Cryptography and Network Security

Fourth Edition

by William Stallings

Lecture slides by Lawrie Brown, Marius Zimand

Modified by Sanchita Mal-Sarkar

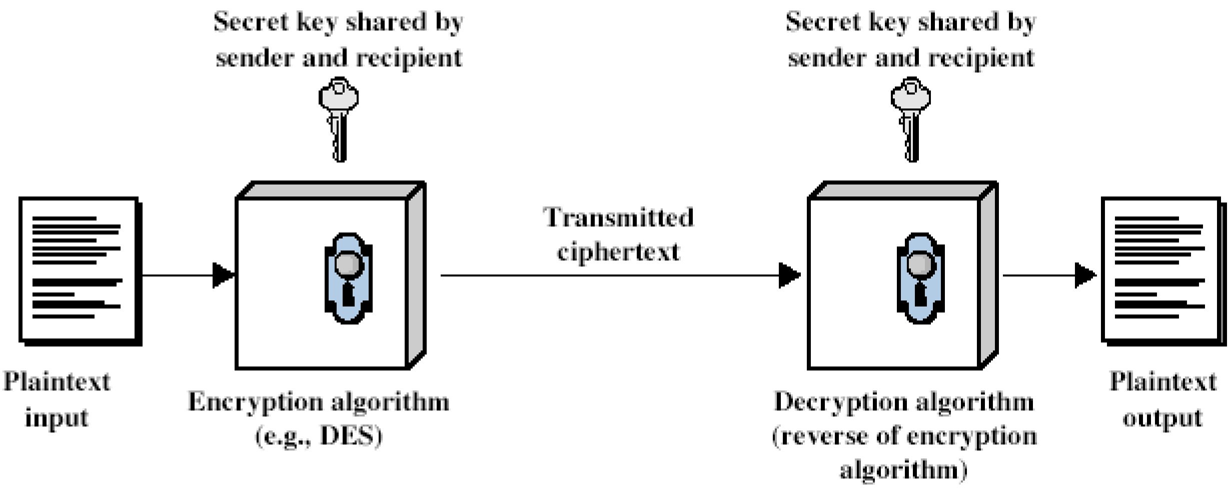
Symmetric Encryption

- Symmetric encryption is also referred to as conventional / private-key / single-key encryption.
- Sender and recipient share a common key.
- All classical encryption algorithms are private-key.
- It was the only type of encryption in use prior to the invention of public-key encryption in 1970s.
- It is by far most widely used of the two types of encryption.

Some Basic Terminology

- **Plaintext** - original message
- **Ciphertext** - coded message
- **Cipher** - algorithm for transforming plaintext to ciphertext
- **Key** - info used in cipher known only to sender/receiver
- **Encipher (encrypt)** - converting plaintext to ciphertext
- **Decipher (decrypt)** - recovering ciphertext from plaintext
- **Cryptography** - study of encryption principles/methods
- **Cryptanalysis (codebreaking)** - study of principles/ methods of deciphering ciphertext *without* knowing key
- **Cryptology** - field of both cryptography and cryptanalysis

Symmetric Cipher Model


Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Symmetric Encryption

- Symmetric encryption is the oldest and the best-known technique. A secret key (a number, a word, or just a string of random letters) is applied to the plain text to change the content in a particular way – shifting each letter by a number of places.
- As long as the sender and the receiver know the key, they can encrypt and decrypt the messages. It’s is faster than asymmetric encryption.
- But the problem is exchanging the secret key over the Internet or a large network so that it does not fall into the wrong hands.
- Anyone who has the secret key, can decrypt the message.
- Example: DES, AES, Blowfish, and Skipjack.

Asymmetric Encryption

- In asymmetric encryption, there are two related keys--a key pair. A public key is made freely available to anyone who might want to send a message.
- A second, private key is kept secret, so that only designated person know it and decrypt it. Any message (text, binary files, or documents) that are encrypted by using the public key can only be decrypted by applying the same algorithm, but by using the matching private key.
- Any message that is encrypted by using the private key can only be decrypted by using the matching public key. This means that you do not have to worry about passing public keys over the Internet (the keys are supposed to be public).
- A problem with asymmetric encryption, however, is that it is slower than symmetric encryption. It requires far more processing power to both encrypt and decrypt the content of the message. RSA is the most widely used asymmetric encryption algorithm. SSL uses asymmetric algorithm.

