

# Cryptography and Network Security

Third Edition  
by William Stallings

Lee, Hoon -Jae

2003 -04 -21

CNSL -Internet -Dongseo

1

## Chapter 5 –Advanced Encryption Standard

*"It seems very simple."*

*"It is very simple. But if you don't know  
what the key is it's virtually  
indecipherable."*

**—Talking to Strange Men, Ruth Rendell**

2003 -04 -21

CNSL -Internet -Dongseo

2

# Origins

- clear a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple -DES – but slow with small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug -99
- Rijndael was selected as the AES in Oct -2000
- issued as FIPS PUB 197 standard in Nov -2001

2003 -04 -21

CNSL -Internet -Dongseo

3

## advanced encryption standard (AES)

- NIST Requirements:
  - Block cypher
  - 128 bit blocks
  - 128/192/256 bit keys
  - Strength equal to or better than 3DES at greatly improved efficiency
    - smartcard, hardware, software
    - flexibility
    - simplicity and elegance
  - Royalty free worldwide
  - security for over 30 years
  - may protect sensitive data for over 100 years
  - public confidence in the cypher

2003 -04 -21

CNSL -Internet -Dongseo

4

# AES candidates

- 15 submissions from international field
- Some strong candidates:

Name	Type	Rounds	Rel. Speed (cycles)	Gates
Twofish	Feistel	16	1254	23000
Serpent	SP-network	32	1800	70000
Mars	Ext. Feistel	32	1600	70000 cells
Rijndael	Square	10, 12, 14	1276	???
RC6	Feistel	20	1436	???

2003 -04 -21

CNSL -Internet -Dongseo

5

# AES

- Rijndael (pronounced "rain-dahl") announced Oct 2000
- Operates on 128 bit blocks
- Key length is variable: 128, 192 or 256 bits
- An SP-network
- Uses a single S-box which acts on a byte input to give a byte output (think of as a 256 byte lookup table):

$$S(x) = M(1/x) + b \text{ over the field } GF(2^8)$$

(M is a matrix, b is a constant)

- Construction gives tight differential and linear bounds.

2003 -04 -21

CNSL -Internet -Dongseo

6



## AES Requirements

- private key symmetric block cipher
- 128 -bit data, 128/192/256 -bit keys
- stronger & faster than Triple -DES
- active life of 20 -30 years (+ archival use)
- provide full specification & design details
- both C & Java implementations
- NIST have released all submissions & unclassified analyses

2003 -04 -21

CNSL -Internet -Dongseo

7

## AES Evaluation Criteria

- initial criteria:
  - security – effort to practically cryptanalyse
  - cost – computational
  - algorithm & implementation characteristics
- final criteria
  - general security
  - software & hardware implementation ease
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)

2003 -04 -21

CNSL -Internet -Dongseo

8

## AES Evaluation Criteria -1(1997)

### SECURITY

- Actual security:** compared to other submitted algorithms (at the same key and block size).
- Randomness:** The extent to which the algorithm output is indistinguishable from a random permutation on the input block.
- Soundness:** of the mathematical basis for the algorithm's security.
- Other security factors:** raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter.

### COST

- Licensing requirements:** NIST intends that when the AES is issued, the algorithm(s) specified in the AES shall be available on a worldwide, non-exclusive, royalty-free basis.
- Computational efficiency:** The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination (128-128); more attention will be paid to hardware implementations and other supported key-block size combinations during Round 2 analysis. Computational efficiency essentially refers to the speed of the algorithm. Public comments on each algorithm's efficiency (particularly for various platforms and applications) will also be taken into consideration by NIST.
- Memory requirements:** The memory required to implement a candidate algorithm--for both hardware and software implementations of the algorithm--will also be considered during the evaluation process. Round 1 analysis by NIST will focus primarily on software implementations; more attention will be paid to hardware implementations during Round 2. Memory requirements will include such factors as gate counts for hardware implementations, and code size and RAM requirements for software implementations.

