

# Lecture 5

## AES Standard

R. Rhouma

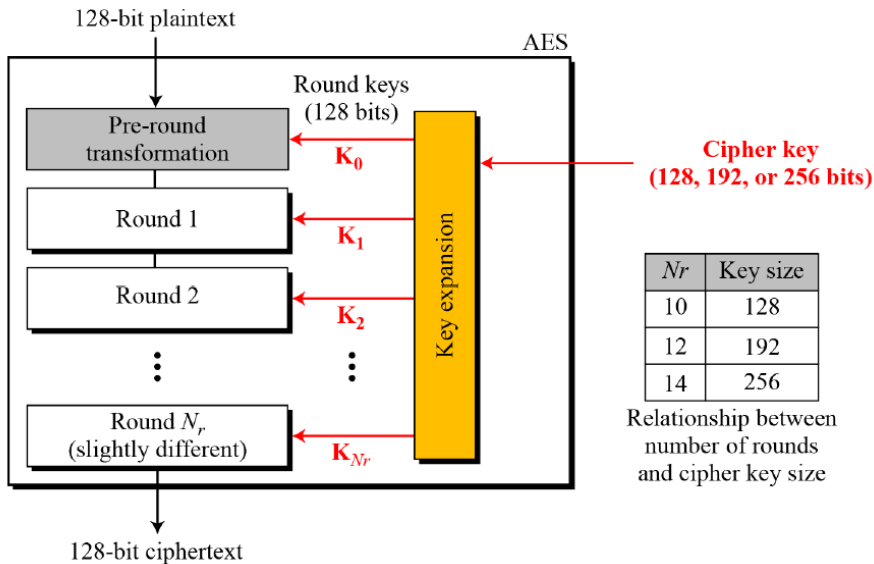
UTAS  
Sultanate of Oman  
September 2022

CSSY2201 : Introduction to Cryptography

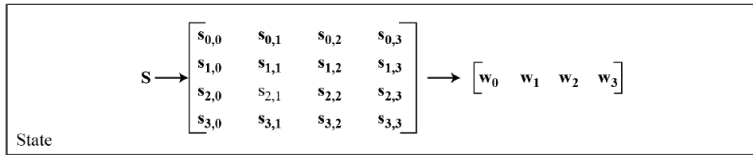
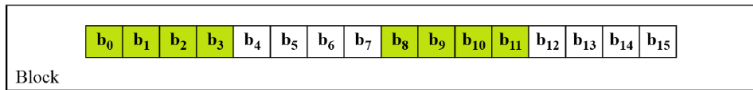
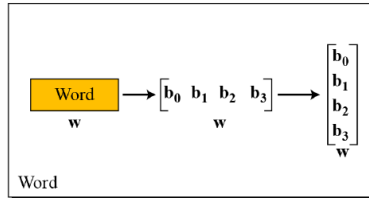
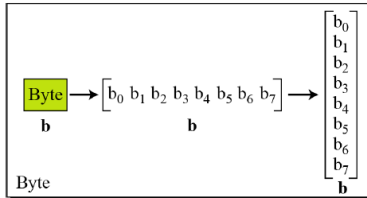
- 1 General Overview
- 2 AES Transformations
  - Substitution
  - Permutation
  - Mixing
  - Key Adding
- 3 KEY EXPANSION
- 4 Summary

- In February 2001, NIST announced that a draft of the Federal Information Processing Standard (FIPS) was available for public review and comment. Finally, AES was published as FIPS 197 in the Federal Register in December 2001.
- AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits. It uses 10, 12, or 14 rounds. The key size, which can be 128, 192, or 256 bits, depends on the number of rounds.
- AES has defined three versions, with 10, 12, and 14 rounds. Each version uses a different cipher key size (128, 192, or 256).
- Round keys are always 128 bits.

