using substitutions or transpositions are not secure because of language characteristics
- hence consider using several ciphers in succession to make harder, but:
  - two substitutions make a more complex substitution
  - two transpositions make more complex transposition
  - but a substitution followed by a transposition makes a new much harder cipher
- this is bridge from classical to modern ciphers

DES

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). DES is a block cipher; it encrypts data in 64-bit blocks. A 64-bit block of plaintext goes in one end of the algorithm and a 64-bit block of ciphertext comes out the other end. DES is a symmetric algorithm: The same algorithm and key are used for both encryption and decryption (except for minor differences in the key schedule). The key length is 56 bits. (The key is usually expressed as a 64-bit number, but every eighth bit is used for parity checking and is ignored. These parity bits are the least- significant bits of the key bytes.) The key can be any 56-bit number and can be changed at any time.

At its simplest level, the algorithm is nothing more than a combination of the two basic techniques of encryption: confusion and diffusion. The fundamental building block of DES is a single combination of these techniques (a substitution followed by a permutation) on the text, based on the key. This is known as a round. DES has 16 rounds; it applies the same combination of techniques on the plaintext block 16 times

General Structure of DES is depicted in the following illustration –

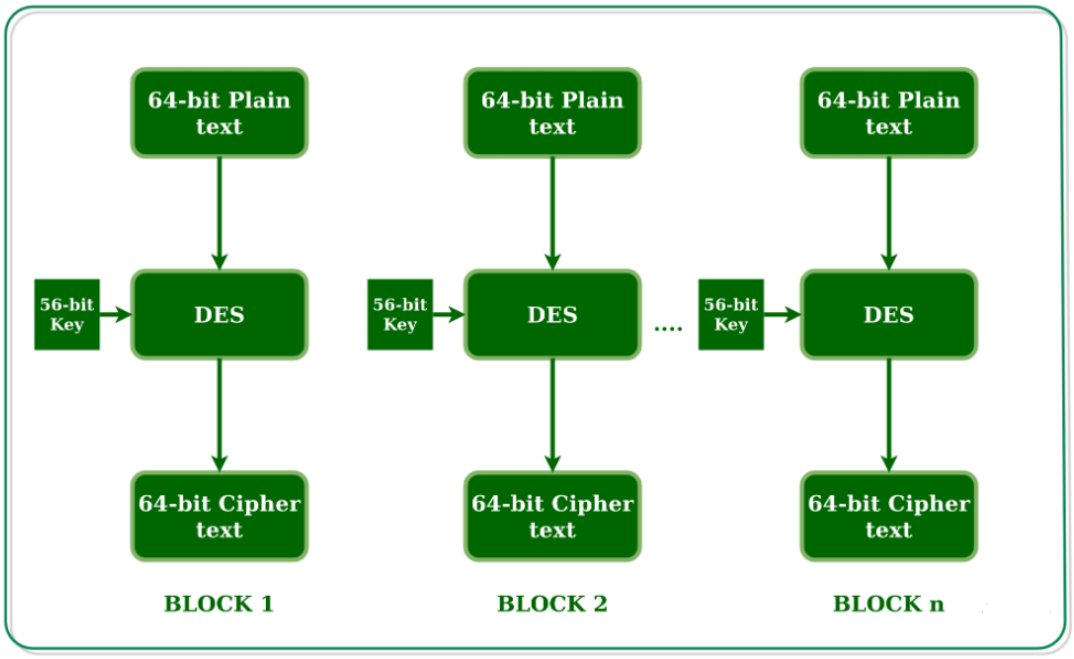


Figure 1 : general structure of DES

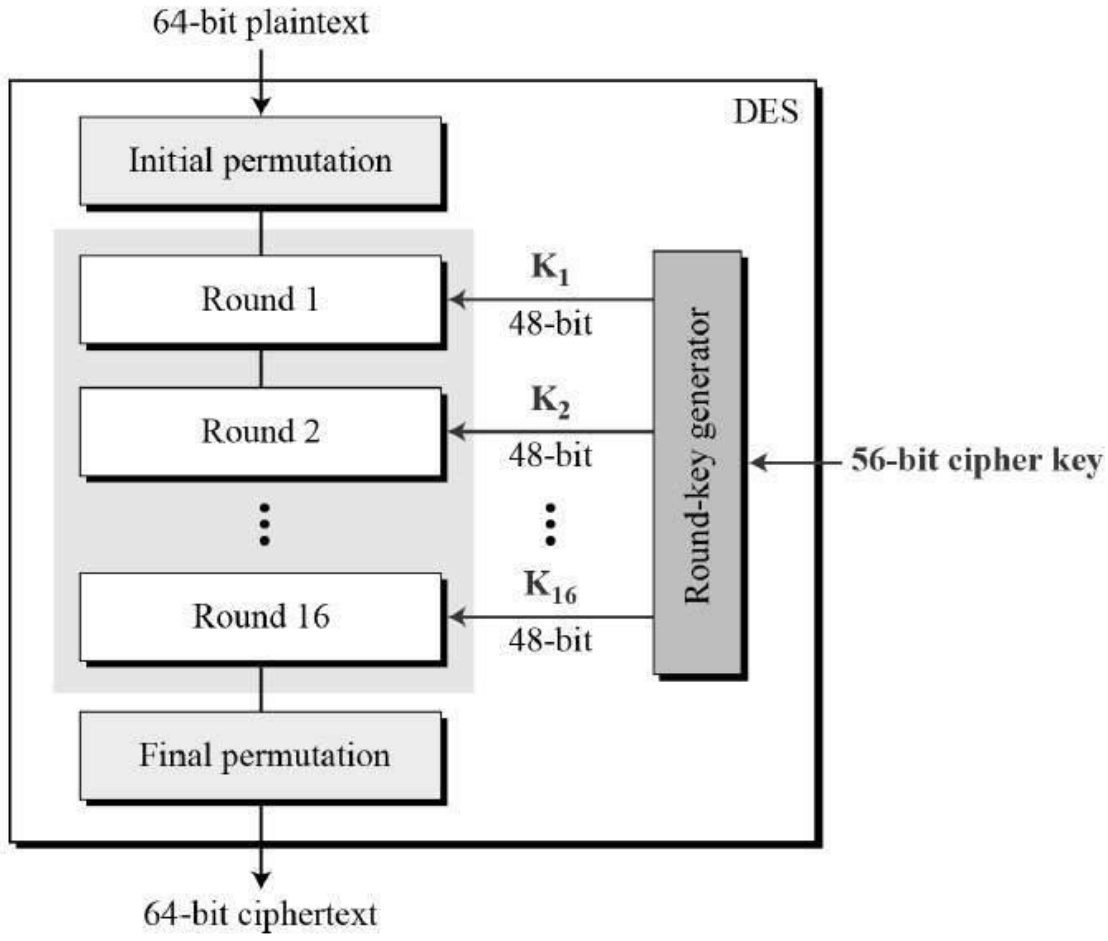


Figure2:- explain DES Basic operation

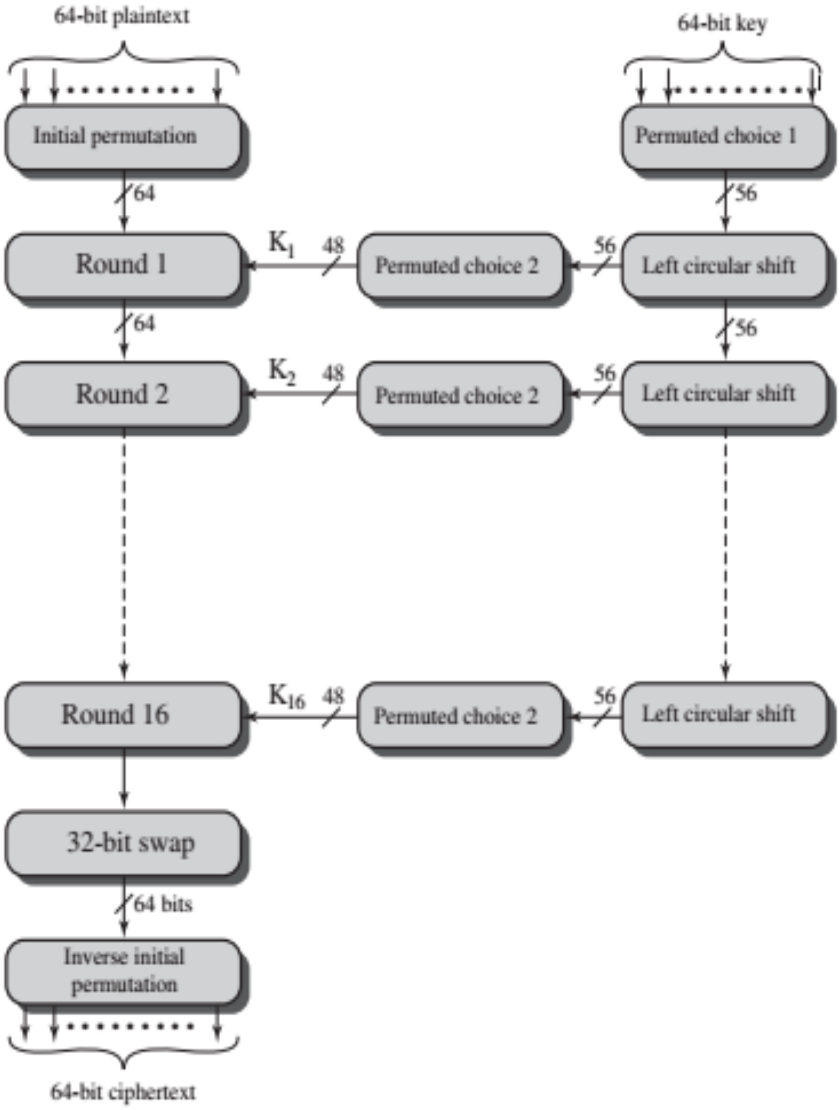


Figure3:- General Depiction of DES Encryption Algorithm

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography

significance in DES. The initial and final permutations are shown as follows −

Round Function

The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

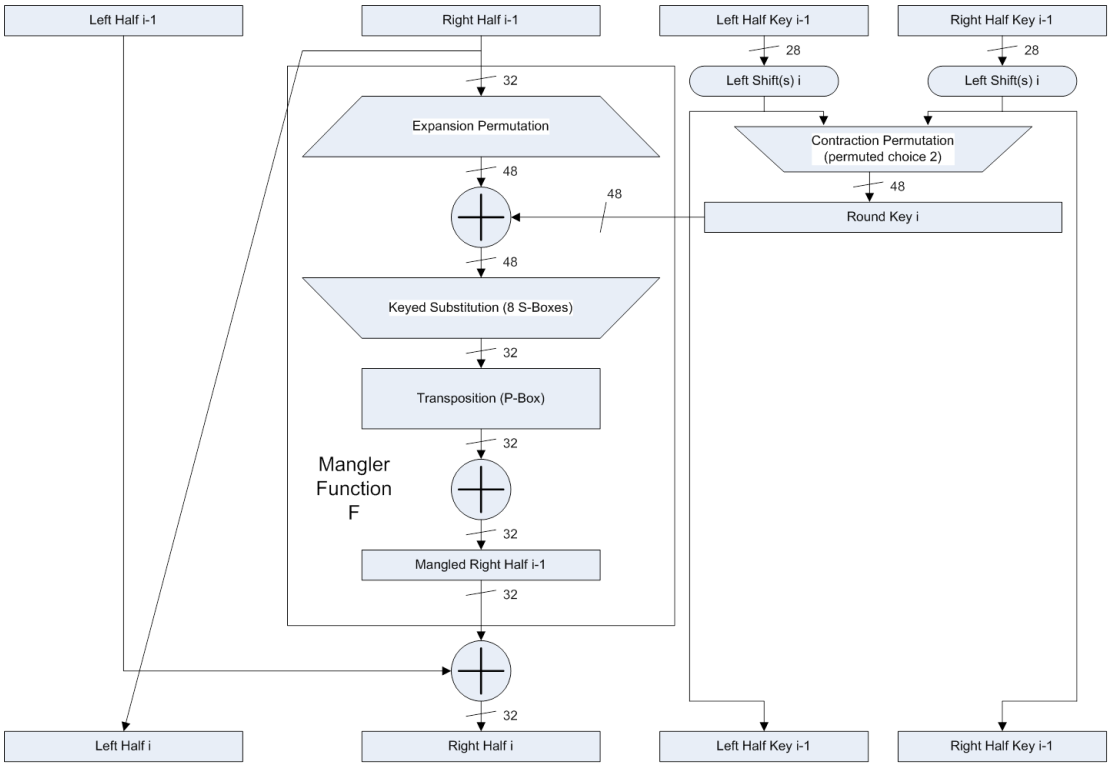
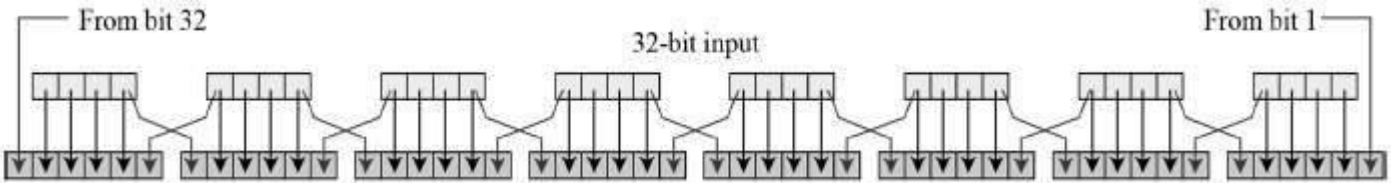