## AES Evaluation Criteria -1(1997)

### ALGORITHM AND IMPLEMENTATION CHARACTERISTICS

- Flexibility:** Candidate algorithms with greater flexibility will meet the needs of more users than less flexible ones, and therefore, inter alia, are preferable. However, some extremes of functionality are of little practical application (e.g., extremely short key lengths); for those cases, preference will not be given. Some examples of flexibility may include (but are not limited to) the following:
  - a. The algorithm can accommodate additional key- and block-sizes (e.g., 64-bit block sizes, key sizes other than those specified in the Minimum Acceptability Requirements section, [e.g., keys between 128 and 256 that are multiples of 32 bits, etc.])
  - b. The algorithm can be implemented securely and efficiently in a wide variety of platforms and applications (e.g., 8-bit processors, ATM networks, voice & satellite communications, HDTV, B-ISDN, etc.).
  - c. The algorithm can be implemented as a stream cipher, message authentication code (MAC) generator, pseudorandom number generator, hashing algorithm, etc.
- Hardware and software suitability:** A candidate algorithm shall not be restrictive in the sense that it can only be implemented in hardware. If one can also implement the algorithm efficiently in firmware, then this will be an advantage in the area of flexibility.
- Simplicity:** A candidate algorithm shall be judged according to relative simplicity of design.

# AES Evaluation Criteria -Final(2000)

## **General Security**

Rijndael has no known security attacks. Rijndael uses S-boxes as nonlinear components. Rijndael appears to have an adequate security margin, but has received some criticism suggesting that its mathematical structure may lead to attacks. On the other hand, the simple structure may have facilitated its security analysis during the timeframe of the AES development process.

## **Software Implementations**

Rijndael performs encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms, and DSPs. However, there is a decrease in performance with the higher key sizes because of the increased number of rounds that are performed. Rijndael's high inherent parallelism facilitates the efficient use of processor resources, resulting in very good software performance even when implemented in a mode not capable of interleaving. Rijndael's key setup time is fast.

## **Restricted-Space Environments**

In general, Rijndael is very well suited for restricted-space environments where either encryption or decryption is implemented (but not both). It has very low RAM and ROM requirements. A drawback is that ROM requirements will increase if both encryption and decryption are implemented simultaneously, although it appears to remain suitable for these environments. The key schedule for decryption is separate from encryption.

## **Hardware Implementations**

Rijndael has the highest throughput of any of the finalists for feedback modes and second highest for non-feedback modes. For the 192 and 256-bit key sizes, throughput falls in standard and unrolled implementations because of the additional number of rounds. For fully pipelined implementations, the area requirement increases, but the throughput is unaffected.

# AES Evaluation Criteria -Final(2000)

## **Attacks on Implementations**

The operations used by Rijndael are among the easiest to defend against power and timing attacks. The use of masking techniques to provide Rijndael with some defense against these attacks does not cause significant performance degradation relative to the other finalists, and its RAM requirement remains reasonable. Rijndael appears to gain a major speed advantage over its competitors when such protections are considered.

## **Encryption vs. Decryption**

The encryption and decryption functions in Rijndael differ. One FPGA study reports that the implementation of both encryption and decryption takes about 60% more space than the implementation of encryption alone. Rijndael's speed does not vary significantly between encryption and decryption, although the key setup performance is slower for decryption than for encryption.

## **Key Agility**

Rijndael supports on-the-fly subkey computation for encryption. Rijndael requires a one-time execution of the key schedule to generate all subkeys prior to the first decryption with a specific key. This places a slight resource burden on the key agility of Rijndael.

## **Other Versatility and Flexibility**

Rijndael fully supports block sizes and key sizes of 128 bits, 192 bits and 256 bits, in any combination. In principle, the Rijndael structure can accommodate any block sizes and key sizes that are multiples of 32, as well as changes in the number of rounds that are specified.

## **Potential for Instruction-Level Parallelism**

Rijndael has an excellent potential for parallelism for a single block encryption.



## AES Shortlist

- after testing and evaluation, shortlist in Aug -99:
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin
- then subject to further analysis & comment
- saw contrast between algorithms with
  - few complex rounds verses many simple rounds
  - which refined existing ciphers verses new proposals

2003 -04 -21

CNSL -Internet -Dongseo

13

## The AES Cipher - Rijndael

- designed by Rijmen -Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an iterative rather than feistel cipher
  - treats data in 4 groups of 4 bytes
  - operates an entire block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