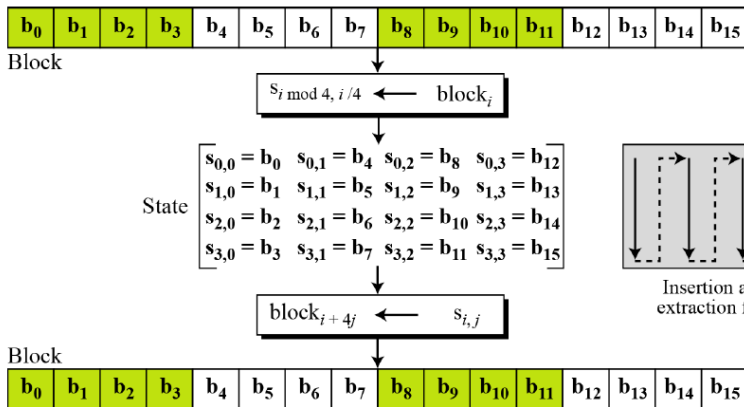
# General design of AES encryption cipher



# Data units used in AES



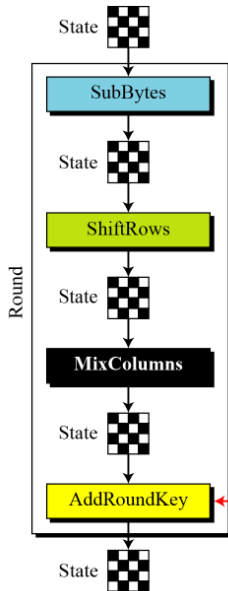
# Block-to-state and state-to-block transformation



# Example : Changing plaintext to state

Text	A E S U S E S A M A T R I X Z Z															
Hexadecimal	00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19															
	<div><div><div>00120C08</div><div>04040023</div><div>12121319</div><div>14001119</div></div><div>State</div></div>															

# Structure of Each Round



Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.



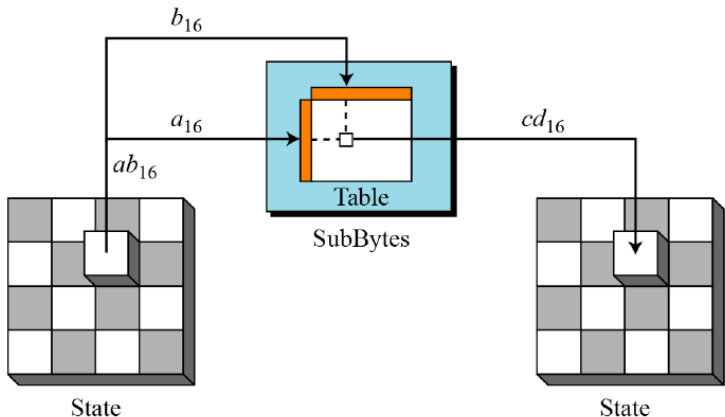
# TRANSFORMATIONS

To provide security, AES uses four types of transformations : substitution, permutation, mixing, and key-adding.

- Substitution : SubBytes and InvSubBytes
- Permutation :
- Mixing
- Key Adding

# Substitution : SubBytes

- SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.
- The SubBytes operation involves 16 independent byte-to-byte transformations.



# SubByte Transformation Table : 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

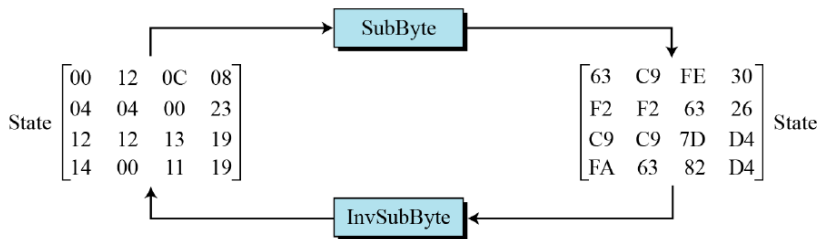
# InvSubBytes Transformation Table : Inverse S-Box

		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	83	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



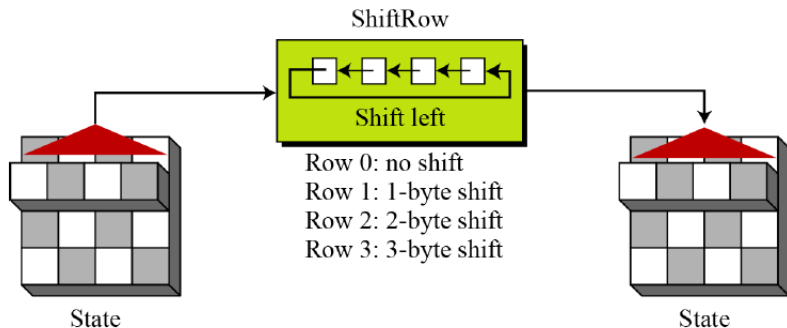
# Example using SubByte and InvSubByte

the figure shows how a state is transformed using the SubBytes transformation. The figure also shows that the InvSubBytes transformation creates the original one.

