Symmetric Ciphers II

Computer and Network Security

Emilio Coppa

Advanced Encryption Standard (AES)

- AES is the most widely used symmetric cipher today

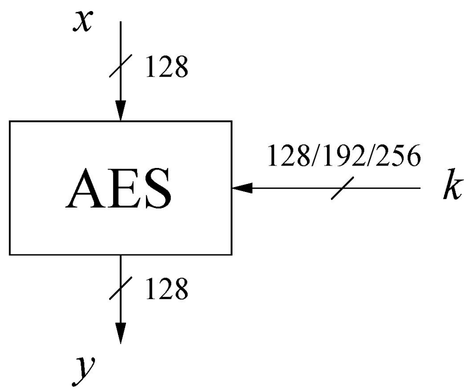
- The algorithm for AES was chosen by the US *National Institute of Standards and Technology* (NIST) in a multi-year selection process.

- 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

Chronology of the AES Selection

- Open call 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.
  NSA allows to use AES with 192/256 bit key.

AES: overview

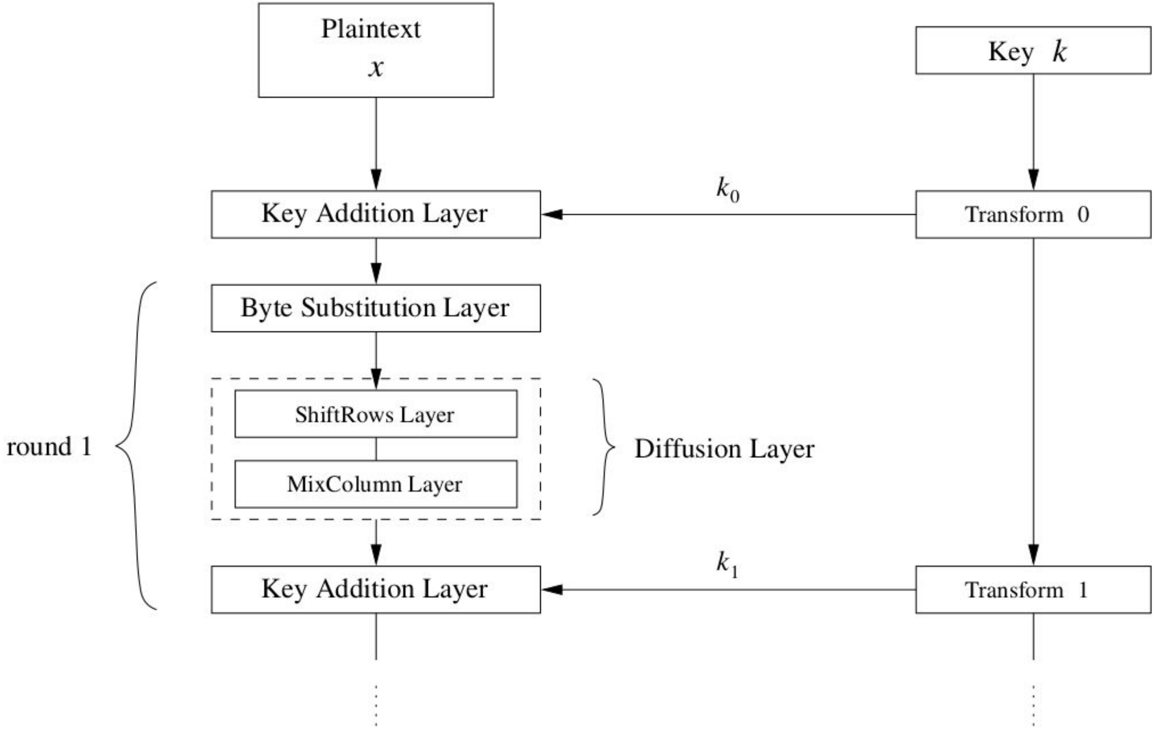


The number of rounds depends on the chosen key length:

