Unit II. Symmetric Cryptography
Advanced Encryption Standard

Presented by: Er. Kobid Karkee
Himalaya College of Engineering

Advanced Encryption Standard (AES)

- AES is the most widely used symmetric cipher today.
  - Internet security standard IPsec, TLS, the Wi-Fi encryption standard IEEE 802.11i, the secure shell network protocol SSH (Secure Shell), the Internet phone Skype etc. use AES.

The AES selection procedure:

- On Sep. 12, 1997, NIST called for proposals for a new Advanced Encryption Standard (AES).
- Fifteen candidate algorithms were submitted by researchers from several countries by August 20, 1998.
- On August 9, 1999, five finalist algorithms were selected: -
  - Mars by IBM Corporation
  - RC6 by RSA Laboratories
  - Rijndael, by Joan Daemen and Vincent Rijmen
  - Serpent, by Ross Anderson, Eli Biham and Lars Knudsen
  - Twofish, by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, ChrisHall and Niels Fergus

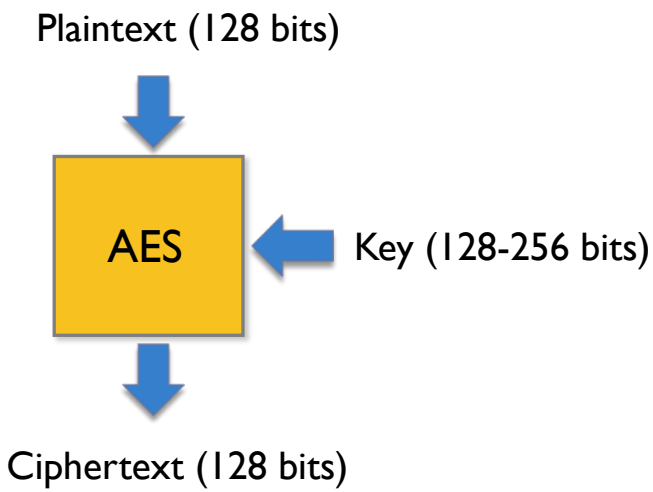
Advanced Encryption Standard (AES)

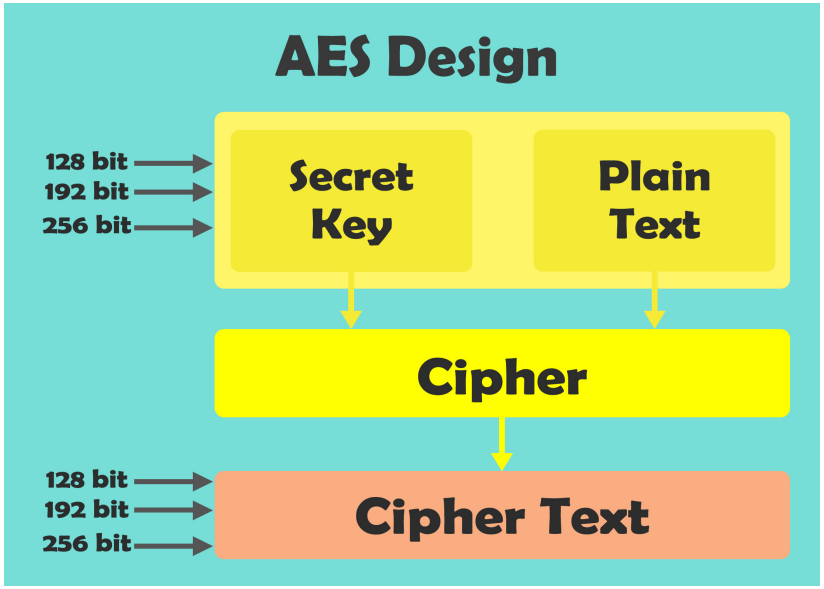
- On October 2, 2000, NIST announced that it had chosen Rijndael as the AES.
- On November 26, 2001, AES was formally approved as a US federal standard.
- In 2003, the US National Security Agency (NSA) announced that it allows AES to encrypt classified documents up to the level SECRET for all key lengths, and up to the TOP SECRET level for key lengths of either 192 or 256 bits.
- Before, only non-public algorithms had been used for the encryption of classified document.

