Cryptography and Network Security

Unit-III

Session 16

Dr. V. K. Pachghare

ENCRYPTION TECHNIQUES

Advanced Encryption Standard

(AES)

Introduction

- DES suffers by brute force attack
- Advanced encryption standard (AES) is also called Rijndael algorithm, emerges as the alternative option
- A variable number of rounds
- The number of rounds depends on the key size

- Design of AES algorithm does not based on Feistel structure.

- Design of AES algorithm does not based on Feistel structure.
- It is based on linear transformation.

- Design of AES algorithm does not based on Feistel structure.
- It is based on linear transformation.
- AES uses different transformations such as
  - substitution,
  - permutation,
  - the mix column and
  - round key addition

- This transformation forms a state.
- A state defines the current condition of the block during encryption.
- A state is nothing but the block of 4 x 4 matrix of bytes which is currently being processed on.

Terminology

State: Defines the current condition (state) of the block. That is the block of bytes that are currently being worked on. The state starts off being equal to the block, however it changes as each round of the algorithms executes. This is the block in progress.

Block: AES can currently encrypt blocks of 128 bits at a time; no other block sizes are presently a part of the AES standard.

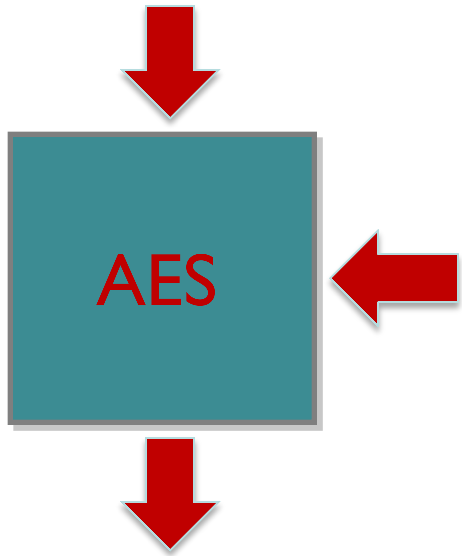
AES

- Plaintext block size: 128 bits
- Key size: 128 bits/192 bits/256 bits
- Number of Rounds: 10/12/14

Key Size [bits]	Number of Rounds
128	10
192	12
256	14

AES Conceptual Scheme

Plaintext (128 bits)



Key (128-256 bits)

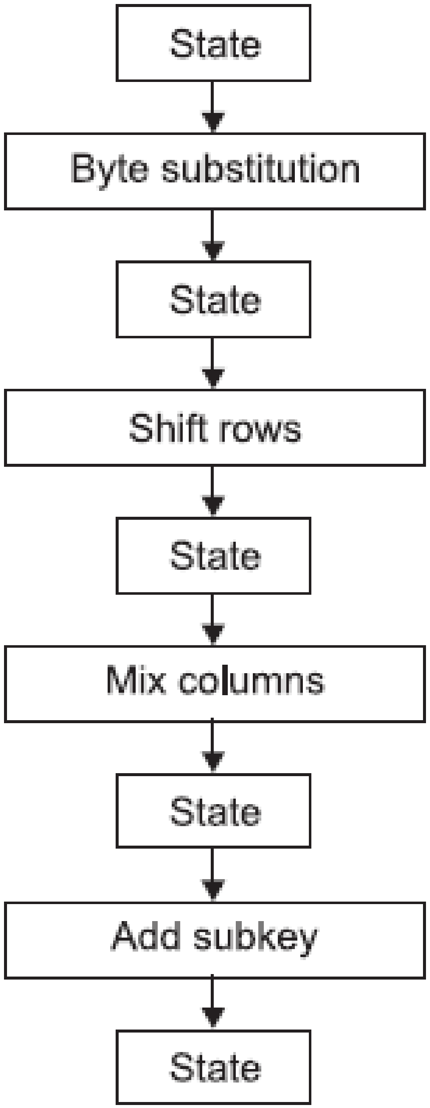
Ciphertext (128 bits)