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

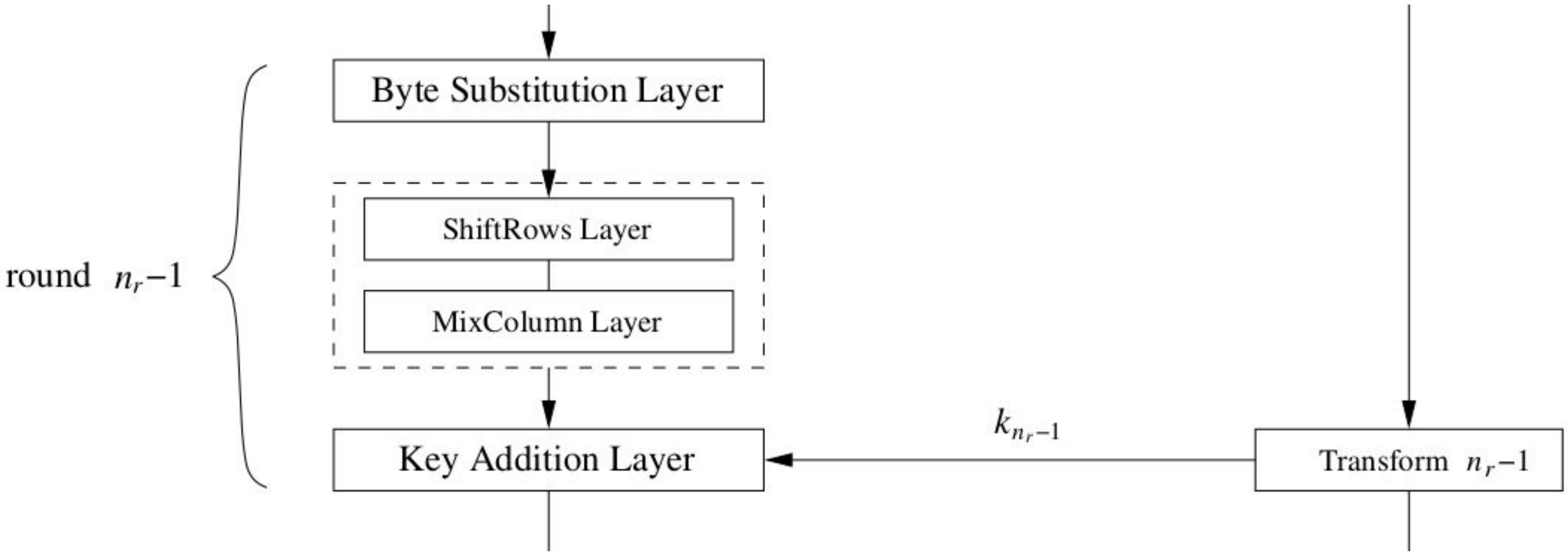
AES: Overview

Each round consists of different “layers”



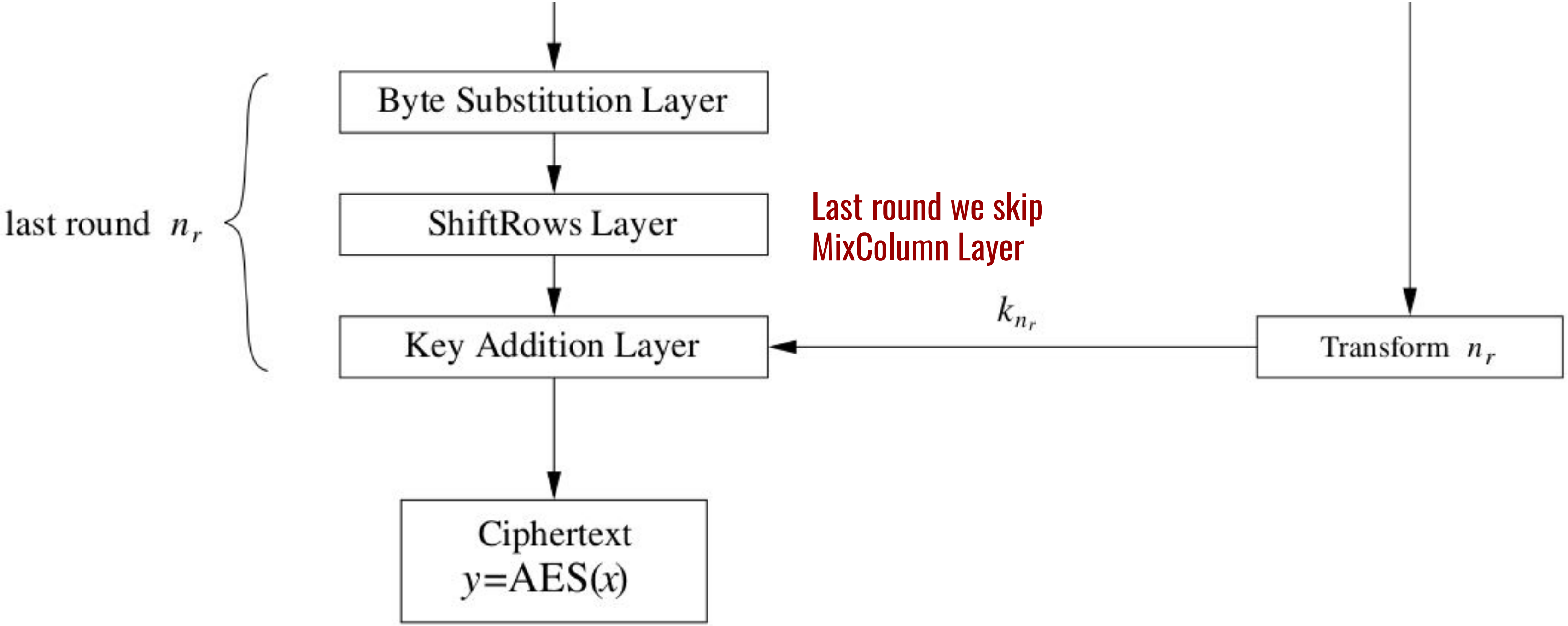
Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