AES:Rijndael

- It is not a Feistel cipher.
  - It works in parallel over the whole input block.
- Designed to be efficient both in hardware and software across a variety of platforms.
- It's a block cipher which works iteratively.
  - Block size: 128 bit (but can also 192 or 256 bit)
  - Key length: 128, 192 or 256 bit
  - Number of rounds: 10, 12 or 14
  - Key scheduling: 44, 52 or 60 subkeys having length = 32 bit

Advanced Encryption Standard (AES)





	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Note: 1 word = 32 bits

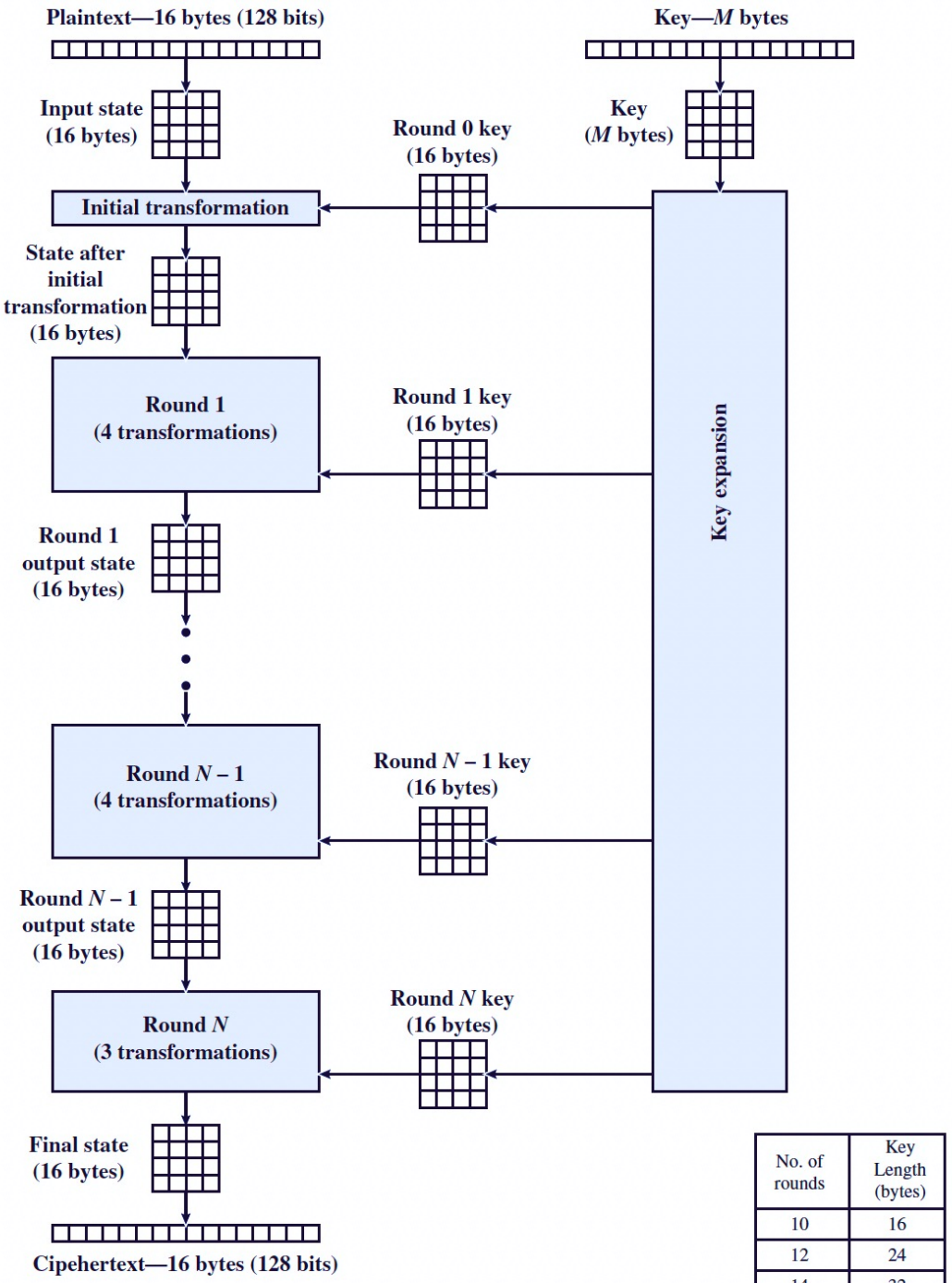
For AES-128:

Keylength N_k= 4 words = 4X32 = 128 bits

Block size N_B = 4 words = 4X32 = 128 bits

Advanced Encryption Standard (AES)

- Each round (except the last one) is a uniform and parallel composition of 4 steps
  - SubBytes (byte-by-byte substitution using an S-box)
  - ShiftRows (a permutation, which cyclically shifts the last three rows in the State)
  - MixColumns (substitution that uses Galois Fields, corps de Galois, GF(2^8) arithmetic)
  - AddRound key (bit-by- bit XOR with an expanded key)
- And the final/last round consists of:
  - SubBytes
    ShiftRows
    AddRound key



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

Figure 5.1 AES Encryption Process

AES Keys

- With 128 bit: 2^{128} = 3.4×10^{38} possible keys
  - A PC that tries 2^{55} keys per second needs 149.000 billion years to break AES
- Con 192 bit: 2^{192} = 6.2×10^{57} possible keys
- .....
- Con 256 bit: 2^{256} = 1.1×10^{77} possible keys
- .....
  - Probably AES will stay secure for at least 15-20 years