Figure4 :Single Round of DES Alg

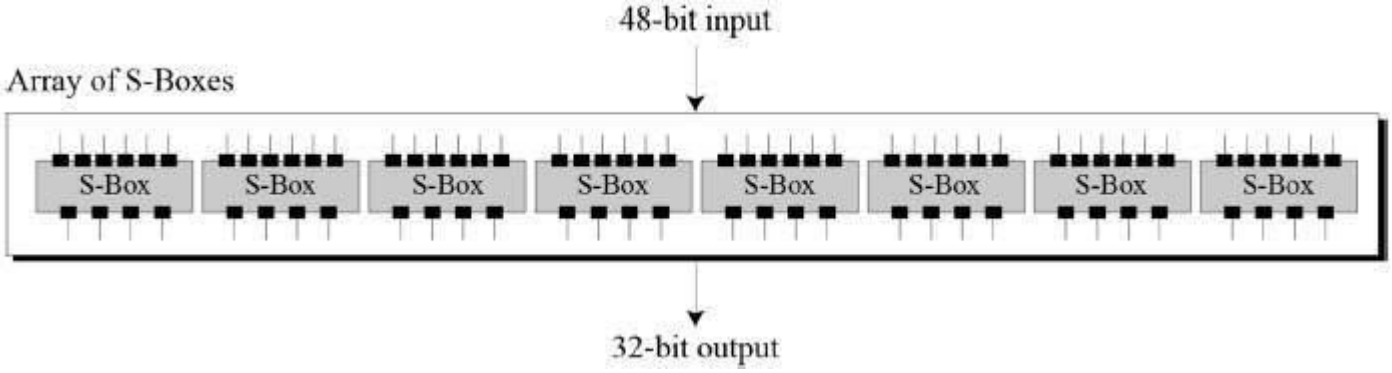
- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration −



- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown −

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	31	31	32	01

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −



- The S-box rule is illustrated below −

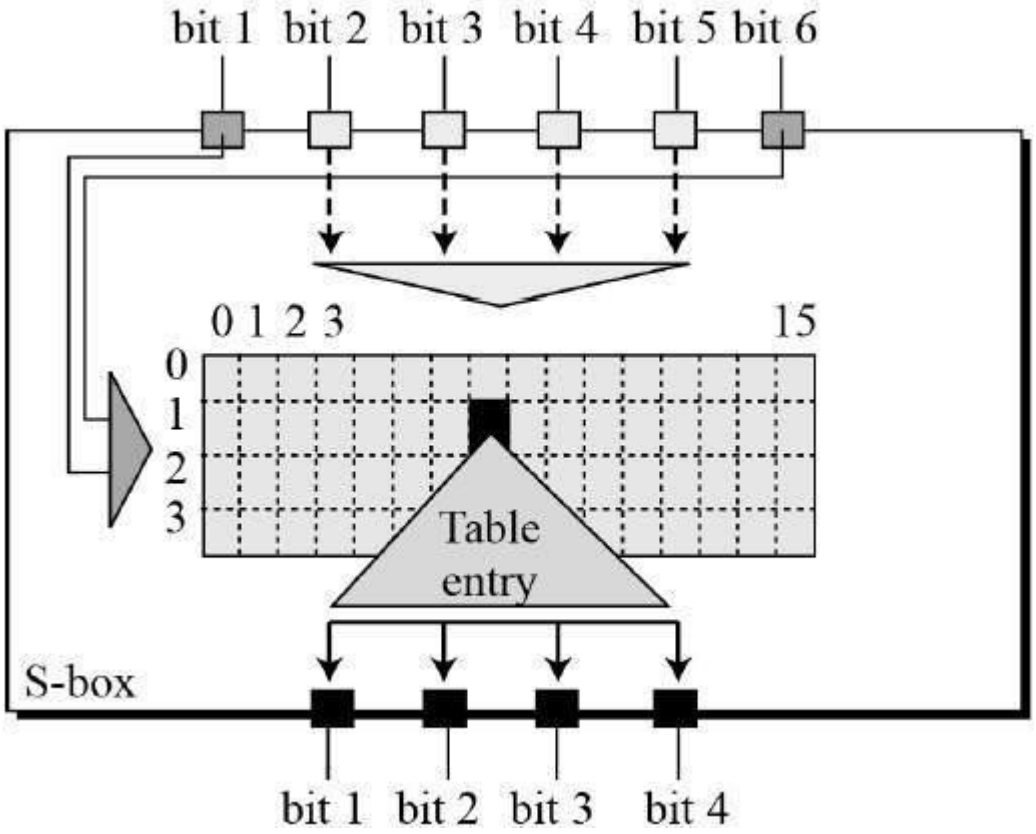


Figure5 :- S-box operation

- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration –

effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption

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

Figure - discording of every 8th bit of original key

We now form the 56 bit key, after that we do the following permutation (**PC-2**) to produce 48 bit keys according to the following table **.**