2003 -04 -21

CNSL -Internet -Dongseo

14

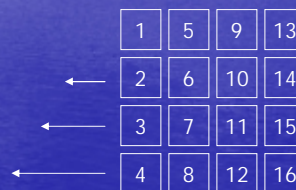
# Rijndael - Parameter

Table 5.3 AES Parameters

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

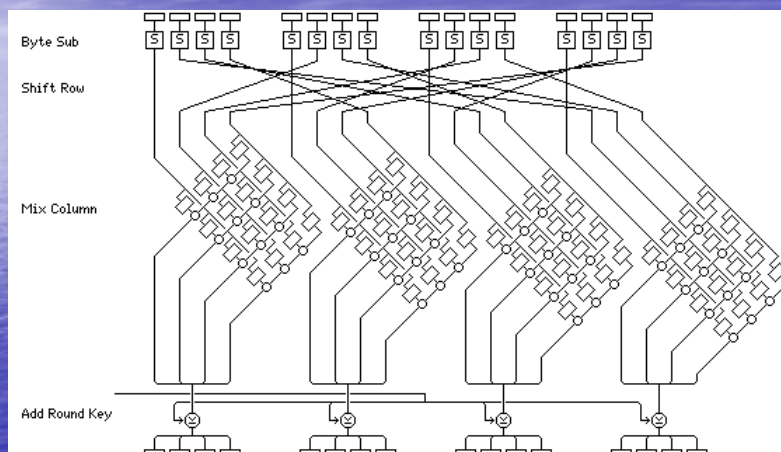
## AES overview

- Linear transformation arranging 16 bytes of the value being encoded in a square and doing bitwise shuffling and mixing.
- Step 1: Byte-sub
  - S-box substitution
- Step 2: Shuffle (or shift-row)
  - top row of four bytes is unchanged
  - second row is shifted one place left
  - third row is shifted two places left
  - fourth row is shifted three places left
- Step 3: Mix Column
  - four bytes in a column are mixed using a matrix multiplication
- Step 4: Add Round Key
  - simply XORs in the subkey for the current round
- Result: Change in the input effects all of output in 2 rounds





# AES overview



2003 -04 -21

CNSL -Internet -Dongseo

17

# AES overview

- Number of rounds variable:
  - 10 for 128-bit keys
  - 12 for 192-bit keys
  - 14 for 256-bit keys
- Gives 50% margin of safety based on current known attacks
  - attack for 6 round 128 bit keys
  - attack for 7 round 192 bit keys
  - attack for 9 round 256 bit keys
  - however require enormous amount of texts (certificational)
- Safety against feasible attacks believed to currently be ~100%

2003 -04 -21

CNSL -Internet -Dongseo

18

# Rijndael

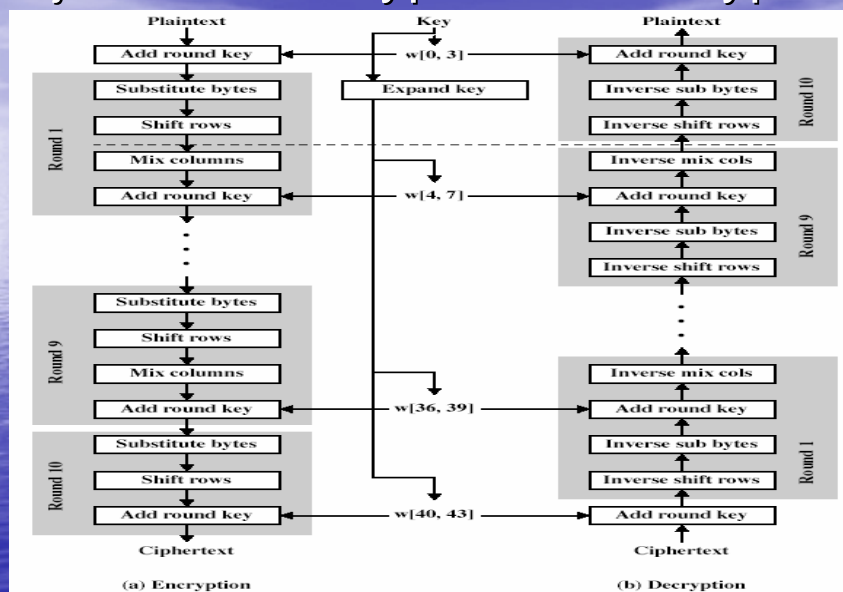
- processes data as 4 groups of 4 bytes (state)
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiply of groups)
  - add round key (XOR state with key material)
- initial XOR key material & incomplete last round
- all operations can be combined into XOR and table lookups - hence very fast & efficient