Key and Block size

- Key with variable length (128,192, 256 bit)
  - Represented with a matrix (*array*) of bytes with 4 rows and Nk columns, Nk=key length / 32
    - key of 128 bits= 16 bytes → Nk=4
    - key of 192 bits= 24 bytes → Nk=6
    - key of 256 bits= 32 bytes → Nk=8

$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,3}$
$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$
$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$
$K_{3,0}$	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$

- Block of length 128 bits=16 bytes
  - Represented with a matrix (*array*) of bytes with 4 rows and Nb columns, Nb=block length / 32
    - Block of 128 bits= 16 bytes → Nb=4

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

in=input

AES: State

- Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the State
  - 4 rows, each containing Nb bytes
  - Nb columns, constituted by 32-bit words
  - $S_{r,c}$ denotes the byte in row r and column c

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

- The array of bytes in input is copied in the State matrix

$$S_{r,c} \leftarrow in$$

- At the end, the State matrix is copied in the output matrix

$$out \leftarrow S_{r,c}$$

Rijndael Design

- Operations performed on State (4 rows of bytes).
- The 128 bit key is expanded as an array of 44 entries of 32 bits words; 4 distinct words serve as a round key for each round; key schedule relies on the S-box
- Algorithms composed of three layers
  - Linear Diffusion
  - Non-linear Diffusion
  - Key Mixing

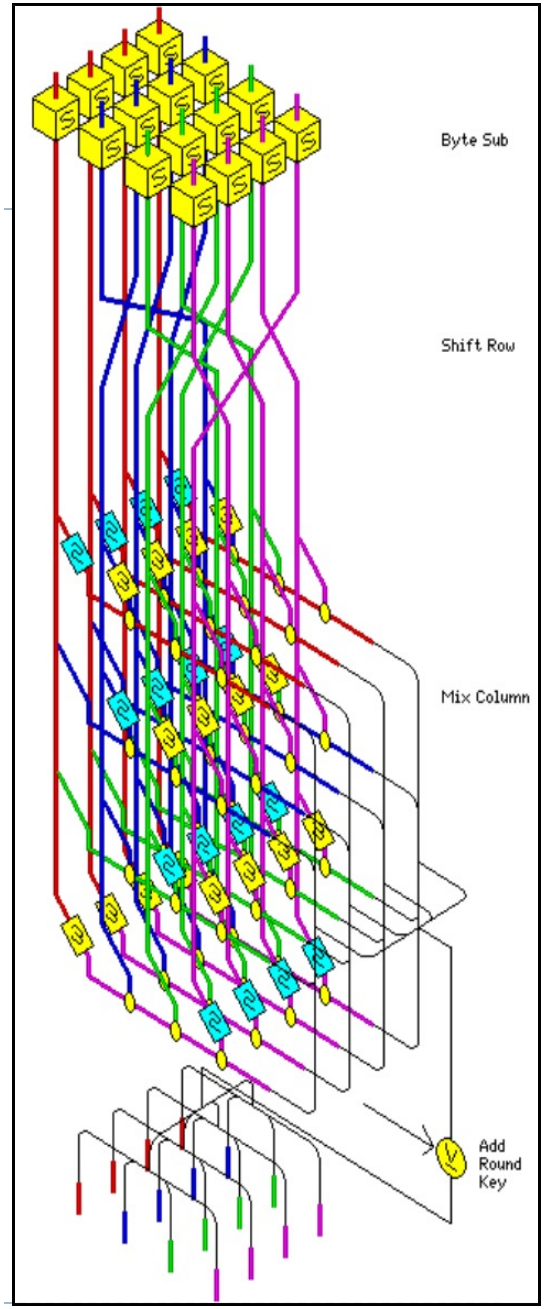
Rijndael: High Level Description

State = X

1. AddRoundKey(State, Key_0)
2. for r = 1 to (Nr - 1)
   a. SubBytes(State, S-box)
   b. ShiftRows(State)
   c. MixColumns(State)
   d. AddRoundKey(State, Key_r)

   end for
3. SubBytes(State, S-box)
4. ShiftRows(State)
5. AddRoundKey(State, Key_{Nr})

