AES (Advanced Encryption Standard)

암호이론 및 정보보안 실무

서울과학기술대학교
발표자: 김인영

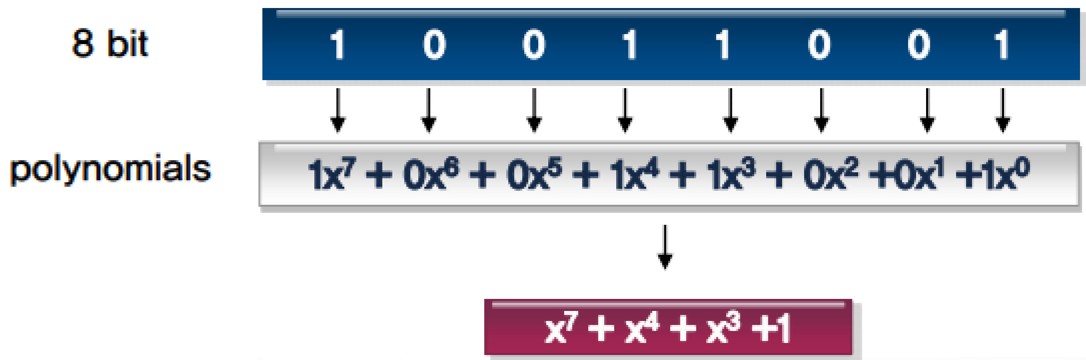
목차

1. Finite Field Arithmetic

1. Finite Field Arithmetic

Cryptography Information Security Laboratory

- AES is based in $GF(2^8)$.
  - Finite field(called Galois Fields) are “field” that contains a finite number of element
- Field
  - Informally, field is a set, along with two operations defined on that set
  - There are axioms in additional and multiplication operation
- AES is presented using polynomials :
  - $f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ ... + a_1 x + a_0 = \sum_{i=0}^{n-1} a_i x^i$



- Additional operation in $GF(2^8)$: bitwise XOR($\oplus$)
- Multiplication operation in $GF(2^8)$: Irreducible polynomials and moduler

2. AES Structure

1. General Structure
2. Detail Structure

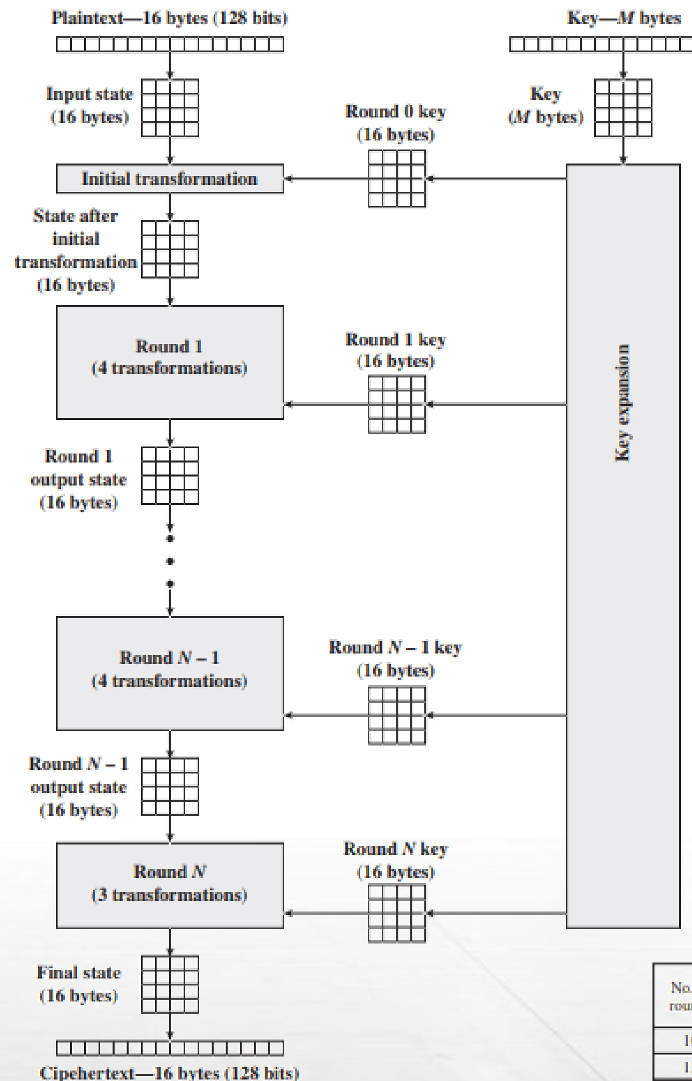
2. AES Structure - General

- AES is included in the ISO/IEC 18033-3 standard.

- Original name is Rijndael.
  - Developed by to Belgian – Vincent Rijmen and John Daemen
  - Submitted a proposal to NIST during AES selection process.