Structure of AES

Key expansion	Subkeys are generated from original key for each round
Initial round	XOR operation between the state and the round key
Rounds 1 to 9	Each round has four steps: byte substitution shift rows, mix columns, add subkey
Final round	This round has three steps: byte substitution, shift rows, add subkey

Each round consists of four stages

- Byte substitution (SubBytes)
- Shift Rows
- Mix Columns
- Add Subkey (AddRoundKey)


State
Byte substitution
State
Shift rows
State
Mix columns
State
Add subkey
State

1. Byte substitution (SubBytes)

- It is also called SubBytes step.
- Uses an S-box to perform a byte-by-byte substitution of the block
- This is a non-linear operation.
- It uses S-box structure similar to DES

S-box

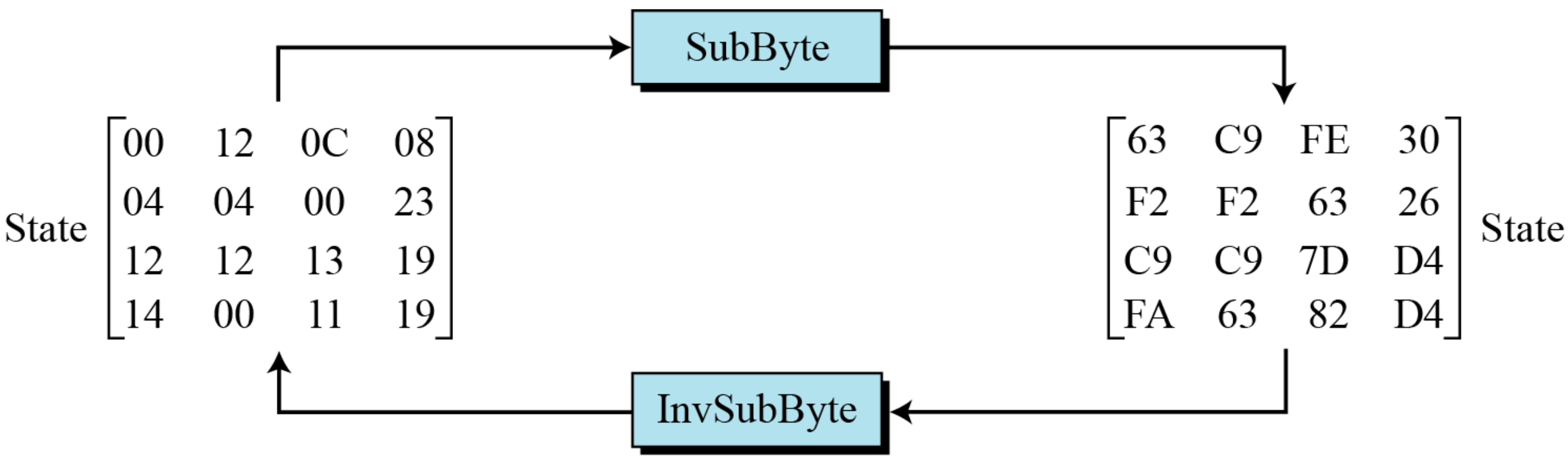
x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	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

Inv. S-box (Decryption)

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

Byte substitution (SubBytes)

The SubBytes and Inv SubBytes transformations are inverses of each other



2. Shift Rows

- A simple permutation
- Provide diffusion to the cipher
- 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.

- A circular byte shift in each each
  - 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

Shift Row

$$\begin{bmatrix} 87 & F2 & 4D & 97 \\ EC & 6E & 4C & 90 \\ 4A & C3 & 46 & E7 \\ 8C & D8 & 95 & A6 \end{bmatrix}$$

$$Before\ \ Shift$$

Shift Row

$$\begin{bmatrix} 87 & F2 & 4D & 97 \\ EC & 6E & 4C & 90 \\ 4A & C3 & 46 & E7 \\ 8C & D8 & 95 & A6 \end{bmatrix} \qquad \begin{bmatrix} 87 & F2 & 4D & 97 \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