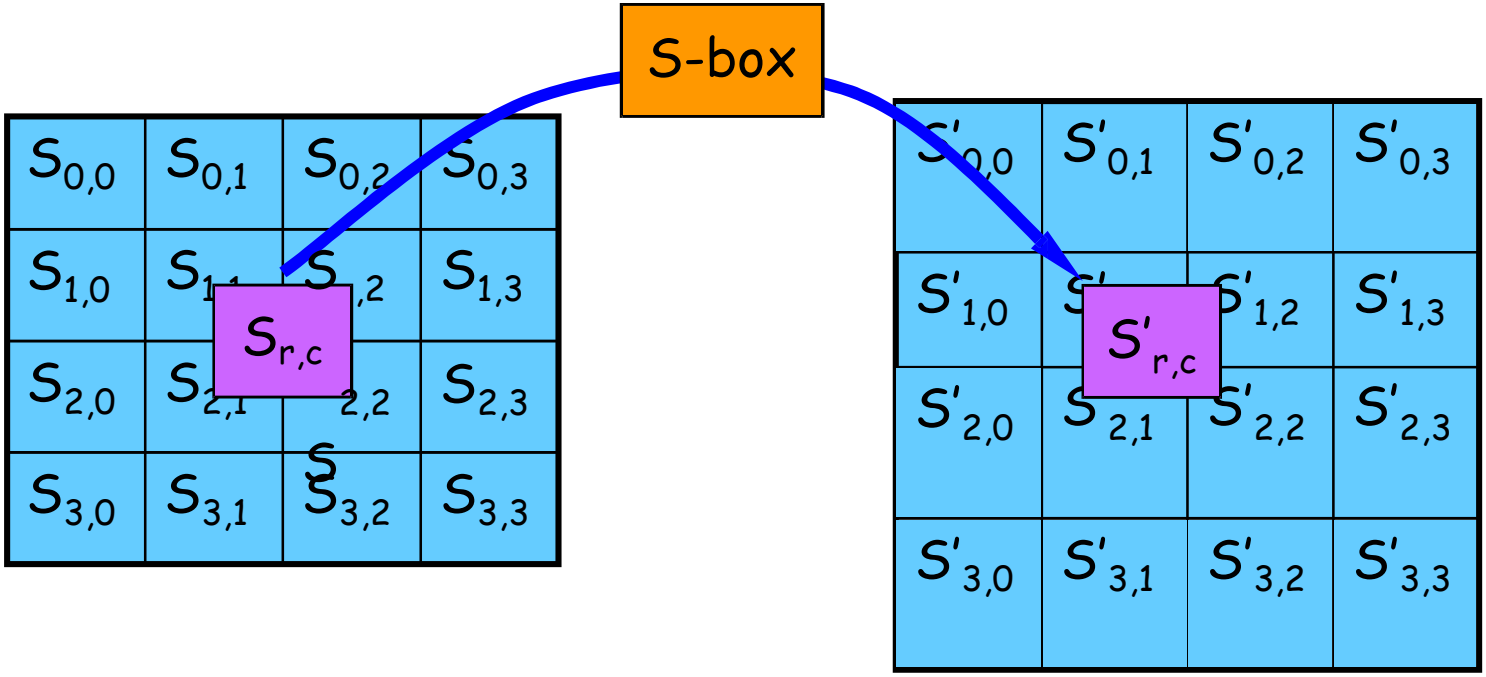
Y = State



SubBytes Transformation

- Bytes are transformed using a *non-linear S-box*

$$S'_{r,c} \leftarrow \text{S-box}(S_{r,c})$$



SubBytes

- Byte substitution using a non-linear (but *invertible*) S-Box (independently on each byte).
- S-box is represented as a 16x16 array, rows and columns indexed by hexadecimal bits
- 8 bytes replaced as follows: 8 bytes define a hexadecimal number **rc**, then $s'_{r,c}$ = binary(S-box(**r, c**))
- How is AES S-box different from DES S-boxes?
  - Only <u>one</u> S-box
  - S-boxes based on modular arithmetic with polynomials, can be defined algebraically
  - Easy to analyze, prove attacks fail

Rijandael S-box Table

	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	3	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

Example: hexa **53** is replaced with hexa **ED**

(The first 4 bits in the byte(the first hexadecimal value, hence) individuate the row, the last 4 bits individuate the column)

SubBytes Transformation

Rationale for S-box:

- The S-box is designed to be resistant to known cryptanalytic attacks.
- The design that has a low correlation between input bits and output bits and the property that the output is not a linear mathematical function of the input.
- The nonlinearity is due to the use of the multiplicative inverse.
- Example of a SubByte transformation:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

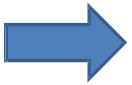
→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

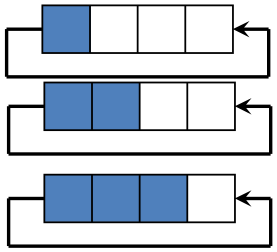
ShiftRows

- Circular Left Shift of a number of bytes equal to the row number. (1st row =0 shifts, 2nd row: 2 byte shifts,……)

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$



$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$



Rationale:

- Transformation ensures that the 4 bytes of one column are spread out to four different columns
- Example of ShiftRows:

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

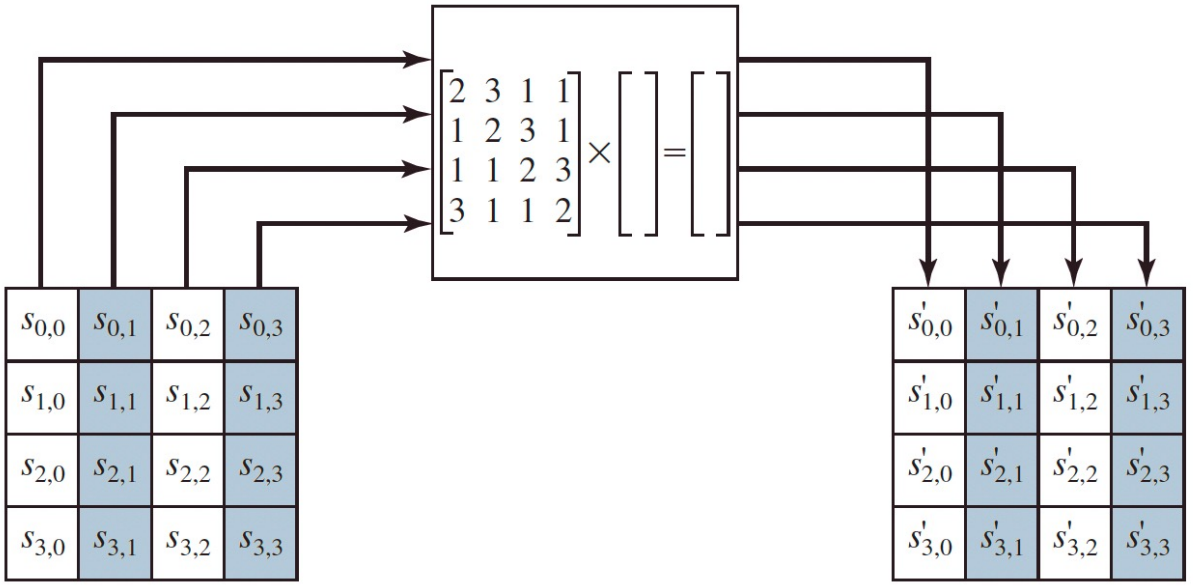
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

MixColumns

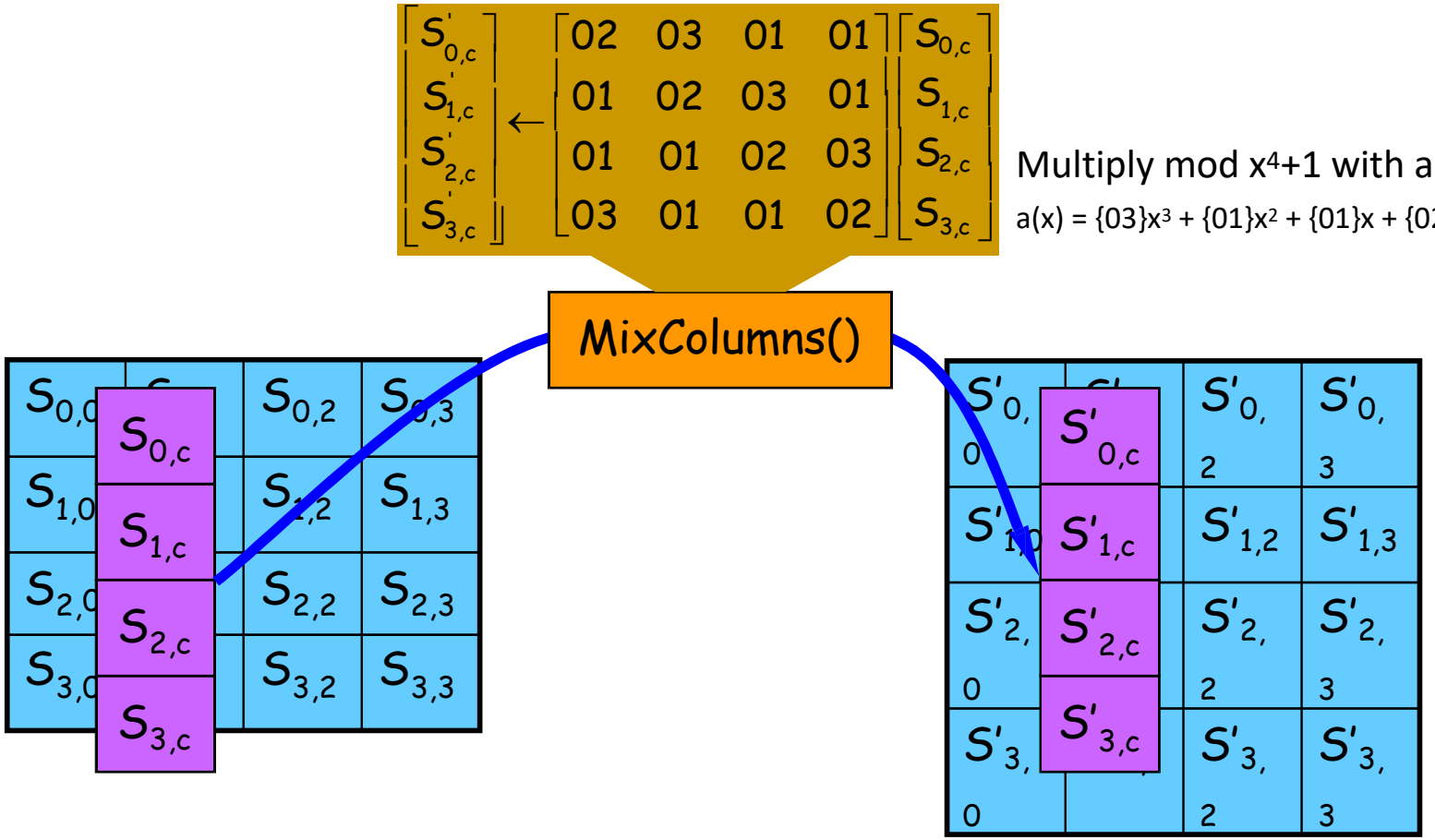
- Interpret each column as a vector of length 4.
- Each column of State is replaced by another column obtained by multiplying that column with a matrix in a particular field (Galois Field).

