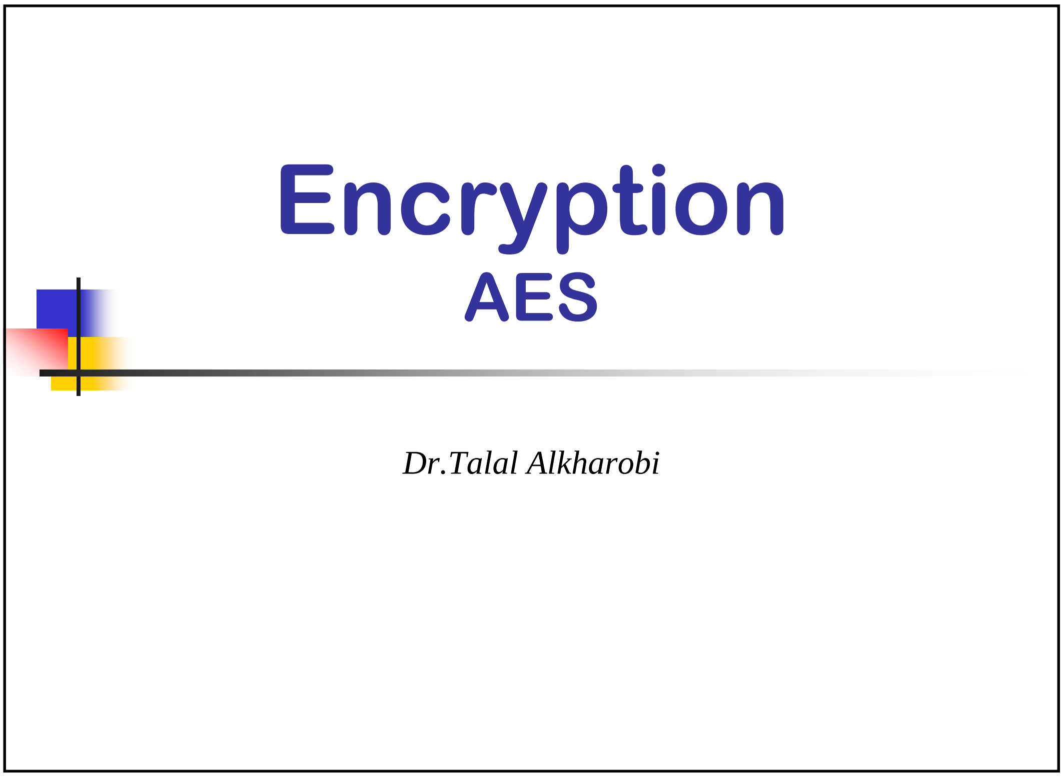
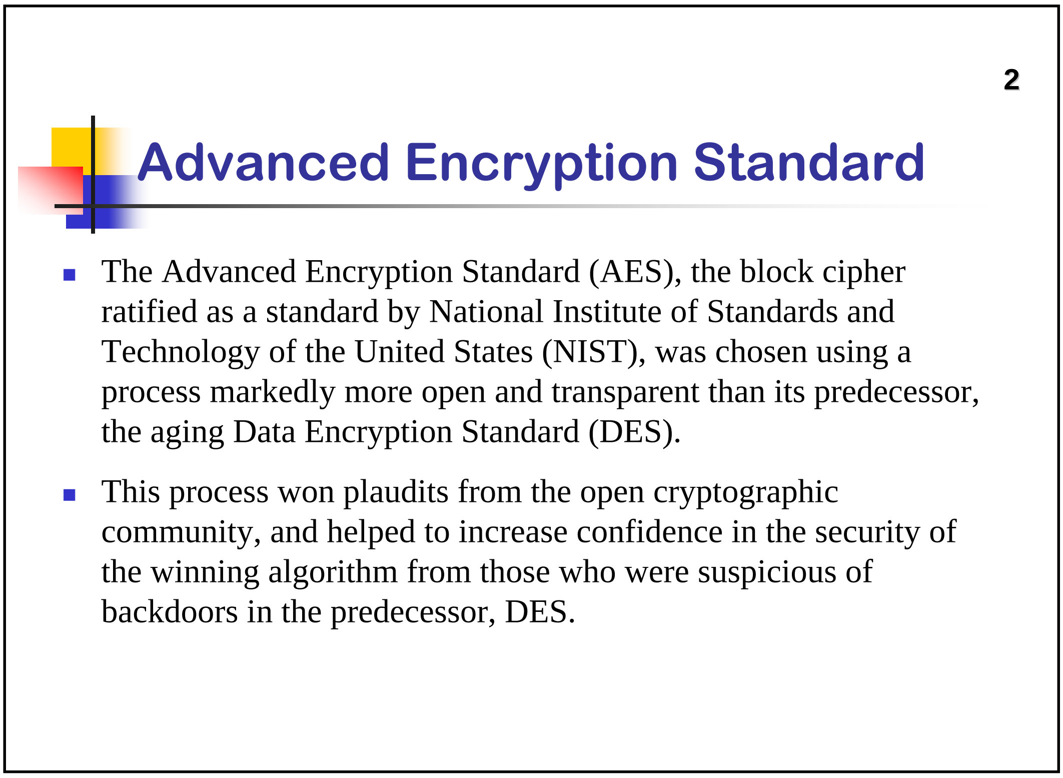


Encryption

AES

Dr.Talal Alkharobi



Advanced Encryption Standard

- The Advanced Encryption Standard (AES), the block cipher ratified as a standard by National Institute of Standards and Technology of the United States (NIST), was chosen using a process markedly more open and transparent than its predecessor, the aging Data Encryption Standard (DES).
- This process won plaudits from the open cryptographic community, and helped to increase confidence in the security of the winning algorithm from those who were suspicious of backdoors in the predecessor, DES.

