UNIT – 2

2. SYMMETRIC ENCRYPTION

2.1 MATHEMATICS OF SYMMETRIC KEY CRYPTOGRAPHY

The mathematics of symmetric key cryptography can be applied on different sets of elements as follows:

Z: Set of Integers

Z_n: Set of Residues with Modulo n.

Z_n^*: Set of Residues with Modulo n and relatively prime with n.

Z_p: Set of Residues with Modulo p.

Z_p^*: Set of Residues with Modulo n and relatively prime with p.

Where n is an arbitrary integer and p is a prime number.

2.1.1 Algebraic Structures

Cryptography requires sets of integers and specific operations that are defined for those sets. The combination of the set and the operations that are applied to the elements of the set is called an algebraic structure. There are three common algebraic structures: Groups, Rings and Fields as shown in Fig. 2.1 below.

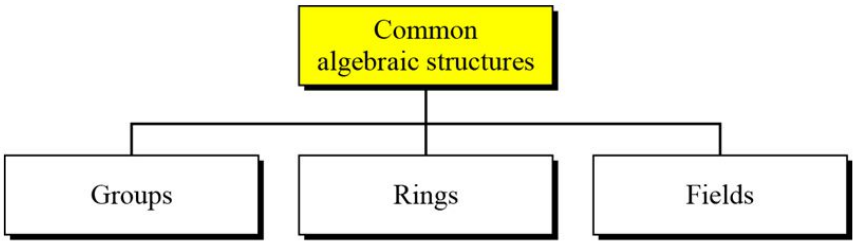


Fig 2.1: Algebraic Structures

Based on five different primitive axioms (Properties) it is decided that, whether the given set of elements and the operator(s) come under Group, Ring or Field.

- **Closure:** If a and b are elements of given set, then c = a • b (spell it as a operated with b) is also an element of the same set.

- **Associativity:** If a, b and c are elements of given set, then (a • b) • c = a • (b • c).
- **Existence of Identity:** For all a in a given set, there exists an element e, called the identity element, such that e • a = a • e = a.
- **Existence of Inverse:** For each a in a given set, there exists an element a', called the inverse of a, such that a • a' = a' • a = e.
- **Commutativity:** For all a and b in a given set, we have a • b = b • a.

Group

A Group G = <{..}, •> is a set of elements with a binary operation "•" that satisfies four axioms, Closure, Associativity, Existence of Identity and Existence of Inverse. A Commutative Group, also called an abelian group, is a group in which the operator satisfies the fifth property Commutativity also.

Example – 1: Given the Group G = $<Z_n, +>$, Prove that this group is an abelian group.

Solution: Let us take the Z_n as Z_{10} in this example, as if we prove for one set of residues, it applies for all.

$$Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Let us have a = 3, b = 4, c = 7.

- Closure: 3 + 4 = 7, 7 is also the member of the same group. So, Closure property is satisfied.
- Associativity: (3 + 4) + 7 = 3 + (4 + 7), So Associativity is satisfied.
- Existence of Identity: Let be identity element e = 0, then 3 + 0 = 0 + 3 = 3, So, Existence of Identity is satisfied.
- Existence of Inverse: The additive inverse of a in Z_{10} is – a, 3+(-3) = -3 + 3 = 0. So, Existence of inverse is satisfied.
- Commutative: 3 + 4 = 4 + 3, So Commutative is satisfied.

As, all the five axioms are satisfied, G = $<Z_n, +>$ is an Abelian Group.

You can work on, G = $<Z_n^*, X>$.

FINITE GROUP: A Group is called a FINITE GROUP if the set has finite number of elements, otherwise, it is an INFINITE GROUP.

ORDER OFA GROUP: The order of a group (G), is the number of elements in the group. If the group is not finite, its order is infinite, if the group is finite, the order is finite.

SUBGROUPS: A subset H of group G is a subgroup of G, if H itself is a group with respect to the operation on G. In other words, if G = <S, •> is a group, H=<T, •> is a group under the same operation, and T is a nonempty subset of S, then H is a subgroup of G.

Eg: H = $<Z_{10}, +>$ is a subgroup of G = $<Z_{12}, +>$

Ring

A Ring, denoted as R = <{….}, •, □>, is an algebraic structure with two operations. The first operation must satisfy all five axioms required for an abelian group. The second operation must satisfy only the first two, i.e., Closure and Associativity. In addition, the second operation must be distributed over the first. i.e.,

$$a \square (b \bullet c) = (a \square b) \bullet (a \square c) \text{ and } (a \bullet b) \square c = (a \square c) \bullet (b \square c)$$

A commutative ring, is a ring in which the commutative property is also satisfied fort the second operation.

Eg: R = <Z, +, X> is a commutative ring.

Field

A field, denoted by F = <{….}, •, □> is a commutative ring in which the first and second operations satisfies all the five axioms.

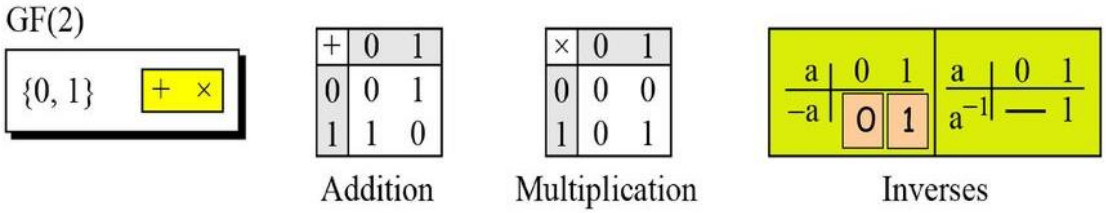
FINITE FIELDS: A Finite field is a field with finite number of elements, is a very important structure in cryptography.