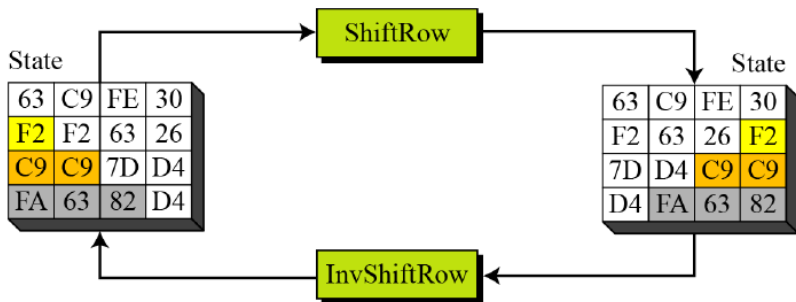


# Permutation : ShiftRows and InvShiftRows

- ShiftRows permutes the bytes.
- In the encryption, the transformation is called ShiftRows (Shift to the left)
- in the decryption, it is called InvShiftRows (Shift to the right)



the Figure shows how a state is transformed using ShiftRows transformation. The figure also shows that InvShiftRows transformation creates the original state.



# Mixing

We need an interbyte transformation that changes the bits inside a byte, based on the bits inside the neighboring bytes. We need to mix bytes to provide diffusion at the bit level.

$$\begin{array}{l}
 ax + by + cz + dt \\
 ex + fy + gz + ht \\
 ix + jy + kz + lt \\
 mx + ny + oz + pt
 \end{array}
 \begin{array}{c}
 \rightarrow \\
 \rightarrow \\
 \rightarrow \\
 \rightarrow
 \end{array}
 \left[ \begin{array}{c} \text{Green Box} \\ \text{Green Box} \\ \text{Green Box} \\ \text{Green Box} \end{array} \right]
 =
 \begin{bmatrix}
 a & b & c & d \\
 e & f & g & h \\
 i & j & k & l \\
 m & n & o & p
 \end{bmatrix}
 \times
 \begin{bmatrix}
 \mathbf{x} \\
 \mathbf{y} \\
 \mathbf{z} \\
 \mathbf{t}
 \end{bmatrix}$$

New matrix
**Constant matrix**
Old matrix

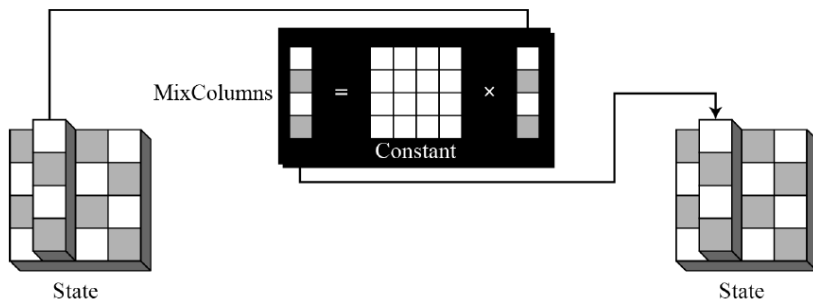


# Constant matrices used by MixColumns and InvMixColumns

$$\begin{array}{c} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \xleftrightarrow{\text{Inverse}} \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \\ C \qquad \qquad \qquad C^{-1} \end{array}$$

# MixColumns

The MixColumns transformation operates at the column level ; it transforms each column of the state to a new column.