$Before\ \ Shift$ $\qquad\qquad$ $After\ \ Shift$

Shift Row

$$\begin{bmatrix} 87 & F2 & 4D & 97 \\ EC & 6E & 4C & 90 \\ 4A & C3 & 46 & E7 \\ 8C & D8 & 95 & A6 \end{bmatrix} \qquad \begin{bmatrix} 87 & F2 & 4D & 97 \\ 6E & 4C & 90 & EC \\ & & & \\ & & & \end{bmatrix}$$

$$\textit{Before Shift} \qquad \qquad \textit{After Shift}$$

Shift Row

$$\begin{bmatrix} 87 & F2 & 4D & 97 \\ EC & 6E & 4C & 90 \\ 4A & C3 & 46 & E7 \\ 8C & D8 & 95 & A6 \end{bmatrix} \qquad \begin{bmatrix} 87 & F2 & 4D & 97 \\ 6E & 4C & 90 & EC \\ 46 & E7 & 4A & C3 \\ & & & \end{bmatrix}$$

Before Shift *After Shift*

Shift Row

$$\begin{bmatrix} 87 & F2 & 4D & 97 \\ EC & 6E & 4C & 90 \\ 4A & C3 & 46 & E7 \\ 8C & D8 & 95 & A6 \end{bmatrix} \qquad \begin{bmatrix} 87 & F2 & 4D & 97 \\ 6E & 4C & 90 & EC \\ 46 & E7 & 4A & C3 \\ A6 & 8C & D8 & 95 \end{bmatrix}$$

Before Shift *After Shift*

3. Mix Columns

- Mix Columns step provides diffusion to the cipher
- Each column is processed separately
- Each byte is replaced by a value dependent on all 4 bytes in the column

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \\ b_4 & b_8 & b_{12} & b_{16} \end{bmatrix}$$

Multiplication Matrix *State Matrix*

```
b1 = (b1 * 2) XOR (b2*3) XOR (b3*1) XOR (b4*1)
b2 = (b1 * 1) XOR (b2*2) XOR (b3*3) XOR (b4*1)
b3 = (b1 * 1) XOR (b2*1) XOR (b3*2) XOR (b4*3)
b4 = (b1 * 3) XOR (b2*1) XOR (b3*1) XOR (b4*2)
```

Galois Field Multiplication

- Multiplication of a value by 02 can be implemented as
  - Shift all the bits to the left by 1 position

Galois Field Multiplication

- Multiplication of a value by 02 can be implemented as
    - Shift all the bits to the left by 1 position
    - If bit B_7 is 0, then bit B_0 is 0

Galois Field Multiplication

- Multiplication of a value by 02 can be implemented as
  - Shift all the bits to the left by 1 position
  - If bit B_7 is 0, then bit B_0 is 0
  - If bit B_7 is 1, then bit B_0 is 0 and XOR with 1B.

- Multiplication of a value by 03 to N can be implemented as
- $N \oplus 02 . N$

For Ex: 02 * 87

$(87)_{16}$ = 1000 0111

For Ex: 02 * 87

$(87)_{16} = 1000\ 0111$

Shift left by 1 bit position and $B_0 = 0$

0000 1110

For Ex: 02 * 87

$(87)_{16}$ = 1000 0111

Shift left by 1 bit position and $B_0 = 0$

0000 1110

As B_7 bit (in original number) is 1   [1000 0111]

So XOR with 1B (0001 1011)

For Ex: 02 * 87

$(87)_{16}$ = 1000 0111

Shift left by 1 bit position and $B_0 = 0$

0000 1110

As B_7 bit (in original number) is 1

Now XOR this with 1B (0001 1011)

$$
\begin{array}{rl}
 & 0000\ 1110 \\
\oplus & 0001\ 1011 \\
\hline
 & 0001\ 0101\ [15]
\end{array}
$$