Advanced Encryption Standard

- A new standard was needed primarily because DES has a relatively small 56-bit key which was becoming vulnerable to brute force attacks.
- In addition the DES was designed primarily for hardware and is relatively slow when implemented in software.
- While Triple-DES avoids the problem of a small key size, it is very slow in software, is unsuitable for limited-resource platforms, and may be affected by potential security issues connected with the (today comparatively small) block size of 64 bits.

Start of the process

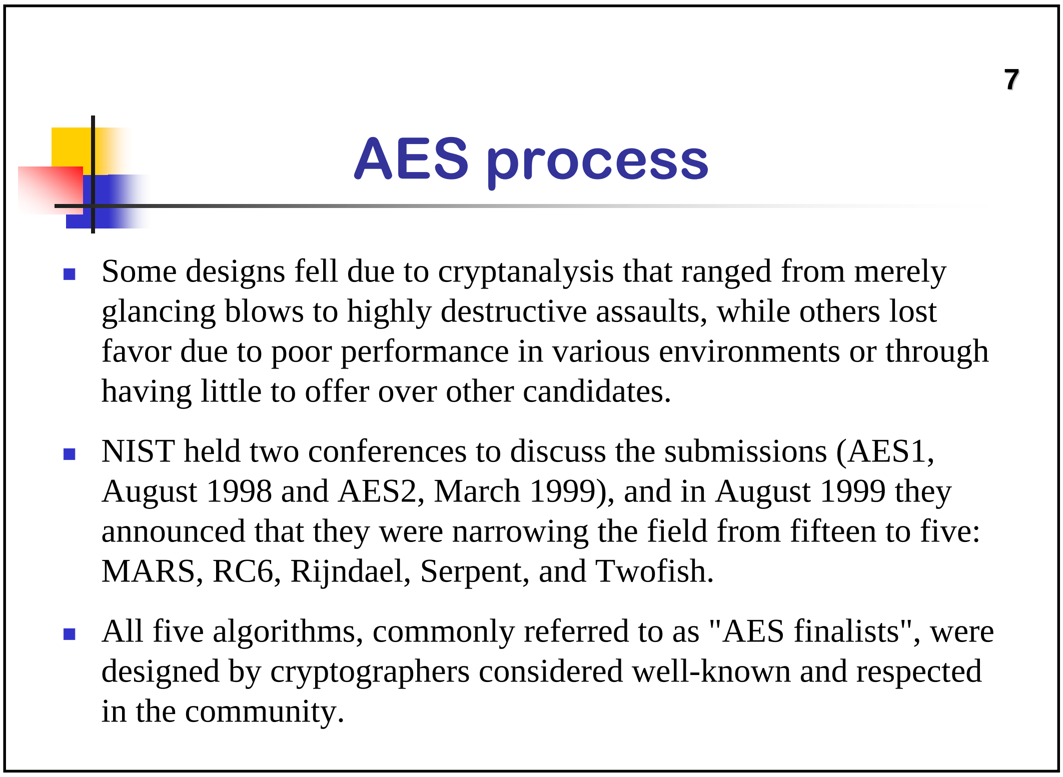
- On January 2, 1997, NIST announced that they wished to choose a successor to the DES to be known as the AES.
- Like DES, this was to be "an unclassified, publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century
- However, rather than simply publishing a successor, NIST asked for input from interested parties on how the successor should be chosen.
- Interest from the open cryptographic community was immediately intense, and NIST received a great many submissions during the three month comment period.

AES process

- The result of this feedback was a call for new algorithms on September 12, 1997
- The algorithms were all to be block ciphers, supporting a block size of 128 bits and key sizes of 128, 192, and 256 bits.
- In the nine months that followed, fifteen different designs were created and submitted from several different countries.
- They were, in alphabetical order: CAST-256 | CRYPTON | DEAL | DFC | E2 | FROG | HPC | LOKI97 | MAGENTA | MARS | RC6 | Rijndael | SAFER+| Serpent | Twofish