Requirements

- Two requirements for secure use of symmetric encryption:
  - A strong encryption algorithm
  - A secret key known only to sender / receiver
- Mathematically:

  $Y = \mathrm{E}_K(X)$

  $X = \mathrm{D}_K(Y)$
- Assume encryption algorithm is known
- Implies a secure channel to distribute key

Classification of Cryptography

- Cryptographic system is characterized by:
  - Type of encryption operations used
    - Substitution / Transposition / Product
  - Number of keys used
    - Single-key or private / Two-key or public
  - Way in which plaintext (original message) is processed
    - Block / Stream

Cryptanalysis

- Objective is to recover key in use rather than simply to recover the plaintext (original message).
- There are two general approaches:
  - Cryptanalytic attack - Cryptanalytic attacks rely on the nature of the algorithm and some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs.
  - Brute-force attack - tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On an average, half of all possible keys must be tried to achieve success.

Cryptanalytic Attacks

- **Ciphertext only**
  - Only know algorithm & ciphertext, is statistical, know or can identify plaintext
- **Known plaintext**
  - Know/suspect plaintext & ciphertext
- **Chosen plaintext**
  - Select plaintext and obtain ciphertext
- **Chosen ciphertext**
  - Select ciphertext and obtain plaintext
- **Chosen text**
  - Select plaintext or ciphertext to en/decrypt

More Definitions

- **Unconditional security**
  - No matter how much computer power or time is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext.
- **Computational security**
  - Given limited computing resources (eg time needed for calculations is greater than age of universe), the cipher cannot be broken.

Brute Force Search

- Always possible to simply try every key
- Most basic attack, proportional to key size
- Assume either know / recognize plaintext

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/µs	Time required at 10^6 decryptions/µs
32	$2^{32} = 4.3 \times 10^9$	2^{31} µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} µs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} µs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} µs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} µs = 6.4×10^{12} years	6.4×10^6 years

Stream and Block ciphers

- An important distinction in symmetric cryptographic algorithms is between stream and block ciphers.
- Stream ciphers: coverts one symbol of plaintext directly into a symbol of ciphertext.
- Block ciphers: encrypts a group of plaintext symbols as one block.
- Simple substitution is an example of a stream cipher. Columnar transposition is a block cipher.
- Most modern symmetric encryption algorithms are block ciphers. Block sizes vary - 64 bits for DES (Data Encryption Standard) and 128 bits for AES (Advanced Encryption System).

Classical Substitution Ciphers

- Where letters of plaintext are replaced by other letters or by numbers or symbols
- Or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

Caesar Cipher

- Earliest known substitution cipher
- Used by Julius Caesar
- First attested use in military affairs
- Replaces each letter by 3rd letter on
- Example:

```
meet me after the toga party
PHHW PH DIWHU WKH WRJD SDUWB
```

Caesar Cipher

- Can define transformation as:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- Mathematically gives each letter a number

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- Then have Caesar cipher as:

  $c = \mathrm{E}(p) = (p + k) \bmod (26)$

  $p = \mathrm{D}(\mathrm{c}) = (\mathrm{c} - k) \bmod (26)$

Cryptanalysis of Caesar Cipher

- Only have 26 possible ciphers
  - A maps to A,B,..Z
- Could simply try each in turn
- A **brute force search**
- Given ciphertext, just try all shifts of letters
- Do need to recognize when have plaintext
- Example: break ciphertext "GCUA VQ DTGCM"

Monoalphabetic Cipher

- Rather than just shifting the alphabet
- Could shuffle (jumble) the letters arbitrarily
- Each plaintext letter maps to a different random ciphertext letter
- Hence, key is 26 letters long

```
Plain:  abcdefghijklmnopqrstuvwxyz
Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext:  ifwewishtoreplaceletters
Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA
```