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

$6E = 0110\ \ 1110$

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

$6E = 0110 \quad 1110$

$\{02 * 6E\} = 0110 \quad 1110$

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

6E = 0110 1110

{02 * 6E} = 0110 1110

Shift left by 1 bit position and $B_0 = 0$

1101 1100

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

$6E = 0110 \quad 1110$

$\{02 * 6E\} = 0110\ 1110$

So, $\{02 * 6E\} = 1101 \quad 1100$

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

$6E \oplus 11011100$

03 . 6E can be written as

$6E \oplus \{02 * 6E\}$

$0110\ 1110 \oplus 11011100$

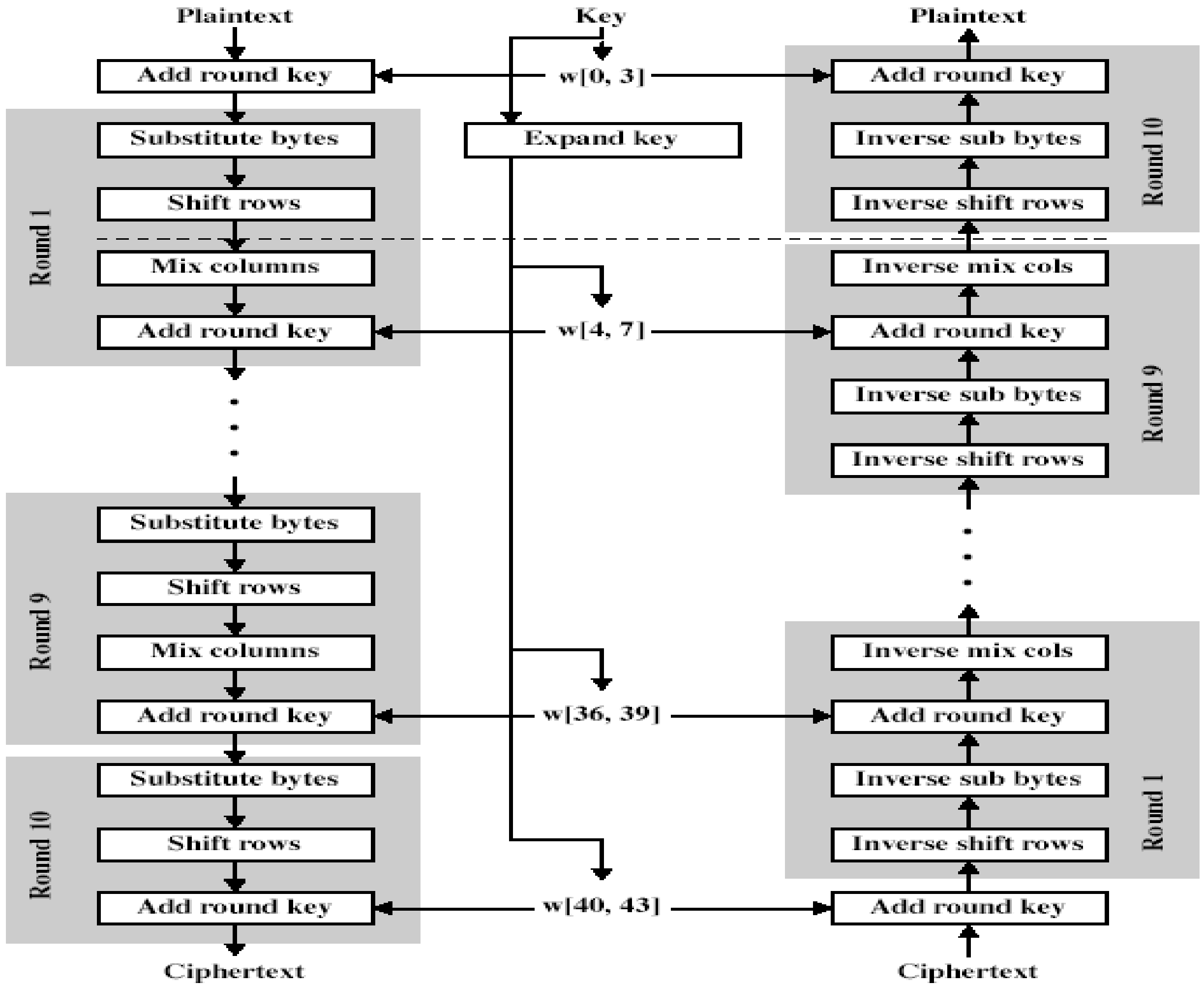
	0110	**1110**	**(6E)**
$\oplus$	**1101**	**1100**	**(02 * 6E)**
	1011	**0010**	