AES process

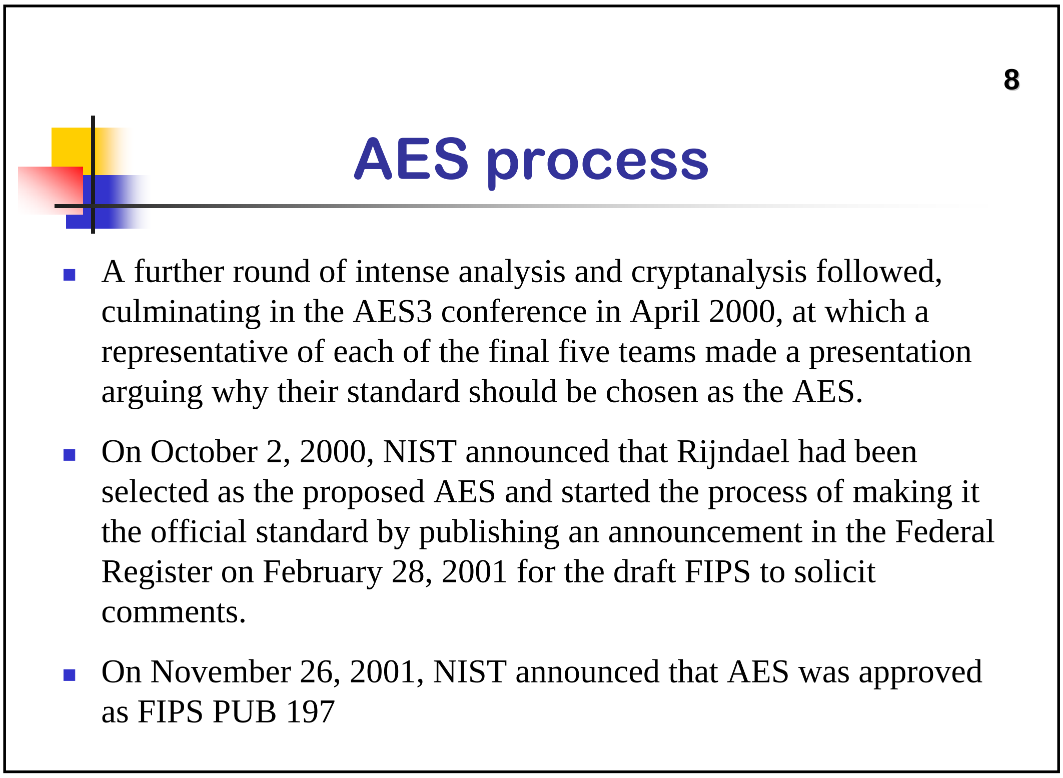
- Algorithms were assessed in
  - Security
  - Performance in a variety of settings (PCs of various architectures, smart cards, hardware implementations)
  - Feasibility in limited environments (smart cards with very limited memory, low gate count implementations, FPGAs).
- In the ensuing debate, many advantages and disadvantages of the different candidates were investigated by cryptographers;



7

AES process

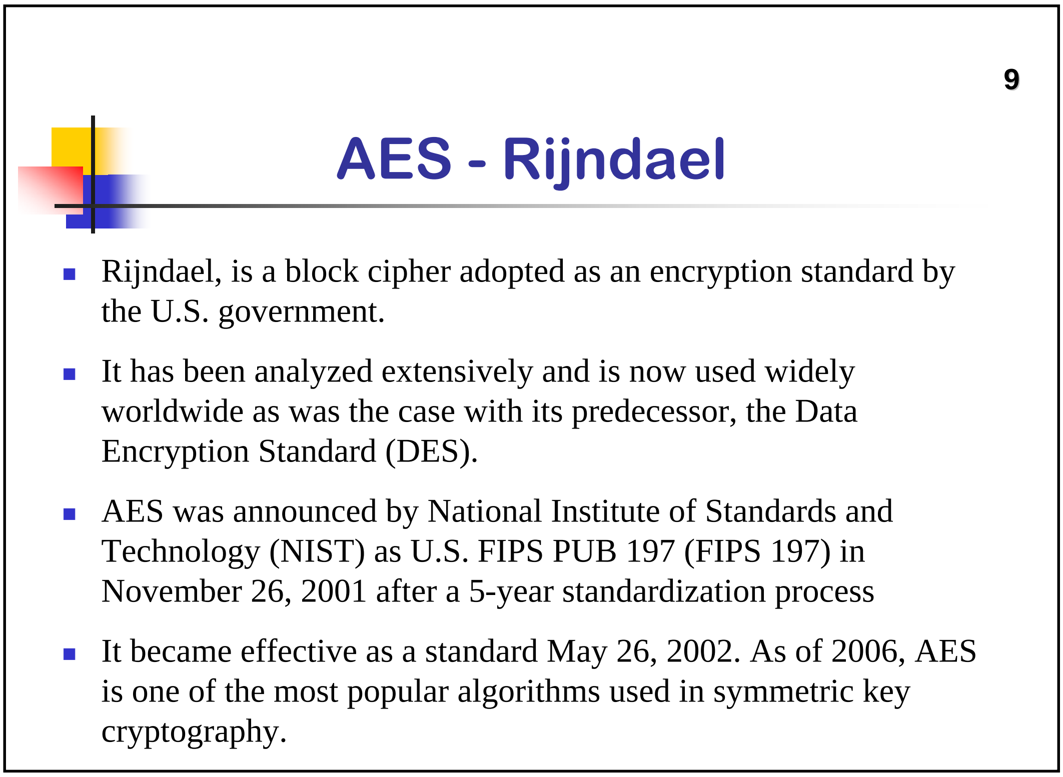
- Some designs fell due to cryptanalysis that ranged from merely glancing blows to highly destructive assaults, while others lost favor due to poor performance in various environments or through having little to offer over other candidates.
- NIST held two conferences to discuss the submissions (AES1, August 1998 and AES2, March 1999), and in August 1999 they announced that they were narrowing the field from fifteen to five: MARS, RC6, Rijndael, Serpent, and Twofish.
- All five algorithms, commonly referred to as "AES finalists", were designed by cryptographers considered well-known and respected in the community.