- Block size : 128 bits(16bytes)

- Key length : 128, 192, 256 bits(16,24,32 bytes)
  - AES-128, AES-192, AES-256 depending in key length



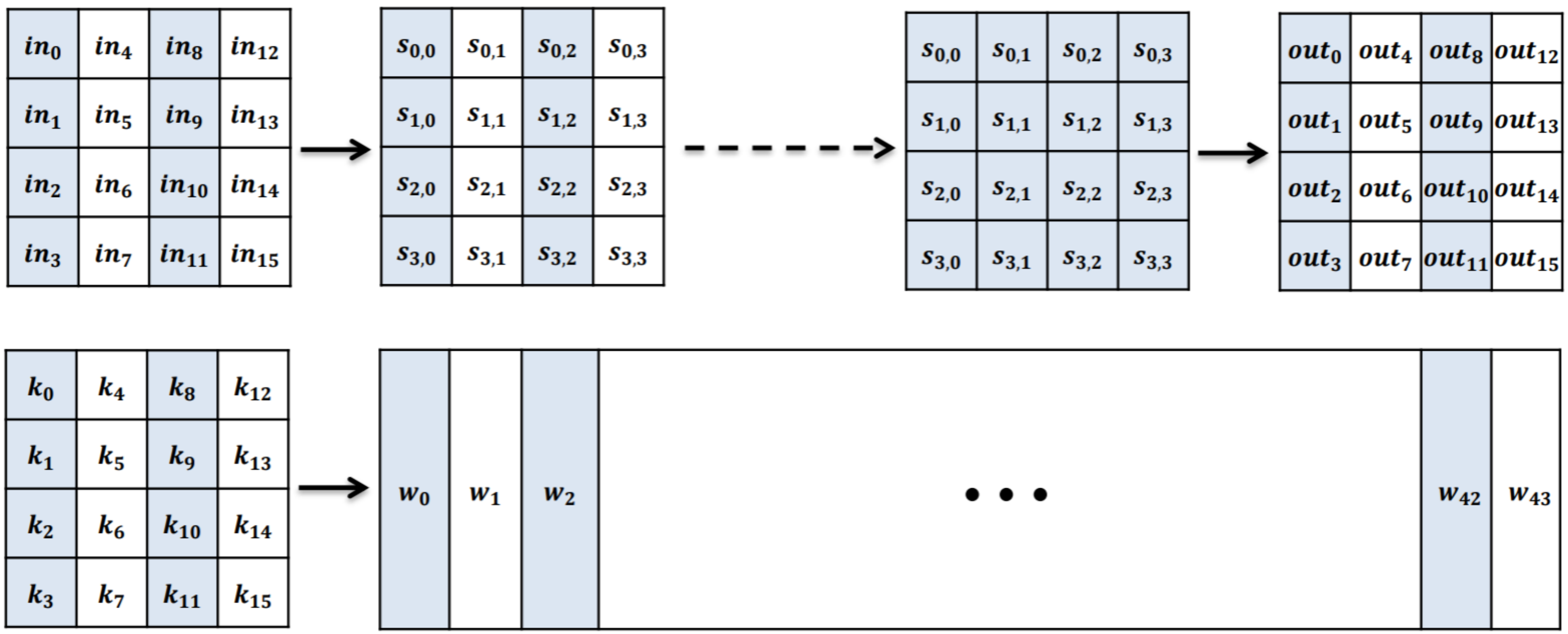
No. of rounds	Key Length (bytes)
10	16
12	24
14	32