Galois showed that for a finite field to be finite, the number of elements should be p^n, where p is a prime and n is a positive integer. The finite fields are usually called Galois Fields and denoted as $GF(p^n)$.

GF(p) FIELDS: When n=1, we have GF(p) field. This field can be the set Z_p, {0, 1, 2, .., p-1}, with two arithmetic operations, addition and multiplication. Where each element in the set has an additive and multiplicative inverse (for non-zero elements).

Example: GF(2), with set Z_2 = {0, 1} and two operations, addition and multiplication.

Solution:



Addition/ Subtraction in GF(2) is the same as the XOR operation, multiplication/ Division is the same as the AND operation.

GF(2^n) FIELDS: The GF(2^n) uses a set of 2^n elements. The elements in the set are n-bit words. For example, if n = 3, the set is

$$GF(2^3) = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Let us define a GF(2^2) field in which the set has four 2 bit words: {00, 01, 10, 11}. We can apply addition and multiplication for this field in such a way that all axioms are satisfied, as shown below.

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

X	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

2-bit binary Addition can be computed as: 00 + 01 = 0 + 0, 0 + 1 = 01

2-bit binary Multiplication can be computed as:

10	01
ab	cd

=

1	0
ad + bc + ac	ac + bd

2.1.2. Polynomials

As we already defined the rules for addition and multiplication operations on n-bit words that satisfy the axioms in GF(2^n), but it is easier to work with a polynomial of degree n – 1. A polynomial of degree n-1 is an expression of the form as below

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x^1 + a_0x^0.$$

Where x^i is called the ith term and a_i is called coefficient of the ith term.

Example: n – bit word: 10011001 can be represented in polynomial as

1	0	0	1	1	0	0	1
$1x^7$	$0x^6$	$0x^5$	$1x^4$	$1x^3$	$0x^2$	$0x^1$	$1x^0$

So, the polynomial is $x^7 + x^4 + x^3 + 1$.

POLYNOMIAL ADDITION:

Now, let us see how addition operation can be performed on polynomials. The addition operation is simple as such that we need to add (XOR) the quotients of the polynomial to get the result.

Example:

	$x^7 + x^5 + x^3 + x + 1$
	$x^6 + x^5 + x^4 + x^3 + 1$
+	$x^7 + x^6 + x^4 + x$

Note: The shortcut of polynomial addition is to eliminate the common exponents and take the different exponents.

Addition of two polynomials never creates a polynomial out of the degree. The additive identity in a polynomial is a zero polynomial (a polynomial with all coefficients set to zero) because adding a polynomial with itself results a zero polynomial. The additive inverse of a polynomial with coefficients in GF(2) is the polynomial itself. This means that the subtraction operation is the same as the addition operation. Addition and subtraction operations on polynomials are the same operations and leads to XOR operation.

POLYNOMIAL MULTIPLICATION:

The sum of multiplication of each term in the first polynomial to each term of the second polynomial is the multiplication of two polynomials. The rules for polynomial multiplication is mentioned below.

1. The coefficient multiplication is done in GF(2).
2. Multiplying x^i with x^j leads to x^{i+j}.
3. The multiplication may create terms with degree more than n-1, which means we need to reduce using the modulus operation.

Example: Multiply $x^5 + x^2 + 1$ with $x^6 + x^4 + x$ in GF(2^8)

	$x^5 + x^2 + 1$
	$x^6 + x^4 + x$
Multiplication	$x^5(x^6 + x^4 + x) + x^2(x^6 + x^4 + x) + 1(x^6 + x^4 + x)$
	$x^{11} + x^9 + \cancel{x^6} + x^8 + \cancel{x^6} + x^3 + x^6 + x^4 + x$
Addition of common terms makes the coefficients to zero	$x^{11} + x^9 + x^8 + x^6 + x^4 + x^3 + x$

Now, after multiplying $x^5 + x^2 + 1$ with $x^6 + x^4 + x$ in GF(2^8) we get the resultant polynomial as $x^{11} + x^9 + x^8 + x^6 + x^4 + x^3 + x$. This resultant polynomial is having the degree more than n – 1 i.e., 8 – 1 = 7. So, we need to reduce using the modulus operation.

The modulus operation on the polynomials can be performed using a prime polynomial. This means a polynomial that cannot be factored into a polynomial with degree of less than n. Such polynomials are also referred as irreducible polynomials. The example irreducible polynomials are shown in table below.

Degree	*Irreducible Polynomials*
1	$(x+1), (x)$
2	(x^2+x+1)
3	$(x^3+x^2+1), (x^3+x+1)$
4	$(x^4+x^3+x^2+x+1), (x^4+x^3+1), (x^4+x+1)$
5	$(x^5+x^2+1), (x^5+x^3+x^2+x+1), (x^5+x^4+x^3+x+1),$ $(x^5+x^4+x^3+x^2+1), (x^5+x^4+x^2+x+1)$

Now, we need to add the resultant polynomial with irreducible polynomial with the degree n to get the remainder as the polynomial of degree less than n-1.