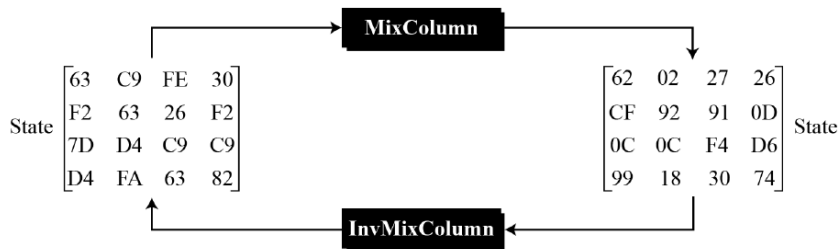


The InvMixColumns transformation is basically the same as the MixColumns transformation.

### Note

The MixColumns and InvMixColumns transformations are inverses of each other.

the Figure shows how a state is transformed using the MixColumns transformation. The figure also shows that the InvMixColumns transformation creates the original one.

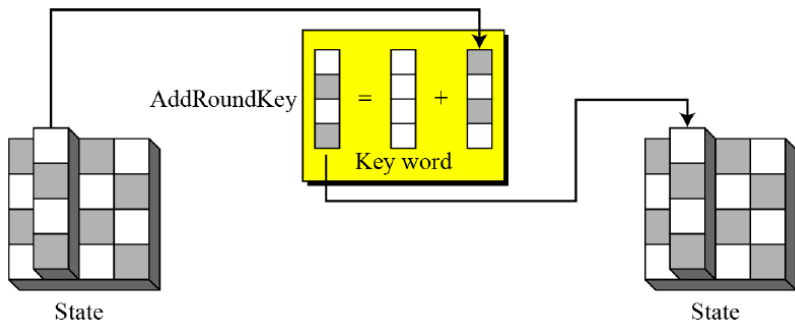


# Key Adding

AddRoundKey proceeds one column at a time. AddRoundKey adds a round key word with each state column matrix ; the operation in AddRoundKey is matrix addition.

## Note

The AddRoundKey transformation is the inverse of itself.



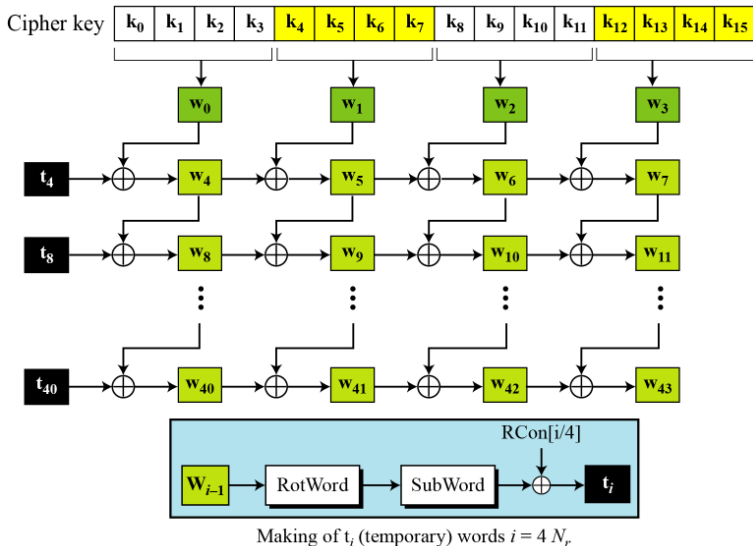
# KEY EXPANSION

To create round keys for each round, AES uses a keyexpansion process. If the number of rounds is  $N_r$ , the key-expansion routine creates  $N_r + 1$  128-bit round keys from one single 128-bit cipher key.

Figure – words for each round

<i>Round</i>	<i>Words</i>			
Pre-round	$w_0$	$w_1$	$w_2$	$w_3$
1	$w_4$	$w_5$	$w_6$	$w_7$
2	$w_8$	$w_9$	$w_{10}$	$w_{11}$
...	...			
$N_r$	$w_{4N_r}$	$w_{4N_r+1}$	$w_{4N_r+2}$	$w_{4N_r+3}$

# Key Expansion in AES-128



# the Rcon constants

<i>Round</i>	<i>Constant (RCon)</i>	<i>Round</i>	<i>Constant (RCon)</i>
1	( <u>01</u> 00 00 00) <sub>16</sub>	6	( <u>20</u> 00 00 00) <sub>16</sub>
2	( <u>02</u> 00 00 00) <sub>16</sub>	7	( <u>40</u> 00 00 00) <sub>16</sub>
3	( <u>04</u> 00 00 00) <sub>16</sub>	8	( <u>80</u> 00 00 00) <sub>16</sub>
4	( <u>08</u> 00 00 00) <sub>16</sub>	9	( <u>1B</u> 00 00 00) <sub>16</sub>
5	( <u>10</u> 00 00 00) <sub>16</sub>	10	( <u>36</u> 00 00 00) <sub>16</sub>



The Table shows how the keys for each round are calculated assuming that the 128-bit cipher key agreed upon by Alice and Bob is  $(2475A2B33475568831E2120013AA5487)_{16}$ .