AES encryption Process

2. AES Structure - General

Cryptography Information Security Laboratory

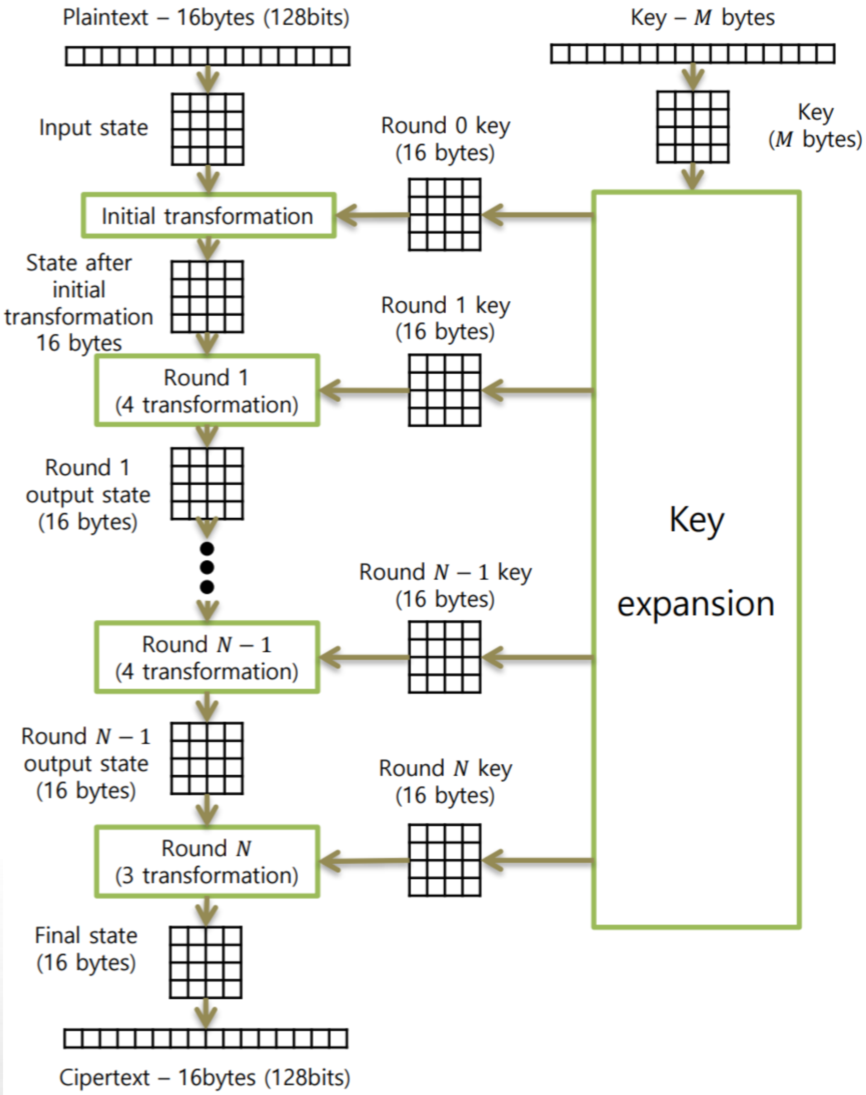
- State that 4*4 bytes array(128 bits block)is used in encryption/decryption process
- Key is presented by array 4*4 array that expanded by key scheduling algorithm
  - 128bit -> 44word(4byte)



Key length	128	192	256
Plain text size	128	128	128
Round	10	12	14
Round key length	128	128	128
Expanded key length	176	208	240

2. AES Structure - detail

1. Not Feistel structure
2. Key expanded to 44 word(32bit). Words are used to round key.
3. Consist of 1 permutation and 3 substitution process
4. One round consist of 9 round that consist of 4 process
   - Substitute Bytes
   - Shift Row
   - Mix Columns
   - Add Round Key
5. Key used only in "Add round key" process
6. SB, SR, MC provides confusion, diffusion and nonlinearity, not security. ADD round key process only provides security.
7. The whole process is reversible. Reverse function is used to decryption process.