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

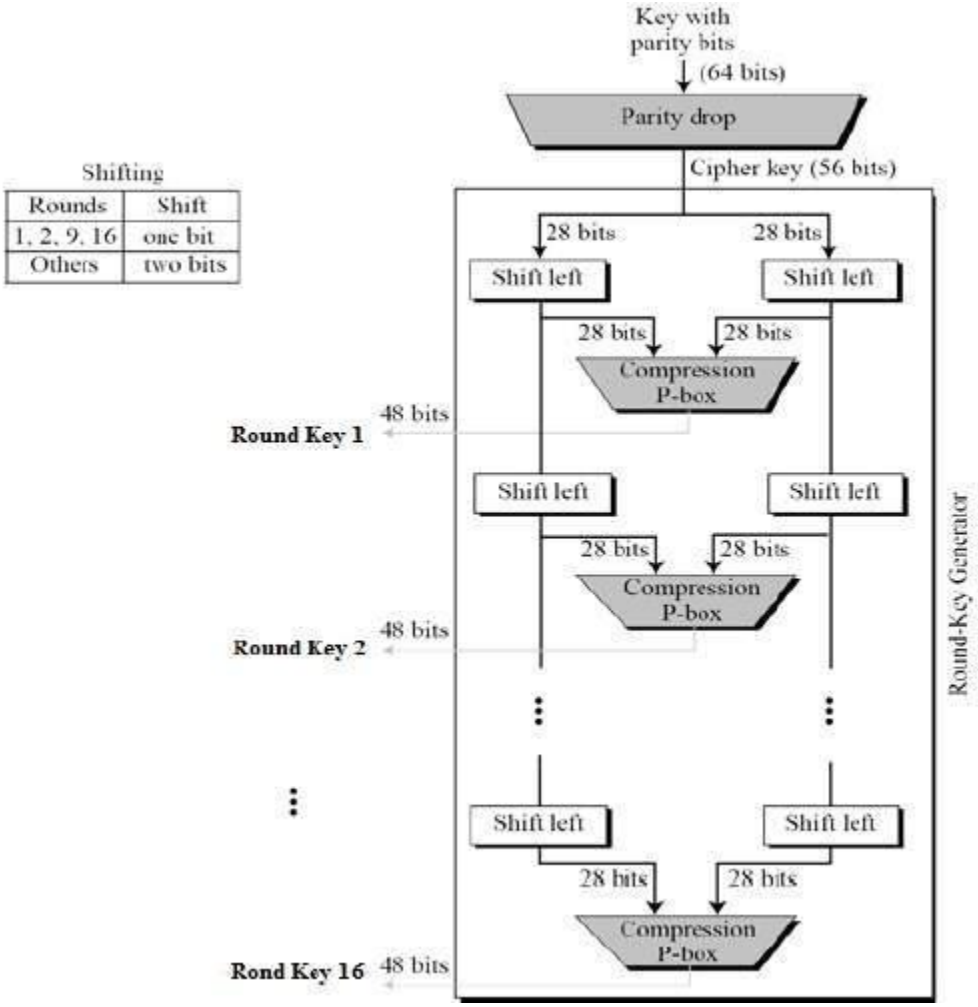


Figure 6 : Key Generation Operation

The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- **Avalanche effect** − A small change in plaintext results in the very great change in the ciphertext.
- **Completeness** − Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys should be avoided.

DES has 4 weak keys

- 01010101 01010101
- FEFEFEFE FEFEFEFE
- E0E0E0E0 F1F1F1F1
- 1F1F1F1F 0E0E0E0E

There are several analytic attacks on DES

- these analytic attack do the following:-
  - gather information about encryptions
  - to recover some/all of the sub-key bits
  - if necessary then exhaustively search for the rest
- these attacks depends on statistical attacks
  - differential cryptanalysis
  - linear cryptanalysis
  - related key attacks