Rationale:

- The mix column transformation combined with the shift row transformation ensures that after a few rounds all output bits depend on all input bits



MixColumns Transformation



Bytes in columns are combined linearly

AddRound Key

- The 128 bits of State are bitwise XORed with the 128 bits of the round key
- Operation is viewed as a column-wise operation between the 4 bytes of a State column and one word of the round key

Rationale:

- Simple as possible and affecting every bit
- The complexity of the round key expansion plus the complexity of the other stages of AES ensure security

AddRound Key

- **State** is represented as follows (16 bytes):

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

- **AddRoundKey(State, Key):**
  - Example

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

State

$\oplus$

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

Round Key

$=$

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

Next State

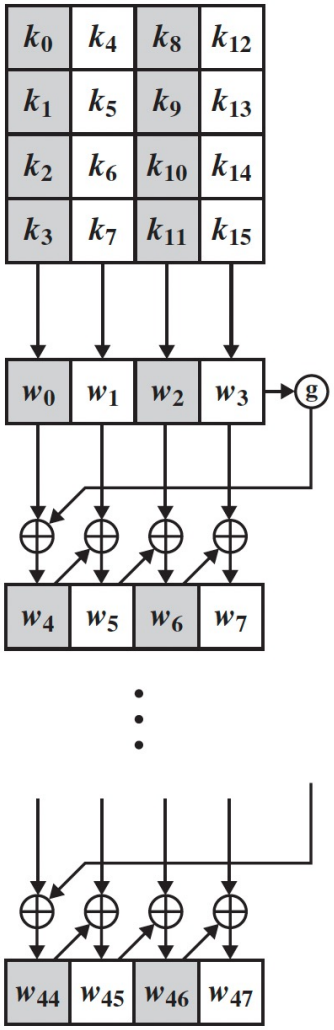
AES Key Expansion

- Takes as input a 4-word (16 byte) key and produces a linear array of 44 words (176) bytes.
  - Sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.
- Key is copied into the first four words of the expanded key.
  - The remainder of the expanded key is filled in four words at a time.

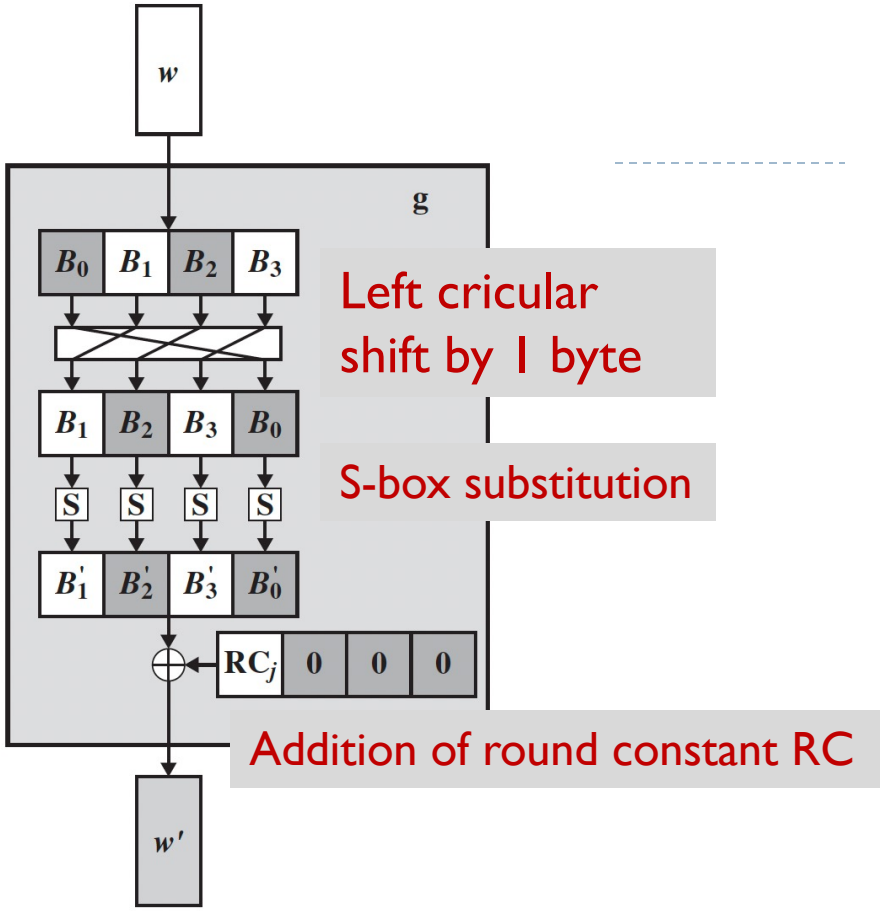
Rationale:

- The expansion key algorithm is designed to be resistant to known cryptanalytic attacks
- The specific criteria that were used are:
  - Knowledge of a part of the round key does not enable calculation of many other round-key bits
  - An invertible transformation
  - Uses round constants to eliminate symmetries
  - Diffusion of round key differences into the round keys
  - Simplicity of description

AES Key Expansion



(a) Overall algorithm



(b) Function g

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Figure 5.9 AES Key Expansion

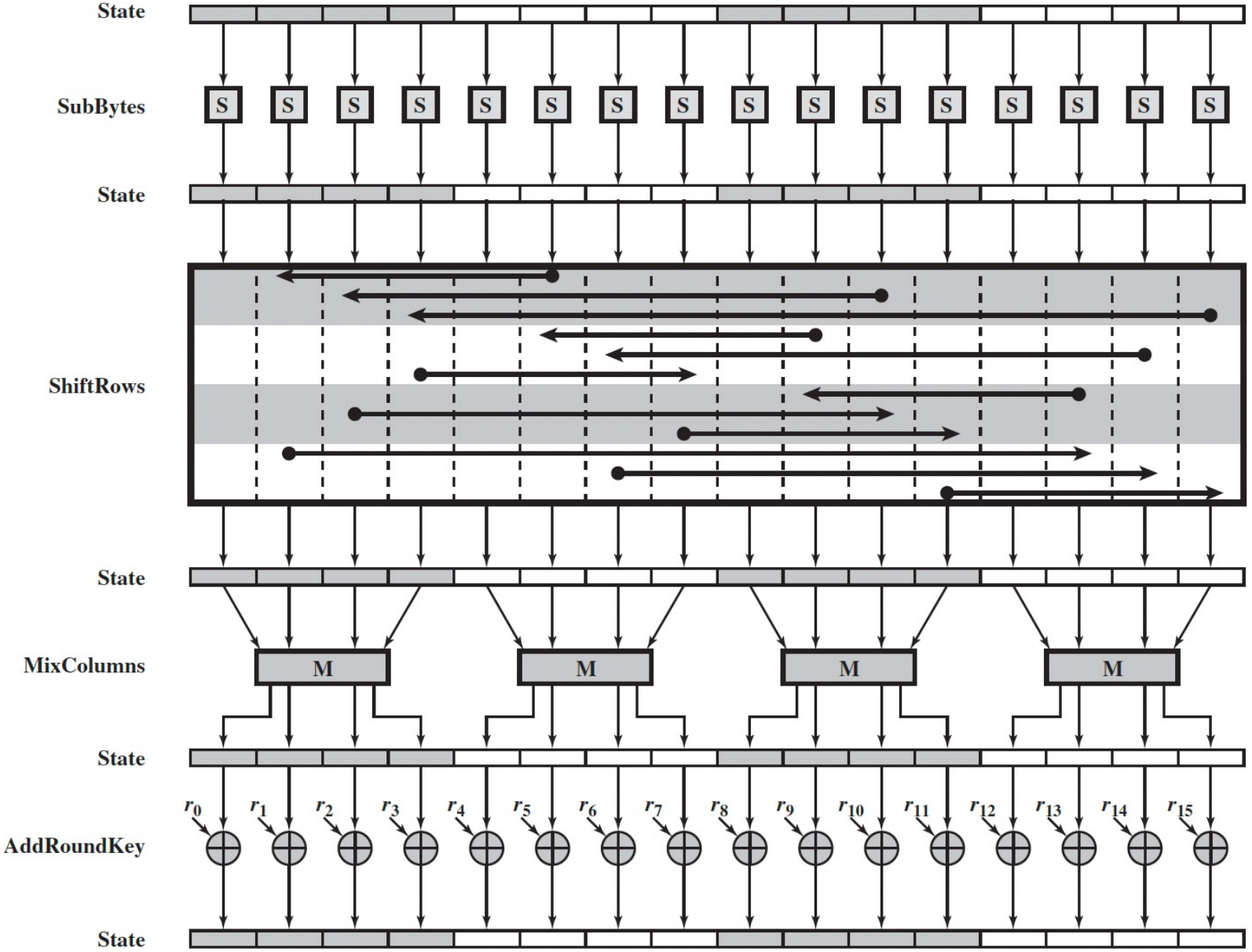


Figure 5.4 AES Encryption Round

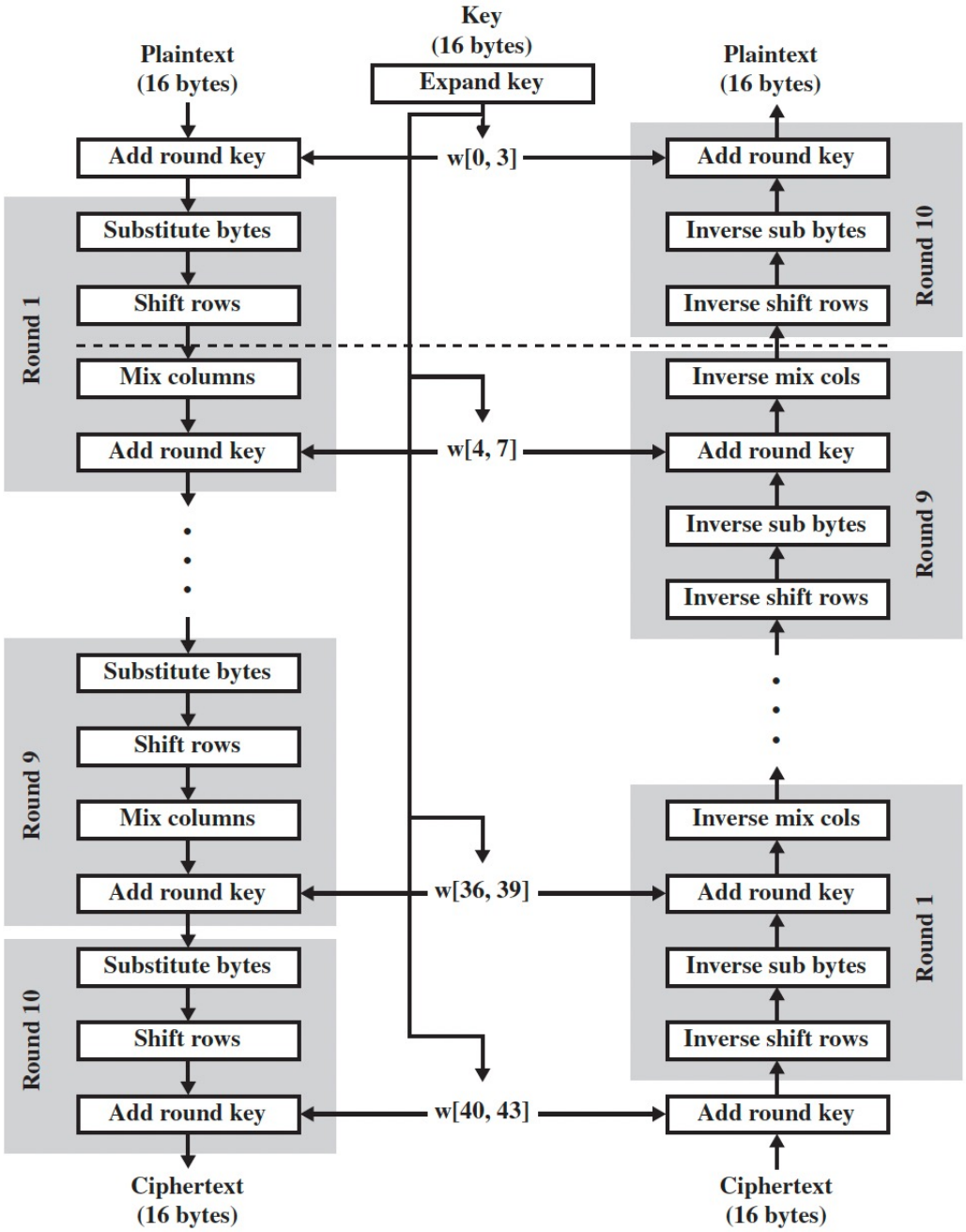


Figure 5.3 AES Encryption and Decryption

AES Decryption

- The decryption algorithm makes use of the expanded key in reverse order; however, the decryption algorithm is not identical to the encryption algorithm.
- State is the same for both encryption and decryption.
- Final round of both encryption and decryption consists of only three stages.

Equivalent Inverse

- AES decryption cipher is not identical to the encryption cipher
  - The sequence of transformations differs although the form of the key schedules is the same
  - Has the disadvantage that two separate software or firmware modules are needed for applications that require both encryption and decryption
- Two separate changes are needed to bring the decryption structure in line with the encryption structure (refer slide 25):
  - The first two stages of the decryption round need to be interchanged
  - The second two stages of the decryption round need to be interchanged

Rijndael Cryptanalysis

- Resistant to linear and differential cryptanalysis
  - Academic break on weaker version of the cipher, 9 rounds.
  - Requires 2^{224} work and 2^{85} chosen related-key plaintexts.
  - Attack not practical.