AES encryption Process

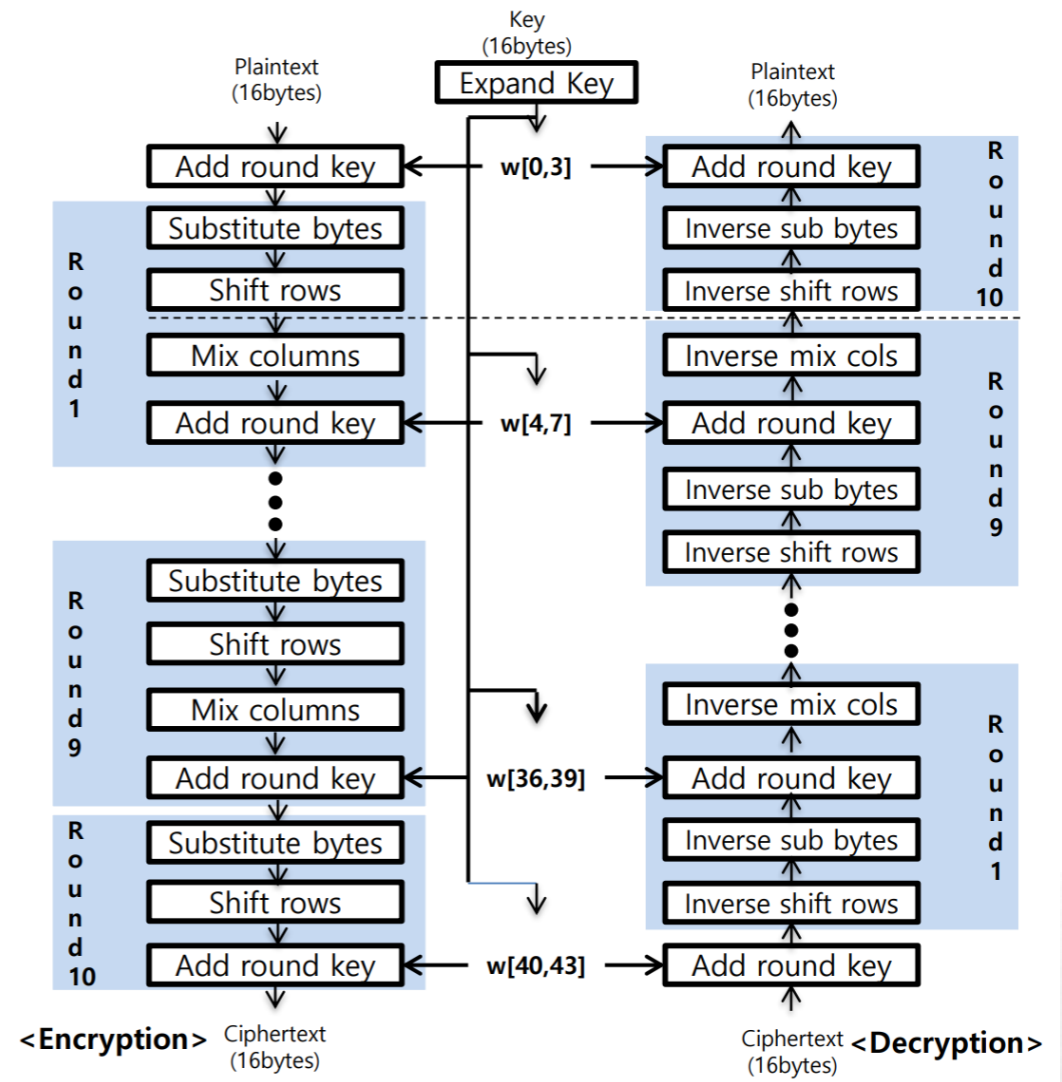
2. AES Structure - detail

Cryptography Information Security Laboratory

8. Decryption algorithm is not the same as the encryption algorithm. Use the reverse order of the expanded key.

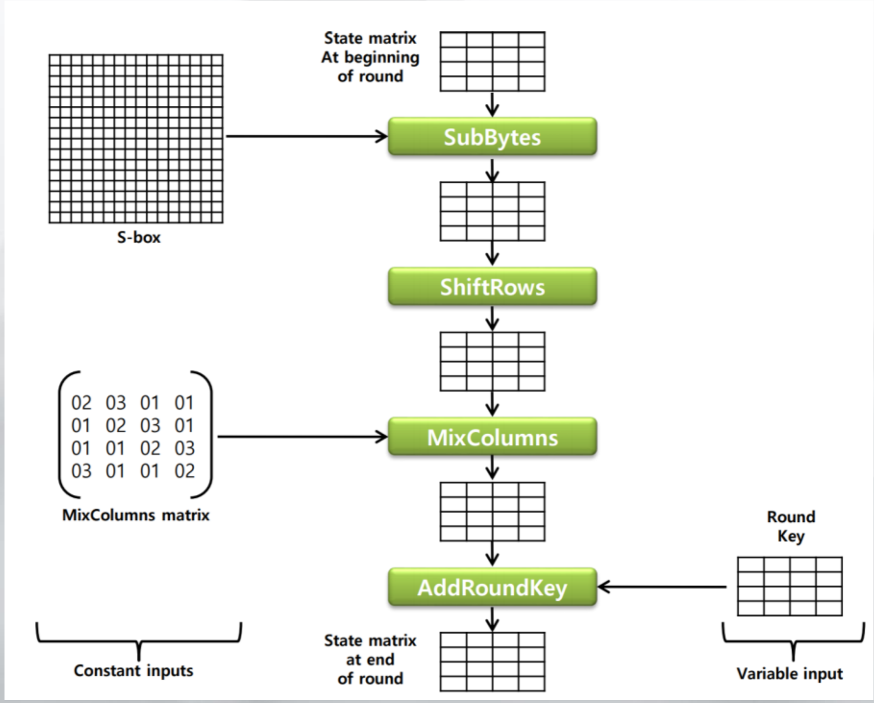
9. At each point in the horizontal direction in the right-hand figure, the state arrangement for encryption and decoding is the same.

10. For both encryption and decryption, the final round consists of only 3 process. It is necessary to be able to reverse.



AES encryption/decryption Process

3. AES Transformation Functions



1. Substitute Byte Transformation
2. Shift Row Transformation
3. Mix Columns Transformation
4. Add Round Key Transformation

Substitute Bytes

- Called SubBytes.
- S-box is 16*16 array.
- Byte in state substitute another byte in S-box.