AES: Overview (2)



Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

AES: Overview (3)



Last round we skip
MixColumn Layer

Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

AES: Layers

Each round consists of four main layers:

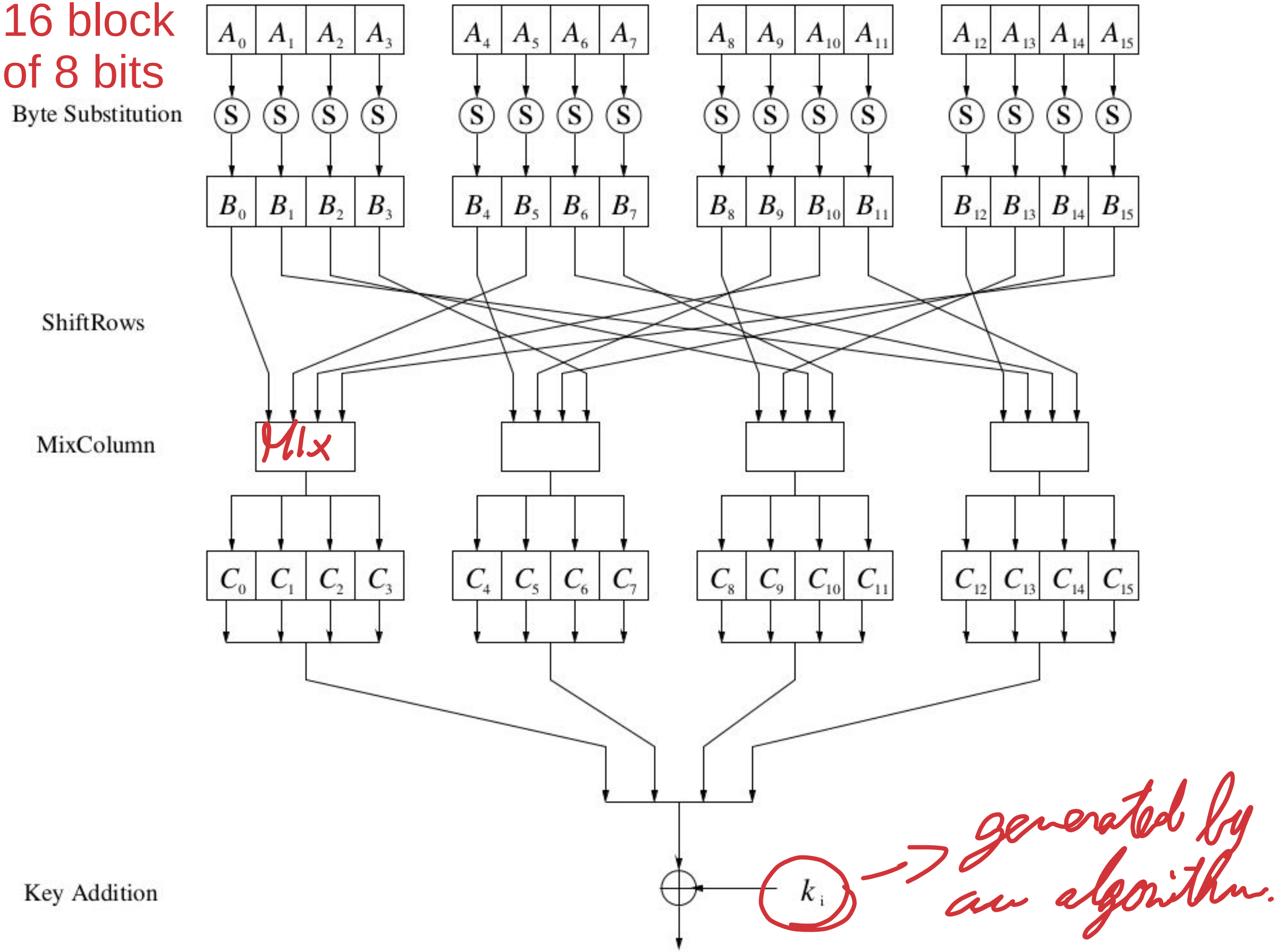
1. ByteSub ⇒ CONFUSION
2. ShiftRow
3. MixColumn ⇒ DIFFUSION
4. Key Addition ⇒ KEY WHITENING