8

AES process

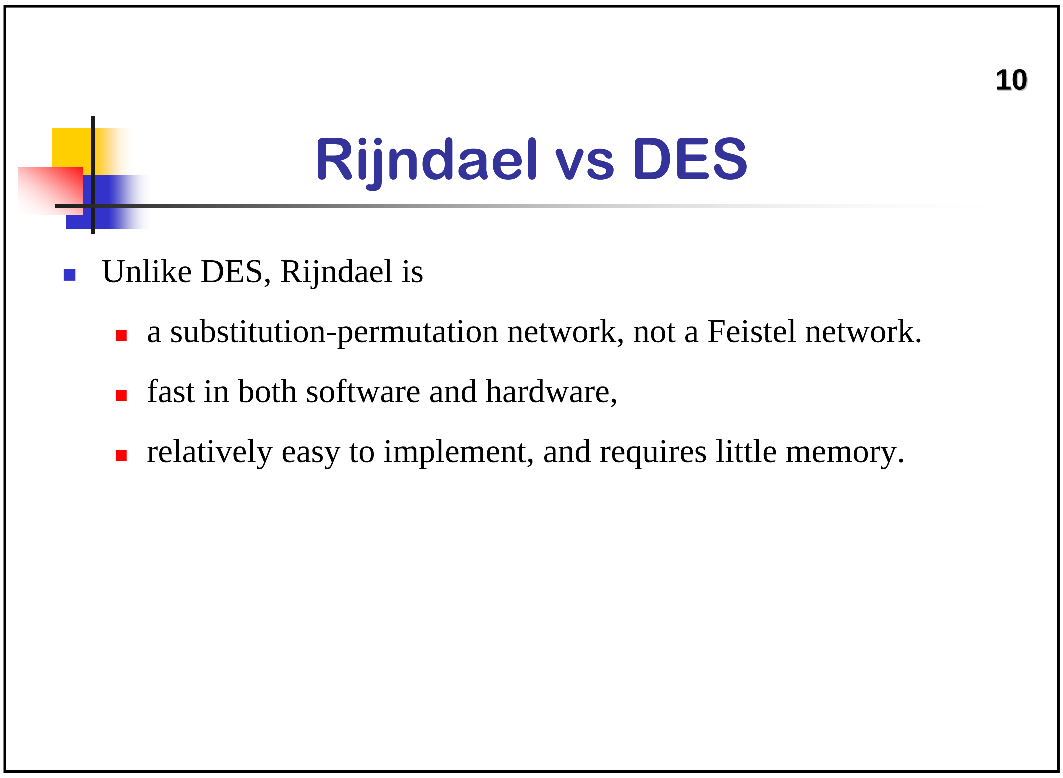
- A further round of intense analysis and cryptanalysis followed, culminating in the AES3 conference in April 2000, at which a representative of each of the final five teams made a presentation arguing why their standard should be chosen as the AES.
- On October 2, 2000, NIST announced that Rijndael had been selected as the proposed AES and started the process of making it the official standard by publishing an announcement in the Federal Register on February 28, 2001 for the draft FIPS to solicit comments.
- On November 26, 2001, NIST announced that AES was approved as FIPS PUB 197



9

AES - Rijndael

- Rijndael, is a block cipher adopted as an encryption standard by the U.S. government.
- It has been analyzed extensively and is now used widely worldwide as was the case with its predecessor, the Data Encryption Standard (DES).
- AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) in November 26, 2001 after a 5-year standardization process
- It became effective as a standard May 26, 2002. As of 2006, AES is one of the most popular algorithms used in symmetric key cryptography.



10

Rijndael vs DES

- Unlike DES, Rijndael is
  - a substitution-permutation network, not a Feistel network.
  - fast in both software and hardware,
  - relatively easy to implement, and requires little memory.

11

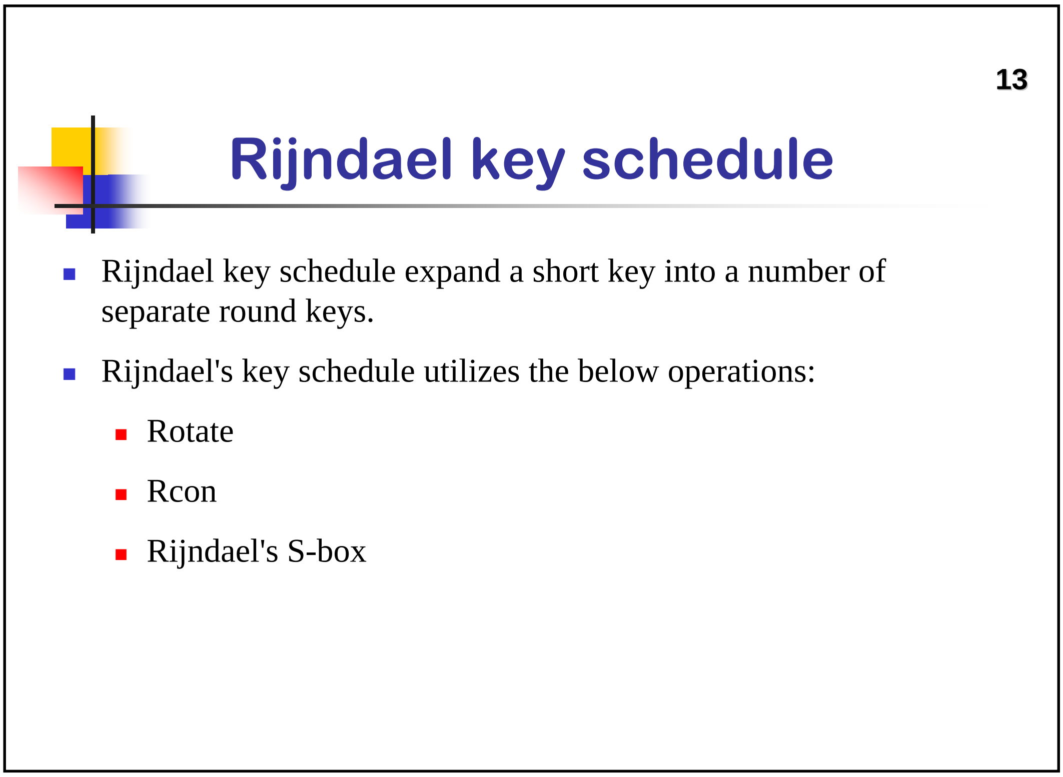
Description of the AES

- Strictly speaking, AES is not precisely Rijndael (although in practice they are used interchangeably) as Rijndael supports a larger range of block and key sizes;
- AES has a fixed block size of 128 bits and a key size of 128, 192 or 256 bits, whereas Rijndael can be specified with key and block sizes in any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits, respectively.
- The key is expanded using Rijndael's key schedule.
- Most of AES calculations are done in a special finite field.

12

Description of the AES

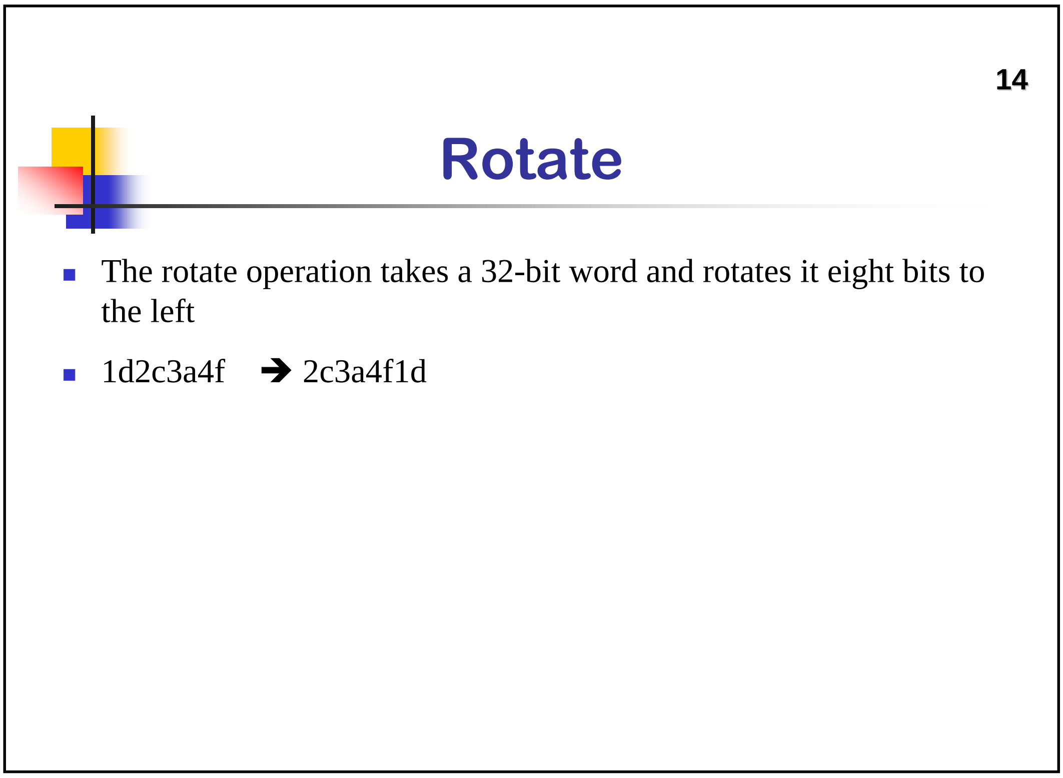
- AES operates on a 4×4 array of bytes, termed the state (versions of Rijndael with a larger block size have additional columns state).
- For encryption, each round of AES (consists of four stages:
  - AddRoundKey
  - SubBytes
  - ShiftRows
  - MixColumns
- The final round replaces the MixColumns stage with another instance of AddRoundKey.



13

Rijndael key schedule

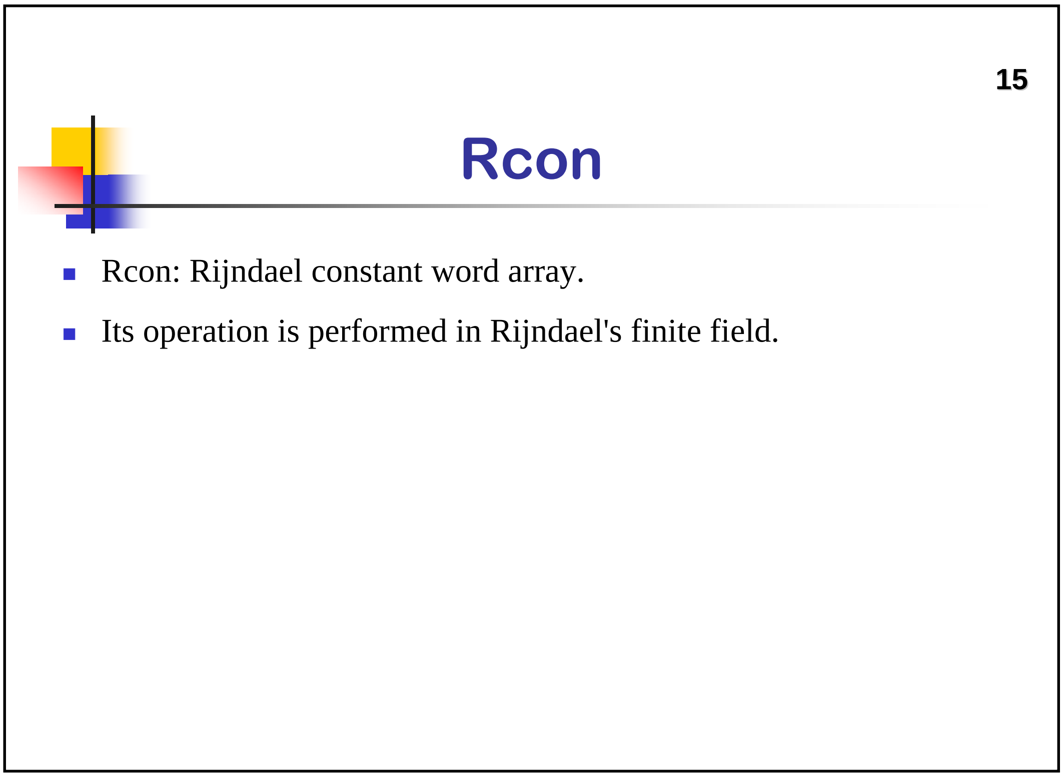
- Rijndael key schedule expand a short key into a number of separate round keys.
- Rijndael's key schedule utilizes the below operations:
  - Rotate
  - Rcon
  - Rijndael's S-box



14

Rotate

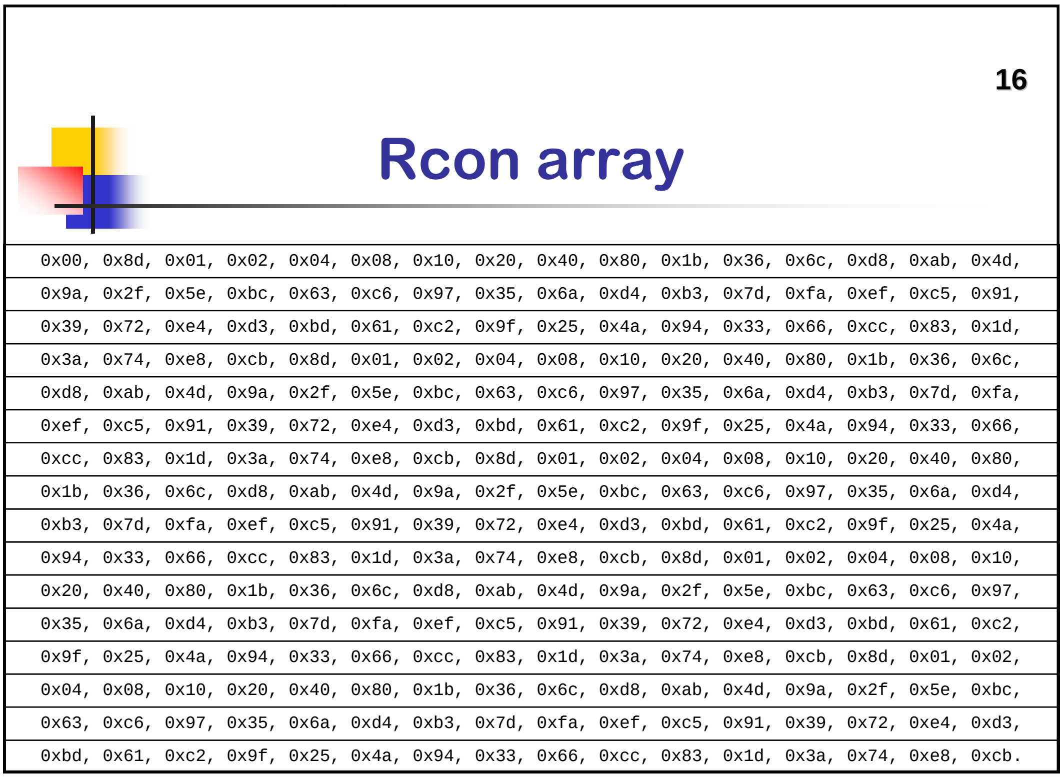
- The rotate operation takes a 32-bit word and rotates it eight bits to the left
- 1d2c3a4f ➔ 2c3a4f1d



15

Rcon

- Rcon: Rijndael constant word array.
- Its operation is performed in Rijndael's finite field.



16

Rcon array

0x00, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d,
0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91,
0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d,
0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c,
0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa,
0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66,
0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80,
0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4,
0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a,
0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10,
0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97,
0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2,
0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02,
0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc,
0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3,
0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb.

17

Rijndael S-box

- The S-box is generated by determining the multiplicative inverse for a given number in Rijndael's finite field (zero, which has no inverse, is sent to zero).
- The multiplicative inverse is then transformed using the following affine transformation:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

18

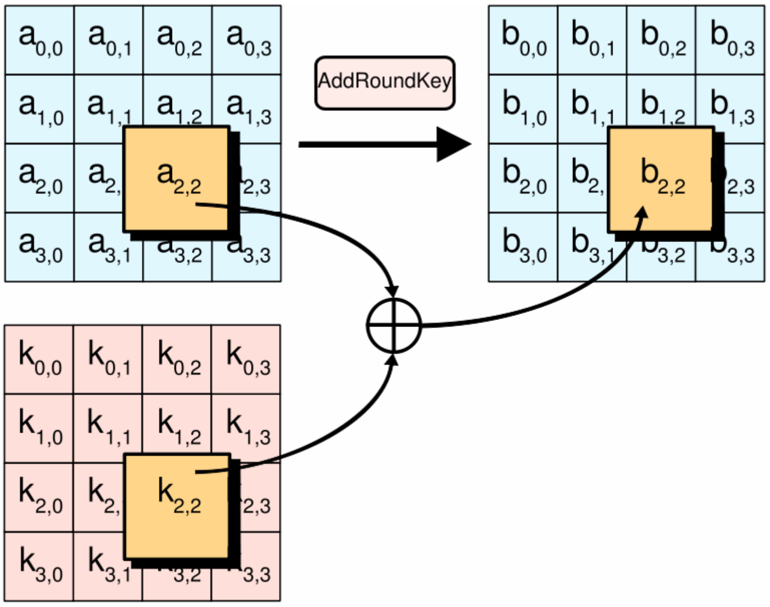
Key schedule core

- Used as an inner loop in the key schedule, and is done thus:
  - Input: 32-bit word, i. Output: 32-bit word.
  - Copy the output over to the input.
  - Rotate the output eight bits to the left
  - Apply Rijndael's S-box on all four individual bytes in the output word
  - Perform the rcon operation with i as the input,
  - XOR the rcon output with the first byte of the output word