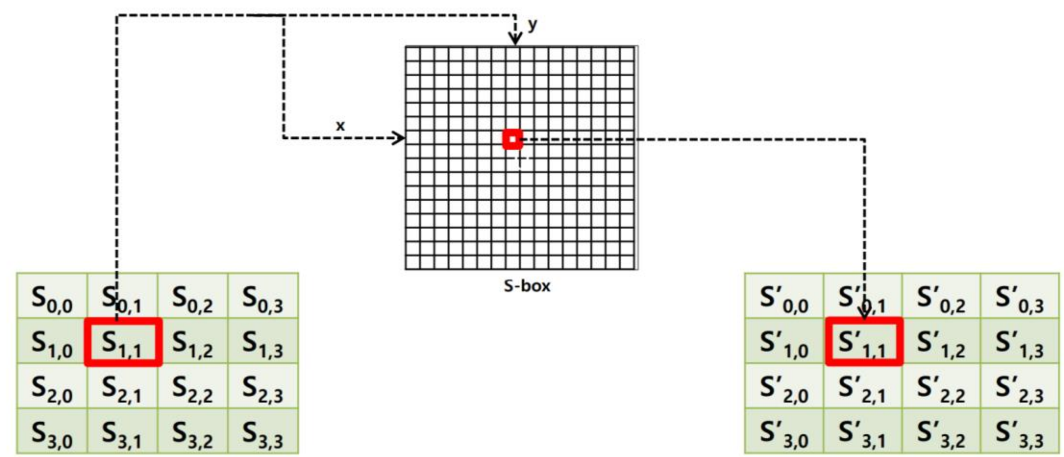
EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

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

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

S-box



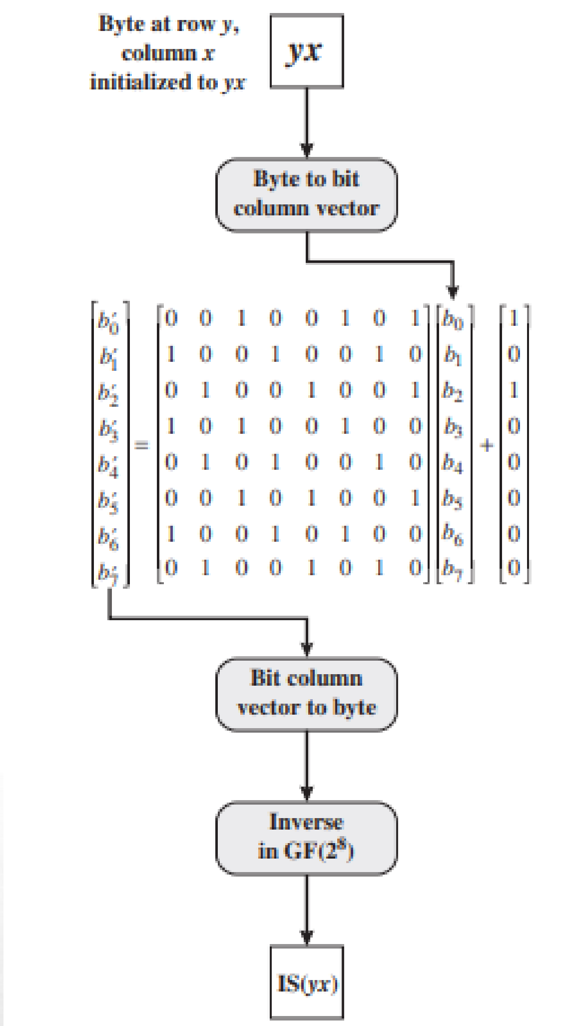
Substitute byte transformation

Substitute Bytes

- The S-box consists of the following methods:
  1. Byte at row y, column x initialized to yx
  2. yx is mapped to its multiplicative inverse in $GF(2^8)$
  3. Byte to bit column vector and transformed using following affine transformation :

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- An example, {95}
- $\{95\}^{-1} = \{8A\}$. 8A = 10001010
- Using upper affine transformation, result is 01010100
- 01010100 is {2A}