So, we divide $x^{11}+x^9+x^8+x^6+x^4+x^3+x$ with prime polynomial of degree n-1 i.e., 7 as $x^7+x^3+x^2+x+1$.

$$
\begin{array}{l}
x^7+x^3+x^2+x+1\;)\;\cancel{x^{11}}+x^9+x^8+\cancel{x^6}+\cancel{x^4}+x^3+x\;(\;x^4+x^2+x \\
\quad\underline{\cancel{x^{11}}+x^7+\cancel{x^6}+x^5+\cancel{x^4}} \\
\qquad\quad \cancel{x^9}+x^8+x^7+\cancel{x^5}+\cancel{x^3}+x \\
\qquad\quad\underline{\cancel{x^9}+\cancel{x^5}+x^4+\cancel{x^3}+x^2} \\
\qquad\qquad\quad \cancel{x^8}+x^7+\cancel{x^4}+\cancel{x^2}+x \\
\qquad\qquad\quad\underline{\cancel{x^8}+\cancel{x^4}+x^3+\cancel{x^2}+x} \\
\qquad\qquad\qquad\quad x^7+x^3
\end{array}
$$

The final result after multiplying x^5+x^2+1 with x^6+x^4+x in GF(2^8) with prime polynomial $x^7+x^3+x^2+x+1$ is x^7+x^3.

The multiplicative identity is always x^0 i.e., 1. For example, in GF(2^8), the multiplicative identity is the bit pattern 00000001. The multiplicative inverse is a calculated using extended Euclidean algorithm.

2.2 INTRODUCTION TO MODERN SYMMETRIC CIPHERS

A user, Alice, can send a message to another user, Bob, over an insecure channel with the assumption that an attacker, Eve, cannot understand the contents of the message by simply snooping over the channel. Alice converts his plaintext into ciphertext with the help of Encryption algorithm and a shared secret key. The ciphertext is then transferred from

Alice to Bob over an insecure channel, where an attacker 'Eve' is one of the participant of the same channel. When the message reaches the Bob, he converts the ciphertext to plaintext using the Decryption algorithm and the same shared secret key. As the Eve is one of the participant of the same insecure channel he can snoop into the channel, but, the data transfer between Alice and Bob is in the ciphertext format, and can be converted into plaintext only with the help of Decryption algorithm and shared secret key, Eve doesn't have any idea of shared secret key, he cannot convert it into plaintext and read the message.

The Encryption and Decryption algorithms are called as Ciphers. A Key is a set of values that the cipher, as an algorithm, operates on. In Symmetric Ciphers, the algorithm is openly announced to all the users but, the key should be kept secret. If the key gets compromised, the overall intention of symmetric cryptography was lost. So, to maintain secrecy, the users frequently changes their secret key.

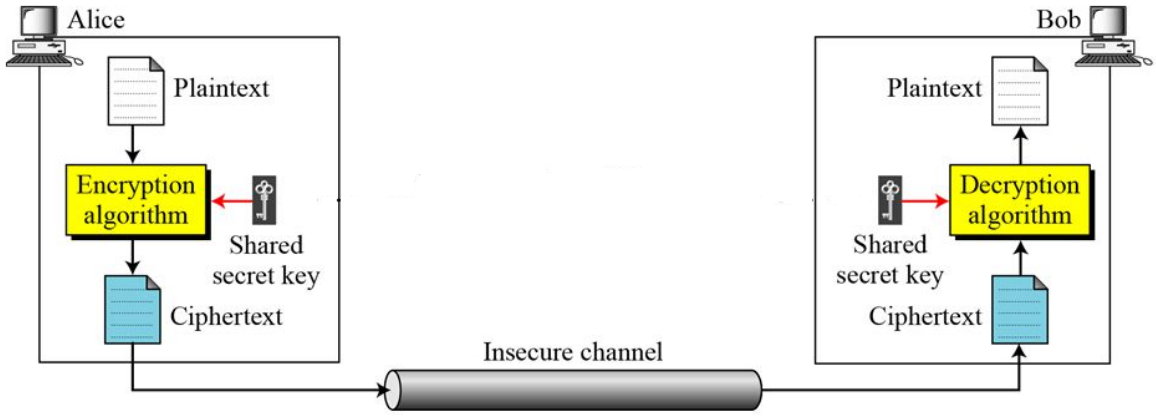


Figure 2.1: General Idea of Symmetric key cipher

$$Encryption\ at\ Alice: C = E_k(P),\ Decryption\ at\ Bob: P = D_k(C)$$

Symmetric Cryptography doesn't provide any process to share the secret key between two users. It is upto the users to secretly share the key and make use of symmetric ciphers to transfer secret message between them.

2.2.1 Categories of Traditional Ciphers

Traditional Ciphers can be categorized into two types: Substitution Ciphers and Transposition Ciphers.

SUBSTITUTION CIPHERS

A substitution cipher replaces one symbol with another. For example, we can replace letter A with letter D, and digit 3 with 7.

Caesar Cipher:

The basic example of substitution ciphers is Caesar Cipher. This Cipher works with a numerical key starting from 0 to 25 in Z_{26}. Alphabets are numbered with 0 to 25 from a to z as shown in table below.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	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

The ciphertext can be computed as follows:

$$C = (P + k) mod\ 26$$

And at the receiver side plain text can be computed as follows:

$$P = (C - k) mod\ 26$$

Example: Encrypt and Decrypt "Welcome to VVIT" using key 3.

Solution:

Encryption:

w	e	l	c	o	m	e	t	o	v	v	i	t
+3	+3	+3	+3	+3	+3	+3	+3	+3	+3	+3	+3	+3
Z	H	O	F	R	P	H	W	R	Y	Y	L	W

Decryption:

Z	H	O	F	R	P	H	W	R	Y	Y	L	W
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
w	e	l	c	o	m	e	t	o	v	v	i	t

Caesar cipher can be easily cracked using brute force attack, as there are only 26 possible keys. The attacker applies all the possible keys and extracts the plaintext easily.