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
62 63 63 63 62 63 63 63 62 63 63 63 62 63 63 63
9b 98 98 c9 f9 fb fb aa 9b 98 98 c9 f9 fb fb aa
90 97 34 50 69 6c cf fa f2 f4 57 33 0b 0f ac 99
ee 06 da 7b 87 6a 15 81 75 9e 42 b2 7e 91 ee 2b
7f 2e 2b 88 f8 44 3e 09 8d da 7c bb f3 4b 92 90
ec 61 4b 85 14 25 75 8c 99 ff 09 37 6a b4 9b a7
21 75 17 87 35 50 62 0b ac af 6b 3c c6 1b f0 9b
0e f9 03 33 3b a9 61 38 97 06 0a 04 51 1d fa 9f
b1 d4 d8 e2 8a 7d b9 da 1d 7b b3 de 4c 66 49 41
b4 ef 5b cb 3e 92 e2 11 23 e9 51 cf 6f 8f 18 8e

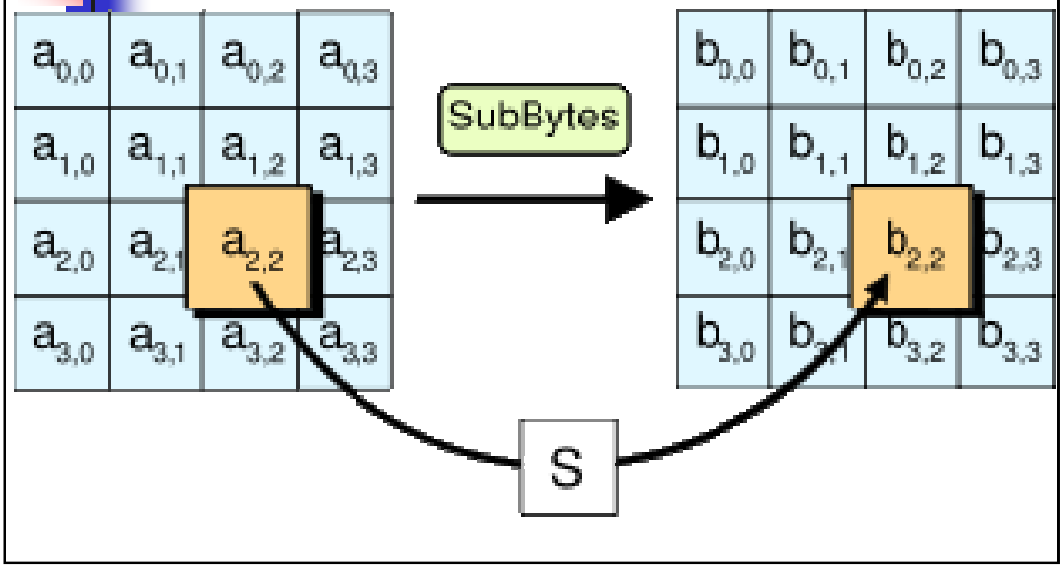
AddRoundKey

- The subkey is combined with the state.
- For each round, a subkey is derived from the main key using Rijndael's key schedule;
- Each subkey is the same size as the state.
- The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

SubBytes

- In the SubBytes step, each byte in the array is updated using an 8-bit S-box.
- This operation provides the non-linearity in the cipher.
- The S-box used is derived from the multiplicative inverse over GF(2^8), known to have good non-linearity properties.
- To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation.
- The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points.



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

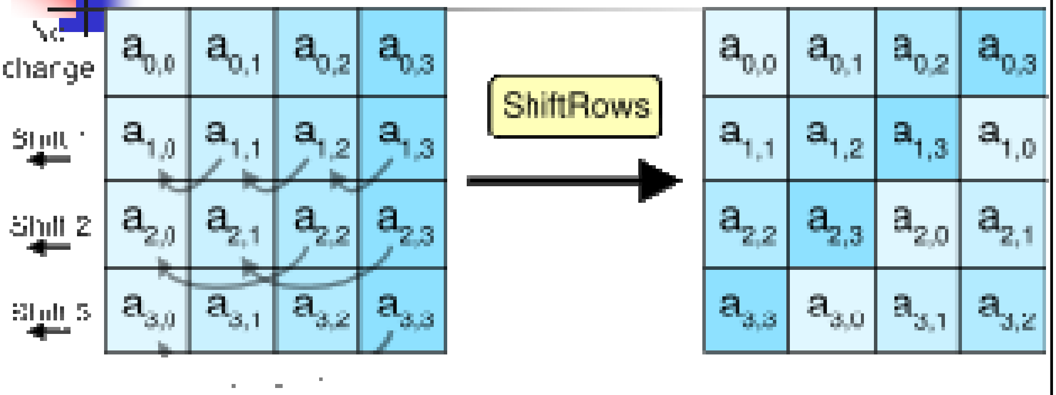
SubBytes

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

S

ShiftRows

- Operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset.
- First row is left unchanged.
- Each byte of the second row is shifted one to the left.
- Similarly, the third and fourth rows are shifted by offsets of two and three respectively.