Creating inverse S-box

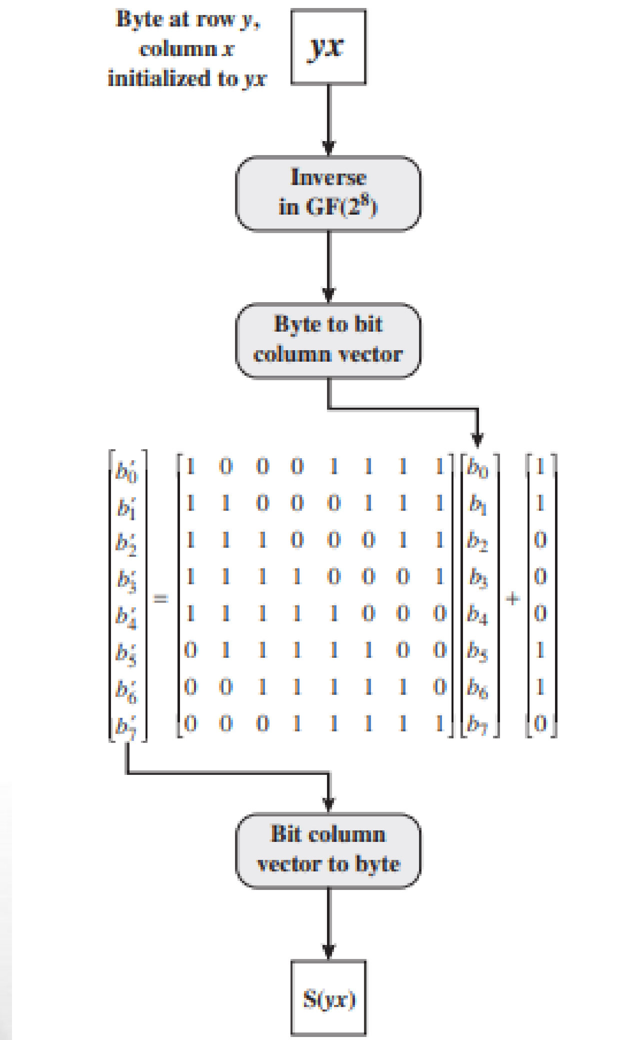
Substitute Bytes

- The inverse S-box consists of the following methods:
  1. Byte to bit column vector and transformed using following affine transformation :

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

  2. Bit Column vector to byte
  3. Inverse $GF(2^8)$

- {2A} is 01010100
- Using upper affine transformation, result is 01010100
- 8A = 10001010, $\{8A\}^{-1} = \{95\}$,

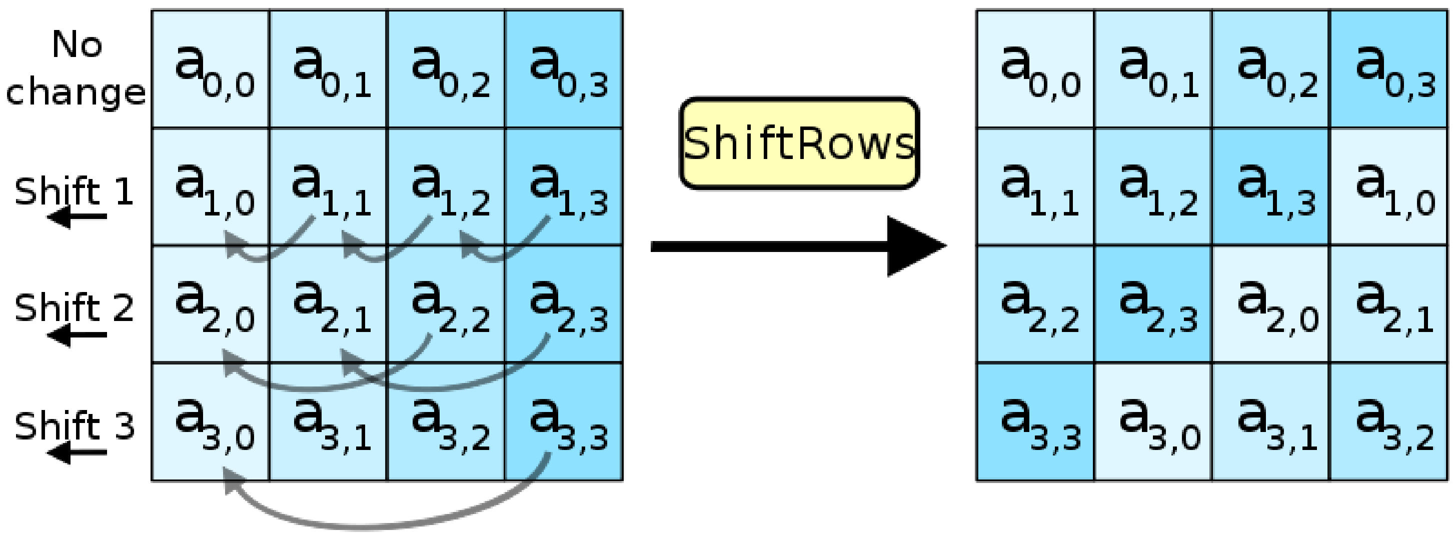


Creating S-box

Shift Row

Cryptography Information Security Laboratory

- Called ShiftRows
- The first row of a state array does not change
- The second row moves by 1byte, The third row by 3 bytes, and the fourth row by 4 bytes.



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

Mix Cloumns

Cryptography Information Security Laboratory

- Called MixColumns
- It appears as a matrix multiplication of the state array.

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

$$\begin{aligned} s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j} \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{aligned}$$

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

→

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

Mix Cloumns

$$\begin{bmatrix} \boxed{02 \quad 03 \quad 01 \quad 01} \\ 01 \quad 02 \quad 03 \quad 01 \\ 01 \quad 01 \quad 02 \quad 03 \\ 03 \quad 01 \quad 01 \quad 02 \end{bmatrix}$$