Round	Values of $r$ 's	First word in the round	Second word in the round	Third word in the round	Fourth word in the round
—		$w_{00} = 2475A2B3$	$w_{01} = 34755688$	$w_{02} = 31E21200$	$w_{03} = 13AA5487$
1	AD20177D	$w_{04} = 8955B5CE$	$w_{05} = BD20E346$	$w_{06} = 8CC2F146$	$w_{07} = 9F68A5C1$
2	470678DB	$w_{08} = CE53CD15$	$w_{09} = 73732E53$	$w_{10} = FFB1DF15$	$w_{11} = 60D97AD4$
3	31DA48D0	$w_{12} = FF8985C5$	$w_{13} = 8CFAAB96$	$w_{14} = 734B7483$	$w_{15} = 2475A2B3$
4	47AB5B7D	$w_{16} = B822deb8$	$w_{17} = 34D8752E$	$w_{18} = 479301AD$	$w_{19} = 54010FFA$
5	6C762D20	$w_{20} = D454F398$	$w_{21} = E08C86B6$	$w_{22} = A71F871B$	$w_{23} = F31E88E1$
6	52C4F80D	$w_{24} = 86900B95$	$w_{25} = 661C8D23$	$w_{26} = C1030A38$	$w_{27} = 321D82D9$
7	E4133523	$w_{28} = 62833EB6$	$w_{29} = 049FB395$	$w_{30} = C59CB9AD$	$w_{31} = F7813B74$
8	8CE29268	$w_{32} = EE61ACDE$	$w_{33} = EAFE1F4B$	$w_{34} = 2F62A6E6$	$w_{35} = D8E39D92$
9	0A5E4F61	$w_{36} = E43FE3BF$	$w_{37} = 0EC1FCF4$	$w_{38} = 21A35A12$	$w_{39} = F940C780$
10	3FC6CD99	$w_{40} = DBF92E26$	$w_{41} = D538D2D2$	$w_{42} = F49B88C0$	$w_{43} = 0DDB4F40$

Each round key in AES depends on the previous round key. The dependency, however, is nonlinear because of SubWord transformation. The addition of the round constants also guarantees that each round key will be different from the previous one. Example : The two sets of round keys can be created from two cipher keys that are different only in one bit.

Cipher Key 1: 12 45 A2 A1 23 31 A4 A3 B2 CC AA 34 C2 BB 77 23  
 Cipher Key 2: 12 45 A2 A1 23 31 A4 A3 B2 CC AB 34 C2 BB 77 23

# Comparing two sets of round keys

R.	Round keys for set 1	Round keys for set 2	B. D.
—	1245A2A1 2331A4A3 B2CCA <u>A</u> 34 C2BB7723	1245A2A1 2331A4A3 B2CCAB <u>B</u> 34 C2BB7723	01
1	F9B08484 DA812027 684D8 <u>A</u> 13 AAF6F <u>D</u> 30	F9B08484 DA812027 684D8 <u>B</u> 13 AAF6F <u>C</u> 30	02
2	B9E48028 6365A00F 0B282A1C A1DED72C	B9008028 6381A00F 0BCC2B1C A13AD72C	17
3	A0EAF11A C38F5115 C8A77B09 6979AC25	3D0EF11A 5E8F5115 55437A09 F479AD25	30
4	1E7BCEE3 DDF49FF6 1553E4FF 7C2A48DA	839BCEA5 DD149FB0 8857E5B9 7C2E489C	31
5	EB2999F3 36DD0605 238EE2FA 5FA4AA20	A2C910B5 7FDD8F05 F78A6ABC 8BA42220	34
6	82852E3C B4582839 97D6CAC3 C87260E3	CB5AA788 B487288D 430D4231 C8A96011	56
7	82553FD4 360D17ED A1DBDD2E 69A9BDCD	588A2560 EC0D0DED AF004FDC 67A92FCD	50
8	D12F822D E72295C0 46F948EE 2F50F523	0B9F98E5 E7929508 4892DAD4 2F3BF519	44
9	99C9A438 7EEB31F8 38127916 17428C35	F2794CF0 15EBD9F8 5D79032C 7242F635	51
10	83AD32C8 FD460330 C5547A26 D216F613	E83BDAB0 FDD00348 A0A90064 D2EBF651	52