03 . 6E = 1011 0010 [B2]

4. Add Subkey (AddRoundKey)

- A portion of a key unique to this round is XOR with the round result.
- This operation provides confusion and incorporates the key

- An iteration of the above steps 1 to 4 is called a round.
- The amount of rounds of the algorithm depends on the key size.
- The only exception being that in the last round the Mix Column step is not performed, to make the algorithm reversible during decryption



(a) Encryption

(b) Decryption

Key Generation

- The encryption of AES is initiated by XOR operation between the plaintext block and a key.

- The encryption of AES is initiated by XOR operation between the plaintext block and a key.
- AES has total 10 rounds for a 128-bits key.

- The encryption of AES is initiated by XOR operation between the plaintext block and a key.
- AES has total 10 rounds for a 128-bits key.
- Total 10 subkeys are required, one for each round.

- The encryption of AES is initiated by XOR operation between the plaintext block and a key.
- AES has total 10 rounds for a 128-bits key.
- Total 10 subkeys are required, one for each round.
- These 10 subkeys are generated from the original key.

- One key has 128 bits or 16 bytes

- One key has 128 bits or 16 bytes
- These subkeys will never be reused.

- One key has 128 bits or 16 bytes
- These subkeys will never be reused.
- The logic of subkey generation is designed in such a way that, changes in one bit of a key affects the subkeys of the several rounds.

- One key has 128 bits or 16 bytes
- These subkeys will never be reused.
- The logic of subkey generation is designed in such a way that, changes in one bit of a key affects the subkeys of the several rounds.
- This provides confusion in the AES algorithm.

Steps for subkeys generation

Step 1

- The key for AES is 128 bits or 16 bytes
- This 16 bytes are arranged in the form of 4 x 4 matrix.

Steps for subkeys generation

Step 1

- The key for AES is 128 bits or 16 bytes
- This 16 bytes are arranged in the form of 4 x 4 matrix.
- So, the first column of a matrix is filled by first 4 bytes

Steps for subkeys generation

Step 1

- The key for AES is 128 bits or 16 bytes
- This 16 bytes are arranged in the form of 4 x 4 matrix.
- So, the first column of a matrix is filled by first 4 bytes
- The second column is filled by second 4 bytes

Steps for subkeys generation

Step 1

- The key for AES is 128 bits or 16 bytes
- This 16 bytes are arranged in the form of 4 x 4 matrix.
- So, the first column of a matrix is filled by first 4 bytes
- The second column is filled by second 4 bytes
- The third column filled by third 4 bytes and

Steps for subkeys generation

Step 1

- The key for AES is 128 bits or 16 bytes
- This 16 bytes are arranged in the form of 4 x 4 matrix.
- So, the first column of a matrix is filled by first 4 bytes
- The second column is filled by second 4 bytes
- The third column filled by third 4 bytes and
- The last column is filled by last 4 bytes of a key.

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

$$\begin{bmatrix} w_0 & w_1 & w_2 & w_3 \end{bmatrix}$$

Each column stands as one word of key “w”. Such as:

$w_0 = (b_0; b_1; b_2; b_3)$

$w_1 = (b_4; b_5; b_6; b_7)$

$w_2 = (b_8; b_9; b_{10}; b_{11})$

$w_3 = (b_{12}; b_{13}; b_{14}; b_{15})$

This is the key used for initial round.

Subkey for the next round is generated from this key.