Monoalphabetic Cipher:

In Monoalphabetic Cipher, each plaintext letter is replaced with a ciphertext letter. Each alphabet in English will have a replacement with another alphabet to form ciphertext, for example as shown below.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	F	Z	A	B	E	C	I	L	H	T	U	X	R	Q	S	Y	K	M	O	N	G	J	P	V	W

For Monoalphabetic Cipher, the size of key is 26 alphabets and the key will be the list of replaced alphabets.

Encryption: Asper key mentioned above.

w	e	l	c	o	m	e	t	o	v	v	i	t
J	B	U	Z	Q	X	B	O	Q	G	G	L	O

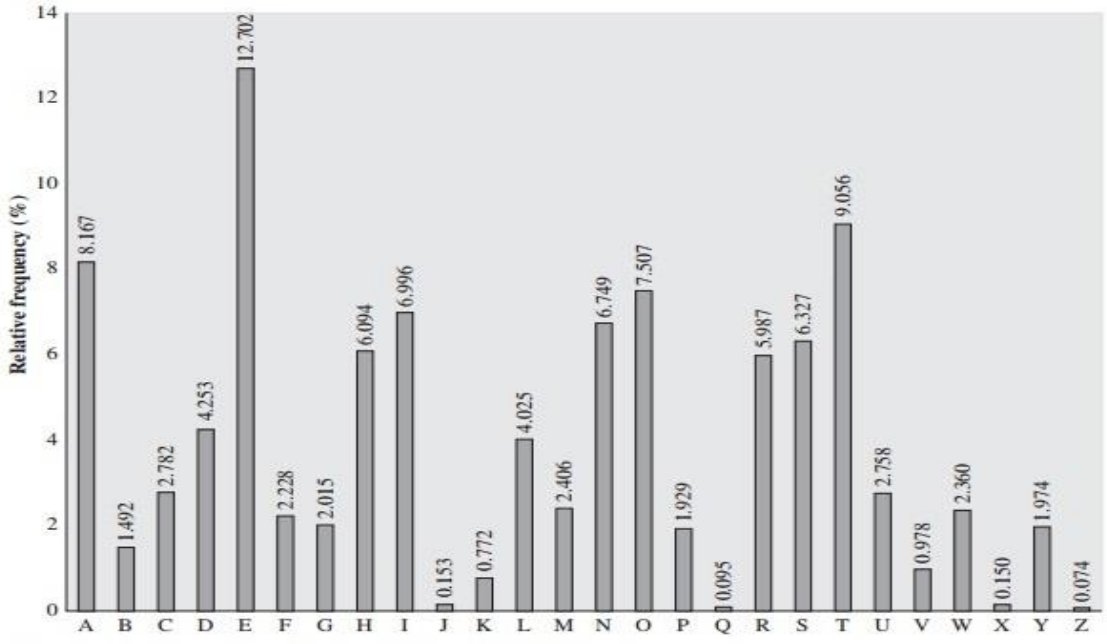
Decryption:

J	B	U	Z	Q	X	B	O	Q	G	G	L	O
w	e	l	c	o	m	e	t	o	v	v	i	t

As the key size increased to 26 characters when compared with Caesar cipher. So, there are 26! possible keys the sender and receiver can use. If an attacker decrypt the ciphertext using the bruteforce attack, he has to try 26! Keys on the ciphertext to retrieve the plaintext, which is highly impossible. So, Monoalphabetic cipher can counter against the bruteforce attack.

But, the attacker knows the nature of the plaintext, then he can exploit the regularities of the language based on the standard frequency distribution for English as shown in below.

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				



The attacker tries to retrieve the plaintext by replacing the most repeated letters in the ciphertext with the letters having equal frequency in English.

TRANSPOSITION CIPHERS

Transposition ciphers, changes the position of the plaintext letters to form the ciphertext. For example, plaintext "VVIT" can be converted to ciphertext as "IVTV".

Rail Fence Cipher

In Rail Fence cipher, the plaintext letters are written in a zigzag pattern to form the ciphertext. The pattern will be organized in such a way to form a huge difference between plaintext and the ciphertext.

Example: Encrypt "Meet me in the party" using Rail Fence Cipher.

Solution:

Encryption:

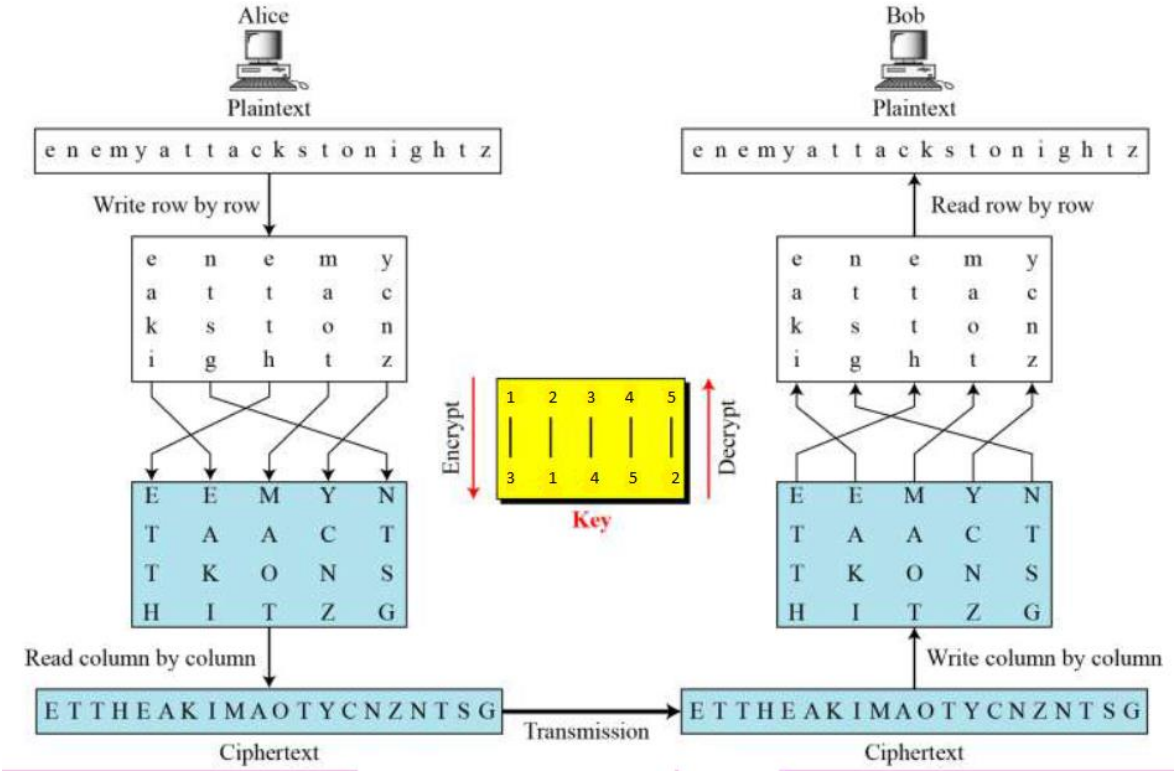
M e m i t e a t
e t e n h p r y

Ciphertext: “MEMITEATETENHPRY”

Column Transposition Cipher

Column Transposition Cipher organizes the plaintext into a number of columns based on the key. It builds the ciphertext by listing the plaintext letters in the order the key is assigned to the columns. The working of column Transposition cipher is clearly shown in below example.

Example: Encrypt the plaintext “enemy attacks tonightz” with the key 31452.

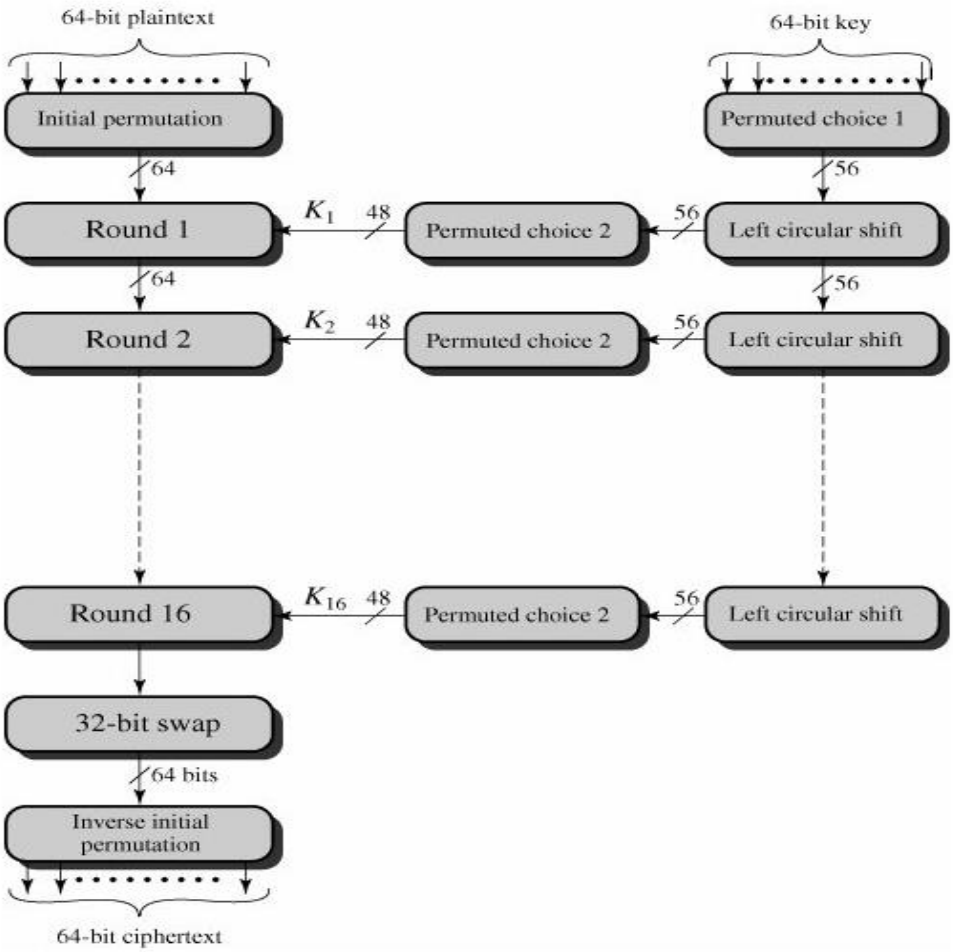


2.3 DATA ENCRYPTION STANDARD

Data Encryption Standard (DES) is a modification of project "LUCIFER" by IBM in 1971. Later it was adopted in 1977 by National Bureau of Standards, and then to National Institute of Standards and Technology.

DES encrypts data in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit ciphertext. As DES is a Fiestel Cipher, the reverse procedure of encryption algorithm can be used as decryption algorithm.