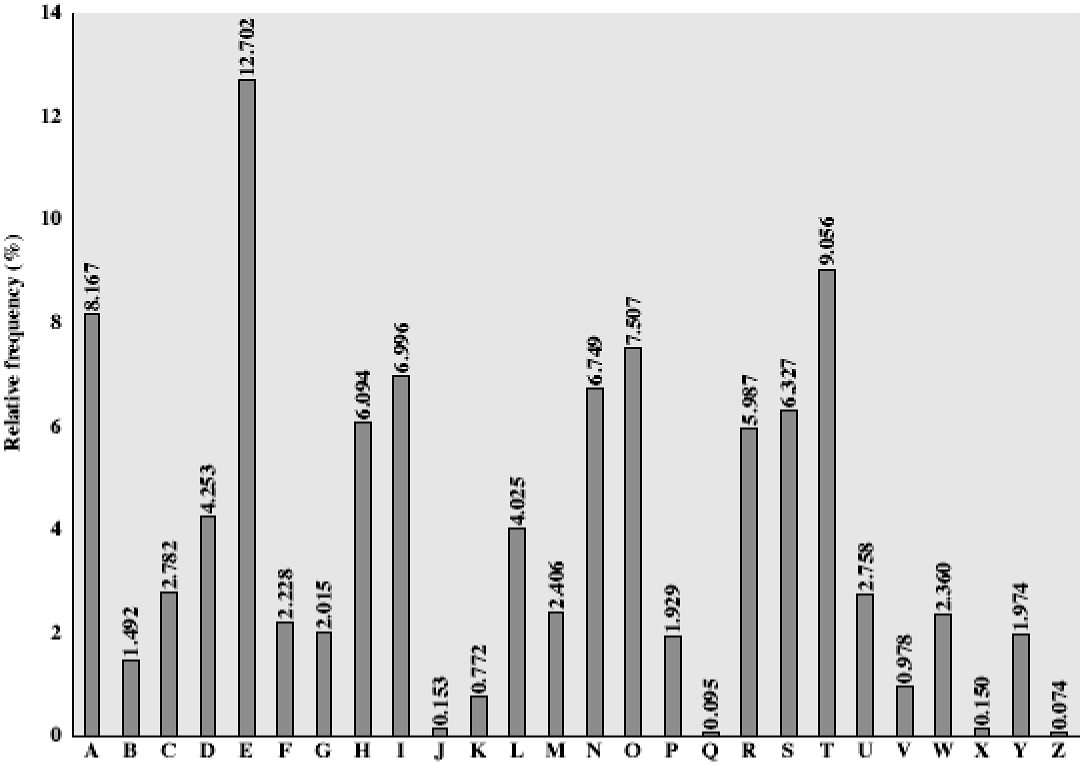
Monoalphabetic Cipher Security

- Now have a total of 26! = 4 x 1026 keys
- with so many keys, might think is secure
- But the problem is language characteristics

Language Redundancy and Cryptanalysis

- Human languages are **redundant: frequencies of occurrences of letters are not the same** in natural languages. Letters are not equally commonly used
- In English E is by far the most common letter
  - followed by T,R,N,I,O,A,S
- Other letters like Z,J,K,Q,X are fairly rare
- Have tables of single, double & triple letter frequencies for various languages

English Letter Frequencies



Use in Cryptanalysis

- Key concept - monoalphabetic substitution ciphers do not change relative letter frequencies
- Discovered by Arabian scientists in 9th century
- Calculates letter frequencies for ciphertext
- Compares counts/plots against known values
- If caesar cipher looks for common peaks/troughs
  - peaks at: A-E-I triple, NO pair, RST triple
  - troughs at: JK, X-Z
- For monoalphabetic must identify each letter
  - tables of common double/triple letters help

Example Cryptanalysis

- Given ciphertext:

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

- Count relative letter frequencies (see text)
- Guess P & Z are e and t
- Guess ZW is th and hence ZWP is the
- Proceeding with trial and error finally get:

```
it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow
```

Polyalphabetic Ciphers

- **Polyalphabetic substitution ciphers**
- Improves security using multiple cipher alphabets
- Makes cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- Uses a key to select which alphabet is used for each letter of the message
- Uses each alphabet in turn
- Repeats from start after end of key is reached

Transposition Ciphers

- Classical **transposition** or **permutation** ciphers: These will hide the message by rearranging the letter order without altering the actual letters used
- Can recognize these letters since they have the same frequency distribution as the original text