Step 2:

Calculate $g[w_3]$ using following steps.

a) Perform circular left shift of the bytes of w_3 (fourth word of a key).

b) Perform substitution of the bytes using S-box.

c) Add round constant

S-Box for Key Generation

	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

Round Constant (RCon)

- RCon is a word in which the three rightmost bytes are zero
- It is different for each round and defined as:

  RCon[j] = (RCon[j],0,0,0)

  where RCon[1] =1 , RCon[j] = 2 * RCon[j-1]
- Multiplication is defined over GF(2^8) but can be implement in Lookup Table

confusion

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\underline{\mathbf{01}}\ 00\ 00\ 00)_{16}$	6	$(\underline{\mathbf{20}}\ 00\ 00\ 00)_{16}$
2	$(\underline{\mathbf{02}}\ 00\ 00\ 00)_{16}$	7	$(\underline{\mathbf{40}}\ 00\ 00\ 00)_{16}$
3	$(\underline{\mathbf{04}}\ 00\ 00\ 00)_{16}$	8	$(\underline{\mathbf{80}}\ 00\ 00\ 00)_{16}$
4	$(\underline{\mathbf{08}}\ 00\ 00\ 00)_{16}$	9	$(\underline{\mathbf{1B}}\ 00\ 00\ 00)_{16}$
5	$(\underline{\mathbf{10}}\ 00\ 00\ 00)_{16}$	10	$(\underline{\mathbf{36}}\ 00\ 00\ 00)_{16}$

Step 3

Generation of round key for first round.

$w_4 = w_0$ XOR $g(w_3)$

$w_4 = w_0$ XOR $g(w_3)$

$w_5 = w_1$ XOR w_4

$w_6 = w_2$ XOR w_5

$w_7 = w_3$ XOR w_6

Therefore, the round key for first round is $[w_4, w_5, w_6, w_7]$.

Subkey 2

$w_0 = w_4$

$w_1 = w_5$

$w_2 = w_6$

$w_3 = w_7$

Repeat steps 1 to 3 above to generate the key for all the keys

AES Example

Key in Hex (128 bits):

54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Plaintext in Hex (128 bits):

54 77 6F 20 4F 6E 65 20 4E 69 6E 65 20 54 77 6F

Roundkey Generation

Key in Hex (128 bits):

54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

$$\begin{bmatrix} 54 & 73 & 20 & 67 \\ 68 & 20 & 4B & 20 \\ 61 & 6D & 75 & 46 \\ 74 & 79 & 6E & 75 \end{bmatrix}$$

$$\begin{bmatrix} w_0 & w_1 & w_2 & w_3 \end{bmatrix}$$

Roundkey Generation

Key in Hex (128 bits):

54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

$w_0 = (54; 68; 61; 74);$

$w_1 = (73; 20; 6D; 79);$

$w_2 = (20; 4B; 75; 6E);$

$w_3 = (67; 20; 46; 75)$

w[3] = (67; 20; 46; 75)

$g(w_3)$:

circular byte left shift of w_3:

(20; 46; 75; 67)

Byte Substitution (S-Box):

(B7; 5A; 9D; 85)

Adding round constant

(01; 00; 00; 00) gives:

$g(w_3)$ = (B6; 5A; 9D; 85)

	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

S-Box

w_0	0101 0100	0110 1000	0110 0001	0111 0100
$\oplus\ g(w_3)$	1011 0110	0101 1010	1001 1101	1000 0101
w_4	1110 0010	0011 0010	1111 1100	1111 0001
	E2	32	FC	F1

$$w_4 = w_0 \oplus g(w_3) = (E2;\ 32;\ FC;\ F1)$$

$$w_5 = w_1 \oplus w_4 = (91;\ 12;\ 91;\ 88)$$

$$w_6 = w_2 \oplus w_5 = (B1;\ 59; E4; E6)$$

$$w_7 = w_3 \oplus w_6 = (D6;\ 79; A2;\ 93)$$

First roundkey:

E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

Round 0

State Matrix and Roundkey 0 Matrix

$$\begin{bmatrix} 54 & 4F & 4E & 20 \\ 77 & 6E & 69 & 54 \\ 6F & 65 & 6E & 77 \\ 20 & 20 & 65 & 6F \end{bmatrix} \oplus \begin{bmatrix} 54 & 73 & 20 & 67 \\ 68 & 20 & 4B & 20 \\ 61 & 6D & 75 & 46 \\ 74 & 79 & 6E & 75 \end{bmatrix} = \begin{bmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{bmatrix}$$

$State \quad Matrix \quad (Pla\,\mathrm{int}\,ext) \qquad\qquad Key$

$$\begin{array}{lllll} & 69 & 0110 & 1001 & \\ \oplus & 4B & \underline{0100} & \underline{1011} & \\ & & 0010 & 0010 & = 22 \end{array}$$

- The next State Matrix is

$$\begin{bmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{bmatrix}$$

- This Matrix uses as current State Matrix for next round

Round 1

1. Byte substitution (SubBytes)

- Substitute each entry of current state matrix by corresponding entry in AES S-Box
- byte 3C is substituted by entry of S-Box in row 3 and column C, i.e. by EB

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

This leads to new State Matrix

$$\begin{bmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{bmatrix} \Rightarrow \begin{bmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{bmatrix}$$

2. Shift Rows

$$\begin{bmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{bmatrix} \Rightarrow \begin{bmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{bmatrix}$$

3. Mix Column

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{bmatrix} = \begin{bmatrix} BA & 84 & E8 & 1B \\ 75 & A4 & 8D & 40 \\ F4 & 8D & 06 & 7D \\ 7A & 32 & 0E & 5D \end{bmatrix}$$

$$(02 \bullet 63) \oplus (03 \bullet 2F) \oplus (01 \bullet AF) \oplus (01 \bullet A2)$$

$$= \quad BA$$

$$(02 \bullet 63) \oplus (03 \bullet 2F) \oplus (01 \bullet AF) \oplus (01 \bullet A2)$$

2 x 63

63 => 0110 0011

Shift left-> 1100 0110

3 x 2F = 2F XOR (2 x 2F)

2 x 2F = 0010 1111

Shift Left => 0101 1110

XOR 2F => 0010 1111

----

3 x 2F 0 111 0001

2 x 63 = 1100 0110
3 x 2F = 0111 0001
1 x AF = AF = 1010 1111
1 x A2 = A2 = 1010 0010

$$
\begin{array}{rr}
 & 11000110 \\
 & 01110001 \\
\oplus & 10101111 \\
 & 10100010 \\
\hline
 & 10111010
\end{array}
$$

4. Add Subkey (AddRoundKey)

State Matrix and Roundkey No. 1 Matrix

$$\begin{bmatrix} BA & 84 & E8 & 1B \\ 75 & A4 & 8D & 40 \\ F4 & 8D & 06 & 7D \\ 7A & 32 & 0E & 5D \end{bmatrix} \oplus \begin{bmatrix} E2 & 91 & B1 & D6 \\ 32 & 12 & 59 & 79 \\ FC & 91 & E4 & A2 \\ F1 & 88 & E6 & 93 \end{bmatrix} = \begin{bmatrix} 58 & 15 & 59 & CD \\ 47 & B6 & D4 & 39 \\ 08 & 1C & E2 & DF \\ 8B & BA & E8 & CE \end{bmatrix}$$

Output after round 1

58 47 08 8B 15 B6 1C BA 59 D4 E2 E8 CD 39 DF CE

Comparison of AES with DES

	AES	DES
Block size (in bits)	128	64
Key size (in bits)	128, 192, 256	56
Speed	High	Low
Encryption primitives	Substitution, shift, bit mixing	Substitution, permutation
Cryptographic primitives	Confusion, Diffusion	Confusion, Diffusion

Diffusion

- The statistical structure of the plaintext is dissipated into long range statistics of the ciphertext
- Each plaintext digit affect the value of many ciphertext digits

Confusion

- Confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible