Rijndael Algorithm
(Advanced Encryption Standard)
AES

AES selection process

- September 12, 1997: the NIST publicly calls for nominees for the new AES
- 1st AES conference, August 20-23, 1998
  - (15 algorithms are candidates for becoming AES)
- Public Review of the algorithms
- 2nd AES conference, March 22-23, 1999
  - (presentation, analysis and testing)
- August 9, 1999: the 5 finalists are announced
  - (MARS, RC6, RINJDAEL, SERPENT, TWOFISH)
- Public Review
- 3rd AES conferece, April 13-14, 2000
  - (presentation, analysis and testing)

AES selection process

- October 2, 2000: the winner is chosen: RINJDAEL
- February 28, 2001: publication of a Draft by *Federal Information Processing Standard* (FIPS)
- Public Review of 90 days
- Proposal to the *Secretary of Commerce* for approval
- Publication on the *Federal Register*, December 6, 2001,
  - Effective starting from May 26, 2002

Pronunciation: Reign Dahl, Rain Doll, Rhine Dahl

Requirements for AES

- In the selection process, NIST asked for:
  - A block cipher
  - Key length: 128, 192, or 256 bit
  - Block length: 128 bit
  - Possible implementation on smart-cards
  - Royalty-free

- NIST platform used to test candidate cipher algorithms:
  - PC IBM-compatible, Pentium Pro 200 MHz, 64 MB RAM, WINDOWS 95
  - Borland C++ 5.0 compiler, and Java Development Kit (JDK) 1.1

- NIST selection of the winning algorithm based on:
  - Security
  - Efficient implementation *both* in hardware and software
  - Code length and memory utilization

Documentation produced by candidates

- Algorithm Description
- Analysis of the algorithm (advantages/disadvantages)
- Estimation of the computation efficiency
- Algorithm analysis with respect to the best known attacks (e.g. with known or chosen plaintext)
- Implementation in ANSI C
- *Optimized implementation* both in ANSI C and Java

Finalists and candidates for AES

RIJNDAEL	**Joan Daemen, Vincent Rijmen**
MARS	IBM
RC6	RSA Laboratories
SERPENT	R. Anderson, E. Biham, L. Knudsen
TWOFISH	B.Schneier, J.Kelsey, D.Whiting, D.Wagner, C.Hall, N.Ferguson
CAST-256	Entrust Technologies, INC.
CRYPTON	Future System, INC.
DEAL	R. Outerbridge, L.Knudsen
DFC	CNRS
E2	Nippon Telegraph and Telephone Corp.
FROG	TecApro Internacional S.A.
HPC	L.Brown, J.Pieprzyk, J.Seberry
LOKI97	L.Brown, J.Pieprzyk, J.Seberry
MAGENTA	Deutsche Telekom AG
SAFER+	Cylink Corp.

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 also 192 or 256 bit)
  - Key length: 128, 192, or 256 bit
  - Number of rounds: 10, 12 o 14
  - Key scheduling: 44, 52 or 60 subkeys having length = 32 bit

➢ 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)

AES Parameters

	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

1 word = 32 bit

AES Keys

- With 128 bit: 2^{128} = 3.4x 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} = 6.2x 10^{57} possible keys
  - …
- Con 256 bit: 2^{256} = 1.1x 10^{77} possible keys
  - …

Probably AES will stay secure for at least 20 years

Key and Block

- Key with variable length (128,192, 256 bit)
  - Rappresented 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

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, costituted 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