2003-04-21

CNSL-Internet-Dongseo

19

## Rijndael – Encryption & Decryption

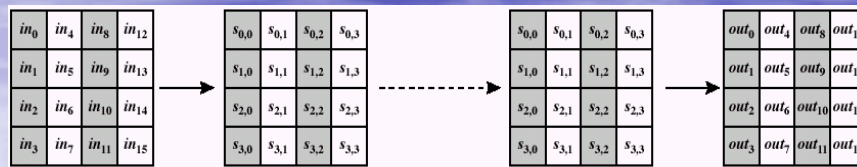


2003-04-21

CNSL-Internet-Dongseo

20

# Rijndael – Data Structure



(a) Input, state array, and output



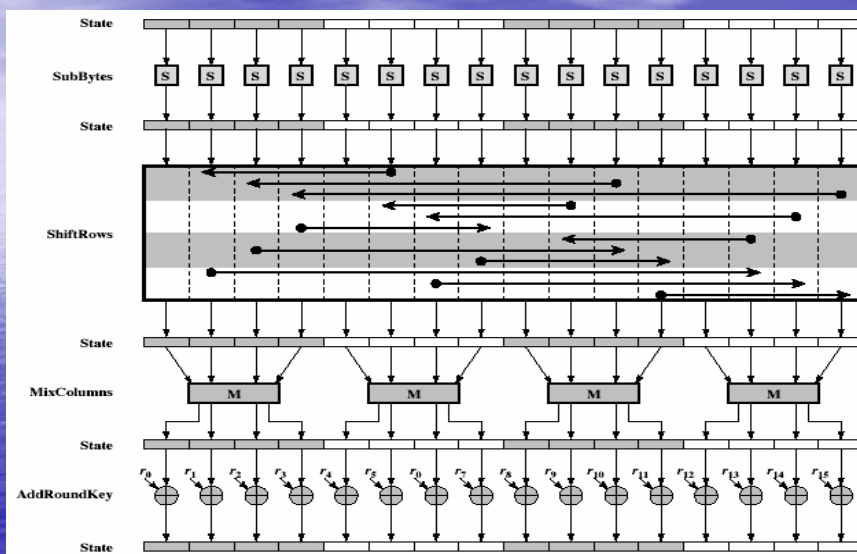
(b) Key and expanded key

2003 -04 -21

CNSL -Internet -Dongseo

21

# AES Round



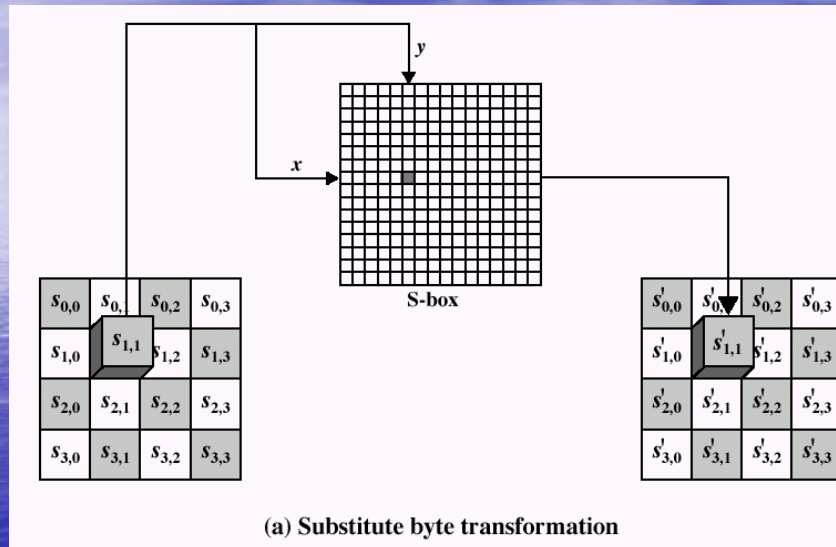
2003 -04 -21

CNSL -Internet -Dongseo

22



## Byte Substitution(1)



2003 -04 -21

CNSL -Internet -Dongseo

23

## Byte Substitution(2)

Table 5.4 AES S-Boxes

(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	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

2003 -04 -21

CNSL -Internet -Dongseo

24

## Byte Substitution(3)

(b) Inverse S-box

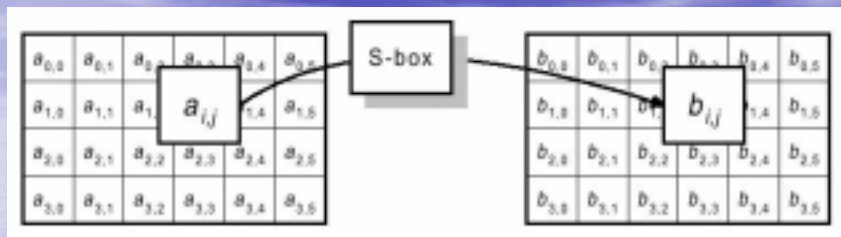
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	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

