Advanced Encryption Standard (AES)

Outline

- Overview of the AES algorithm
- Internal structure of AES
  - Byte Substitution
  - Shift Rows
  - Mix Columns
  - Add Round Key
  - Key schedule
- Decryption

Overview of the AES algorithm

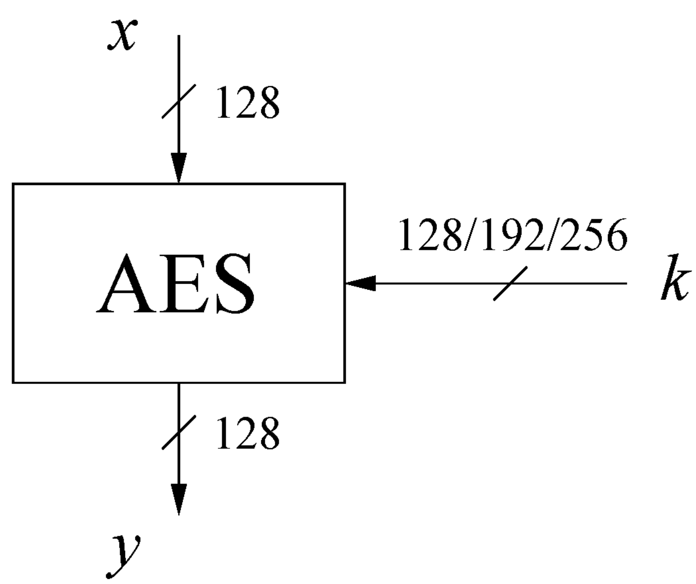
Some Basic Facts

- AES is the most widely used symmetric cipher today
- The algorithm for AES was chosen by the US *National Institute of Standards and Technology* (NIST) in a multi-year selection process
- The requirements for all AES candidate submissions were:
  - Block cipher with **128-bit block size**
  - **Three supported key lengths**: 128, 192 and 256 bit
  - Security relative to other submitted algorithms
  - **Efficiency** in software and hardware implementation

Chronology of the AES Selection

- The need for a new block cipher announced by NIST in January, 1997
- 15 candidates algorithms accepted in August, 1998
- 5 finalists announced in August, 1999:
  - *Mars* – IBM Corporation
  - *RC6* – *RSA* Laboratories
  - *Rijndael* – J. Daemen & V. Rijmen
  - *Serpent* – Eli Biham et al.
  - *Twofish* – B. Schneier et al.
- In October 2000, *Rijndael* was chosen as the AES
- AES was formally approved as a US federal standard in November 2001