As with any encryption scheme, there are two inputs to the encryption function: the plaintext and the key. In this case, the plaintext must be 64 bits in length and the key is 64 bits (later converted to 56 bits when operated with plaintext). The block diagram show below displays the general working of DES algorithm.

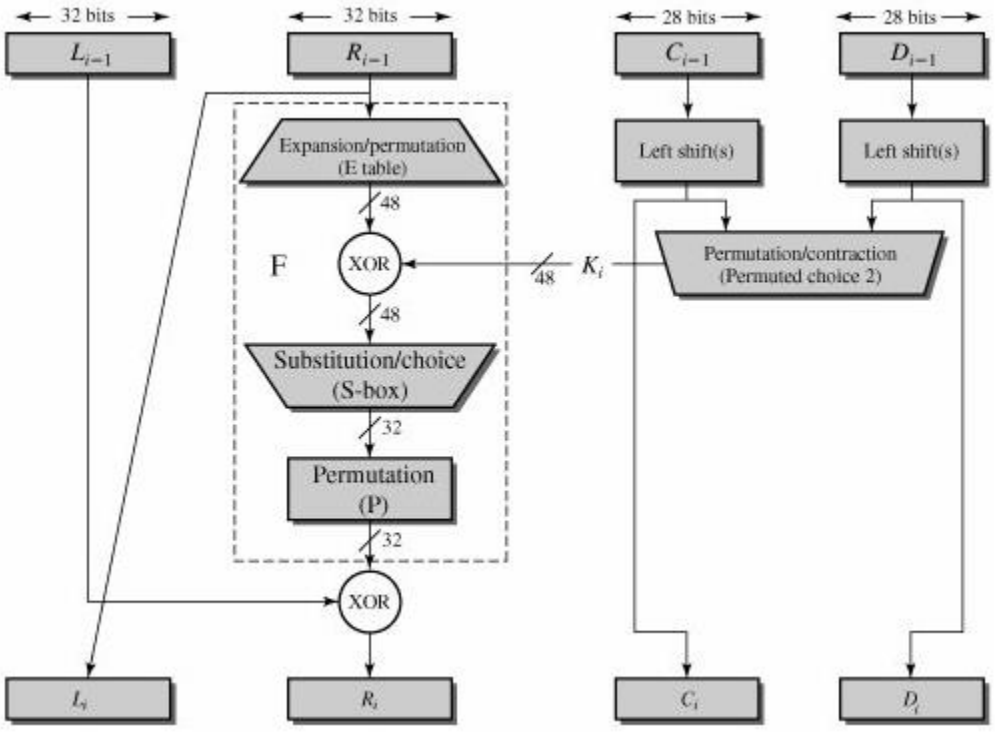


In the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an Initial permutation (IP) that arranges the bits to produce the permuted input. This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions. The diagram below shows the internal structure of single round. Again, focusing on the left-hand side of the diagram. The plaintext of size 64-bit is divided into two halves of each 32-bits and labeled as L (Left) and R (Right). The overall processing at each round can be summarized in the following formulas:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \times F(R_{i-1}, K_i)$$

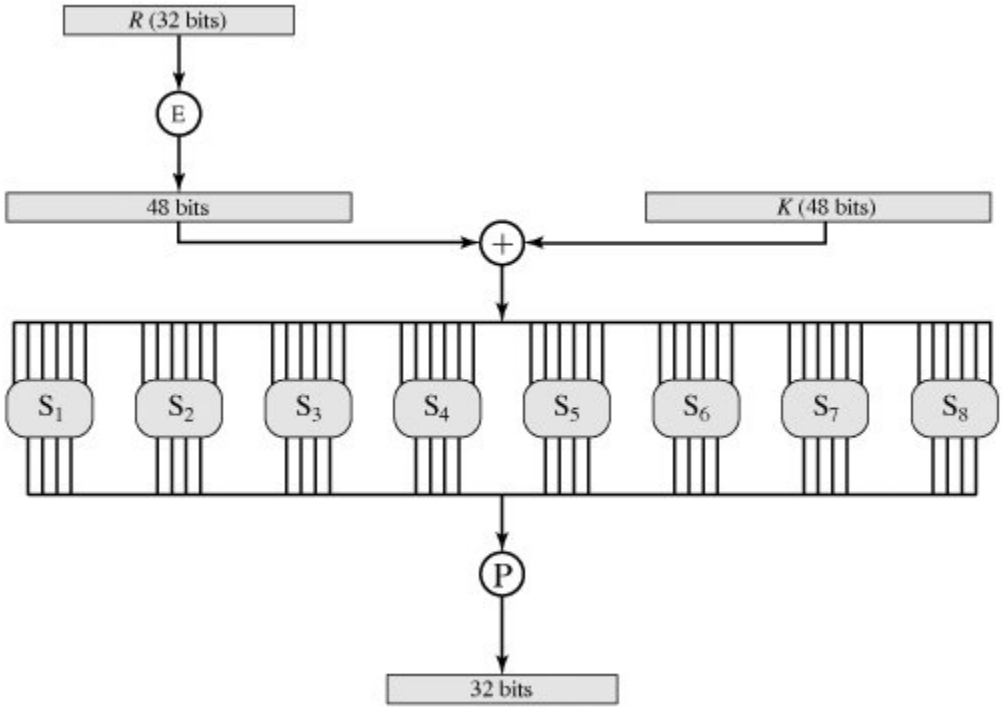
The round key K_i is 48 bits. The R input is 32 bits. This R is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits. The resulting 48 bits are XORed with K_i. This 48-bit result passes through a substitution function (S-Boxes) that produces a 32-bit output.



The role of the S-boxes in the function F, consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. The first and last bits of the input to

box S_i form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S_i. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit binary value to produce the output. For example, in S_1 for input 011001, the row is decided by 01, i.e., row 1 and the column is decided by 1100, i.e., column 12. The value in row 1, column 12 is 9, so the output is 1001. The calculation of substitute bits by S-boxes is clearly shown in the below diagram.

The output of the last round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped in 32-bit swap to produce the preoutput. Finally, the preoutput is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.



The DES Encryption continues the same for every plaintext block of size 64-bits and produces respective ciphertext blocks of size 64-bits.

The input key for DES is of size 64-bits are converted into 56 bit by processing it through Permuted Choice One (PC – 1), which ignores every 8^{th} – bit in the key. This 56 – bit key is then treated as two 26-bit quantities. At each round, the two halves are separately subjected

to a circular left shifts, or rotation, of 1 or 2 bits. These shifted values serve as input key to the next round. They also serve as input to Permuted Choice Two, which produces a 48-bit output that serves as input to the function $F(R_{i-1}, K_i)$.

2.4 ADVANCED ENCRYPTION STANDARD

In the year 1997, National Institute of Science & Technology started looking for alternative to DES, which would be Advanced Encryption Standard (AES). Up on hearing to many proposals, NIST announced "Rijndael", designed by Joan Daemen and Vincent Rijment, was selected as Advanced Encryption Standard in October 2000.

The Rijndael proposal for AES defined a cipher in which the key size can have multiple variants such as 128, 192, or 256 bits, but limits the block length to 128 bits. Different parameters regarding the AES implementation are clearly shown in the below table.

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

The input to the encryption and decryption algorithms is a single 128-bit block, which is organized as a square matrix of 4 X 4 bytes. This matrix is called as state array. A combination of 4 bytes is called as a Word, so, every row or column in a state array represents a word. The example of a state array is shown below.

Example: State Array

$$\begin{bmatrix} EA & 04 & 65 & 85 \\ 83 & 45 & 5D & 96 \\ 5C & 33 & 98 & B0 \\ F0 & 2D & AD & C5 \end{bmatrix}$$

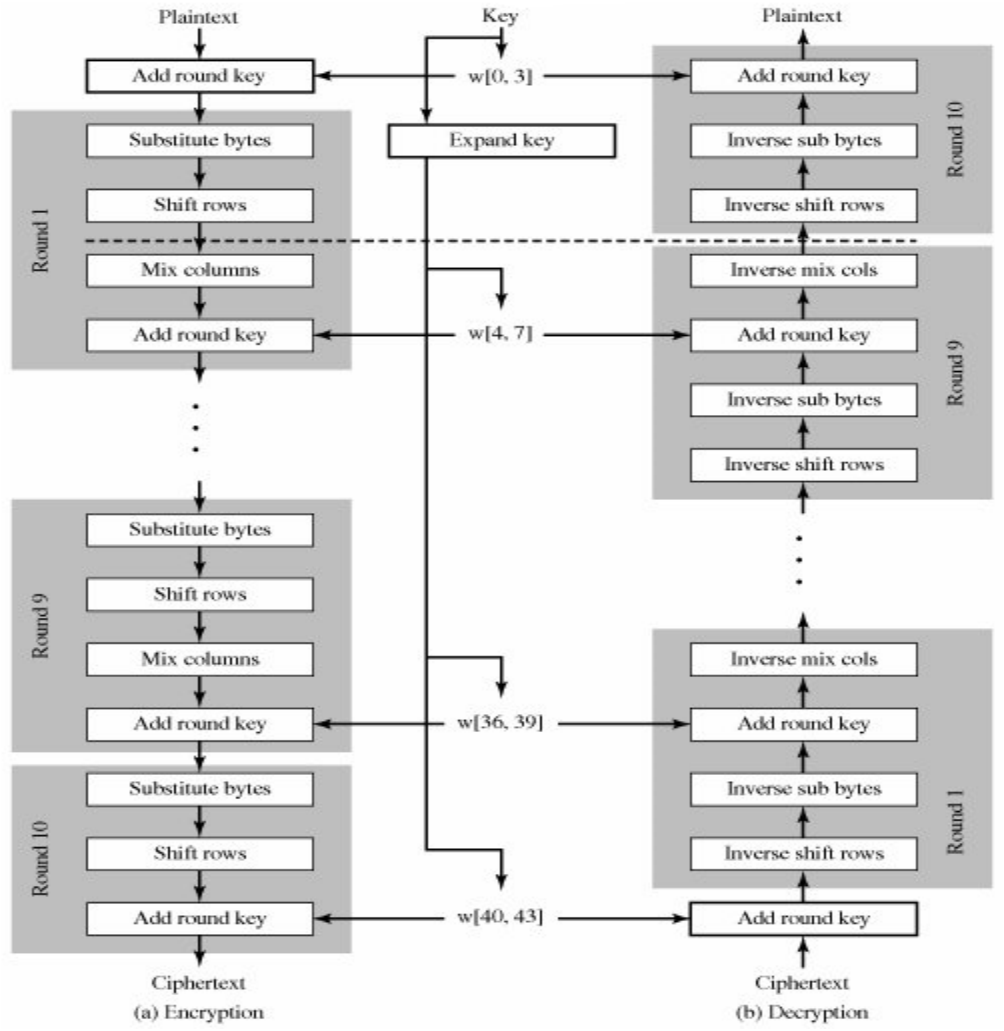
AES had four stages in each round, Substitute bytes, Shift rows, Mix Columns and Add round key. The state array is modified at each stage of the encryption or decryption. After

the final stage, state array is copied to an output array. The structure of AES and stages in each round is clearly shown in below figure.