2003-04-21

CNSL-Internet-Dongseo

25

## Byte Substitution(4) - Examples



EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5



87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

2003-04-21

CNSL-Internet-Dongseo

26

## Byte Substitution(5)

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

2003 -04 -21

CNSL -Internet -Dongseo

27

## Byte Substitution(6)

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by row 9 col 5 byte
  - which is the value {2A}
- S-box is constructed using a defined transformation of the values in GF(2<sup>8</sup>)
- designed to be resistant to all known attacks

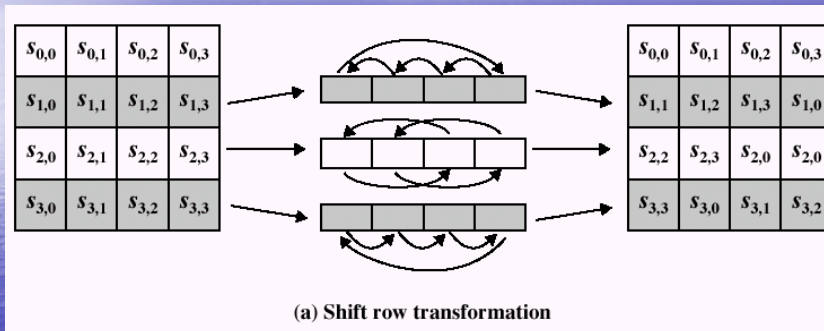
2003 -04 -21

CNSL -Internet -Dongseo

28



## Shift Rows(1)



2003 -04 -21

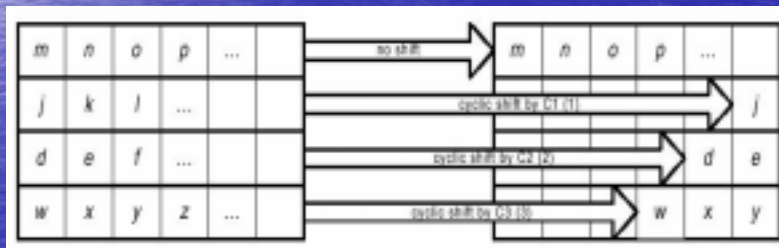
CNSL -Internet -Dongseo

29

## Shift Rows(2)

- Shift offsets for different block lengths

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4




2003 -04 -21

CNSL -Internet -Dongseo

30

## Shift Rows(3) - Examples

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6



87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

2003 -04 -21

CNSL -Internet -Dongseo

31

## Shift Rows(4)

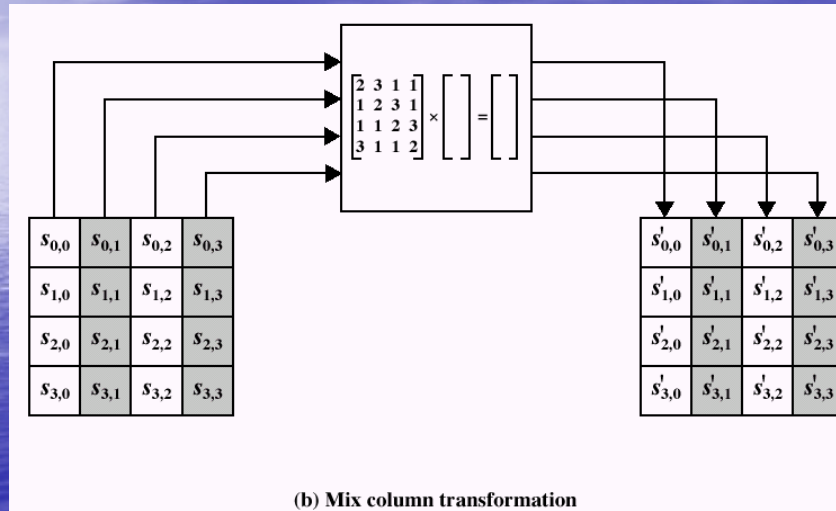
- a circular byte shift in each each
  - 1<sup>st</sup> row is unchanged
  - 2<sup>nd</sup> row does 1 byte circular shift to left
  - 3<sup>rd</sup> row does 2 byte circular shift to left
  - 4<sup>th</sup> row does 3 byte circular shift to left
- decrypt does shifts to right
- since state is processed by columns, this step permutes bytes between the columns