AES Overview

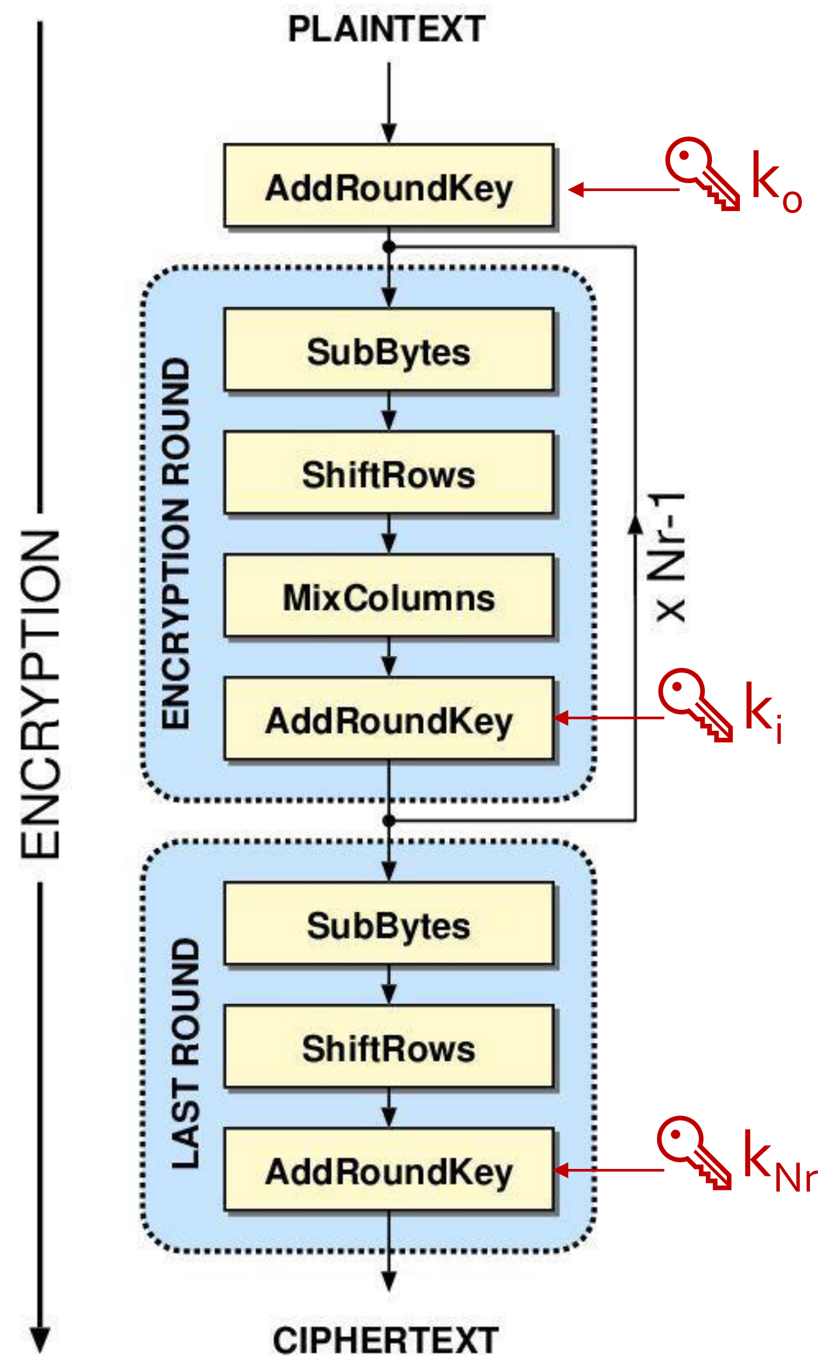


The number of rounds depends on the chosen key length:

Key length (bits)	Number of rounds
128	10
192	12
256	14

AES Overview

- An **iterative** rather than Feistel cipher
- Operates on entire data block in every round
- 10/12/14 rounds depending on the key size.
- Each round consists of Confusion and Diffusion operations
- Note: In the last round, the MixColumns tansformation is omitted



Internal structure of AES

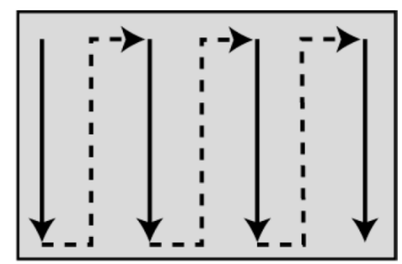
Block to state

- AES is a byte-oriented cipher
- State = Block of bytes that are currently being worked on
- Arranged in 4 x 4 Matrix of **bytes**

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}

Block

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



Insertion and extraction flow

with $b_0, \ldots, b_{15}$ denoting the **16-byte** input of AES arranged in a 4x4 matrix

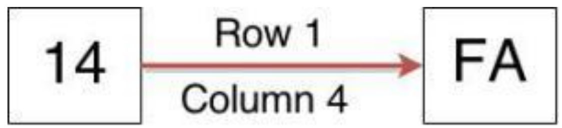
Block to state - example

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

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

SubBytes = Byte Substitution

- Each value of the state is replaced with the corresponding S-Box value => bytewise S-Box substitution
- E.g. HEX 14 would get replaced with HEX FA



	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

SubBytes Example

$$\text{State}\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \xrightarrow{\boxed{\text{SubByte}}} \begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & F2 & 63 & 26 \\ C9 & C9 & 7D & D4 \\ FA & 63 & 82 & D4 \end{bmatrix}\text{State}$$

Shift Rows

- Performs **Left Circular Shift** of the state matrix row:
- This is not a bit wise shift. The circular shift just moves each byte one space over.