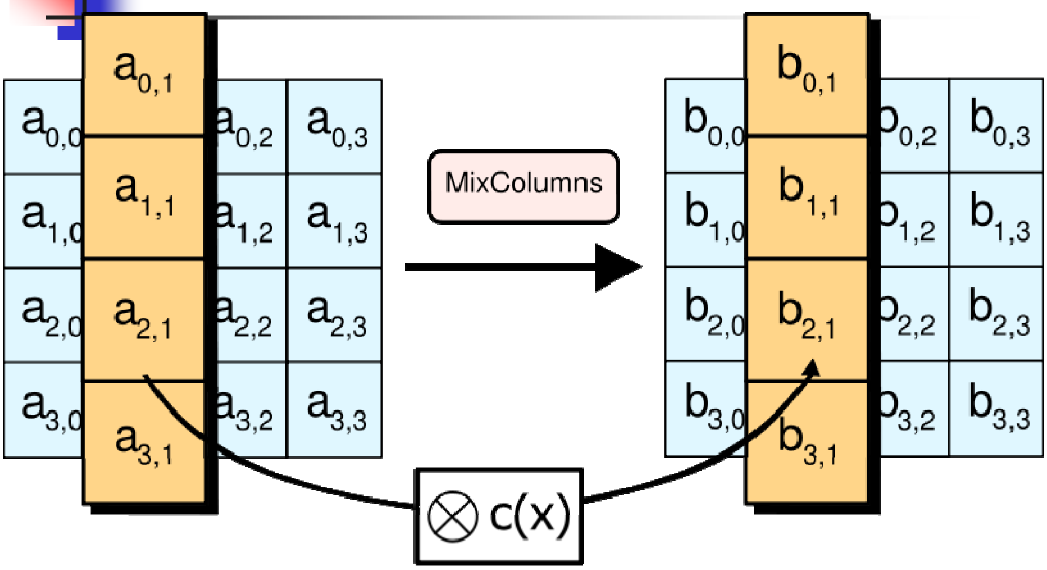
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

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

MixColumns

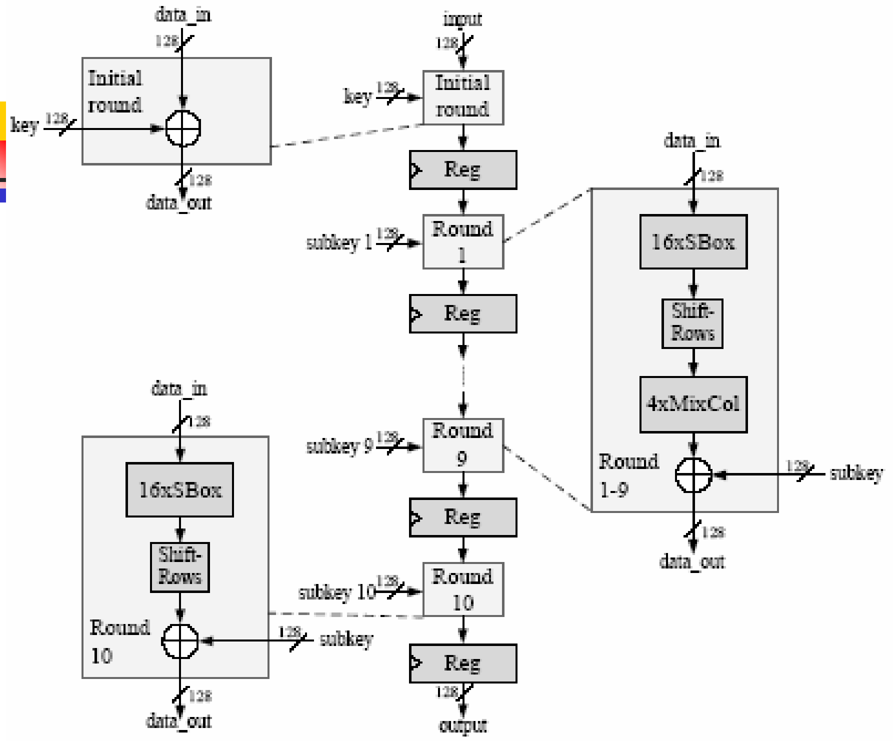
- The four bytes of each column of the state are combined using an invertible linear transformation.
- The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes.
- Together with ShiftRows, MixColumns provides diffusion in the cipher.
- Each column is treated as a polynomial over GF(2^8) and is then multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x) = 3x^3 + x^2 + x + 2$; the inverse of this polynomial is $c'(x) = 11x^3 + 13x^2 + 9x + 14$.

27



$a_{0,0}$
$a_{0,1}$
$a_{0,2}$
$a_{0,3}$
$a_{1,0}$
$a_{1,1}$
$a_{1,2}$
$a_{1,3}$
$a_{2,0}$
$a_{2,1}$
$a_{2,2}$
$a_{2,3}$
$a_{3,0}$
$a_{3,1}$
$a_{3,2}$
$a_{3,3}$
MixColumns
$\otimes$ c(x)
$b_{0,0}$
$b_{0,1}$
$b_{0,2}$
$b_{0,3}$
$b_{1,0}$
$b_{1,1}$
$b_{1,2}$
$b_{1,3}$
$b_{2,0}$
$b_{2,1}$
$b_{2,2}$
$b_{2,3}$
$b_{3,0}$
$b_{3,1}$
$b_{3,2}$
$b_{3,3}$

28



data_in
128
key
128
Initial round
128
data_out
input
128
key
128
Initial round
Reg
subkey 1
128
Round 1
Reg
subkey 9
128
Round 9
Reg
subkey 10
128
Round 10
Reg
128
output
data_in
128
16xSBox
Shift-Rows
4xMixCol
Round 1-9
128
subkey
128
data_out
data_in
128
16xSBox
Shift-Rows
Round 10
128
subkey
128
data_out