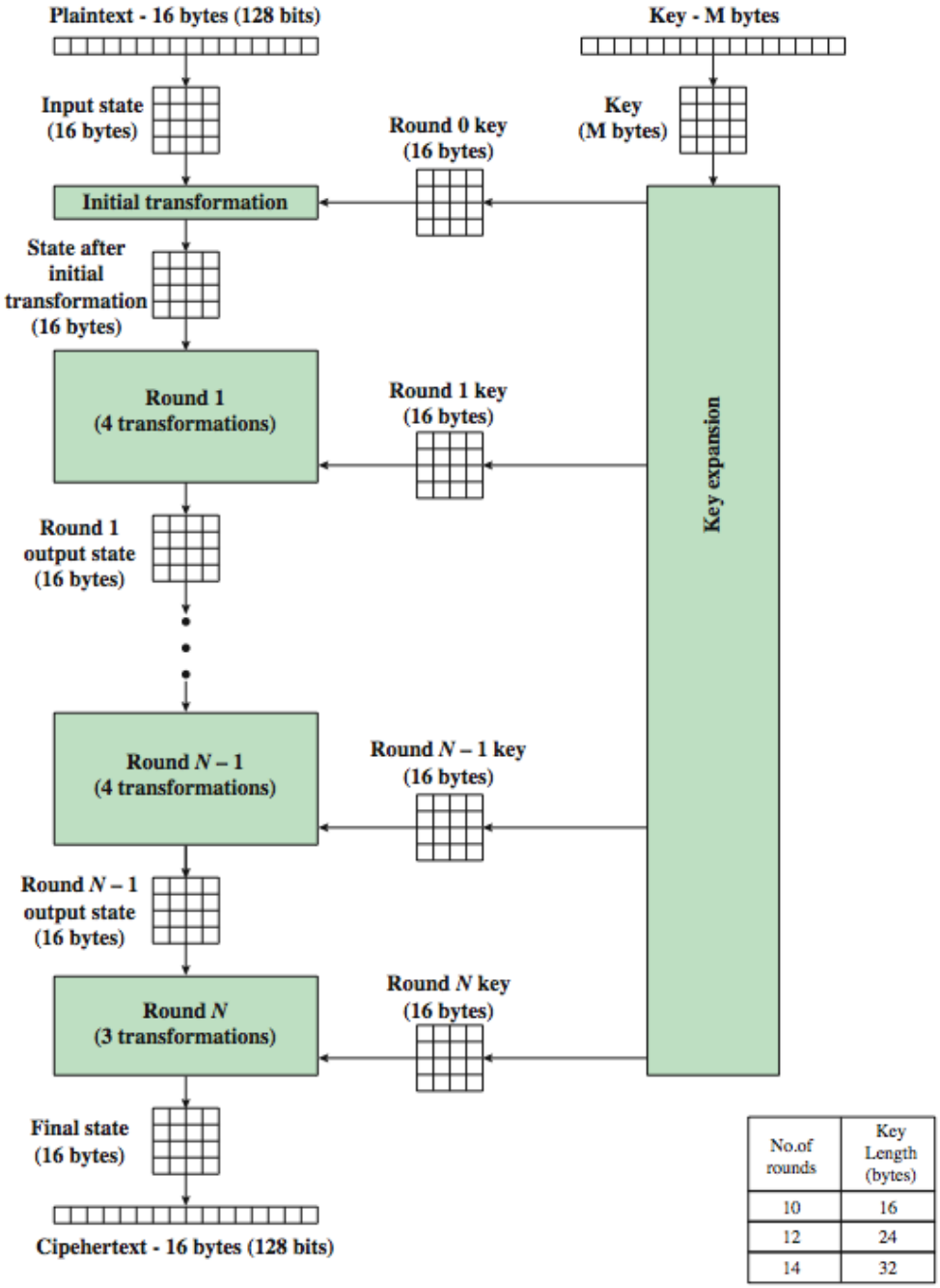
Cryptography

Advanced Encryption Standard (AES)

- The Advanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information. AES is implemented in software and hardware throughout the world to encrypt sensitive data.
- It is essential for government computer security, cybersecurity and electronic data protection.

- clear a replacement for DES was needed
  - have theoretical attacks that can break it

have demonstrated exhaustive key search attacks



No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Encryption Process

AES Cipher

- It has the following bit keys
- 128/192/256
- It has 128 bit data
- an **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to have:
  - resistance against known attacks
  - speed and code compactness on many CPUs
  - simple design

AES Structure

- data block of **4 columns** of **4 bytes** is state
- The cipher takes a plaintext block size of **128 bits**.
- The key length can be **16, 24, or 32 bytes**(**128, 192, or 256 bits**).
- The cipher consists of N rounds, where the number of rounds depends on the key length:
  - **10** rounds for a **16-byte** key
  - **12** rounds for a **24-byte** key
  - **14** rounds for a **32-byte** keyadd round key.

AES Stages

- Four different stages are used, one of permutation and three of substitution:

- **Substitute bytes:** Uses an S-box to perform a byte-by-byte substitution of the block
- **Shift Rows:** A simple permutation.
- **Mix Columns:** A substitution that makes use of arithmetic over **GF**(2^8).
- **Add Round Key:** A simple bitwise XOR of the current block with a portion of the expanded key.
- The final round of both encryption and decryption consists of only three stages.

Strength of the key size

With **AES**, like most modern block ciphers, the **key size** directly relates to the **strength** of the **key** / algorithm. ... Due to the difference in **key** schedule there are related **key** attacks on **AES**-256 but not on **AES**-128 or **AES**-192. The number of rounds is 10, 12 or 14 for the 128, 192 and 256 bit **key size** respectively.

Some comment on AES

1. The AES design is based on

- **a substitution-permutation network (SPN)**
- **does not use the Data Encryption Standard (DES) Feistel network.**

1. each stage is easily reversible

2. **decryption uses keys in reverse order**
3. **decryption does recover plaintext**
4. **final round has only 3 stages**

Substitute Bytes

- **a simple substitution of each byte.**
- **uses one table of 16x16 bytes containing a permutation of all 256 8-bit values**
- **each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)**
  - **eg. byte {19} is replaced by byte in row 1, column 9**
  - **which has value {4D}**
- **S-box constructed using defined transformation of values in GF(2^8)**
- **designed to be resistant to all known attacks**

Substitute Bytes

S- Box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Shift Rows

- a circular byte shift in each row.
  - 1st row is unchanged.
  - 2nd row does 1 byte circular shift to left.
  - 3rd row does 2 byte circular shift to left.
  - 4th row does 3 byte circular shift to left.
- decrypt inverts using shifts to right.
- since state is processed by columns, this step permutes bytes between the columns.

Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF(2^8) using prime poly m(x) =$x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Mix Columns example

(d4.02)⊕(bf.03)⊕(5d.01)⊕(30.01)=04

(1101 0100.02)

1) **if .01 stay the same.**
2) **if .02 :**

- **if left =0 then delete it and add 0 to the Far right.**

  EX: 01011101 ----> 10111010

- **if left =1 then delete it and add 0 to the Far right and make XOR with (1B) or (0001 1011).**

  EX: 11010100 ----> 10101000

  00011011 ⊕

  10110011

(d4.02)⊕(bf.03)⊕(5d.01)⊕(30.01)=04

(1011 1111.03)

3) **if .03 :**

- **Multiply with 01 and 02 and make XOR to the results.**

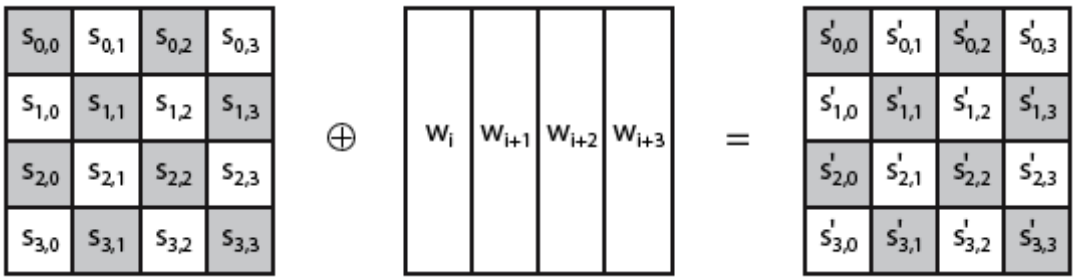
EX: above:

101111111.01 = 10111111

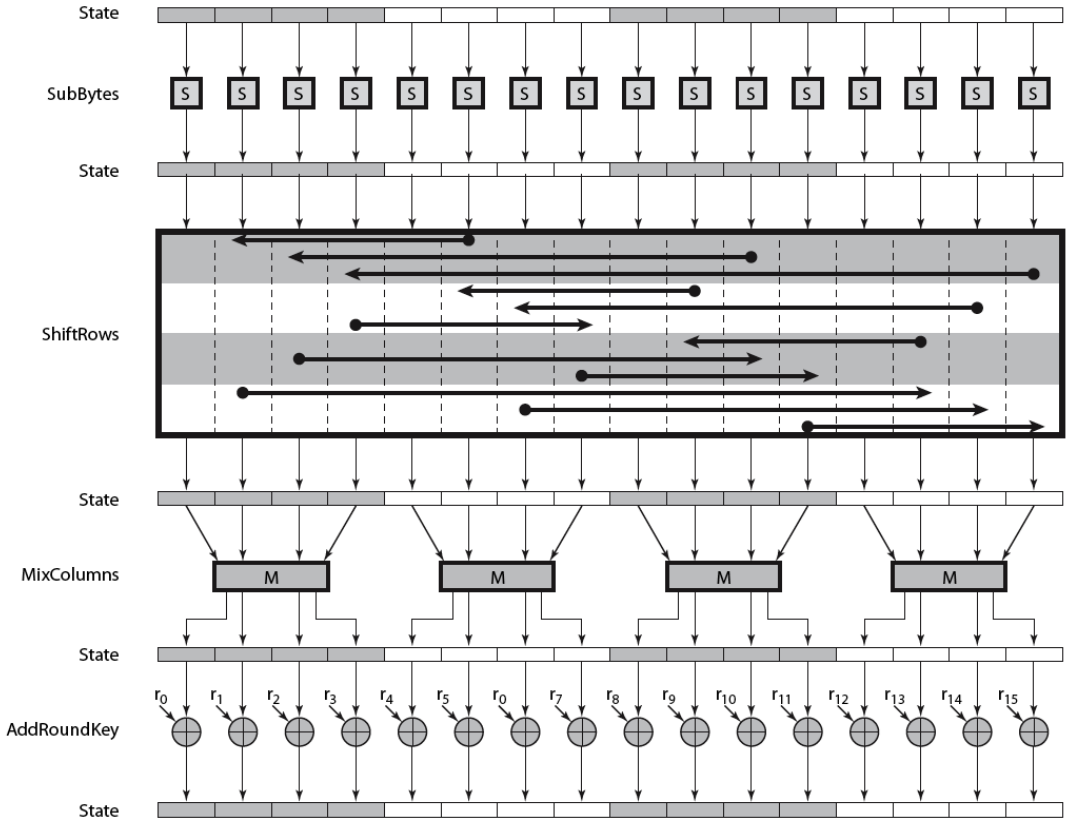
101111111.02 = 01111110 ⊕ 00011011= 01100101

10111111 ⊕ 01100101 = 1101 1010

Add Round Key


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
⊕
w_i w_{i+1} w_{i+2} w_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State