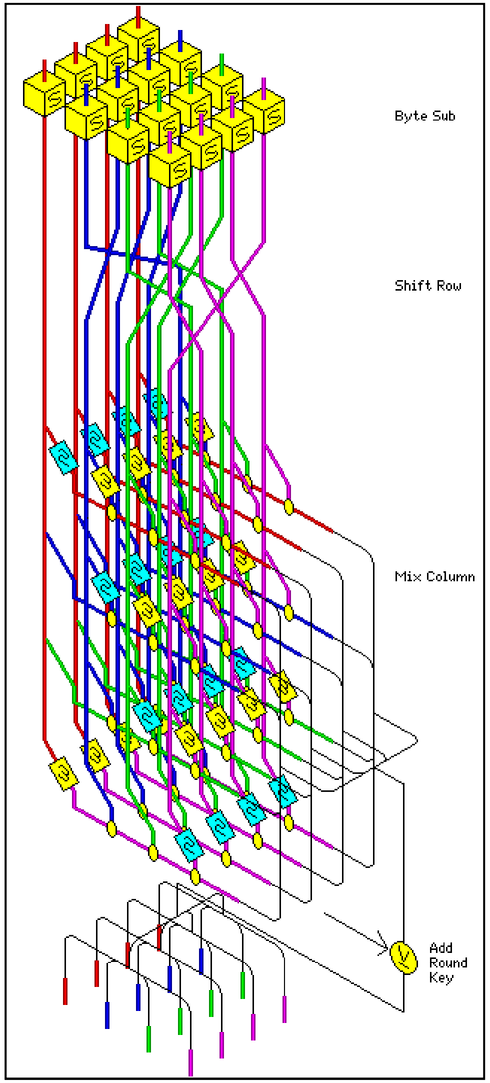
$$out \leftarrow S_{r,c}$$

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

Rijandael: 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
1.  SubBytes(State, S-box)
2.  ShiftRows(State)
3.  AddRoundKey(State, $Key_{Nr}$)
    Y = State
```

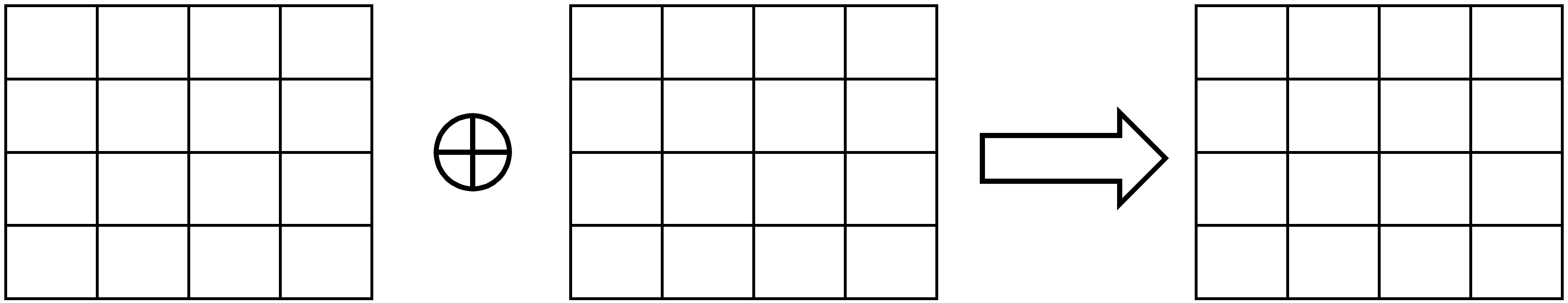


AddRound Key

- **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)**:

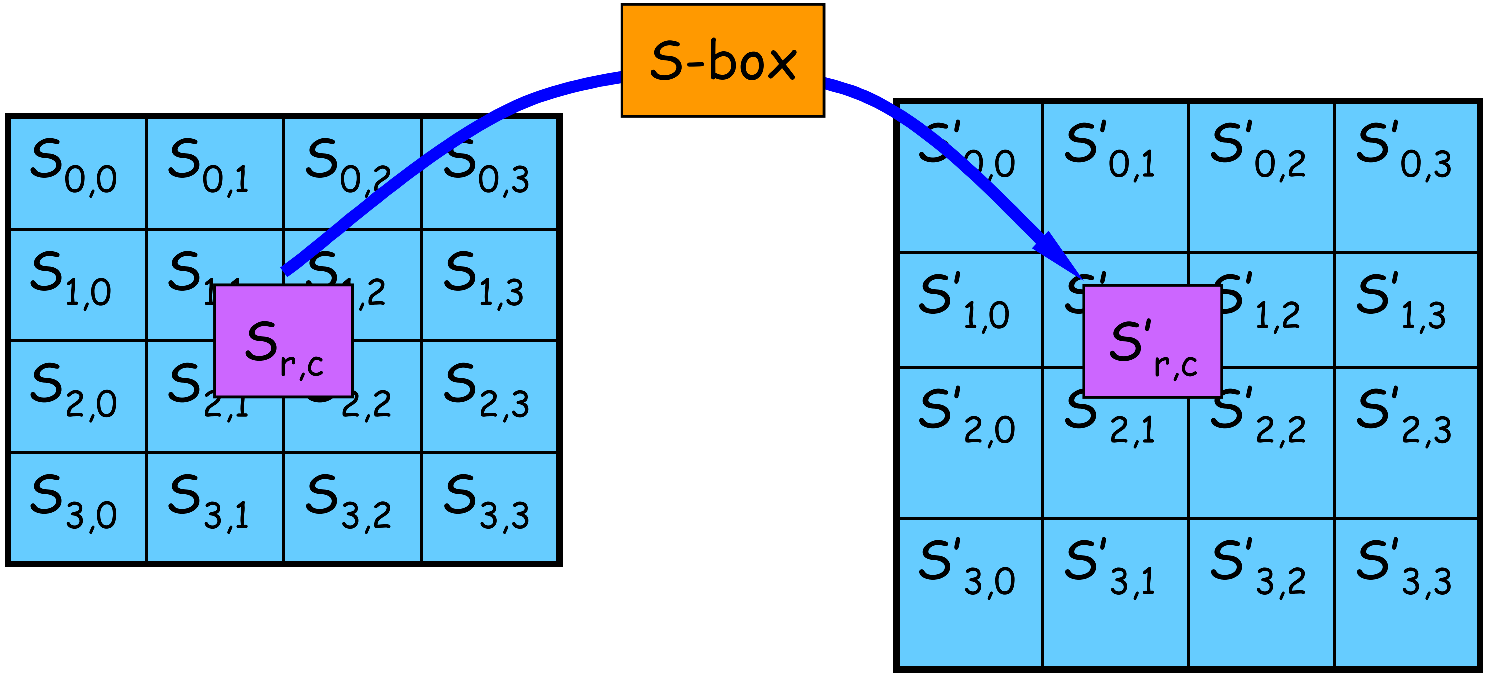


Key **State** **State**

SubBytes Transformation

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

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


S-box
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{r,c}$
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{1,1}$ $S'_{1,2}$ $S'_{1,3}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$