Similarly, the 128-bit key is represented as a square matrix of bytes. This key is expanded into an array of key schedule words and represented as array **w**. Each word is four bytes and the total key schedule is 44 words for the 128-bit key.

The ordering of plaintext bytes in the state array is by column. i.e., the first four bytes of plaintext will occupy the first column and next succeeding bytes occupy the next columns as shown below. Similarly, the same ordering for the key schedule array w.

$$\begin{bmatrix} B0 & B4 & B8 & B12 \\ B1 & B5 & B9 & B13 \\ B2 & B6 & B10 & B14 \\ B3 & B7 & B11 & B15 \end{bmatrix}$$



The structure 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 tenth round of three stages.

2.4.1 Substitute Bytes Transformation

There are forward and inverse substitute byte transformations, forward is used for the encryption process and inverse is used for the decryption process. The substitute byte transformation, called SubBytes, is a simple table lookup. AES defines a 16 X 16 matrix of byte values, called an S-Box, that contains a permutation of all possible 256 byte values. Each individual byte of state array 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 2A is substituted in the place of 95. The same procedure applies for inverse s-box also. The 16 X 16 matrix for s-box and inverse s-box is shown below.

(a) S-box

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**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

(b) Inverse S-box

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

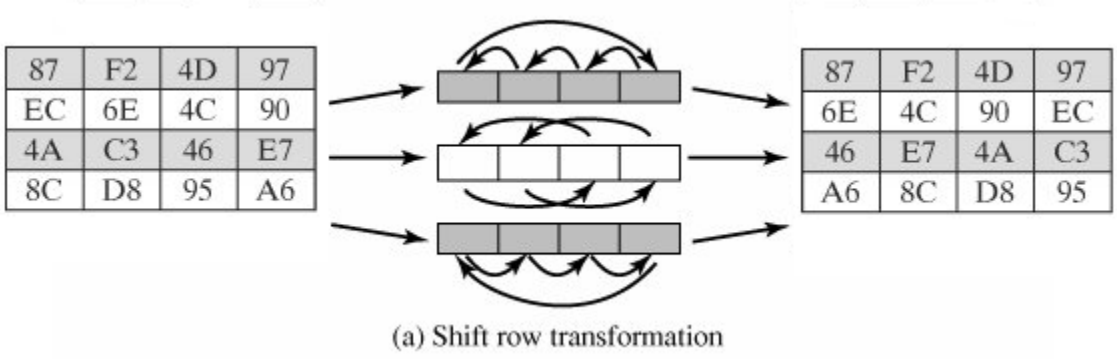
Example: Apply SubBytes for the bytes in the following state array.

$$\begin{bmatrix} EA & 04 & 65 & 85 \\ 83 & 45 & 5D & 96 \\ 5C & 33 & 98 & B0 \\ F0 & 2D & AD & C5 \end{bmatrix} \rightarrow \begin{bmatrix} 87 & F2 & 4D & 97 \\ EC & 6E & 4C & 90 \\ 4A & C3 & 46 & E7 \\ 8C & D8 & 95 & A6 \end{bmatrix}$$

2.4.2 ShiftRows Transformation

Like SubBytes, ShiftRows Transformation have two forms, Forward and Inverse ShiftRow Transformation, called ShiftRows. This operation applies transposition in the state array. 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 operation of forward ShiftRows Transformation is clearly shown in below example.

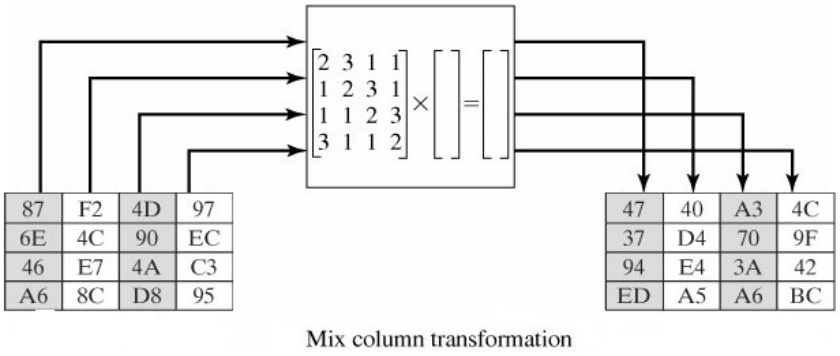
Example: Apply ShiftRows for the state array calculated in SubBytes.



(a) Shift row transformation

2.4.3 MixColumns Transformation

MixColumns, operates on each column individually. Each byte of a column is mapped into a new value by applying matrix bit multiplication between the state array and a predefined matrix. This operation is performed by column wise, i.e., for each word individually and the result will again form a new state array. The operation of MixColumn is clearly shown in following figure.



Mix column transformation

2.4.4 AddRoundKey Transformation

In AddRoundKey transformation, called AddRoundKey, the 128 bits of state array are bitwise XORed with the 128 bits of the round key. This 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. This operation is explained with the following example.

Example:

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

2.4.5 Key Expansion

The AES original key of size can only form 4 words of 16 bytes or 128 bits. The remaining 40 words can be generated with the help of AES Key expansion algorithm. This algorithm generated remaining 40 words with the help of existing 4 words generated from original key. The process of key expansion is clearly shown in the following figure.