2003 -04 -21

CNSL -Internet -Dongseo

32

## Mix Columns(1)



2003 -04 -21

CNSL -Internet -Dongseo

33

## Mix Columns(2)

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in  $GF(2^8)$  using prime poly  $m(x) = x^8 + x^4 + x^3 + x + 1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

2003 -04 -21


CNSL -Internet -Dongseo

34



## Mix Columns(3) - Example

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95



47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

2003 -04 -21

CNSL -Internet -Dongseo

35

## Add Round Key

- XOR state with 128 -bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption is identical since XOR is own inverse, just with correct round key
- designed to be as simple as possible

2003 -04 -21

CNSL -Internet -Dongseo

36

## Add Round Key

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC



AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

2003 -04 -21

CNSL -Internet -Dongseo

37

## AES Key Expansion

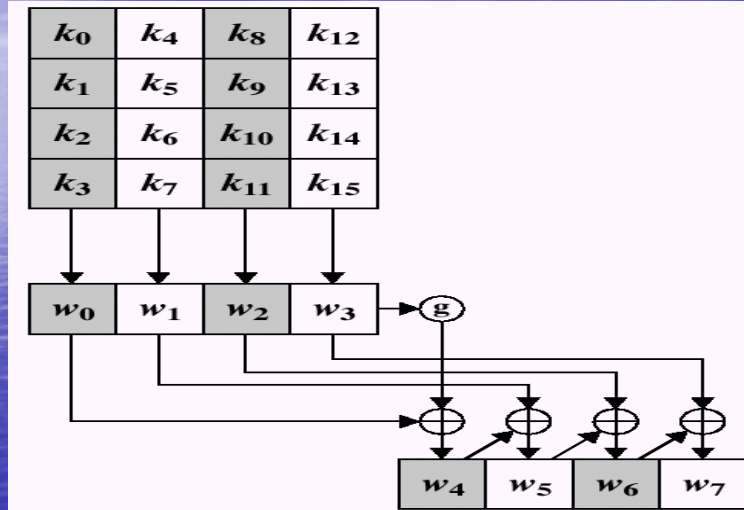
- takes 128 -bit (16 -byte) key and expands into array of 44/52/60 32 -bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - every 4<sup>th</sup> has S -box + rotate + XOR constant of previous before XOR together
- designed to resist known attacks

2003 -04 -21

CNSL -Internet -Dongseo

38

## AES Key Expansion



2003 -04 -21

CNSL -Internet -Dongseo

39

## AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
- works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

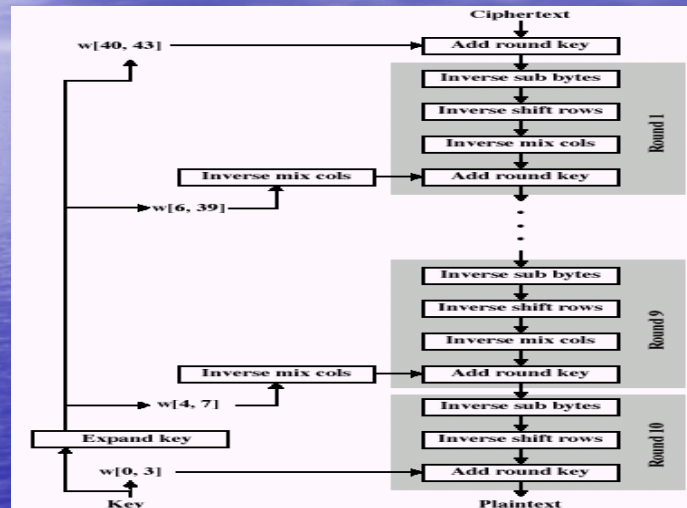
2003 -04 -21

CNSL -Internet -Dongseo

40



# AES Decryption



2003 -04 -21

CNSL -Internet -Dongseo

41

## Implementation Aspects

- can efficiently implement on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shifting
  - add round key works on byte XORs
  - mix columns requires matrix multiply in  $GF(2^8)$  which works on byte values, can be simplified to use a table lookup

2003 -04 -21

CNSL -Internet -Dongseo

42

## Implementation Aspects

- can efficiently implement on 32-bit CPU
  - redefine steps to use 32-bit words
  - can precompute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 16Kb to store tables
- designers believe this very efficient implementation was a key factor in its selection as the AES cipher

2003-04-21

CNSL-Internet-Dongseo

43

## Summary

- have considered:
  - the AES selection process
  - the details of Rijndael – the AES cipher
  - looked at the steps in each round
  - the key expansion
  - implementation aspects

2003-04-21

CNSL-Internet-Dongseo

44