$S'_{r,c}$

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 $s_{\mathbf{r,c}}$ = binary(S-box(**r, c**))
- How is AES S-box different from DES S-boxes?
  - Only one S-box
  - S-boxes based on modular arithmetic with polynomials, can be defined algebraically
  - Easy to analyze, prove attacks fail

Rijandael S-box Table

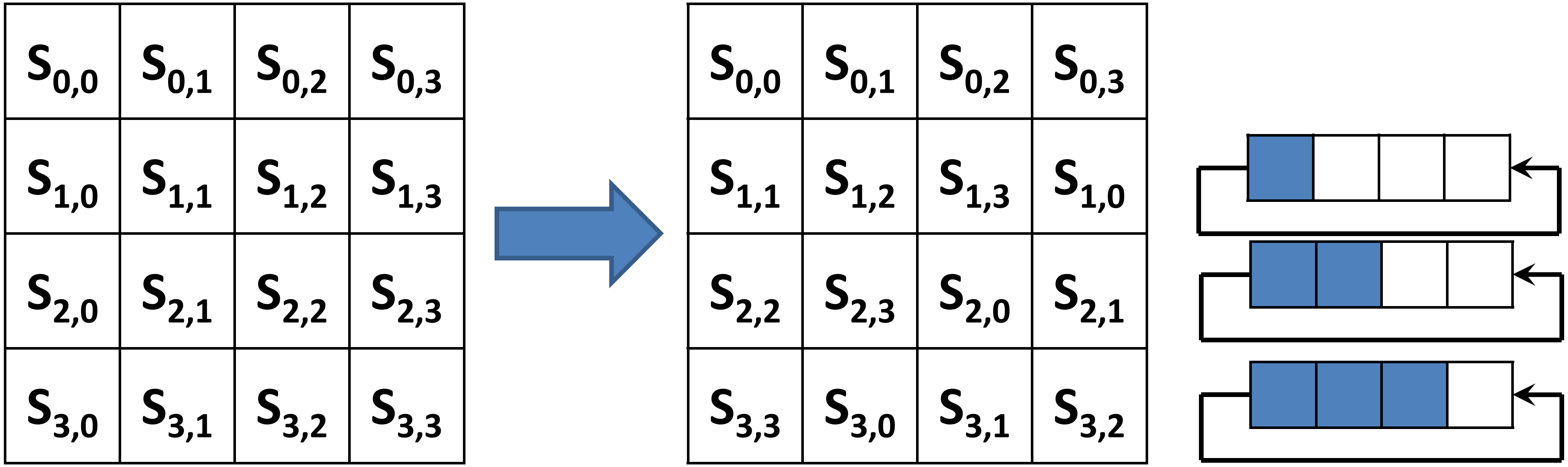
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	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)

ShiftRows

- Circular Left Shift of a number of bytes equal to the row number



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

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

MixColumns

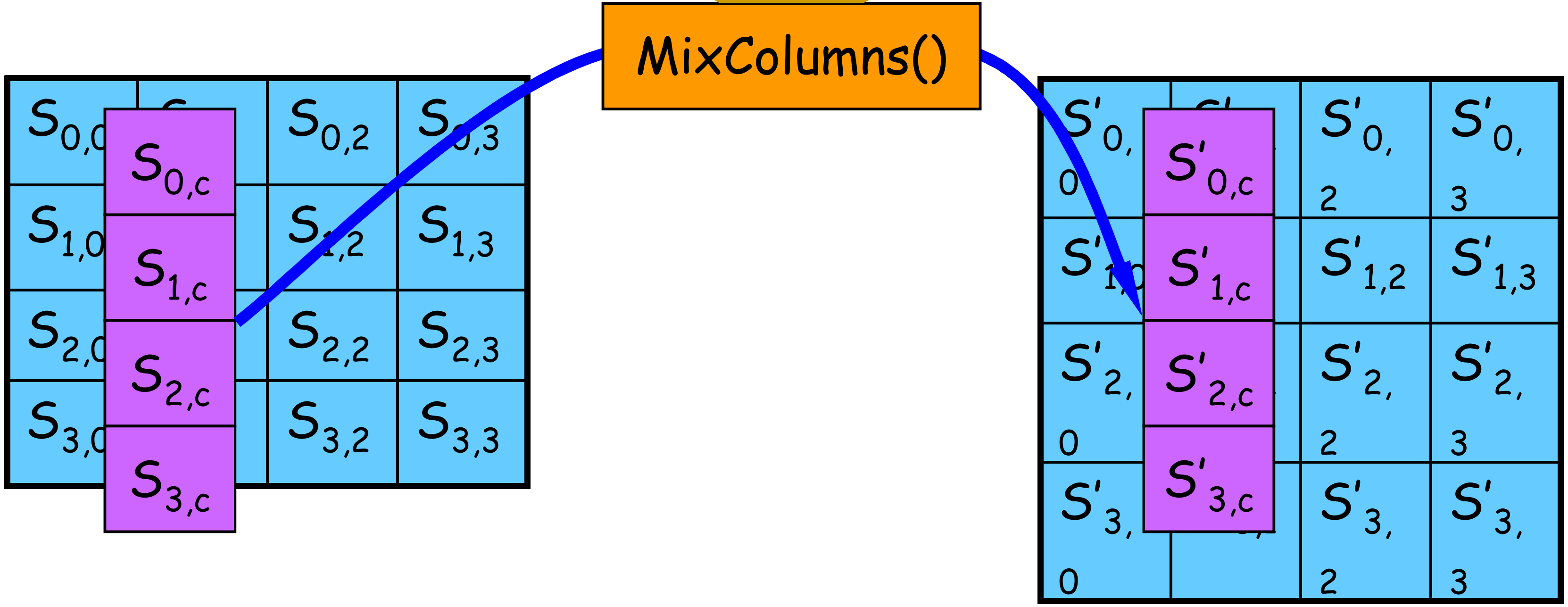
- Interpret each column as a vector of length 4.
- Each column of State is replaced by another column obtained by multiplying that column with a matrix in a particular field (Galois Field).

MixColumns Transformation

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} \leftarrow \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Multiply mod x^4+1 with a(x)

$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$



Bytes in columns are combined linearly

Decryption

- The decryption algorithm is not identical with the encryption algorithm, but uses the same key schedule.
- There is also a way of implementing the decryption with an algorithm that is equivalent to the encryption algorithm (each operation replaced with its inverse), however, in this case, the key schedule must be changed.

Rijandel 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.