The concept of weak keys, does not apply to AES. Assume that all bits in the cipher key are 0s. The following shows the words for some rounds :

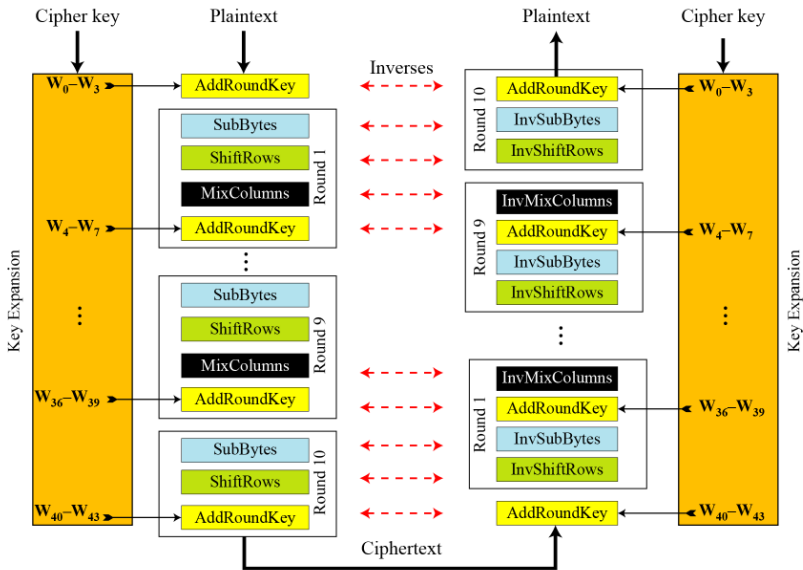
Pre-round:	00000000	00000000	00000000	00000000
Round 01:	62636363	62636363	62636363	62636363
Round 02:	9B9898C9	F9FBFBAA	9B9898C9	F9FBFBAA
Round 03:	90973450	696CCFFA	F2F45733	0B0FAC99
...	...	...	...	...
Round 10:	B4EF5BCB	3E92E211	23E951CF	6F8F188E

The words in the pre-round and the first round are all the same. In the second round, the first word matches with the third ; the second word matches with the fourth. However, after the second round the pattern disappears ; every word is different.

# The AES Cipher : Summary

AES uses four types of transformations for encryption and decryption. In the standard, the encryption algorithm is referred to as the cipher and the decryption algorithm as the inverse cipher.

# AES cipher and inverse cipher



# AES pseudocode

```

Cipher (InBlock [16], OutBlock[16],  $w[0 \dots 43]$ )
{
    BlockToState (InBlock, S)

     $S \leftarrow \text{AddRoundKey} (S, w[0 \dots 3])$ 
    for (round = 1 to 10)
    {
         $S \leftarrow \text{SubBytes} (S)$ 
         $S \leftarrow \text{ShiftRows} (S)$ 
        if (round  $\neq$  10)  $S \leftarrow \text{MixColumns} (S)$ 
         $S \leftarrow \text{AddRoundKey} (S, w[4 \times \text{round}, 4 \times \text{round} + 3])$ 
    }

    StateToBlock (S, OutBlock);
}

```

# Security of AES

- AES was designed after DES. Most of the known attacks on DES were already tested on AES.
- Brute-Force Attack : AES is definitely more secure than DES due to the larger-size key.
- Statistical Attacks : Numerous tests have failed to do statistical analysis of the ciphertext.
- Differential and Linear Attacks : There are no differential and linear attacks on AES as yet.