Last round does not have MixColumn layer.

AES: Layers (2)

16 block
of 8 bits

Visually:



Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

Internal Structure of AES

- AES is a byte-oriented cipher. It is not based on Feistel network (as DES), but on a substitution-permutation network
- The state A (i.e., the 128-bit data path) can be arranged in a 4x4 matrix:

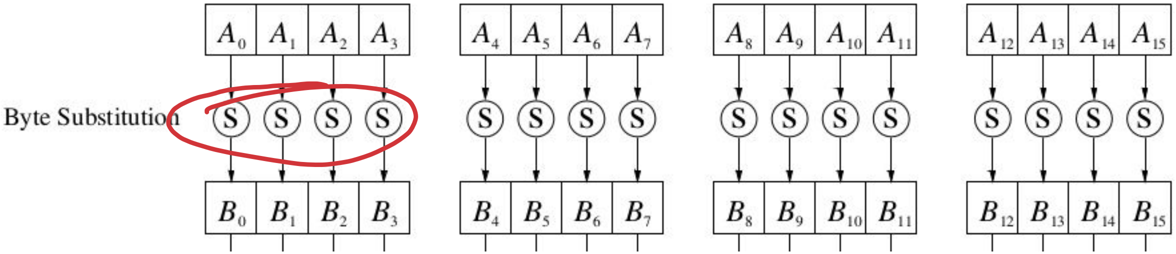
A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

with $A_0, \ldots, A_{15}$ denoting the 16-byte input of AES

Byte Substitution Layer

1st Step

Independent by every byte.



Confusion: if you flip one bit in A_i, it will affect on average 3 or 4 bits in B_i

- The Byte Substitution layer consists of 16 S-Boxes with the following properties:
- The S-Boxes are
  - identical
  - the only nonlinear elements of AES, i.e., $\mathrm{ByteSub}(A_i) + \mathrm{ByteSub}(A_j) \neq \mathrm{ByteSub}(A_i + A_j)$
  - bijective, i.e., there exists a one-to-one mapping of input and output bytes ⇒ S-Box can be uniquely reversed
- In sw implementations, the S-Box is usually realized as a lookup table

Byte Substitution Layer (2)

$$S(A_i) = B_i$$

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

E.g., Using AES S-Box:

$$S(\texttt{0xC2}) = \texttt{0x25}$$

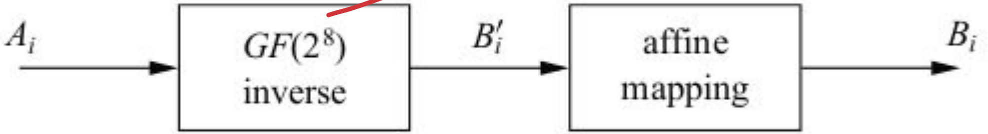
Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

Byte Substitution Layer (3)

Multiplication inverse of galois field

Q. How the S-Box has been built?

A. The S-Box is designed to perform two operations:



Each A_i (8 bit) is seen as an element in GF(2^8):

$$A_i = 1100\,0010 \Rightarrow A_i(x) = x^7 + x^6 + x$$

The first step computes the inverse (which provides the non linearity in AES):

$$B_i'(x) = A(x)^{-1}$$

$$P(x) = x^8 + x^4 + x^3 + x + 1$$

AES irreducible polynomial

such that: $B_i'(x) \cdot A(x)^{-1} \equiv 1 \bmod P(x)$

Byte Substitution Layer (4)