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

$\rightarrow$

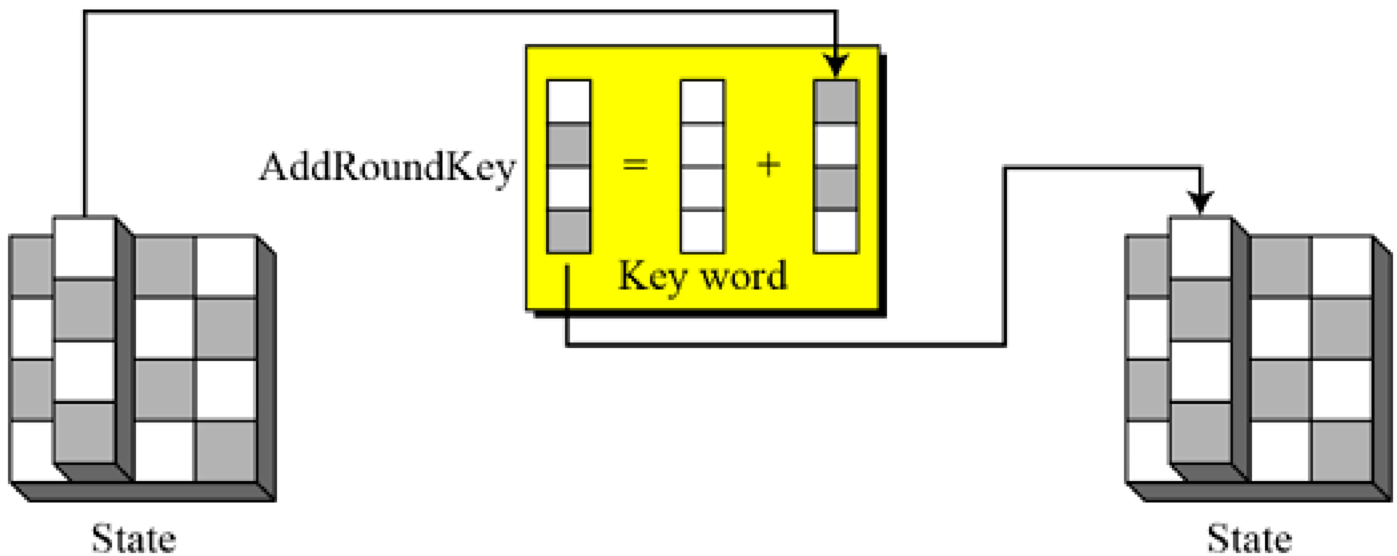
47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$\begin{aligned}
(\{02\} \cdot \{87\}) \oplus (\{03\} \cdot \{6E\}) \oplus \{46\} \oplus \{A6\} &= \{47\} \\
\{87\} \oplus (\{02\} \cdot \{6E\}) \oplus (\{03\} \cdot \{46\}) \oplus \{A6\} &= \{37\} \\
\{87\} \oplus \{6E\} \oplus (\{02\} \cdot \{46\}) \oplus (\{03\} \cdot \{A6\}) &= \{94\} \\
(\{03\} \cdot \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \cdot \{A6\}) &= \{ED\}
\end{aligned}$$

$$\begin{aligned}
\{02\} \cdot \{87\} &= 0001\ 0101 \\
\{03\} \cdot \{6E\} &= 1011\ 0010 \\
\{46\} &= 0100\ 0110 \\
\{A6\} &= \underline{1010\ 0110} \\
& \ 0100\ 0111 = \{47\}
\end{aligned}$$

Add Round Key

- 128bit round key XOR with State.



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	D6

4. AES Key Expansion

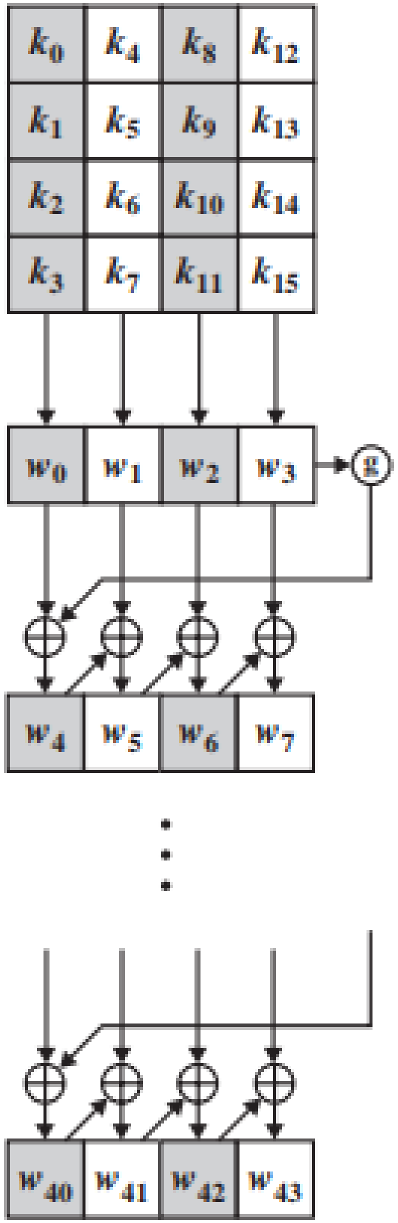
Key Expansion Algorithm

- There are 11 round keys based on 128 bits and each key consists of 4 words, total of 44 words are required

- Key Extension Order
  1. Divide the key by 4 bytes into 4 words.
  2. XOR the previous words and the fourth word before creating a new word.