Input matrix

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}

Output matrix

B_0	B_4	B_8	B_{12}	no shift
B_5	B_9	B_{13}	B_1	← one position left shift
B_{10}	B_{14}	B_2	B_6	← two positions left shift
B_{15}	B_3	B_7	B_{11}	← three positions left shift

MixColumns

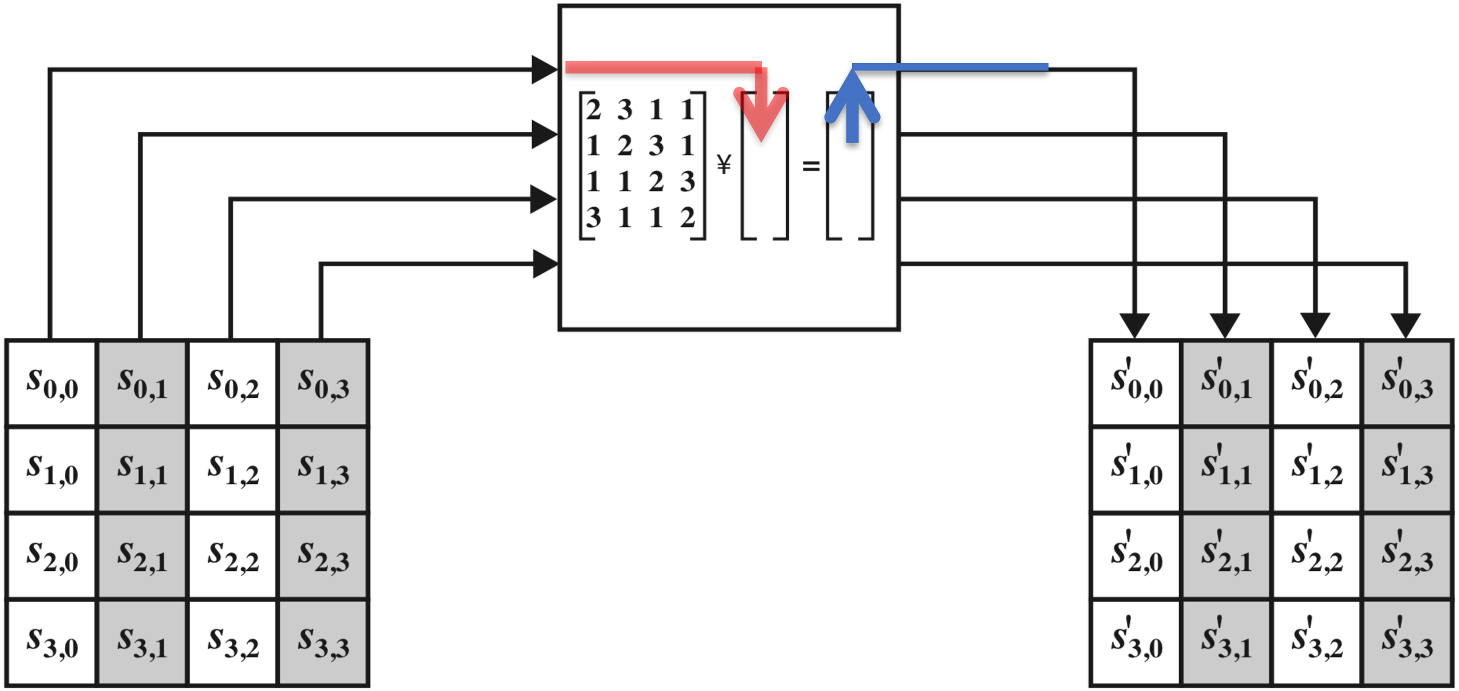
- The MixColumns transformation operates at the column level. It transforms each column of the state to a new column in the next state
- Each 4-byte column is considered as a vector and multiplied by a fixed 4x4 matrix, e.g.,

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\begin{aligned} d_0 &= 2 \bullet b_0 \oplus 3 \bullet b_1 \oplus 1 \bullet b_2 \oplus 1 \bullet b_3 \\ d_1 &= 1 \bullet b_0 \oplus 2 \bullet b_1 \oplus 3 \bullet b_2 \oplus 1 \bullet b_3 \\ d_2 &= 1 \bullet b_0 \oplus 1 \bullet b_1 \oplus 2 \bullet b_2 \oplus 3 \bullet b_3 \\ d_3 &= 3 \bullet b_0 \oplus 1 \bullet b_1 \oplus 1 \bullet b_2 \oplus 2 \bullet b_3 \end{aligned}$$

MixColumns Transformation

$$s'_{0,0} = 2 \cdot s_{0,0} + 3 \cdot s_{1,0} + 1 \cdot s_{2,0} + 1 \cdot s_{3,0}$$



- The MixColumns transformation operates at the column level. It transforms each column of the state to a new column.
- Each 4-byte column is considered as a vector and multiplied by a fixed 4x4 matrix.

Add Round Key

- **State matrix ⊕ Round key matrix**

- Inputs:
  - 16-byte state matrix *C*
  - 16-byte subkey k_i

- Output: $C \oplus k_i$

- The round keys are generated by the key schedule

AES Key Scheduling

- Subkeys are derived recursively from the original 128/192/256-bit input key

- Each round has 1 subkey, plus 1 subkey at the beginning of AES

Key length (bits)	Number of subkeys
128	11
192	13
256	15

AES Key Scheduling

- Takes 128-bits (16-bytes) key and **expands** into array of 44 32-bit words
- 11 subkeys are stored in W[0]…W[3], W[4]…W[7], … , W[40]…W[43]

Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
. . .	. . .			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

AES Key Expansion

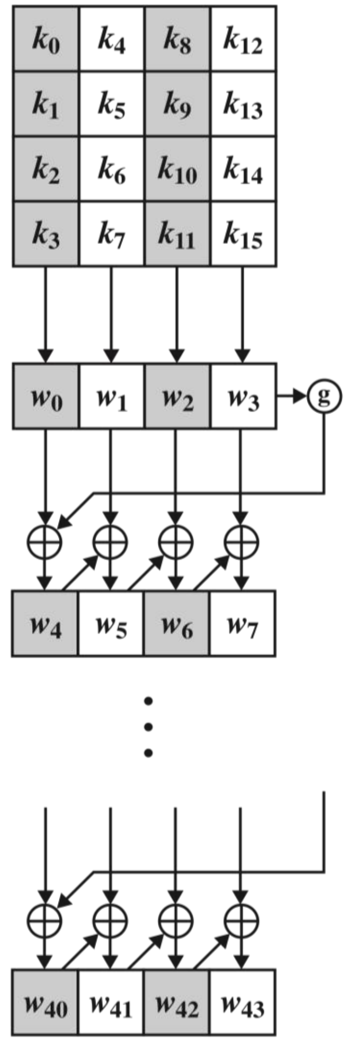
- First subkey *W[0]…W[3]* is the original AES key
- Constructing subsequent groups of 4 words based on the *Previous Word* (W_{i-1}) & *4th back Word* (W_{i-4})

$$W_i = W_{i-1} \oplus W_{i-4}$$

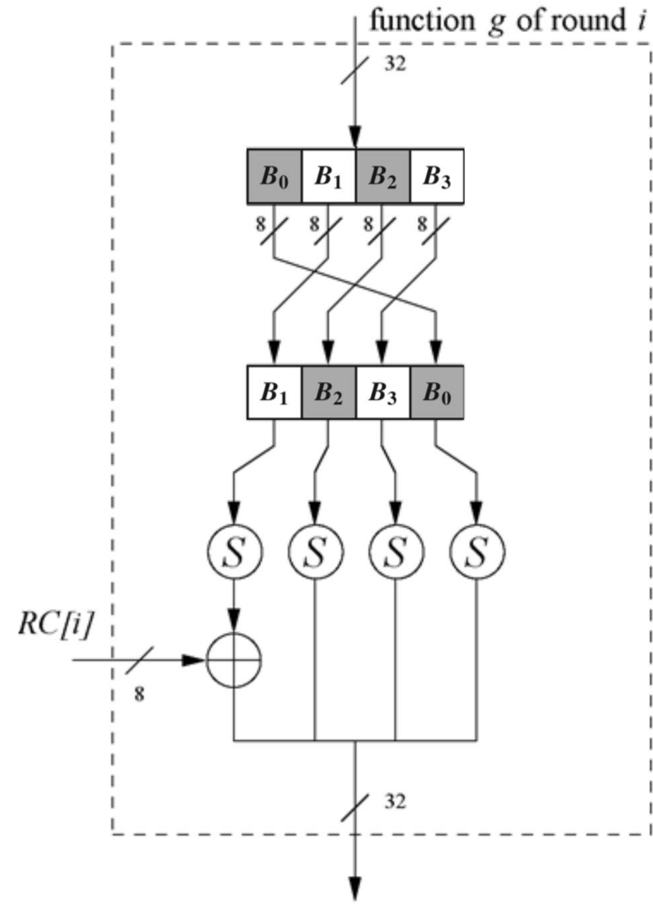
For all values of i that are not multiples of 4.

- **1st word** in each group gets a "special treatment" using function **g** before XOR'ing the *4th back Word* (W_{i-4})

$$W_i = g(W_{i-1}) \oplus W_{i-4}$$



(a) Overall algorithm



(b) Function g

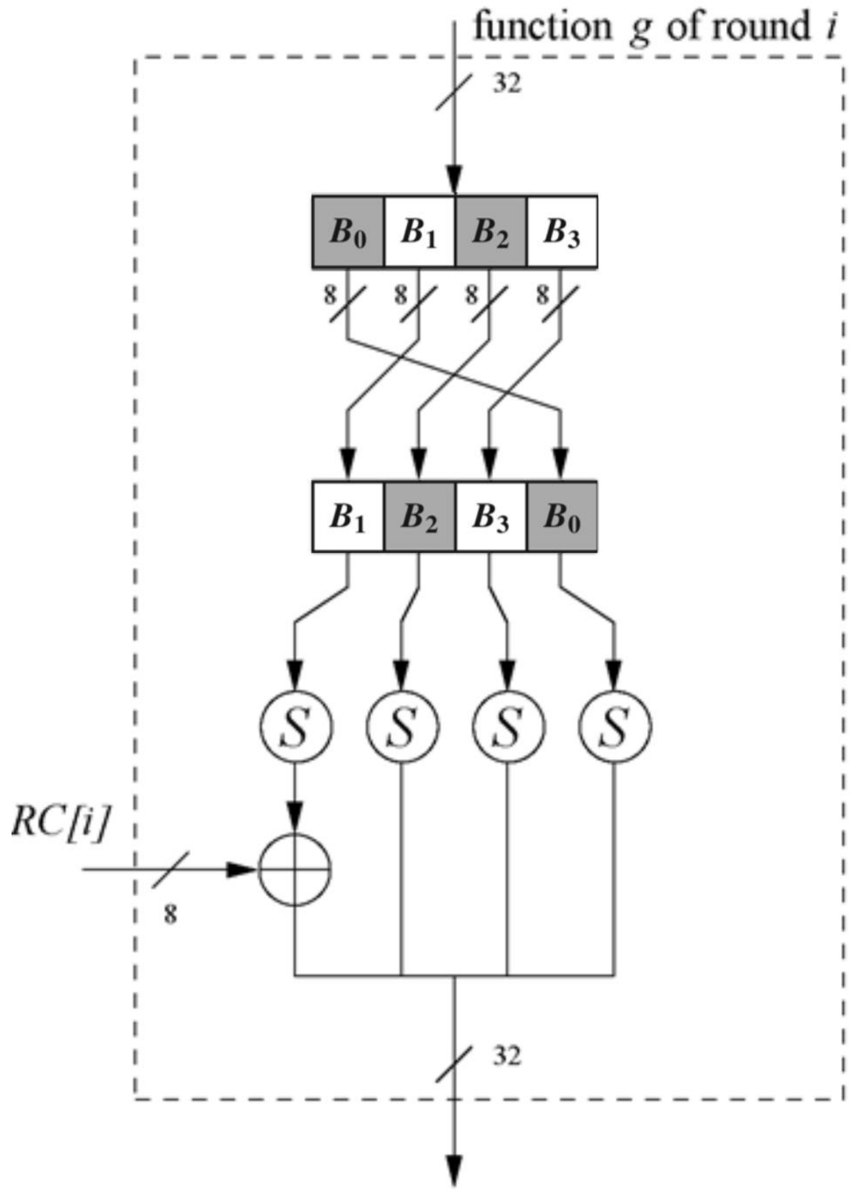
Rotate -> S-box -> XOR a constant

Key Expansion - 1st Word “special treatment”

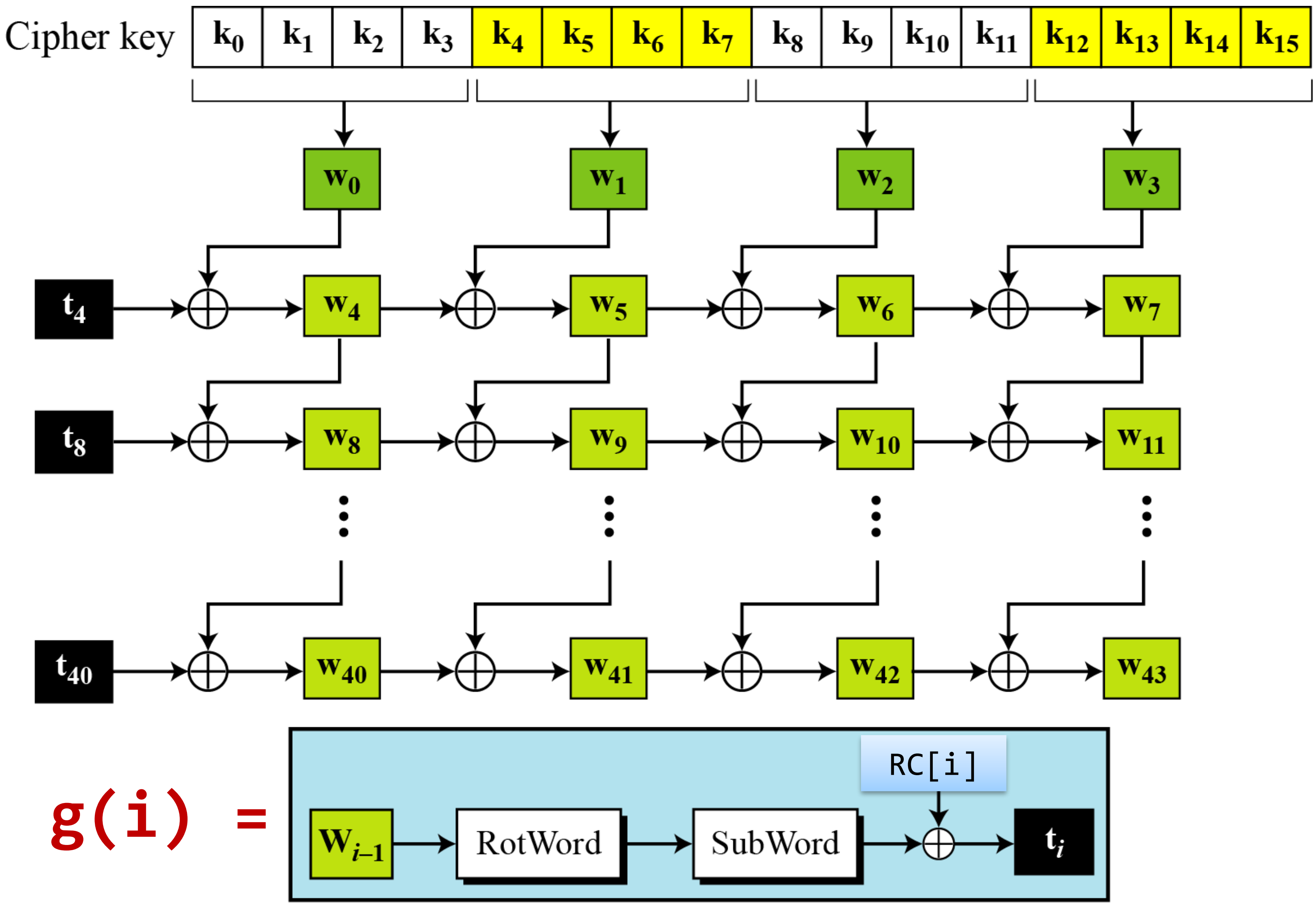
- Function *g* rotates its four input bytes and performs a **bytewise** S-Box substitution

- Leftmost byte is XORed with a **Round Constant** (RC):

Rcon Constants (Base 16)			
Round	Constant(Rcon)	Round	Constant(Rcon)
1	01 00 00 00	6	20 00 00 00
2	02 00 00 00	7	40 00 00 00
3	04 00 00 00	8	80 00 00 00
4	08 00 00 00	9	1B 00 00 00
5	10 00 00 00	10	36 00 00 00



Key Expansion Scheme – Another View



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

Example - First Roundkey

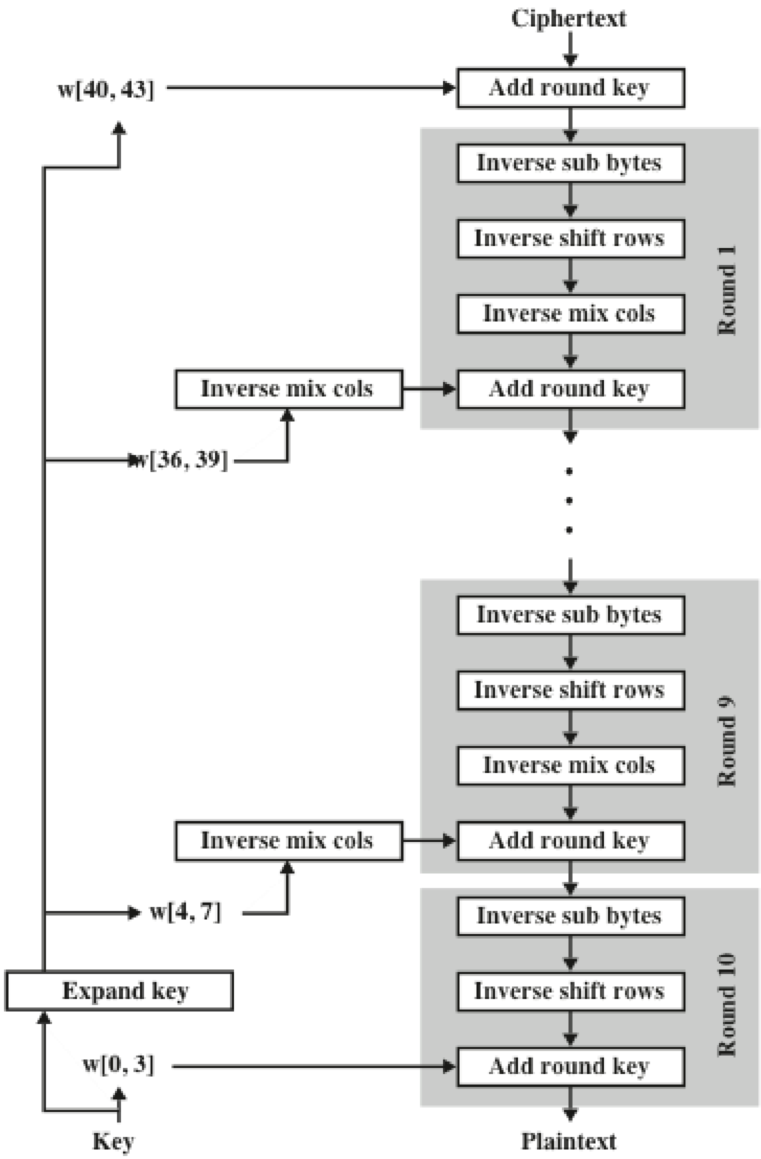
- Key in Hex (128 bits): 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75
- $w[0] = (54, 68, 61, 74), w[1] = (73, 20, 6D, 79), w[2] = (20, 4B, 75, 6E), w[3] = (67, 20, 46, 75)$
- $g(w[3])$:
  - circular byte left shift of $w[3]$: $(20, 46, 75, 67)$
  - Byte Substitution (S-Box): $(B7, 5A, 9D, 85)$
  - Adding round constant $(01, 00, 00, 00)$ gives: $g(w[3]) = (B6, 5A, 9D, 85)$
- $w[4] = w[0] \oplus g(w[3]) = (E2, 32, FC, F1)$:

0101 0100	0110 1000	0110 0001	0111 0100
1011 0110	0101 1010	1001 1101	1000 0101
1110 0010	0011 0010	1111 1100	1111 0001
E2	32	FC	F1

- $w[5] = w[4] \oplus w[1] = (91, 12, 91, 88)$, $w[6] = w[5] \oplus w[2] = (B1, 59, E4, E6)$, $w[7] = w[6] \oplus w[3] = (D6, 79, A2, 93)$
- first roundkey: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

Decryption

Decryption



- AES is not based on a Feistel network

$\Rightarrow$ AES decryption is not identical to encryption. But each step must be inverted for decryption:

- ShiftRows → **Inv ShiftRows**
- MixColumn → **Inv MixColumn**
- Byte Substitution → **Inv Byte Substitution**
- Key Addition uses XOR
- Subkeys are needed in reversed order

Inv ShiftRows

- All rows of the state matrix *B* are shifted to the opposite direction:

Input matrix

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}

Output matrix

B_0	B_4	B_8	B_{12}	no shift
B_{13}	B_1	B_5	B_9	$\rightarrow$ one position right shift
B_{10}	B_{14}	B_2	B_6	$\rightarrow$ two positions right shift
B_7	B_{11}	B_{15}	B_3	$\rightarrow$ three positions right shift

Inv MixColumn

- The MixColumns operation has the following inverse (numbers are decimal):

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Or:

$$b_0 = 14 \bullet d_0 \oplus 11 \bullet d_1 \oplus 13 \bullet d_2 \oplus 9 \bullet d_3$$
$$b_1 = 9 \bullet d_0 \oplus 14 \bullet d_1 \oplus 11 \bullet d_2 \oplus 13 \bullet d_3$$
$$b_2 = 13 \bullet d_0 \oplus 9 \bullet d_1 \oplus 14 \bullet d_2 \oplus 11 \bullet d_3$$
$$b_3 = 11 \bullet d_0 \oplus 13 \bullet d_1 \oplus 9 \bullet d_2 \oplus 14 \bullet d_3$$

InvSubByte

- During decryption each value in the state is replaced with the corresponding inverse of the S-Box
- For example HEX D4 would get replaced with HEX 19

		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	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

AES Security

- **Brute-force attack:** Due to the key length of 128, 192 or 256 bits, a brute-force attack is not possible
- **Analytical attacks:** There is no known analytical attack.
- **Side-channel attacks:**
  - Several side-channel attacks have been published
  - Note that side-channel attacks do not attack the underlying algorithm but its implementation

Summary

- AES is a modern block cipher which supports three key lengths of 128, 192 and 256 bit. It provides excellent long-term security against brute-force attacks.
- AES has been studied intensively since the late 1990s and no attacks have been found.
- AES is not based on Feistel networks. It is an iterative cipher. Each round = 4 steps of SubBytes, ShiftRows, MixColumns, and AddRoundKey (last round no MixColumns)
- Decryption is not the same as encryption (as in DES). Decryption consists of inverse steps.
- AES is efficient in software and hardware.
- It seems likely that the cipher will be the dominant encryption algorithm for many years to come.

Resources

- Crypto Tool 2

https://www.cryptool.org/en/cryptool2

- AES Wikipedia page

https://en.wikipedia.org/wiki/Advanced_Encryption_Standard