E.g., $A_i = 1100\,0010 \Rightarrow A_i(x) = x^7 + x^6 + x$

$$B_i'(x) = A(x)^{-1} = x^5 + x^3 + x^2 + x + 1$$ (computed with EEA)

The second step computed in the S-Box is an affine mapping (this is done to destroy some algebraic properties that could be exploited by an attacker):

$$\begin{array}{cccccc} B_i(x) & & & B_i'(x) & & \\ \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} & \equiv & \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} & \begin{pmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{pmatrix} & + & \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \bmod 2. \end{array}$$

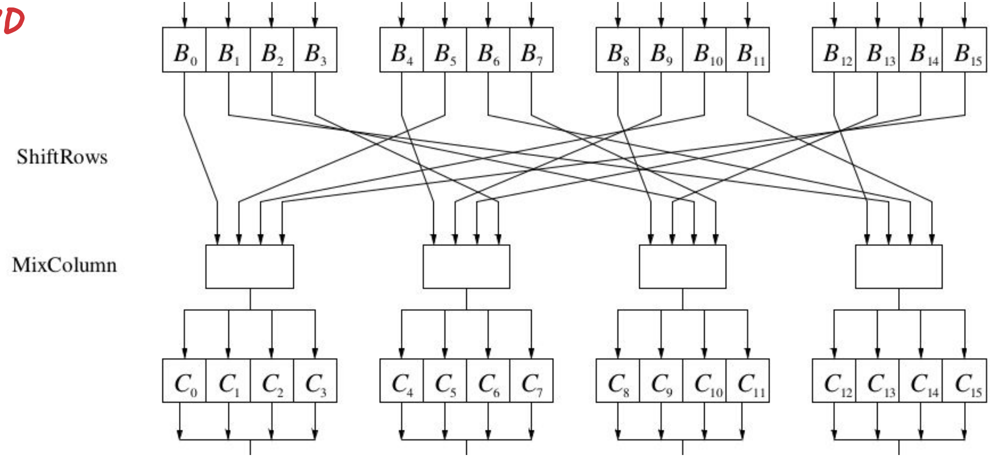
Byte Substitution Layer (5)

Hence, the S-Box is a precomputation of the output for all the possibly A(x)

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

C code that can precompute this table.

Diffusion Layer 2ND



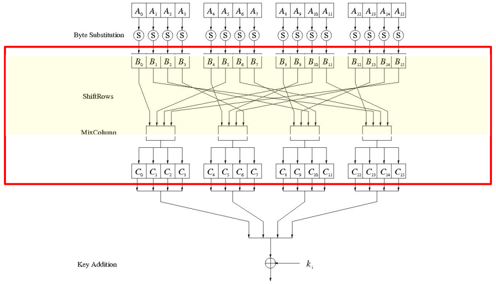
Diffusion: given a byte with some bit flips, it will spread the effect on 32 bits from the state.

- provides diffusion over all input state bits
- consists of two sublayers:
  - ShiftRows Sublayer: Permutation of the data on a byte level
  - MixColumn Sublayer: Matrix operation which combines ("mixes") blocks of four bytes
- performs a linear operation on state matrices A, B, i.e.,

$$\mathrm{DIFF}(A) + \mathrm{DIFF}(B) = \mathrm{DIFF}(A + B)$$

Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

ShiftRows Sublayer



Rows of the state matrix are shifted cyclically:

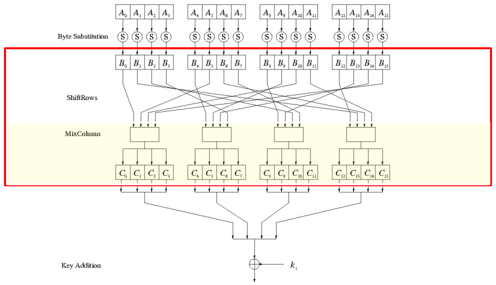
Input matrix

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

MixColumn Sublayer



- Linear transformation which mixes each column of the state matrix
- Each 4-byte column is considered as a vector and multiplied by a fixed 4x4 matrix, e.g., the leftmost mix column box is:

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

  - where 01, 02 and 03 are given in hexadecimal notation
  - each row of the matrix is a shift of the previous row
- All arithmetic is done in the Galois field GF(2^8): e.g.,

$$C_0 = \overset{x}{\underbrace{02}_{\text{byte}}} \cdot \underbrace{B_0}_{\text{byte}} + \overset{x+1}{03} \cdot B_5 + \overset{1}{01} \cdot B_{10} + \overset{1}{01} \cdot B_{15}$$

MixColumn Sublayer (2)

E.g.,
$$C_0 = 02 \cdot B_0 + 03 \cdot B_5 + 01 \cdot B_{10} + 01 \cdot B_{15}$$
$$C_0 = x \cdot B_0 + (x+1) \cdot B_5 + 1 \cdot B_{10} + 1 \cdot B_{15}$$

Addition and multiplication are done as seen in GF(2^8)

E.g., B=(25, ..., 25)

$$\begin{aligned} 02 \cdot 25 &= x \cdot (x^5 + x^2 + 1) \\ &= x^6 + x^3 + x, \\ 03 \cdot 25 &= (x+1) \cdot (x^5 + x^2 + 1) \\ &= (x^6 + x^3 + x) + (x^5 + x^2 + 1) \\ &= x^6 + x^5 + x^3 + x^2 + x + 1. \end{aligned}$$

No modular reduction with P(x) is needed since the result have a degree smaller than 8.

MixColumn Sublayer (3)

Then the addition is, e.g.,:

$$
\begin{array}{rcccccccc}
01 \cdot 25 = & & x^5 + & & x^2 + & & 1 \\
01 \cdot 25 = & & x^5 + & & x^2 + & & 1 \\
02 \cdot 25 = & x^6 + & & x^3 + & & x & \\
03 \cdot 25 = & x^6 + & x^5 + & x^3 + & x^2 + & x + & 1 \\
\hline
C_i = & & x^5 + & & x^2 + & & 1,
\end{array}
$$

MixColumn Sublayer (4)

Another way of defining the MixColumn Sublayer is treat each column as four-term polynomial:

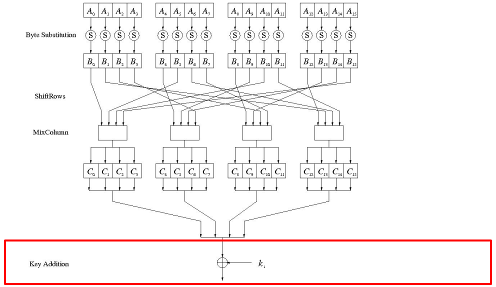
$$b(x) = b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

where each coefficient b_i is in $GF(2^8)$ [this is different from coefficient in GF(2)!] and multiply it by:

$$a(x) = 3x^3 + x^2 + x + 2 \quad \text{modulo} \quad x^4 + 1$$

This is the definition given by the standard and since it is a multiplication with a fixed polynomial can be writte as a matrix multiplication (previous slide).

Key Addition Layer



- Inputs:
  - 16-byte state matrix C
  - 16-byte subkey k_i
- Output: $C \oplus k_i$
- The subkeys are generated in the key schedule

Key Schedule

- 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

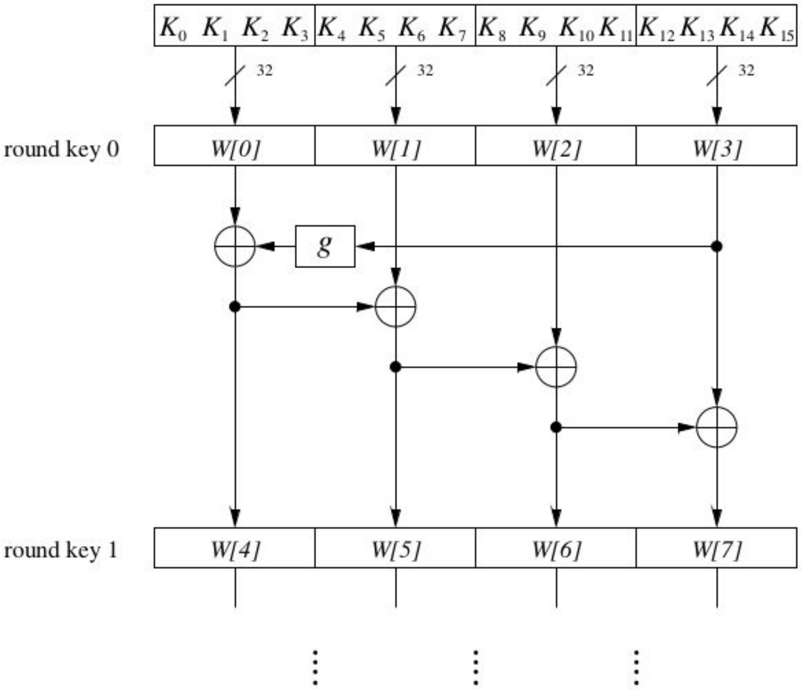
- Key whitening: Subkey is used both at the input and output of AES

$$\Rightarrow \# \text{ subkeys} = \# \text{ rounds} + 1$$

- There are different key schedules for the different key sizes

Key Schedule

Example: Key schedule for 128-bit key AES

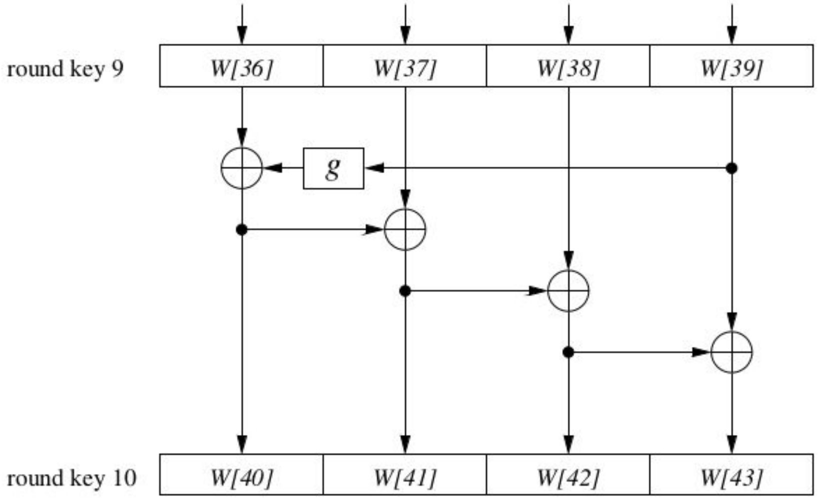


- Word-oriented: 1 word = 32 bits
- 11 subkeys are stored in W[0]…W[3], W[4]…W[7], … , W[40]…W[43]
- First subkey W[0]…W[3] is the original AES key

Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

Key Schedule (2)

Example: Key schedule for 128-bit key AES



Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

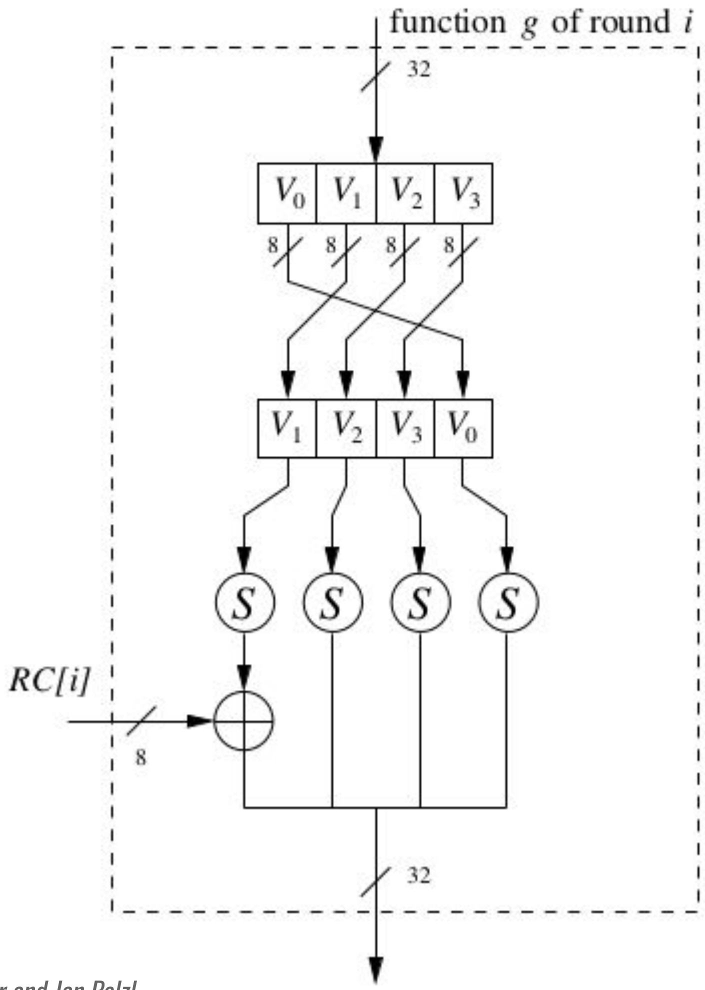
Key Schedule (3)

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

- The round coefficient RC is only added to the leftmost byte and varies from round to round:

  $RC[1] = x^0 = (00000001)_2$
  $RC[2] = x^1 = (00000010)_2$
  $RC[3] = x^2 = (00000100)_2$
  ...
  $RC[10] = x^9 = (00110110)_2$

- x^i represents an element in a Galois field (again, cf. Chapter 4.3 of Understanding Cryptography)



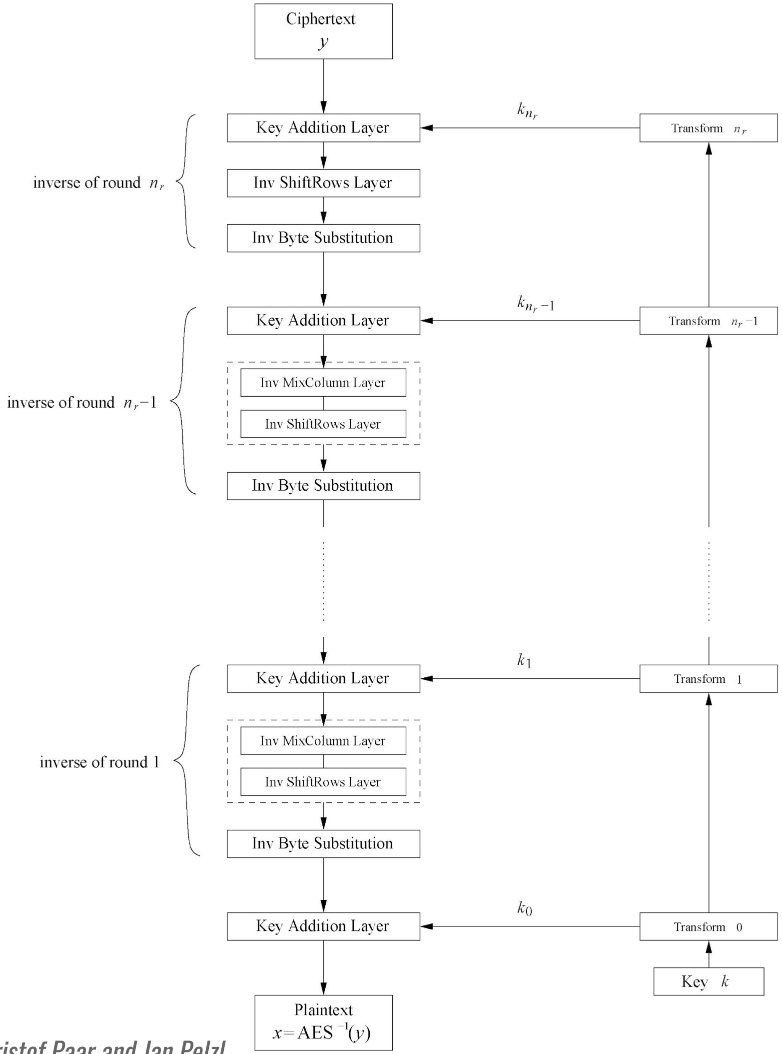
Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

Decryption

- AES is not based on a Feistel network

  $\Rightarrow$ All layers must be inverted for decryption:

  - MixColumn layer $\longrightarrow$ Inv MixColumn layer
  - ShiftRows layer$\longrightarrow$ Inv ShiftRows layer
  - Byte Substitution layer $\longrightarrow$ Inv Byte Substitution layer
  - Key Addition layer is its own inverse



Chapter 4 of Understanding Cryptography by Christof Paar and Jan Pelzl

Credits

These slides are based on material from:

- Slides of Prof. D'Amore from CNS 2019-2020

- Christof Paar and Jan Pelzl. Understanding Cryptography: A Textbook for Students and Practitioners. Springer. http://www.crypto-textbook.com/

- Wikipedia (english version)