  (For the draìnth of 4, pass the previous words to the g function.)

  3. Repeat until 44 words are generated.



Key Expansion Algorithm

Cryptography Information Security Laboratory

- Pseudo-code

```
KeyExpansion (byte key[16],  word w[44])
{
   word temp;
   for (i=0;  i<4;  i++) {
      w[i] = (key[4*i], key[4*i+1, key[4*i+2], key[4*i+3]);
      }
   for (i=4;  i<444;  i++) {
      temp = w[i - 1];
      if (i mod 4 = 0)
         temp = SubWord(RotWord(temp)) XOR Rcon[i/4];
      w[i] = w[i - 4] XOR temp;
      }
}
```

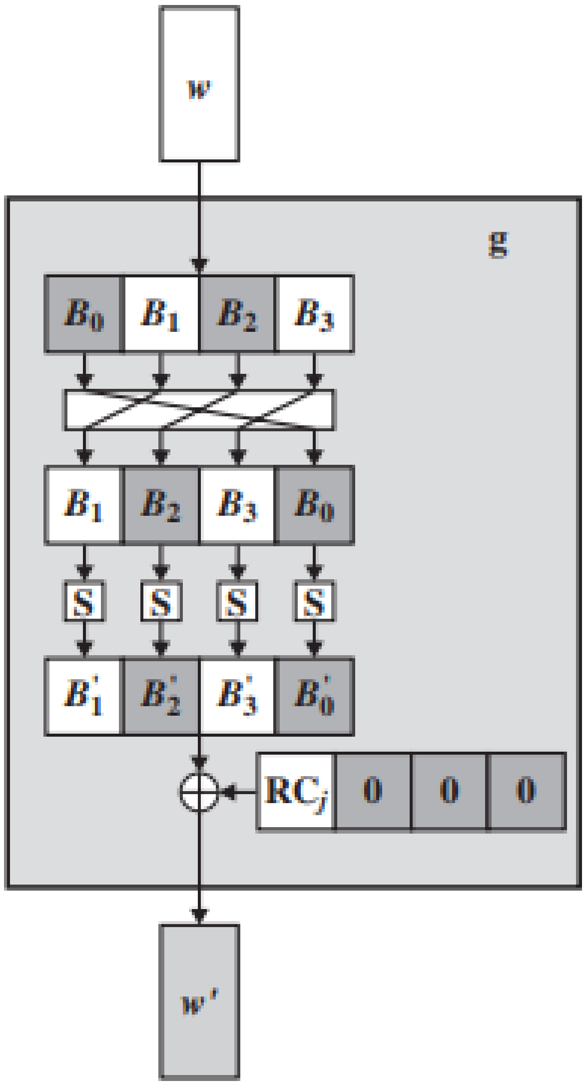
Key Expansion Algorithm

- Key expansion algorithm consists of the following methods

1. $[B_0, B_1, B_2, B_3] \rightarrow [B_1, B_2, B_3, B_0]$
2. S-box substitution
3. XOR Rcon[j]

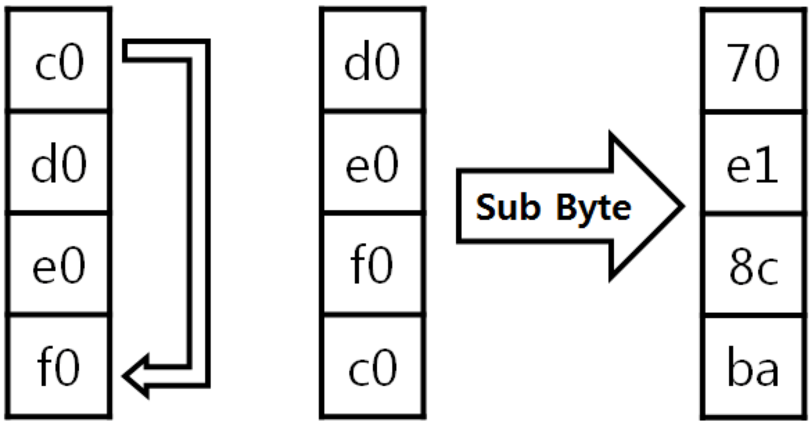
01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

R-Con



Example of key expansion subfunction G

Cryptography Information Security Laboratory



$$\begin{bmatrix} 70 \\ e1 \\ 8c \\ ba \end{bmatrix} \oplus \begin{bmatrix} 01 \\ 00 \\ 00 \\ 00 \end{bmatrix} \oplus \begin{bmatrix} 00 \\ 10 \\ 20 \\ 30 \end{bmatrix} = \begin{bmatrix} 71 & & & \\ f1 & & & \\ ac & & & \\ 8a & & & \end{bmatrix}$$

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

R-Con

5. An AES Example

Avalanche Effect in AES

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Change in plain text

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

Change in key

Thank you