RAMAIAH
Institute of Technology

Why Was the AES Encryption Algorithm necessary?

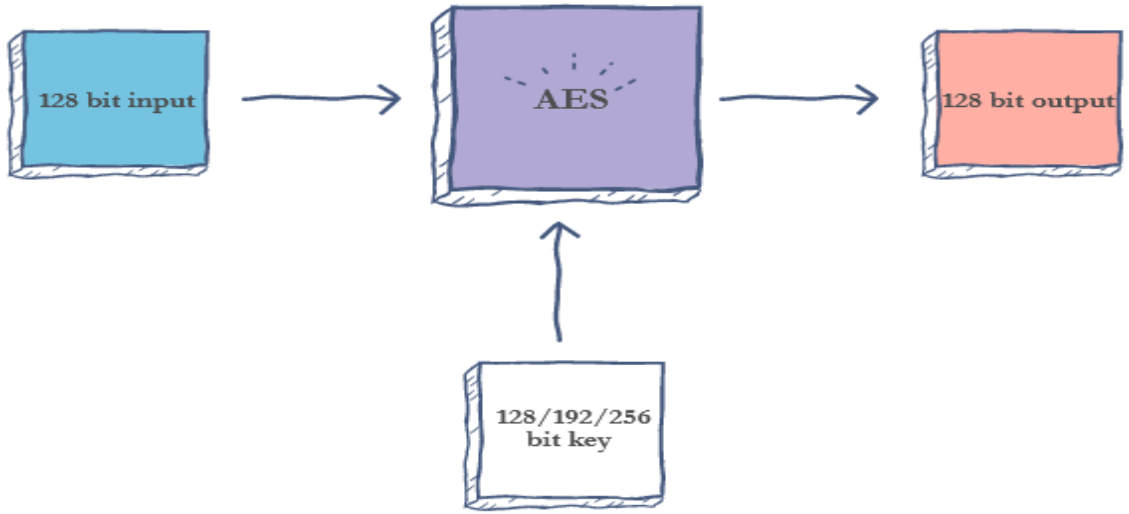
Chronology of DES Cracking	
Broken for the first time	1997
Broken in 56 hours	1998
Broken in 22 hours and 15 minutes	1999
Capable of broken in 5 minutes	2021

RAMAIAH Institute of Technology

Introduction

Advanced Encryption Standard (AES) features are as follows:

- Symmetric key symmetric block cipher
- 128-bit block data, 128/192/256-bit keys
- Stronger and faster than Triple-DES



	DES	AES
Developed	1977	2000
Cipher Type	Symmetric block cipher	Symmetric block cipher
Block size	64 bits	128 bits
Key length	56 bits	128/192/256 bits
Security	Rendered insecure	Considered secure

Introduction

- AES is an iterative rather than Feistel cipher.
- It is based on ‘substitution–permutation network’.
- It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).
- AES performs all its computations on bytes rather than bits.

Introduction

The Advanced Encryption Standard (AES) published by the National Institute of Standards and Technology (NIST) in December 2001.

- History
- Criteria
- Rounds
- Data Units
- Structure of Each Round

Introduction

History

- 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.

Introduction

Criteria

The criteria defined by NIST for selecting AES fall into three areas:

1. Security -128bit key
2. Cost-Computational efficiency and storage requirements
3. Implementation -Flexibility(platform independent)

Introduction

Rounds

- AES is a non Feistel cipher that encrypts and decrypts a block of 128bit data.
- It uses 10,12 or 14 rounds → keysize-128/192/256bits

Introduction

Rounds

AES-128

AES-192

AES-256

But roundkey is

Always 128bits

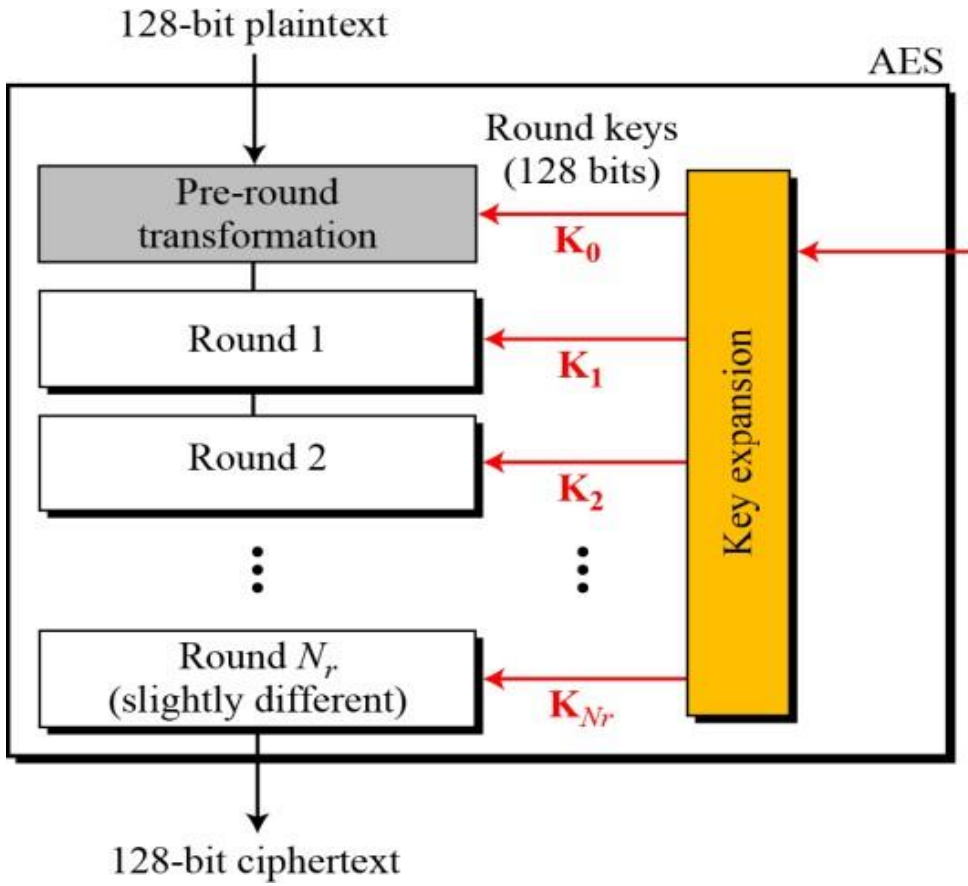
RAMAIAH Institute of Technology

Introduction

Rounds

The number of round key is one more than the number of rounds

Number of Round keys = Nr+1



Nr	Key size
10	128
12	192
14	256

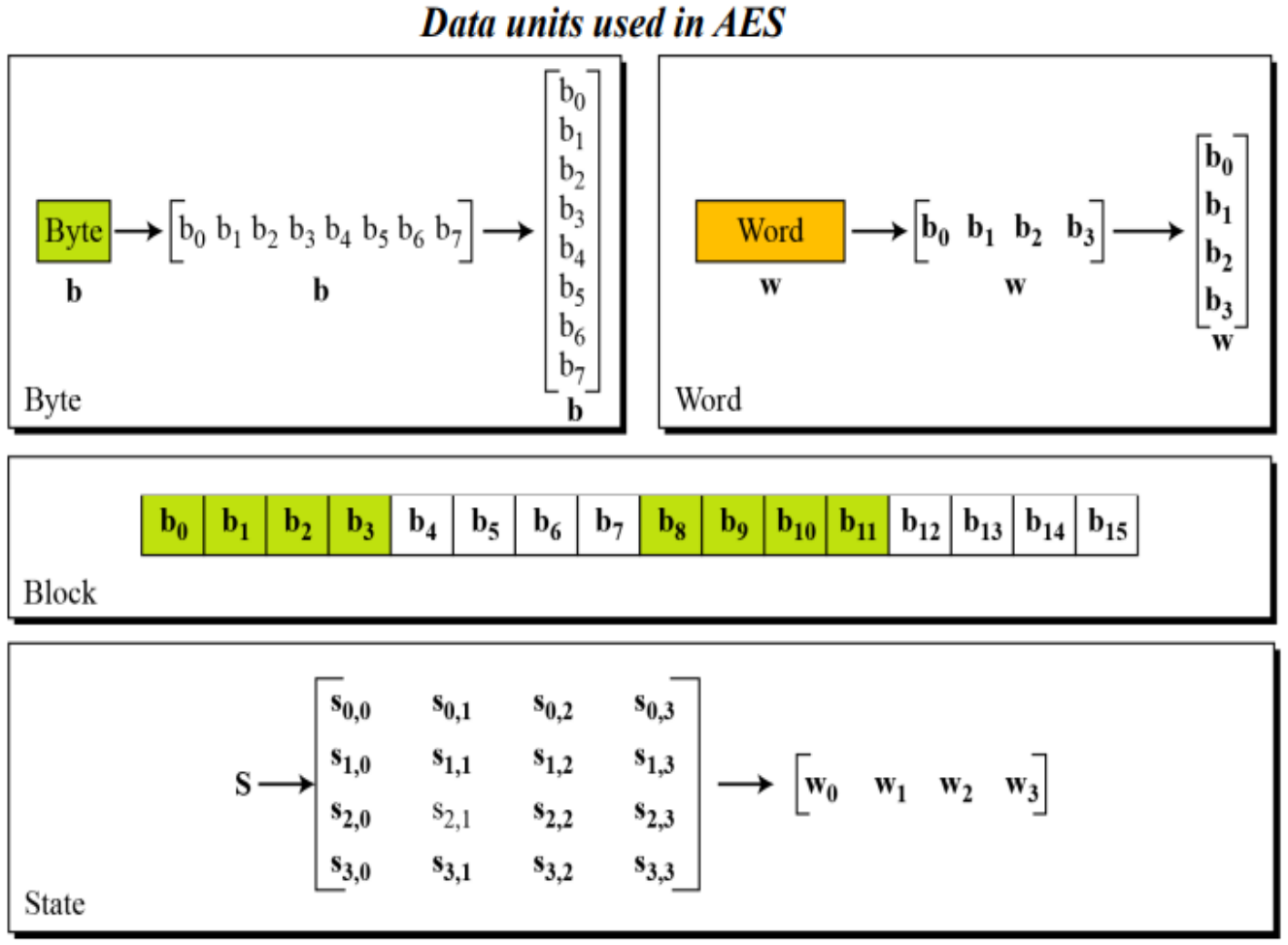
Relationship between number of rounds and cipher key size

Introduction

Data Units

AES uses 5 units of measurement to refer to data

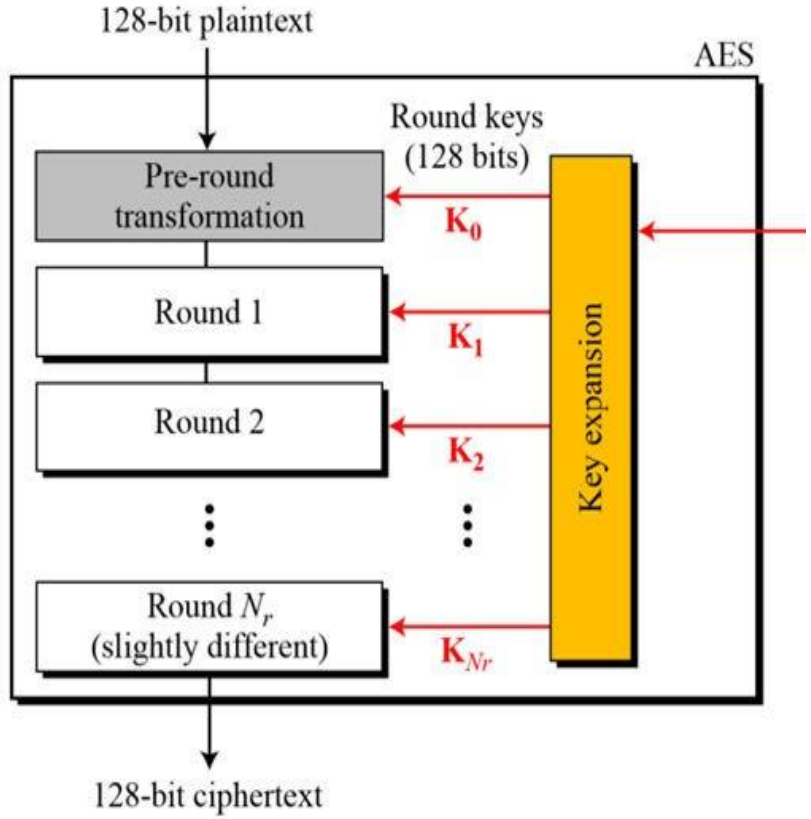
1. Bits
2. Bytes
3. Words
4. Blocks
5. State



Introduction

Data Units – States

- AES uses several rounds
- Each round is made of several stages – Data blocks are transferred from one stage to other
- At the beginning and end of cipher –data blocks
- Before and after each stage – data block is referred as state

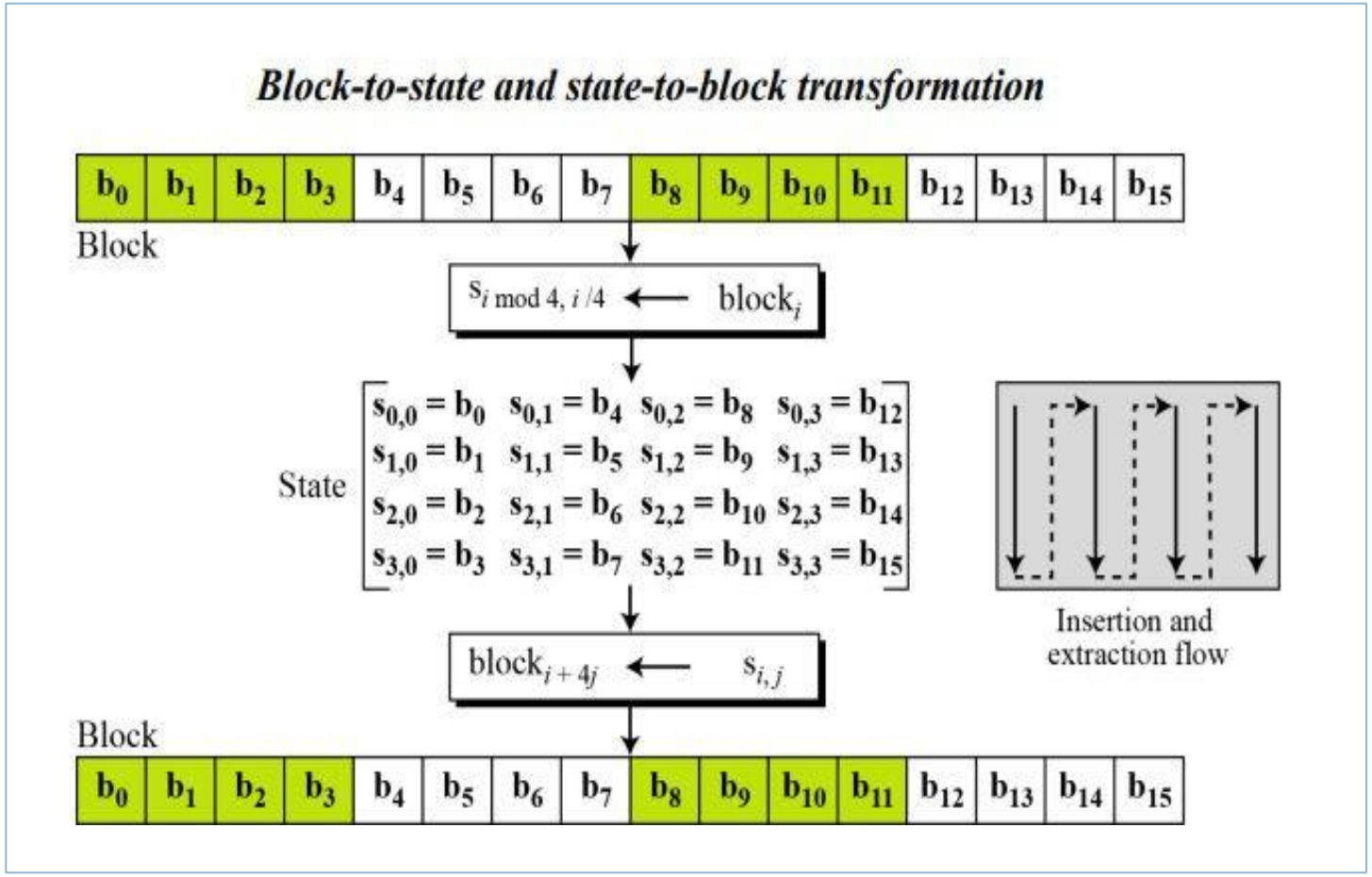


Introduction

Data Units – States

- S – State
- T- temporary state
- States are made up of 16bytes
- Matrix (4x4)

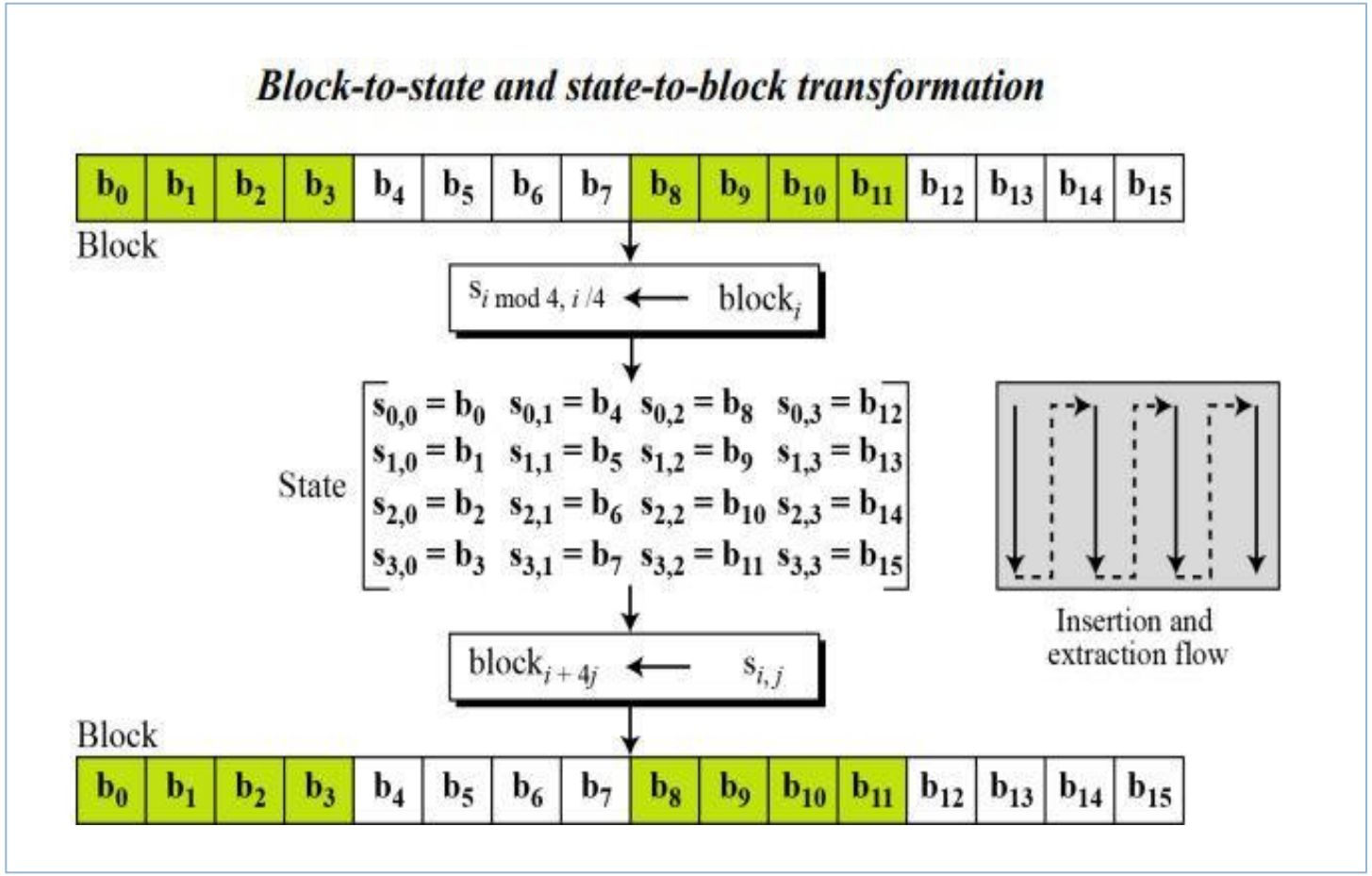
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



Introduction

Data Units – States

- Sate is treated as row matrix(1 x 4) of words



RAMAIAH Institute of Technology

Introduction

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

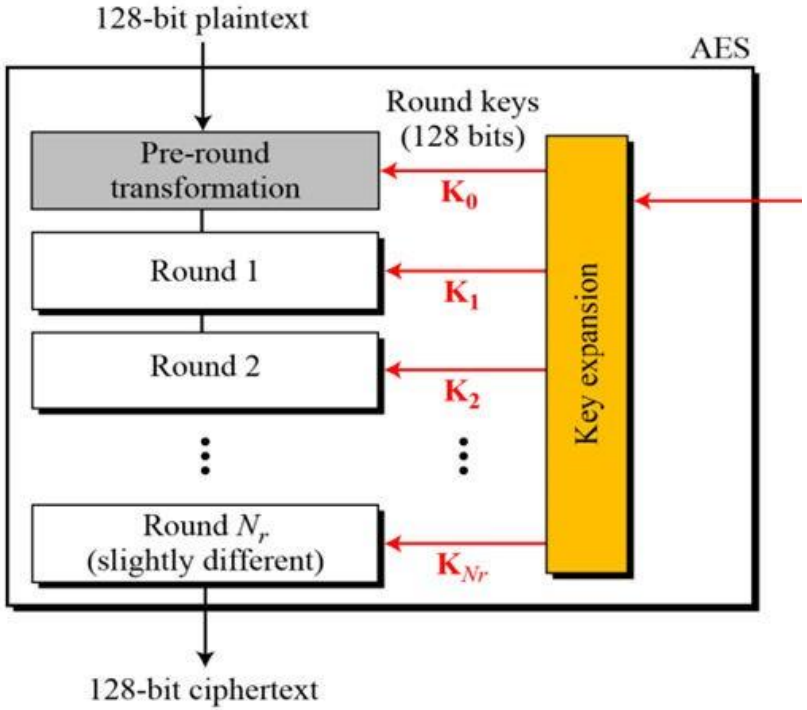
$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$ State

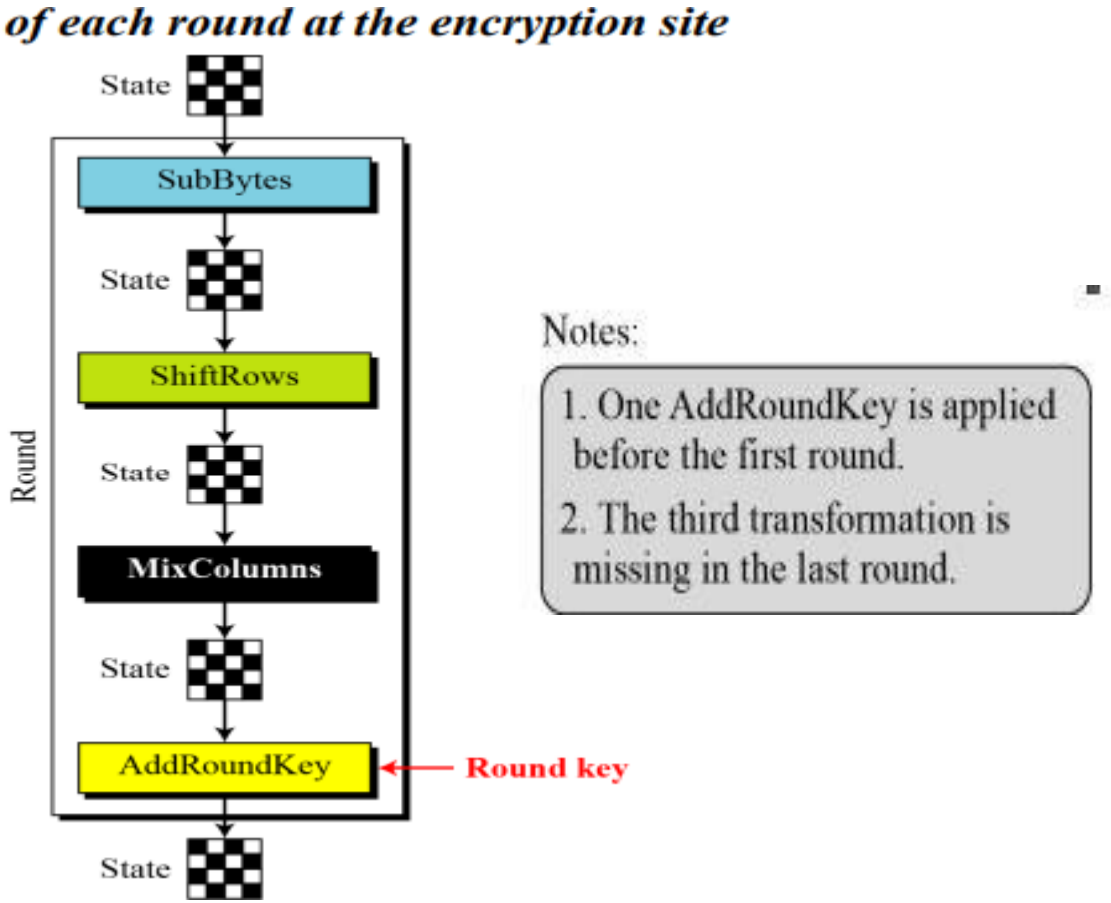
Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Decimal	Hex	Char
0	0	[NULL]
1	1	[START OF HEADING]
2	2	[START OF TEXT]
3	3	[END OF TEXT]
4	4	[END OF TRANSMISSION]
5	5	[ENQUIRY]
6	6	[ACKNOWLEDGE]
7	7	[BELL]
8	8	[BACKSPACE]
9	9	[HORIZONTAL TAB]
10	A	[LINE FEED]
11	B	[VERTICAL TAB]
12	C	[FORM FEED]
13	D	[CARRIAGE RETURN]
14	E	[SHIFT OUT]
15	F	[SHIFT IN]
16	10	[DATA LINK ESCAPE]
17	11	[DEVICE CONTROL 1]
18	12	[DEVICE CONTROL 2]
19	13	[DEVICE CONTROL 3]
20	14	[DEVICE CONTROL 4]
21	15	[NEGATIVE ACKNOWLEDGE]
22	16	[SYNCHRONOUS IDLE]
23	17	[ENG OF TRANS. BLOCK]
24	18	[CANCEL]
25	19	[END OF MEDIUM]
26	1A	[SUBSTITUTE]
27	1B	[ESCAPE]
28	1C	[FILE SEPARATOR]
29	1D	[GROUP SEPARATOR]
30	1E	[RECORD SEPARATOR]
31	1F	[UNIT SEPARATOR]

Introduction

Structure of Round





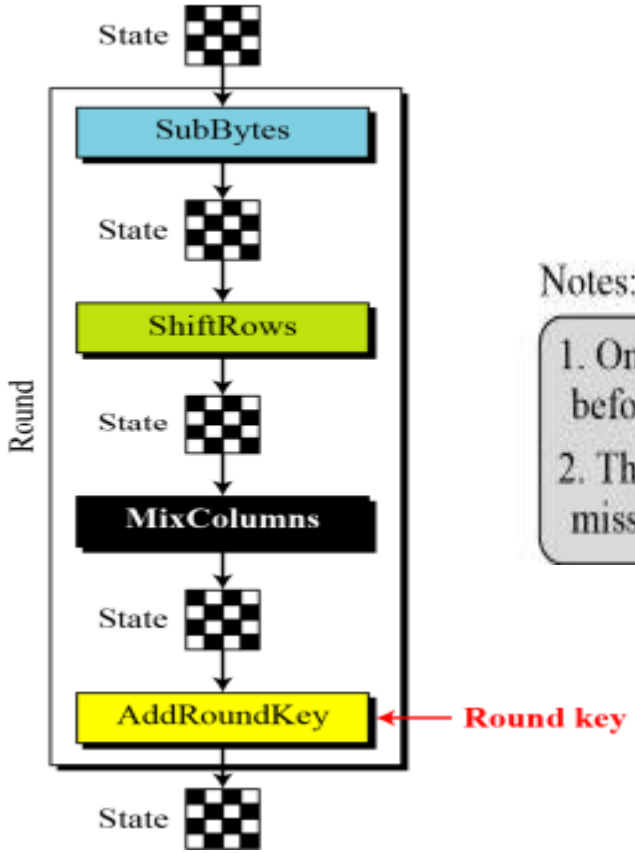
Introduction

Structure of Round

4 Transformations

1. SubBytes
2. ShiftRows
3. MixColumns
4. Add Round Keys

Structure of each round at the encryption site



Notes:

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

Transformation

1. SubBytes(Substitution)
2. ShiftRows (Permutation)
3. MixColumns (Mixing)
4. Add Round Keys (Adding)

Transformation

SubBytes(Substitution)

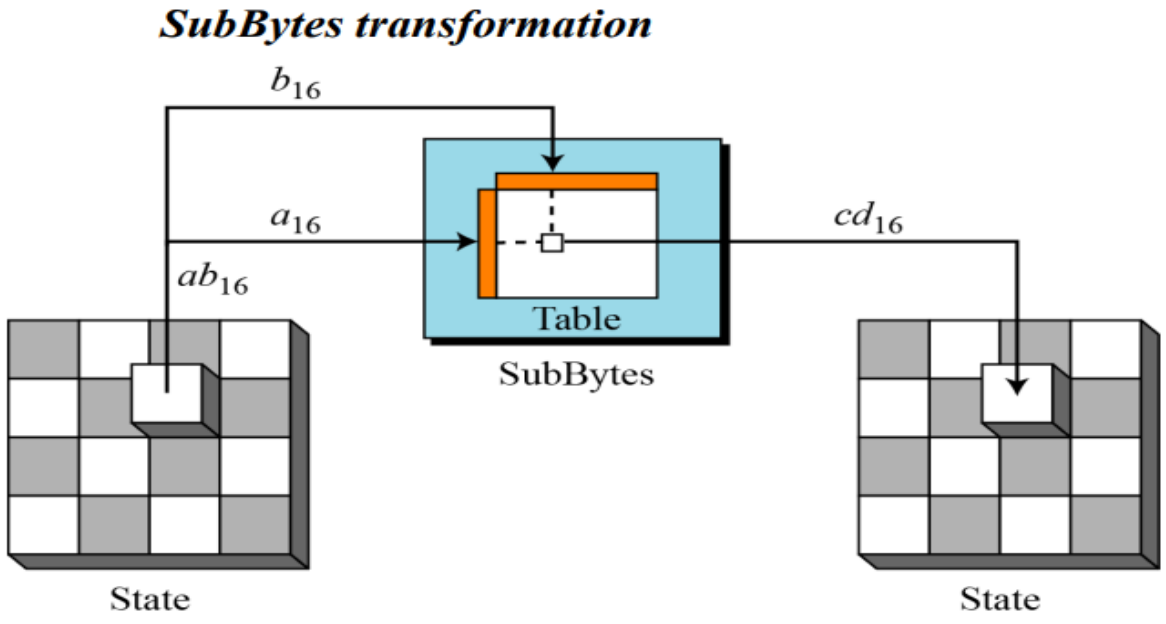
- AES uses substitution
- Mechanism is different
  1. Substitution done for each byte
  2. Table is used for substitution for each byte
  3. Table Lookup process or mathematical calculation in GF(2^8 field)

RAMAIAH Institute of Technology

Transformation

SubBytes

- SubBytes is used at the encryption site.
- To substitute a byte, we interpret the byte as two hexadecimal digits.
- Left digit –row
- Right digit -column



RAMAIAH Institute of Technology

Transformation

SubBytes- Transformation table

SubBytes transformation 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	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	CB	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

	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
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

RAMAIAH Institute of Technology

Transformation

Example 1

$5A_{16} = BE_{16}$

$5B_{16} = 39_{16}$

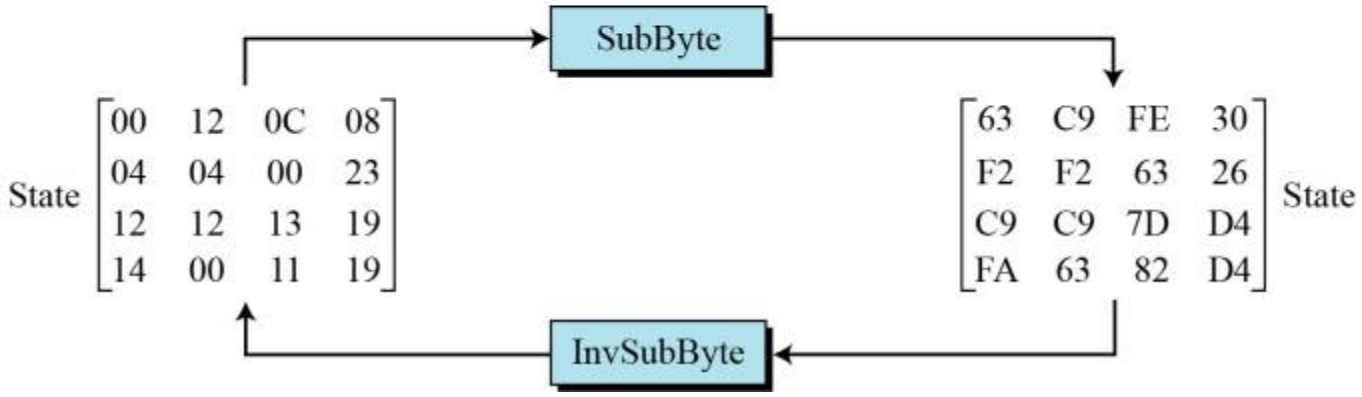
SubBytes transformation table

	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	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	CB	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

Transformation

Example 2

shows how a state is transformed using the SubBytes transformation. The figure also shows that the InvSubBytes transformation creates the original one. Note that if the two bytes have the same values, their transformation is also the same.



	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
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

Transformation

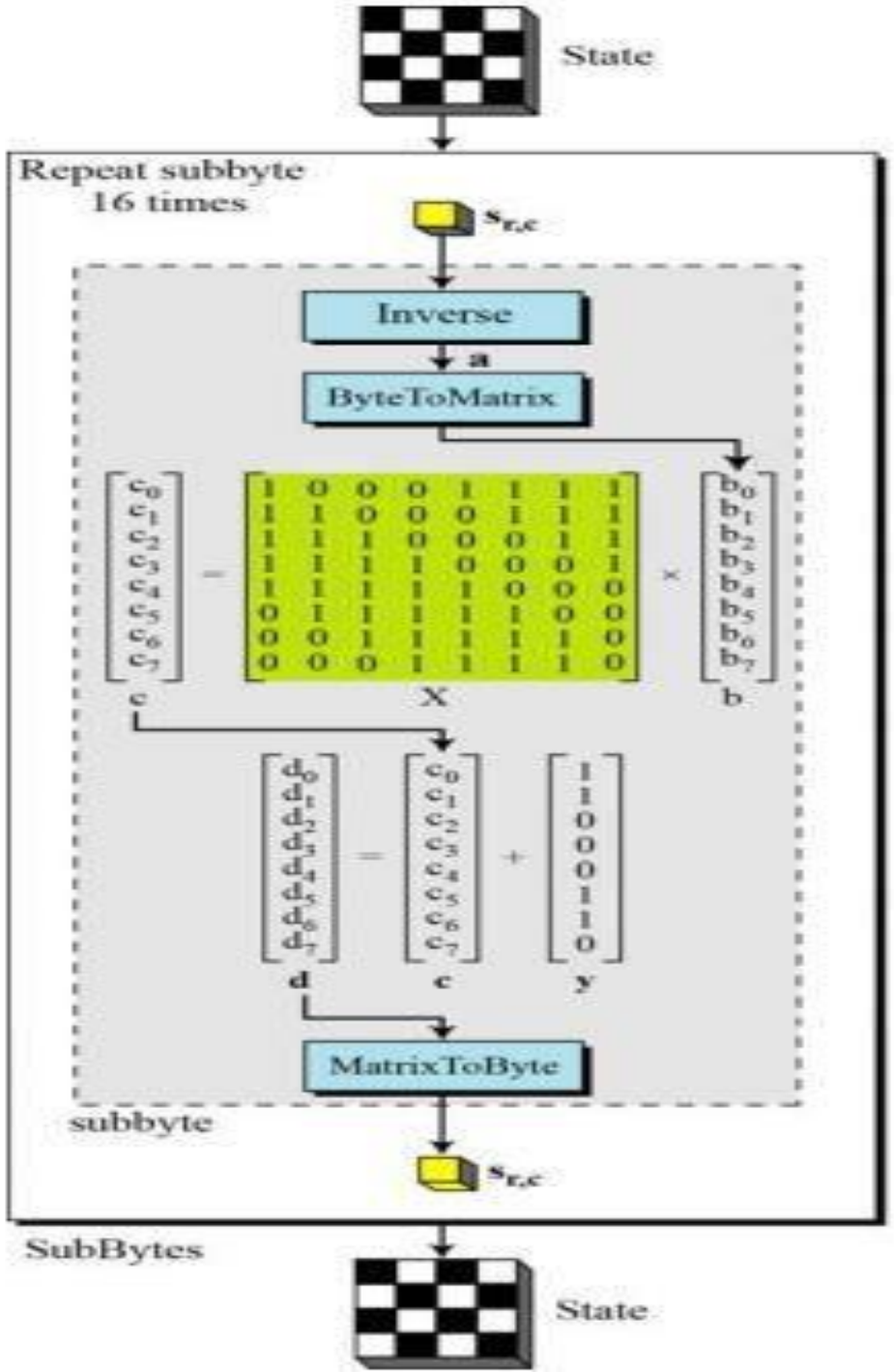
SubBytes

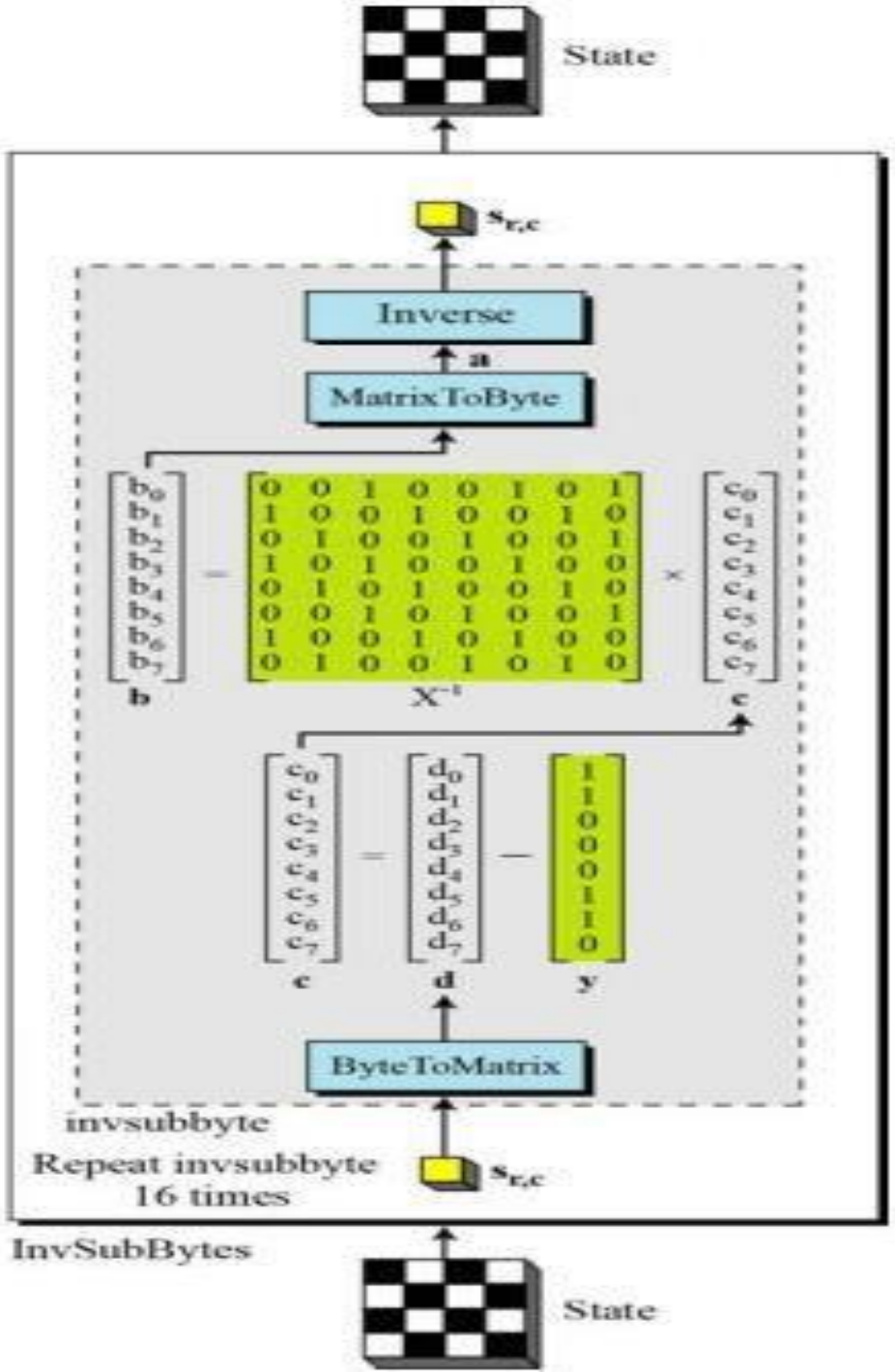
Transformation Using the GF(2^8) Field

AES also defines the transformation algebraically using the GF(28) field with the irreducible polynomials ($x^8 + x^4 + x^3 + x + 1$)

subbyte: $\rightarrow \quad \mathbf{d} = \mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y}$

invsubbyte: $\rightarrow \quad [\mathbf{X}^{-1}(\mathbf{d} \oplus \mathbf{y})]^{-1} = [\mathbf{X}^{-1}(\mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y} \oplus \mathbf{y})]^{-1} = [(s_{r,c})^{-1}]^{-1} = s_{r,c}$


State
Repeat subbyte
16 times
$s_{r,c}$
Inverse
a
ByteToMatrix
X
b
c
d
y
MatrixToByte
subbyte
$s_{r,c}$
SubBytes
State


State
$s_{r,c}$
Inverse
a
MatrixToByte
X^{-1}
b
c
d
y
ByteToMatrix
invsubbyte
Repeat invsubbyte
16 times
$s_{r,c}$
InvSubBytes
State

Transformation

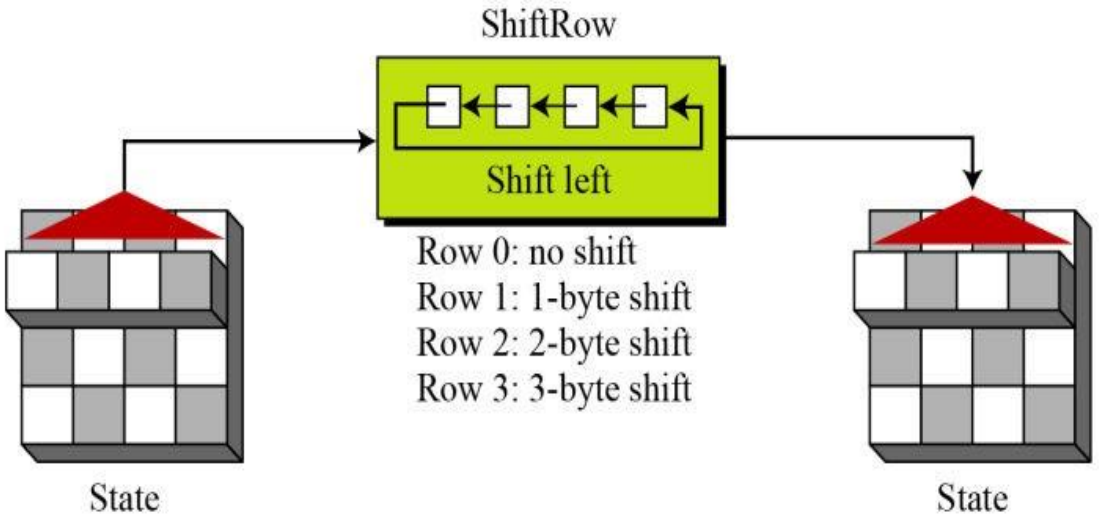
Pseudocode for SubBytes transformation

```
SubBytes (S)
{
   for (r = 0 to 3)
     for (c = 0 to 3)
           S_r,c = subbyte (S_r,c)
}

subbyte (byte)
{
   a ← byte^-1          // Multiplicative inverse in GF(2^8) with inverse of 00 to be 00
    ByteToMatrix (a, b)
    for (i = 0 to 7)
    {
        c_i ← b_i ⊕ b_(i+4)mod 8 ⊕ b_(i+5)mod 8 ⊕ b_(i+6)mod 8 ⊕ b_(i+7)mod 8
        d_i ← c_i ⊕ ByteToMatrix (0x63)
   }
    MatrixToByte (d, d)
    byte ← d
}
```

Transformation

ShiftRows (Permutation)

- Another transformation found in a round is shifting, which permutes the bytes.
- Each row is shifted a particular number of times.



[b0	b1	b2	b3]		[b0	b1	b2	b3]
| b4	b5	b6	b7 |	->	| b5	b6	b7	b4 |
| b8	b9	b10	b11 |		| b10	b11	b8	b9 |
[b12	b13	b14	b15]		[b15	b12	b13	b14]

Transformation

ShiftRows (Permutation)

- InvShiftRows In the decryption, the transformation is called InvShiftRows and the shifting is to the right.

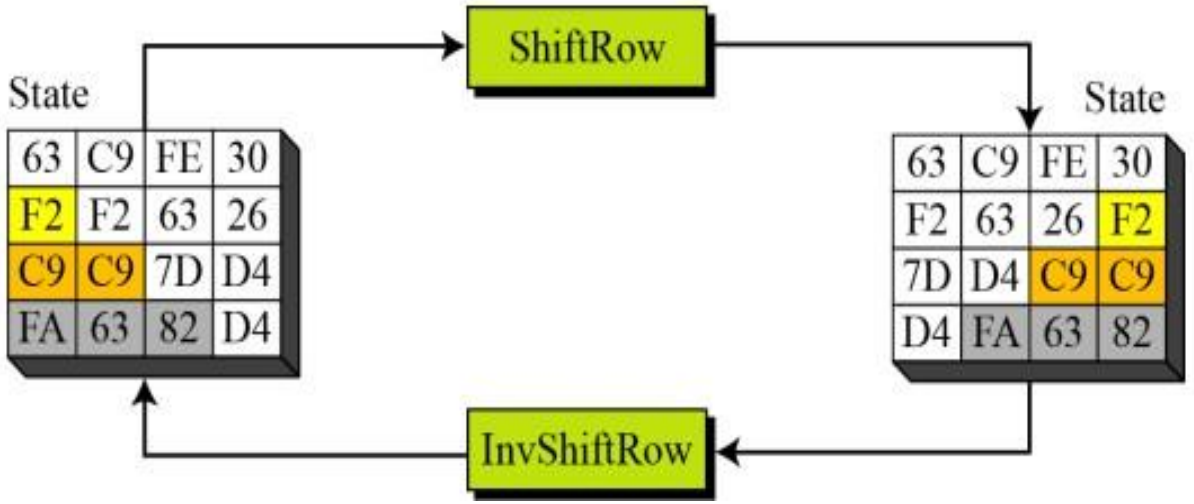
Pseudocode for ShiftRows transformation

```
ShiftRows (S)
{
    for (r = 1 to 3)
        shiftrow (s_r, r)                    // s_r is the rth row
}
shiftrow (row, n)                            // n is the number of bytes to be shifted
{
   CopyRow (row, t)                          // t is a temporary row
   for (c = 0 to 3)
        row_((c – n) mod 4) ← t_c
}
```

Transformation

ShiftRows (Permutation)

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

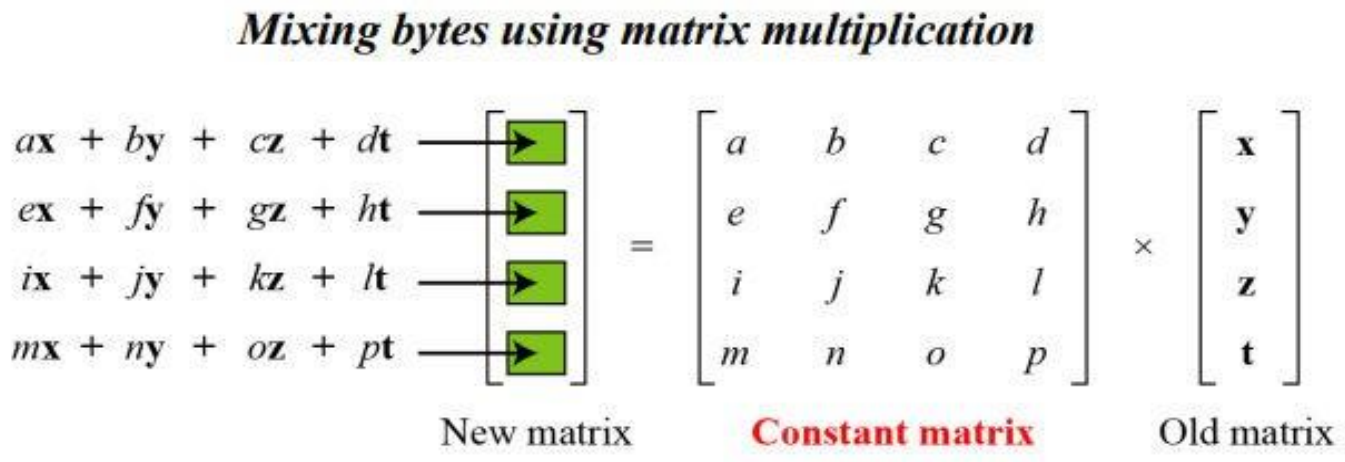


[b0	b1	b2	b3]		[b0	b1	b2	b3]
b4	b5	b6	b7	->	b5	b6	b7	b4
b8	b9	b10	b11		b10	b11	b8	b9
[b12	b13	b14	b15]		[b15	b12	b13	b14]

Transformation

MixColumns (Mixing)

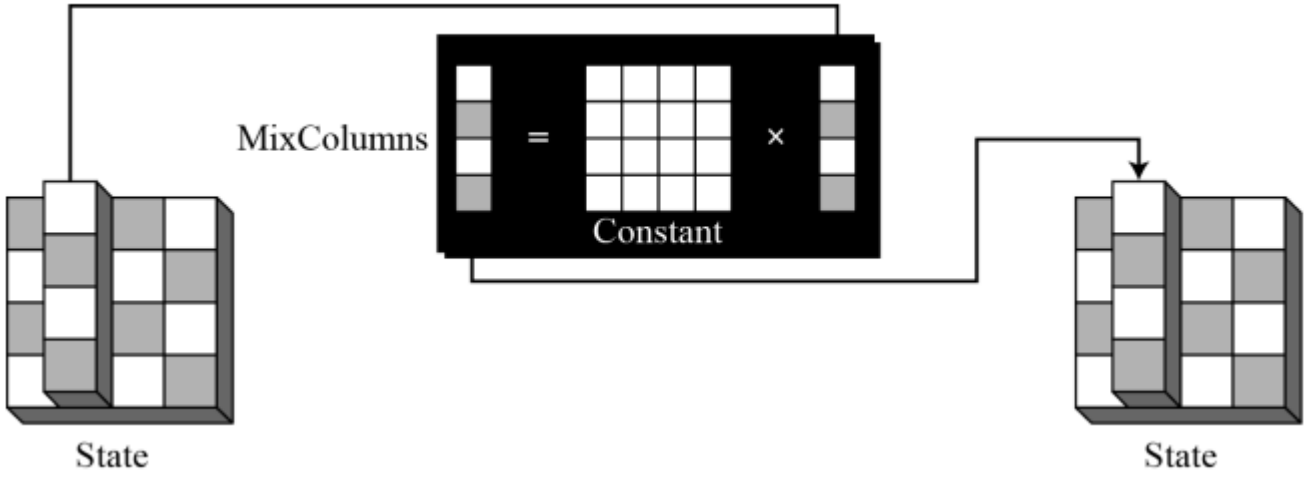
- This step is basically a matrix multiplication.
- Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.



Transformation

MixColumns (Mixing)

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



Transformation

MixColumns (Mixing)

Pseudocode for MixColumns transformation

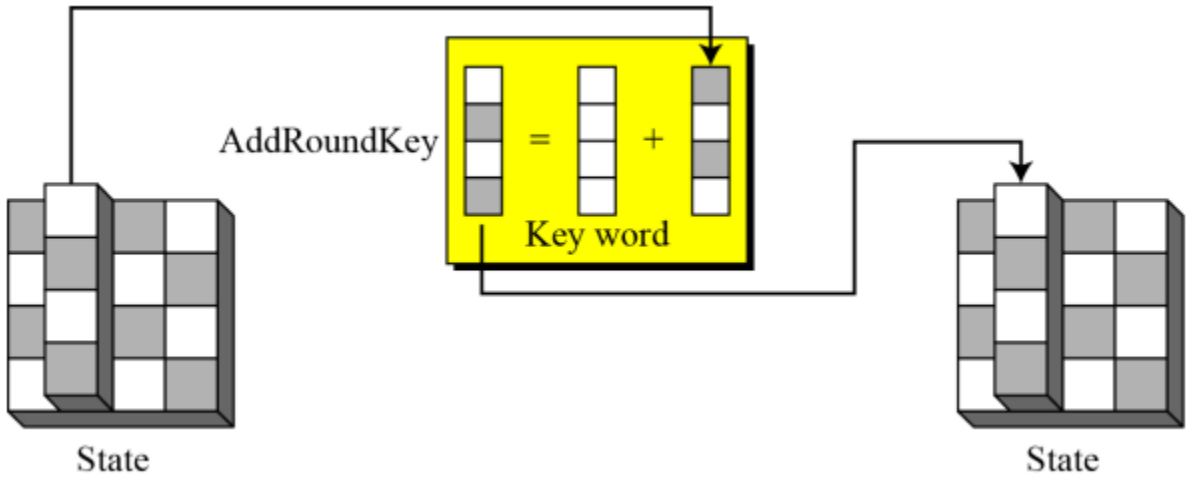
```
MixColumns (S)
{
    for (c = 0 to 3)
        mixcolumn (s_c)
}

mixcolumn (col)
{
   CopyColumn (col, t)                    // t is a temporary column

   col_0 ← (0x02) • t_0 ⊕ (0x03 • t_1) ⊕ t_2 ⊕ t_3

   col_1 ← t_0 ⊕ (0x02) • t_1 ⊕ (0x03) • t_2 ⊕ t_3

   col_2 ← t_0 ⊕ t_1 ⊕ (0x02) • t_2 ⊕ (0x03) • t_3

   col_3 ← (0x03 • t_0) ⊕ t_1 ⊕ t_2 ⊕ (0x02) • t_3
}
```

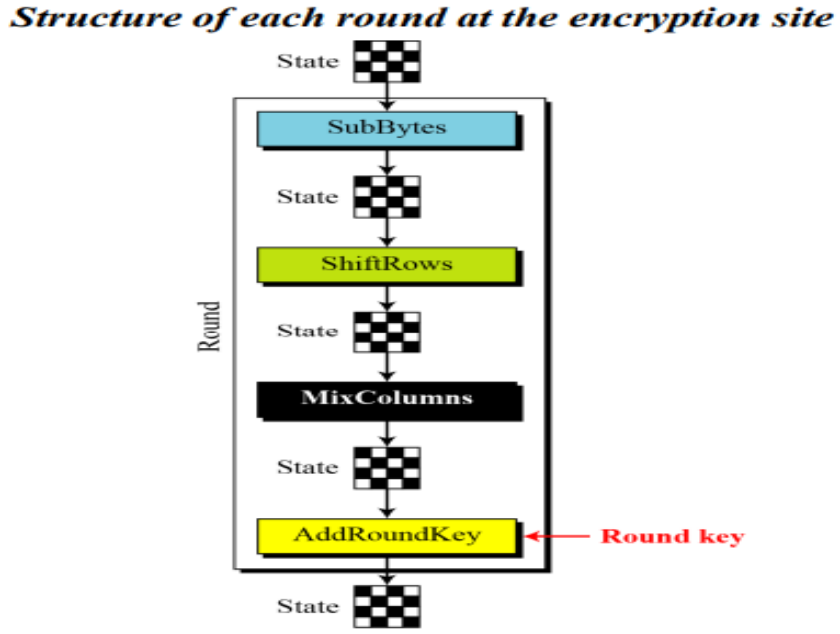
RAMAIAH Institute of Technology

Transformation

Add Round Keys (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.





Transformation

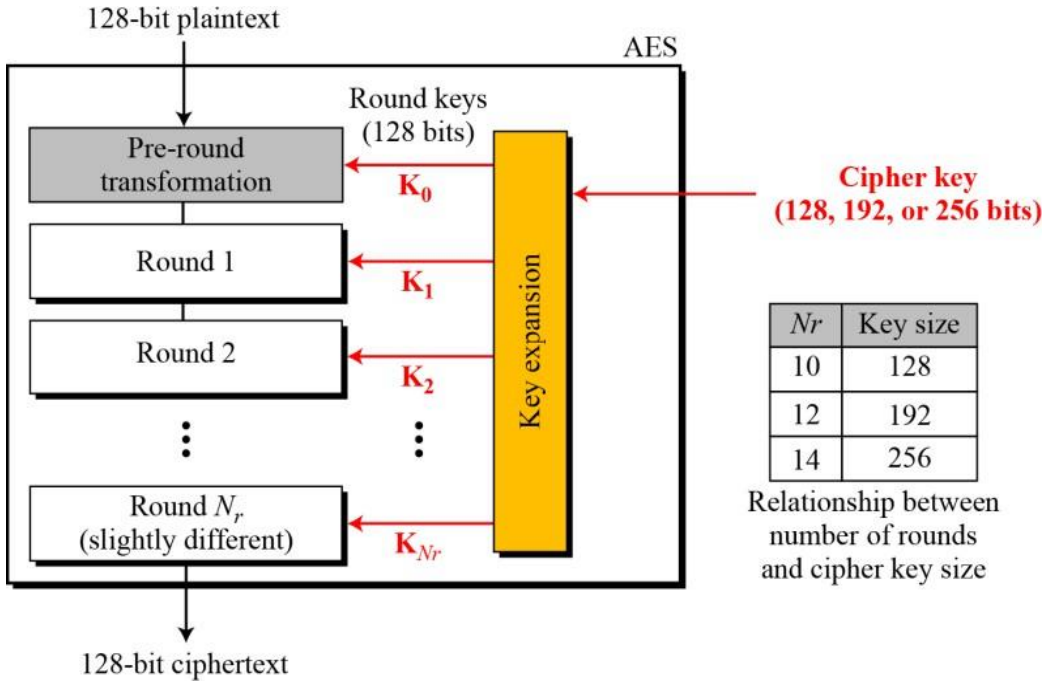
Add Round Keys (Adding)

Pseudocode for AddRoundKey transformation

```
AddRoundKey (S)
{
    for (c = 0 to 3)
        s_c ← s_c ⊕ w_round + 4c
}
```

Key Expansion

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

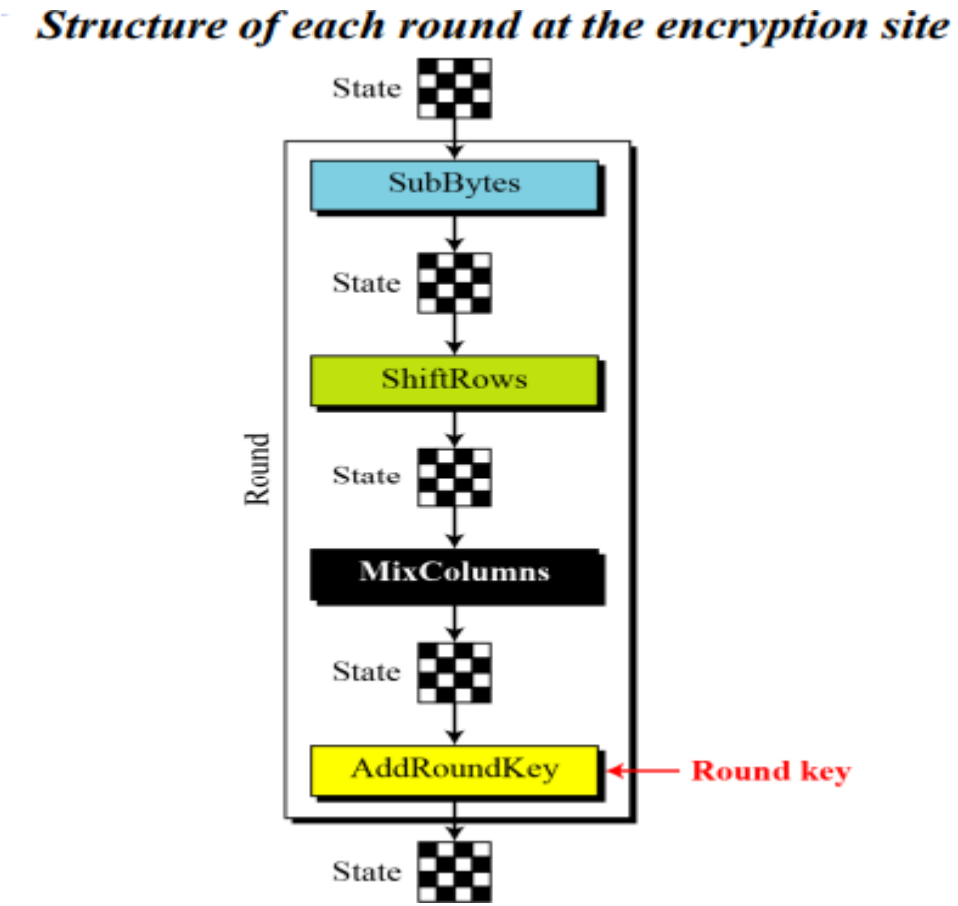


Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

Key Expansion

- First round key is used for pre-round transformation
- Remaining all for every round 4th transformation



RAMAIAH Institute of Technology

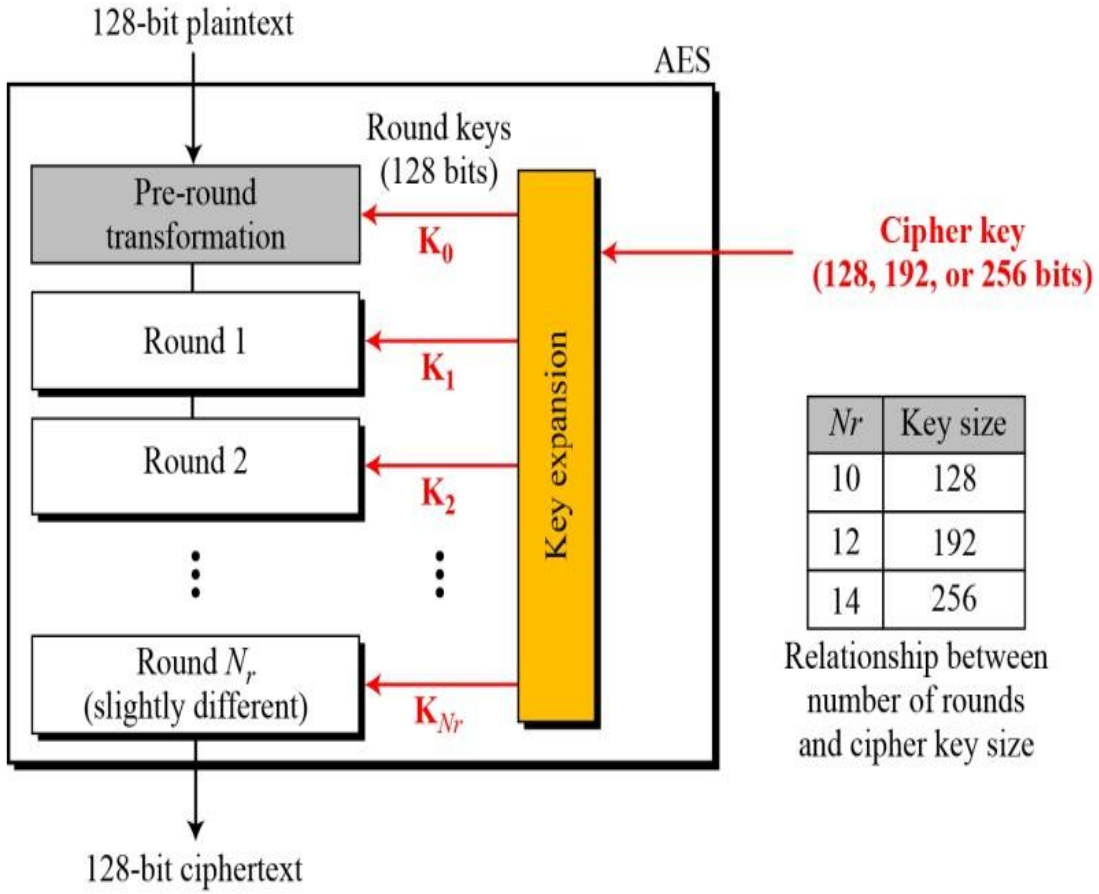
Key Expansion

Key-expansion creates round key word by word, where a word is an array of 4 bytes.

- Key Expansion in AES-128
- Key Expansion in AES-192 and AES-256
- Key-Expansion Analysis

Words for each round

Round	*Words*			
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}



Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

RAMAIAH
Institute of Technology

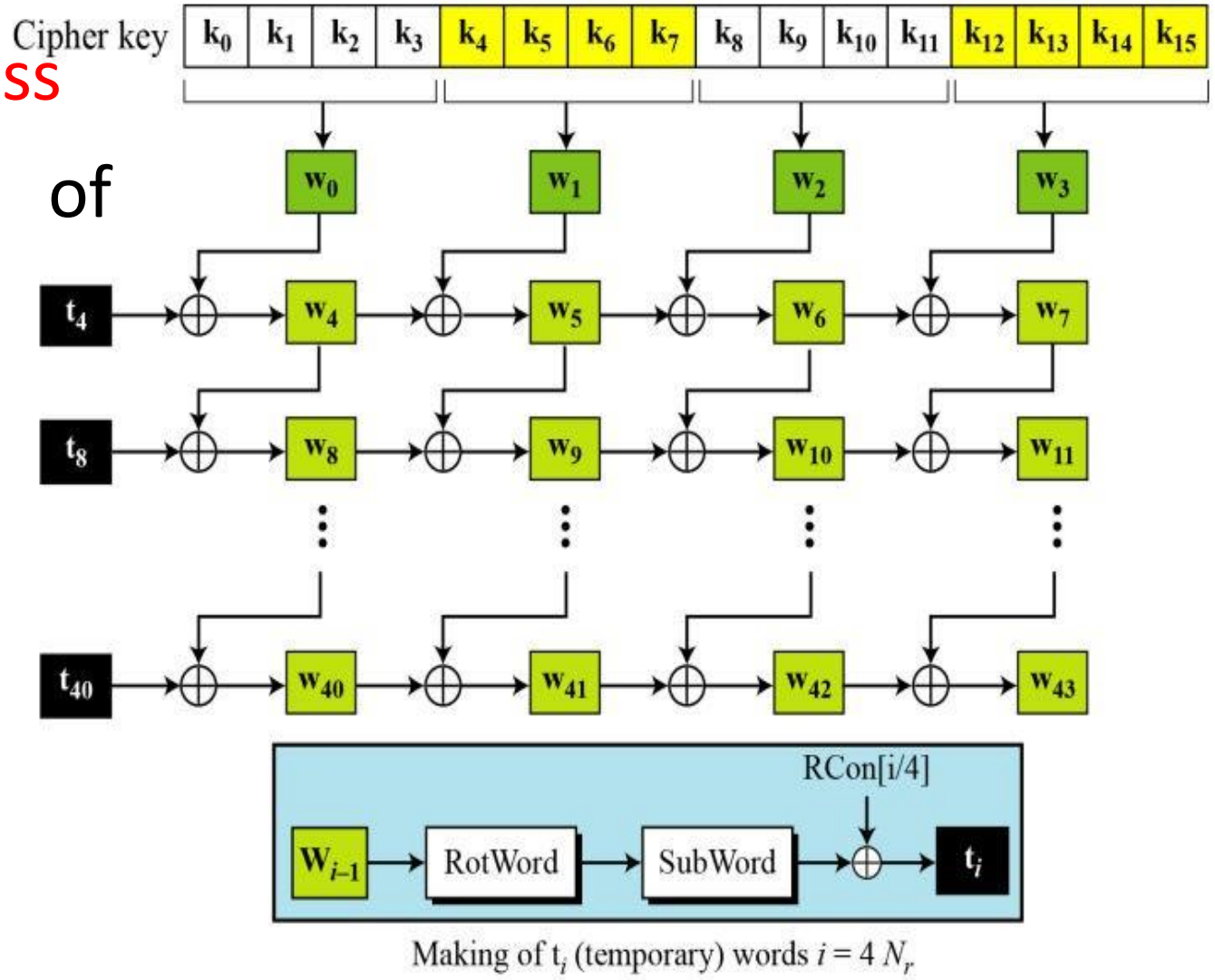
Key Expansion

Key Expansion in AES-128 process

1. Cipher key is an array of 16bytes(k0 to k15)

   The first 4 words(w0,w1,w2,w3) are made from cipher key

   - K0 to k3 -> w0
   - k4 to k7 -> w1
   - K8 to k11 ->w2
   - K12 to k15 ->w3



Making of t_i (temporary) words $i = 4\ N_r$

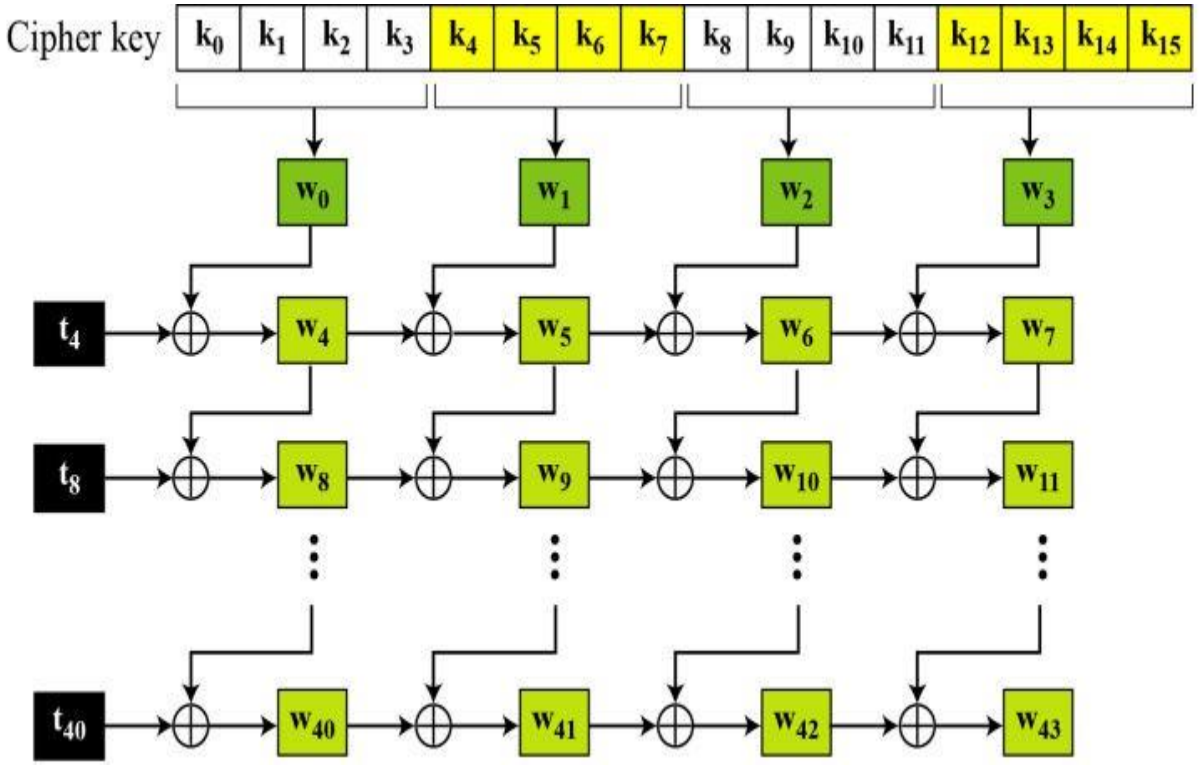
Key Expansion

Key Expansion in AES-128 process

2. The rest of the words (w_i for i : are made as follows

i) if $(i \bmod 4) \neq 0$, $w_i = w_{i-1} \oplus w_{i-4}$

ii) if $(i \bmod 4) = 0$, $w_i = t \oplus w_{i-4}$



Temporary word t

$t_i = \text{subword}(\text{Rotword}(w_{i-1})) \oplus \text{Rcon}_{i/4}$

Key Expansion

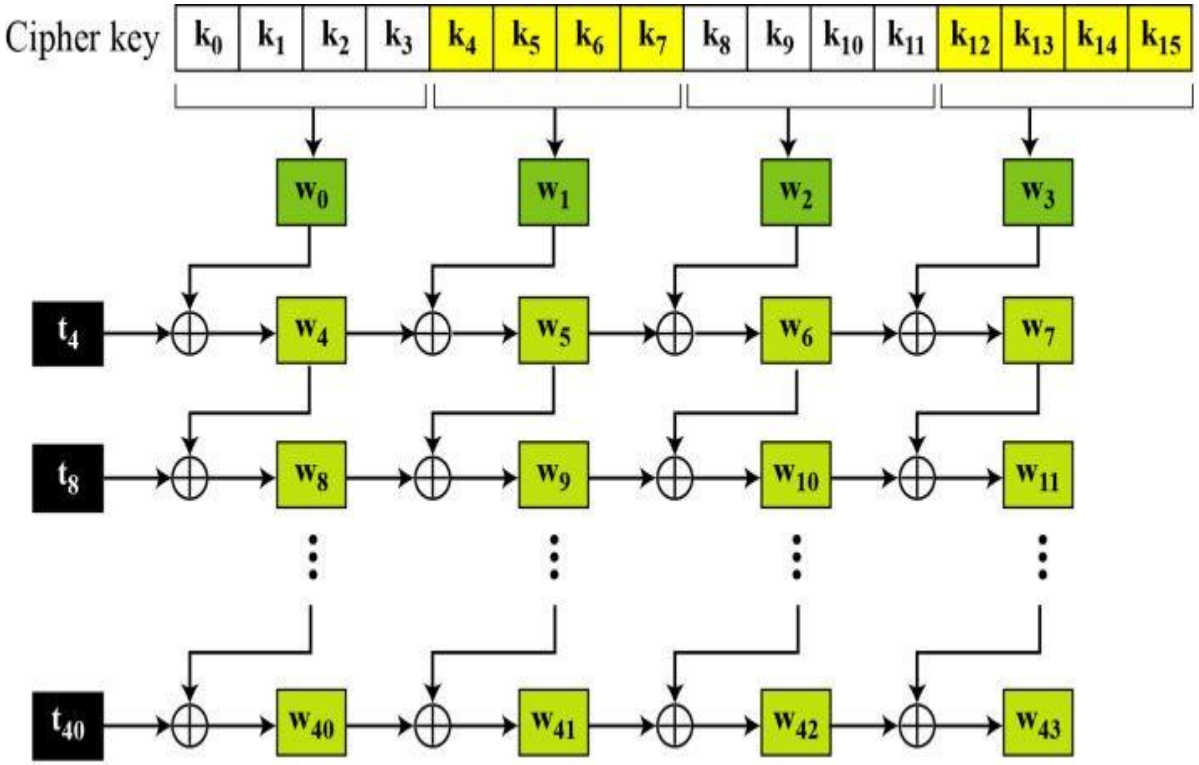
$t_i = subword(Rotword(w_{i-1})) \oplus Rcon_{i/4}$

Rotword : Applied to only one row

Rotate word routine takes a word as an array of 4bytes and shifts each byte to the left with wrapping.

Subword : Applied to 4 bytes.

Substitute word routine takes each byte in the word and substitute another byte for it.



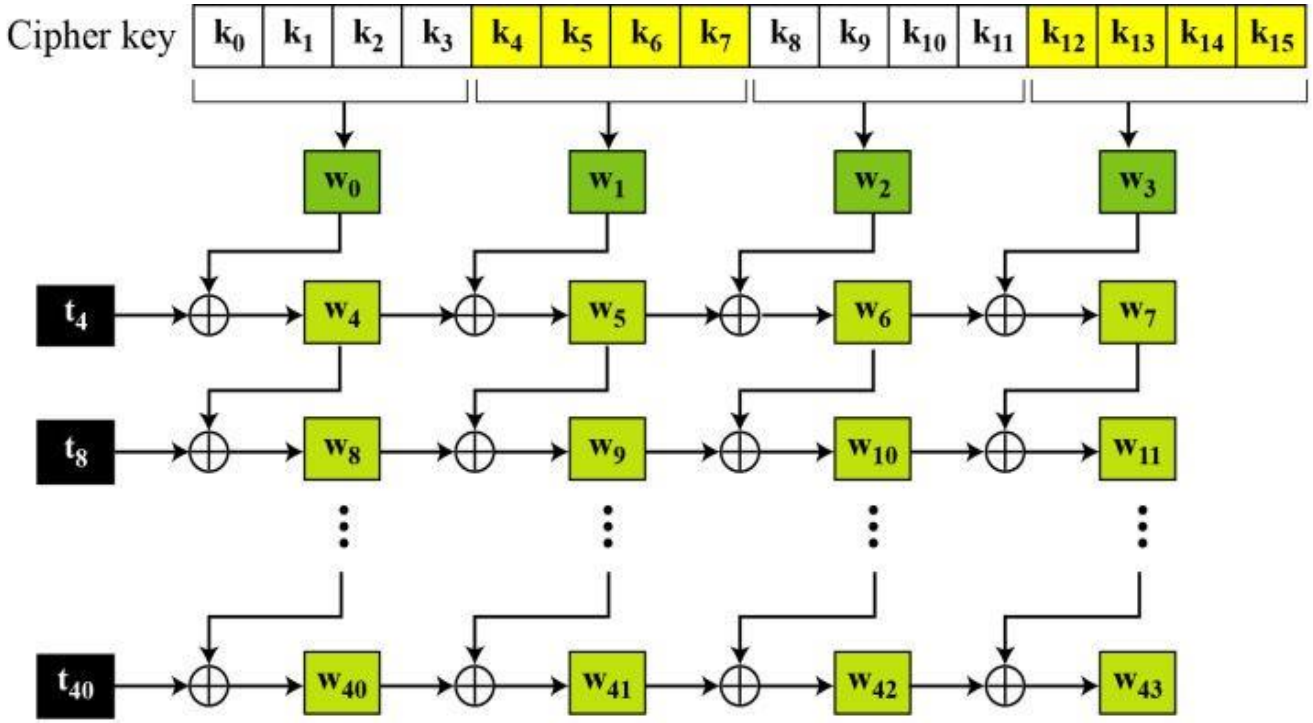
RAMAIAH Institute of Technology

Key Expansion

$t_i = subword(Rotword(w_{i-1})) \oplus Rcon_{i/4}$

Rcon : Round constant is a 4byte value in which the rightmost 3bytes are always zero

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



RAMAIAH
Institute of Technology

Key Expansion

The key-expansion routine can either use the table when calculating the words or use the GF(2^8) field to calculate the leftmost byte dynamically, as shown below

RC_1	$\rightarrow x^{1-1}$	$=x^0$	mod *prime*	$= 1$	$\rightarrow 00000001$	$\rightarrow 01_{16}$
RC_2	$\rightarrow x^{2-1}$	$=x^1$	mod *prime*	$= x$	$\rightarrow 00000010$	$\rightarrow 02_{16}$
RC_3	$\rightarrow x^{3-1}$	$=x^2$	mod *prime*	$= x^2$	$\rightarrow 00000100$	$\rightarrow 04_{16}$
RC_4	$\rightarrow x^{4-1}$	$= x^3$	mod *prime*	$= x^3$	$\rightarrow 00001000$	$\rightarrow 08_{16}$
RC_5	$\rightarrow x^{5-1}$	$= x^4$	mod *prime*	$= x^4$	$\rightarrow 00010000$	$\rightarrow 10_{16}$
RC_6	$\rightarrow x^{6-1}$	$= x^5$	mod *prime*	$= x^5$	$\rightarrow 00100000$	$\rightarrow 20_{16}$
RC_7	$\rightarrow x^{7-1}$	$= x^6$	mod *prime*	$= x^6$	$\rightarrow 01000000$	$\rightarrow 40_{16}$
RC_8	$\rightarrow x^{8-1}$	$=x^7$	mod *prime*	$= x^7$	$\rightarrow 10000000$	$\rightarrow 80_{16}$
RC_9	$\rightarrow x^{9-1}$	$=x^8$	mod *prime*	$= x^4 + x^3 + x + 1$	$\rightarrow 00011011$	$\rightarrow 1B_{16}$
RC_{10}	$\rightarrow x^{10-1}$	$=x^9$	mod *prime*	$= x^5 + x^4 + x^2 + x$	$\rightarrow 00110110$	$\rightarrow 36_{16}$

RAMAIAH
Institute of Technology

Key Expansion

Key Expansion in AES-128 - Algorithm

KeyExpansion ([key_0 to key_{15}], [$\mathbf{w}_0$ **to** $\mathbf{w}_{43}$])

```
{
    for (i = 0 to 3)
        w_i ← key_4i + key_4i+1 + key_4i+2 + key_4i+3

    for (i = 4 to 43)
    {
        if (i mod 4 ≠ 0)    w_i ← w_i−1 + w_i−4
        else
        {
            t ← SubWord (RotWord (w_i−1)) ⊕ RCon_i/4
            w_i ← t + w_i−4
        }
    }
}
```

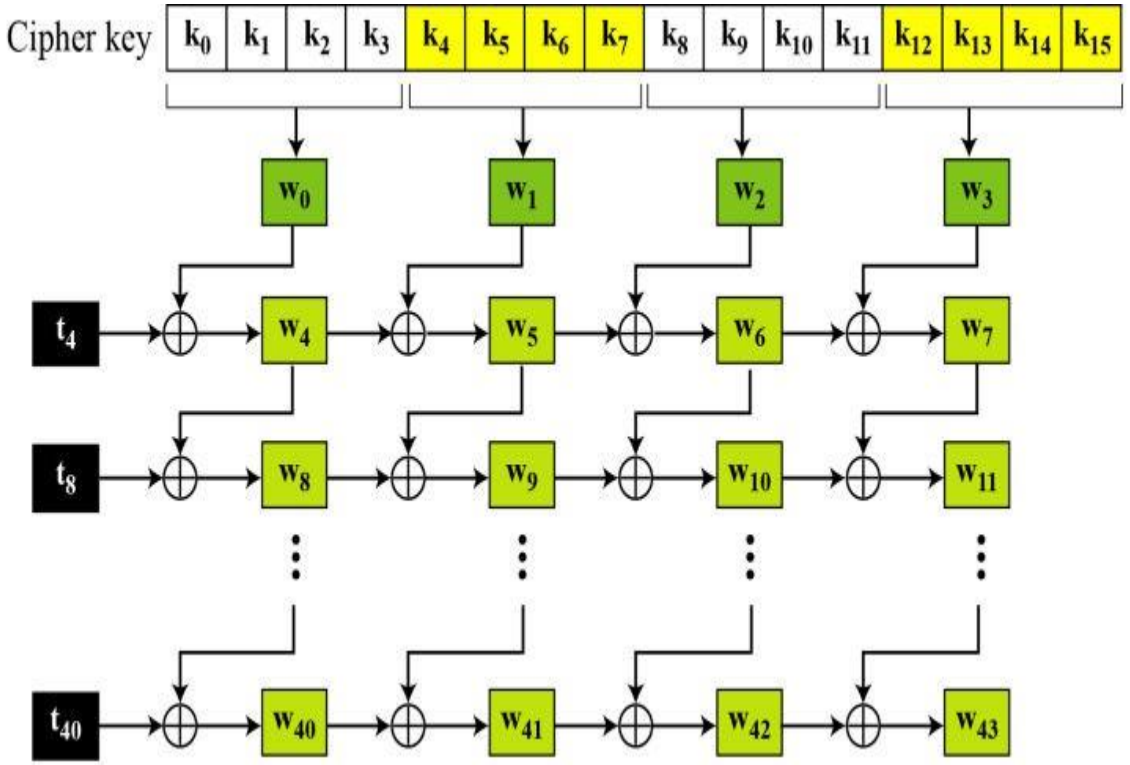
RAMAIAH Institute of Technology

Key Expansion

Table 7.5 shows how the keys for each round are calculated assuming that the 128-bit cipher key agreed upon by Alice and Bob is (24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87)$_{16}$.

Table 7.5 *Key expansion example*

Round	*Values of t'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



Key Expansion

Table 7.5 shows how the keys for each round are calculated assuming that the 128-bit cipher key agreed upon by Alice and Bob is (24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87)$_{16}$.

Table 7.5 *Key expansion example*

Round	*Values of t'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

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

$t_i = \text{subword}(\text{Rotword}(w_{i-1})) \oplus \text{Rcon}_{i/4}$

$t_4 = \text{subword}(\text{Rotword}(w_{4-1})) \oplus \text{Rcon}_{4/4}$

$t_4 = \text{subword}(\text{Rotword}(w_3)) \oplus \text{Rcon}_1$

Rotword(13AA5487) = AA548713

Subword (AA548713) = AC20177D

$t_4 = \text{AC20177D} \oplus \text{Rcon}_1$

$= \text{AC20177D} \oplus$ **01 00 00 00 ➔ AD20177D**

Key Expansion –AES 192 and AES 256

Key-expansion algorithms in the AES-192 and AES-256 versions are very similar to the key expansion algorithm in AES-128, with the following differences:

1. In AES-192, the words are generated in groups of six instead of four.
   a. The cipher key creates the first six words (w_0 to w_5).
   b. If $i \bmod 6 \neq 0$, $w_i \leftarrow w_{i-1} + w_{i-6}$; otherwise, $w_i \leftarrow t + w_{i-6}$.
2. In AES-256, the words are generated in groups of eight instead of four.
   a. The cipher key creates the first eight words (w_0 to w_7).
   b. If $i \bmod 8 \neq 0$, $w_i \leftarrow w_{i-1} + w_{i-8}$; otherwise, $w_i \leftarrow t + w_{i-8}$.
   c. If $i \bmod 4 = 0$, but $i \bmod 8 \neq 0$, then $w_i = \text{SubWord}(w_{i-1}) + w_{i-8}$.

Key expansion analysis

The key-expansion mechanism in AES has been designed to provide several features that thwart the cryptanalyst.

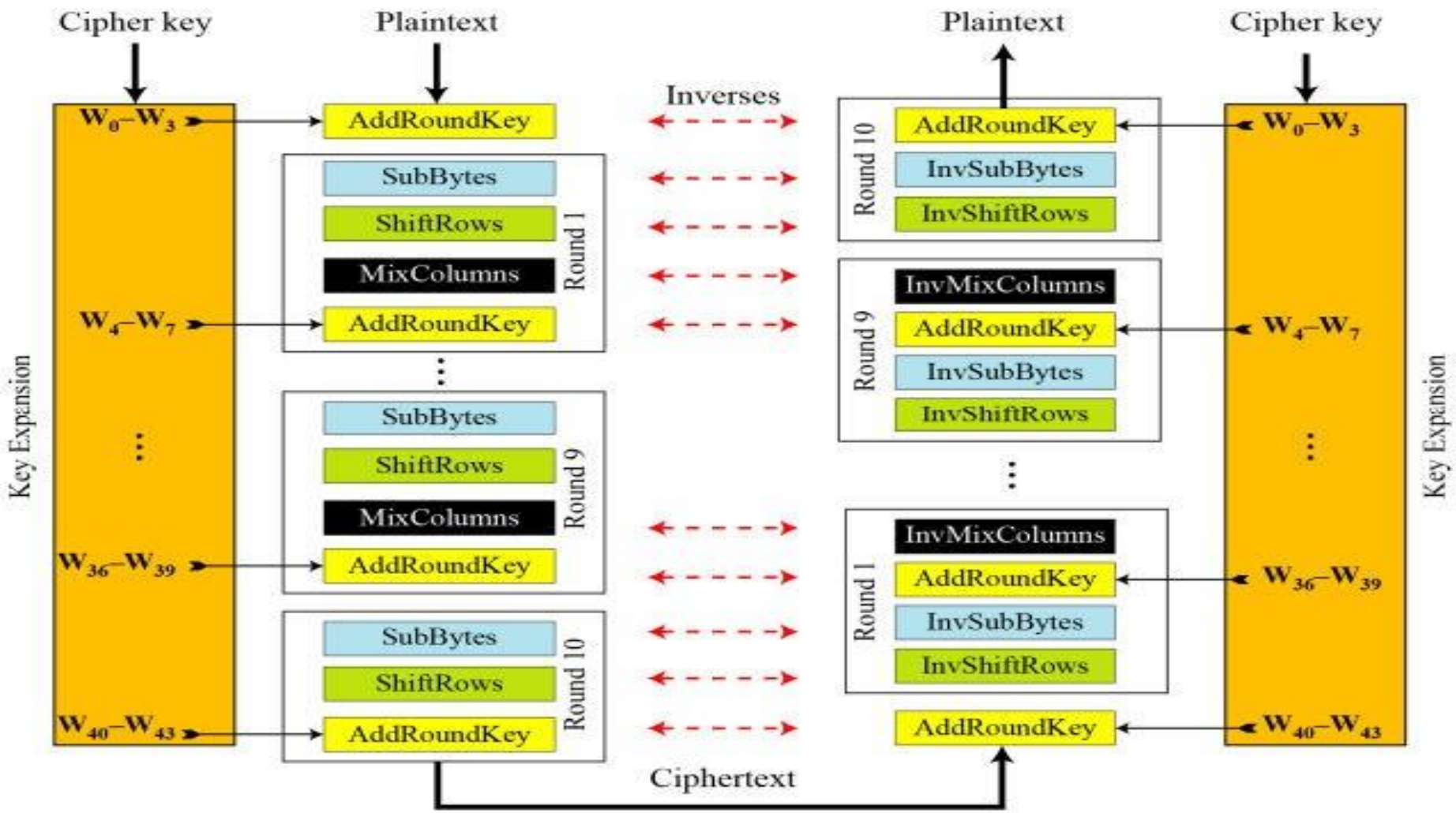
1. Even if Eve knows only part of the cipher key or the values of the words in some round keys, she still needs to find the rest of the cipher key before she can find all round keys. This is because of the nonlinearity produced by SubWord transformation in the key-expansion process.
2. Two different cipher keys, no matter how similar to each other, produce two expansions that differ in at least a few rounds.
3. Each bit of the cipher key is diffused into several rounds. For example, changing a single bit in the cipher key, will change some bits in several rounds.
4. The use of the constants, the RCons, removes any symmetry that may have been created by the other transformations.
5. There are no serious weak keys in AES, unlike in DES.
6. The key-expansion process can be easily implemented on all platforms.
7. The key-expansion routine can be implemented without storing a single table; all calculations can be done using the $GF(2^8)$ and FG(2) fields.

The AES Ciphers

- AES uses four types of transformations for encryption and decryption.
- Encryption algorithm is referred to as the cipher
- Decryption algorithm as the inverse cipher.
- Two different design for implementation
  - Original Design
  - Alternative Design

RAMAIAH
Institute of Technology

The AES Ciphers – Original design



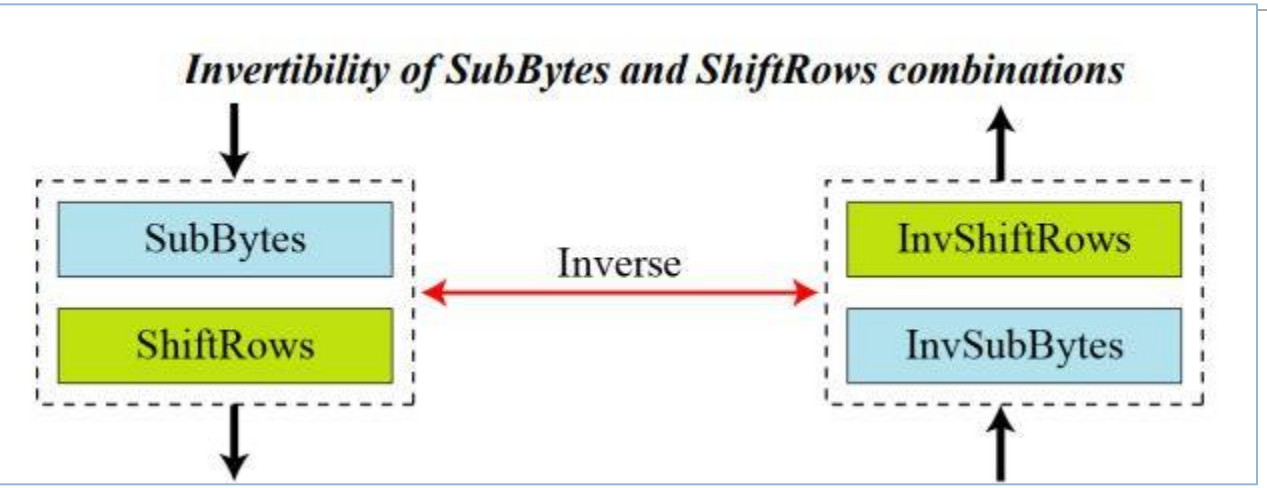
The AES Ciphers –Original design

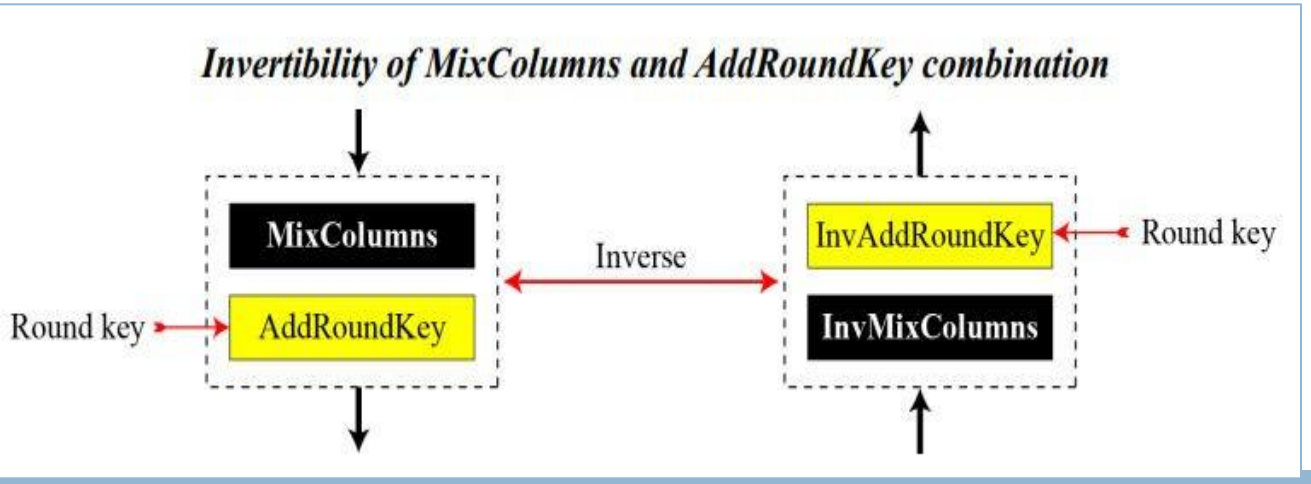
The code for the AES-128 version of this design is shown in Algorithm

Pseudocode for cipher in the original design

```
Cipher (InBlock [16], OutBlock[16], w[0 … 43])
{
   BlockToState (InBlock, S)

   S ← AddRoundKey (S, w[0…3])
    for (round = 1 to 10)
    {
        S ← SubBytes (S)
        S ← ShiftRows (S)
         if (round ≠ 10)  S ← MixColumns (S)
         S ← AddRoundKey (S, w[4 × round, 4 × round + 3])
    }

  StateToBlock (S, OutBlock);
}
```

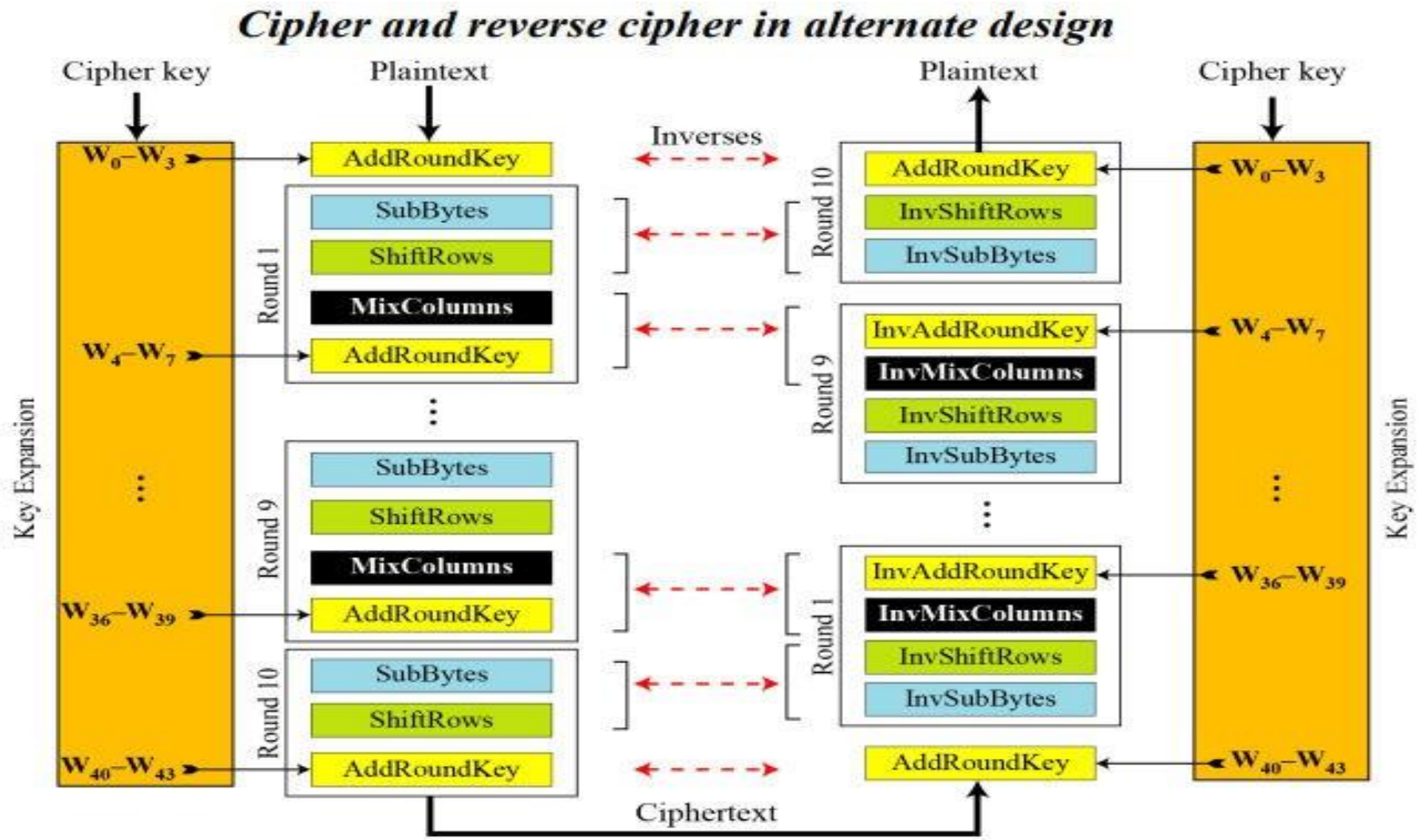
The AES Ciphers –Alternative Design

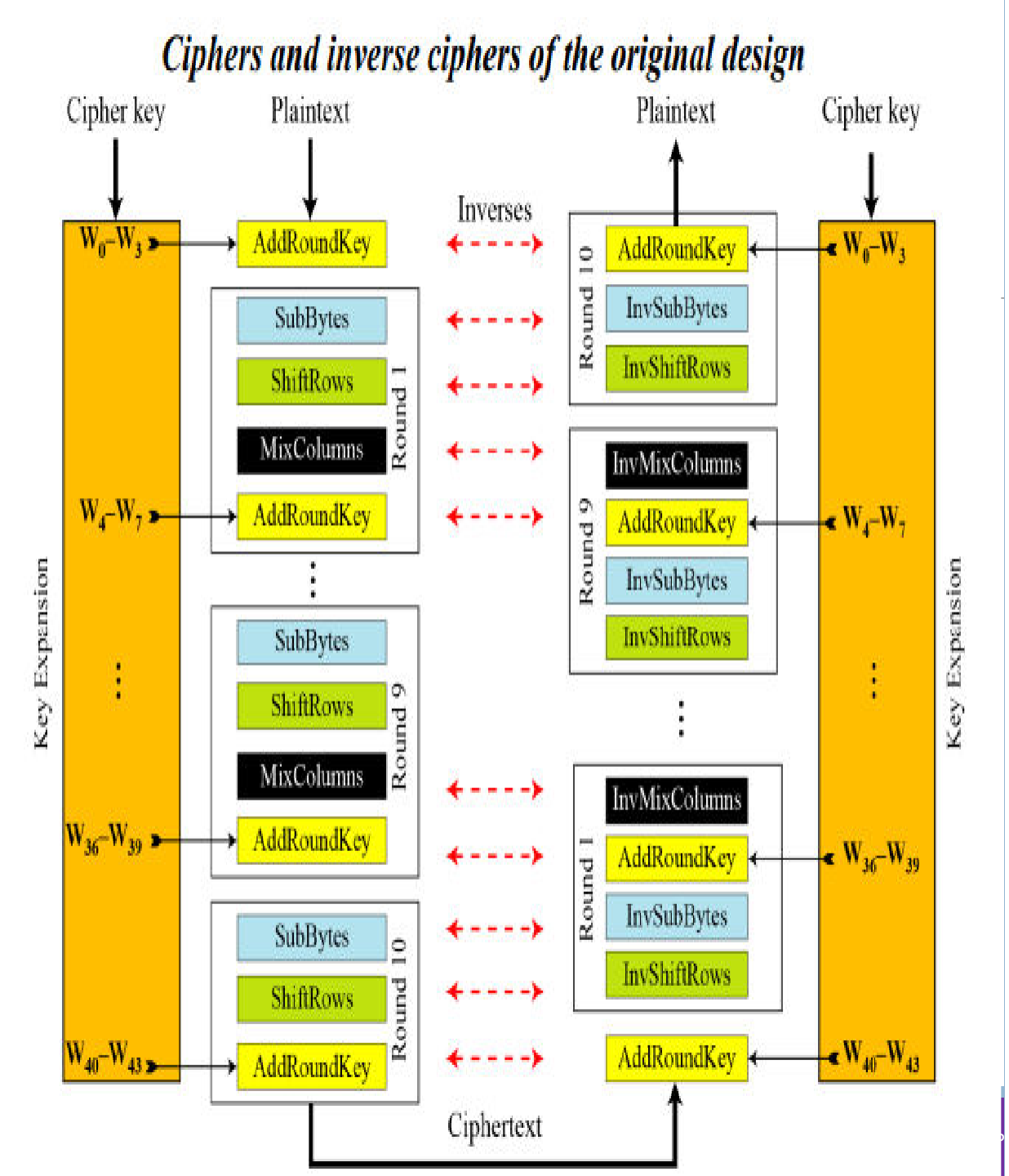


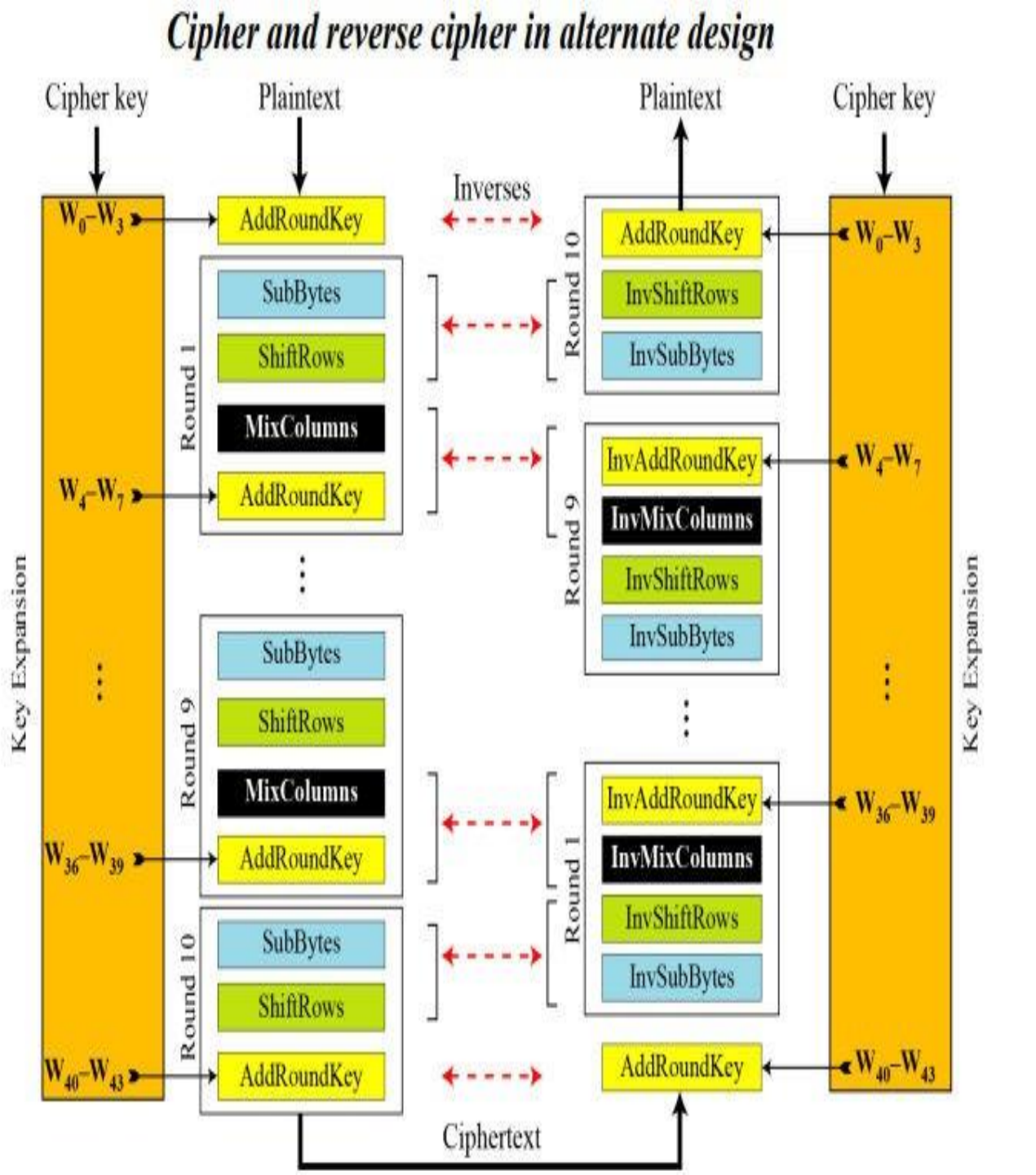


RAMAIAH
Institute of Technology

The AES Ciphers –Alternative Design




Ciphers and inverse ciphers of the original design
Cipher key
Plaintext
Plaintext
Cipher key
Inverses
Key Expansion
Key Expansion
W0–W3
W4–W7
W36–W39
W40–W43
AddRoundKey
SubBytes
ShiftRows
MixColumns
Round 1
Round 9
Round 10
InvMixColumns
InvSubBytes
InvShiftRows
Ciphertext


Cipher and reverse cipher in alternate design
Cipher key
Plaintext
Plaintext
Cipher key
Inverses
Key Expansion
Key Expansion
W0–W3
W4–W7
W36–W39
W40–W43
AddRoundKey
SubBytes
ShiftRows
MixColumns
InvAddRoundKey
InvMixColumns
InvShiftRows
InvSubBytes
Round 1
Round 9
Round 10
Ciphertext

Analysis of AES

This section is a brief review of the three characteristics of AES.

- Security
- Implementation
- Simplicity and Cost

Analysis of AES

Security

- 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.

Analysis of AES

Implementation

AES can be implemented in software, hardware, and firmware.

The implementation can use table lookup process or routines that use a well-defined algebraic structure.

Analysis of AES

Simplicity and Cost

The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.