Row Transposition Ciphers

- A more complex transposition
- Write letters of message out in rows over a specified number of columns
- Then reorder the columns according to some key before reading off the rows

```
Key:        3 4 2 1 5 6 7
Plaintext:  a t t a c k p
            o s t p o n e
            d u n t i l t
            w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Product Ciphers

- Ciphers using substitutions or transpositions are not secure because of language characteristics
- Hence consider using several ciphers in succession to make harder, but:
  - Two substitutions make a more complex substitution
  - Two transpositions make more complex transposition
  - But a substitution followed by a transposition makes a new much harder cipher
- This is bridge from classical to modern ciphers

DES and AES

Origin of AES

Basic AES

Inside Algorithm

Final Notes

Origins

- A replacement for DES was needed
  - Key size is too small
- Can use Triple-DES – but slow, small block
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug 99

AES Competition Requirements

- Private key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES
- Provide full specification & design details
- Both C & Java implementations

AES Evaluation Criteria

- Initial criteria:
  - security – effort for practical cryptanalysis
  - cost – in terms of computational efficiency
  - algorithm & implementation characteristics

- Final criteria
  - general security
  - ease of software & hardware implementation
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)

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

- Found contrast between algorithms with
  - few complex rounds versus many simple rounds
  - Refined versions of existing ciphers versus new proposals

Rijndael: pronounce “Rain-Dahl”

The AES Cipher - Rijndael

- Rijndael was selected as the AES in Oct-2000
  - Designed by Vincent Rijmen and Joan Daemen in Belgium
  - Issued as FIPS PUB 197 standard in Nov-2001

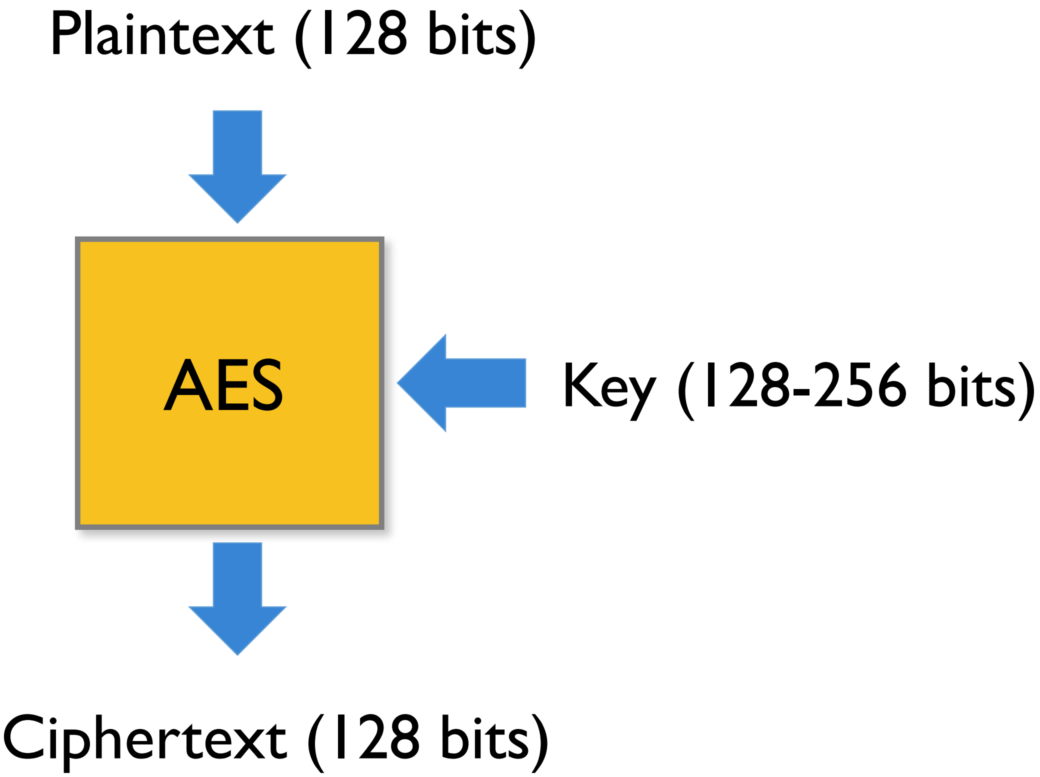
- An **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes (128 bits)
  - operates on entire data block in every round

- Rijndael design:
  - simplicity
  - has 128/192/256 bit keys, 128 bits data
  - resistant against known attacks
  - speed and code compactness on many CPUs

V. Rijmen

J. Daemen

AES Conceptual Scheme



Plaintext (128 bits)

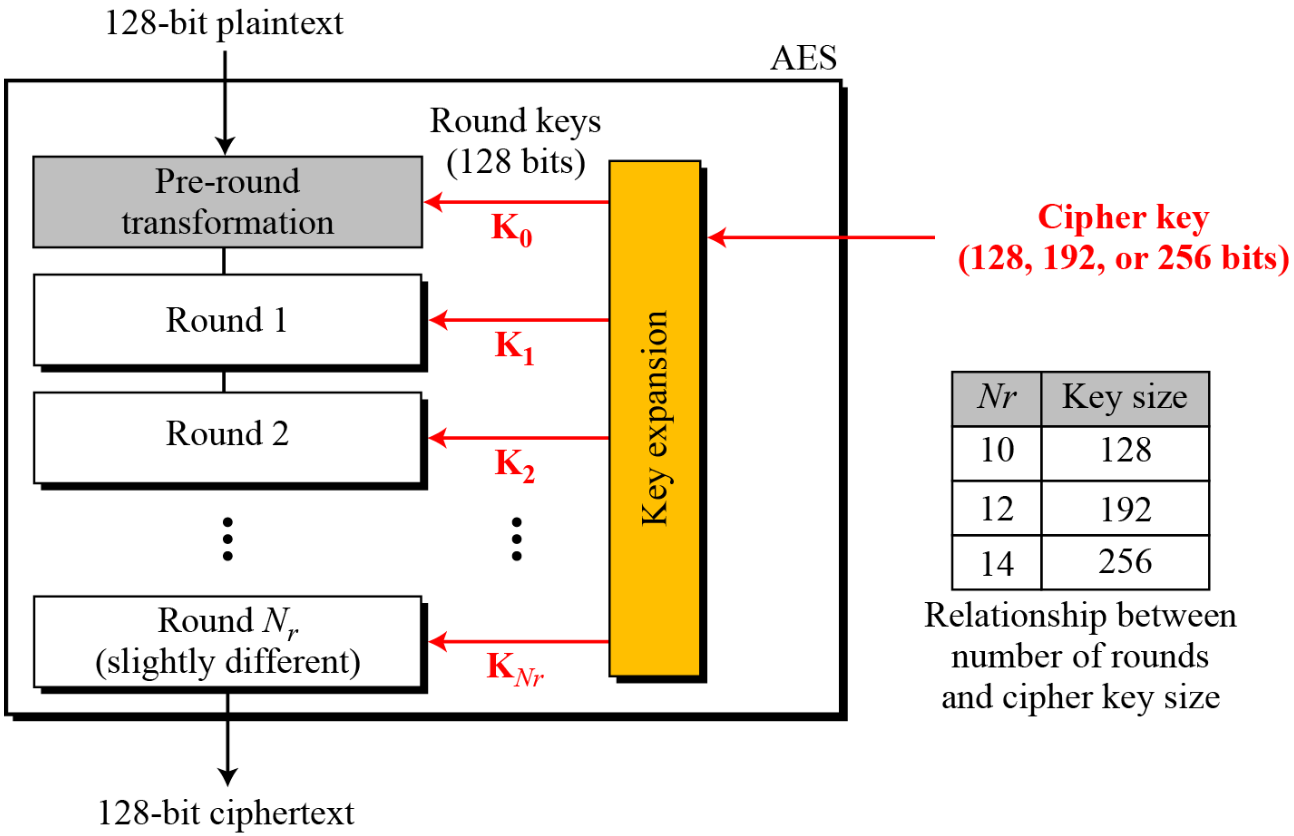
AES

Key (128-256 bits)

Ciphertext (128 bits)

Multiple rounds

- Rounds are (almost) identical
  - First and last round are a little different



Nr	Key size
10	128
12	192
14	256

Relationship between
number of rounds
and cipher key size

High Level Description

Key Expansion

- Round keys are derived from the cipher key using Rijndael's key schedule

Initial Round

- AddRoundKey : Each byte of the state is combined with the round key using bitwise xor

Rounds

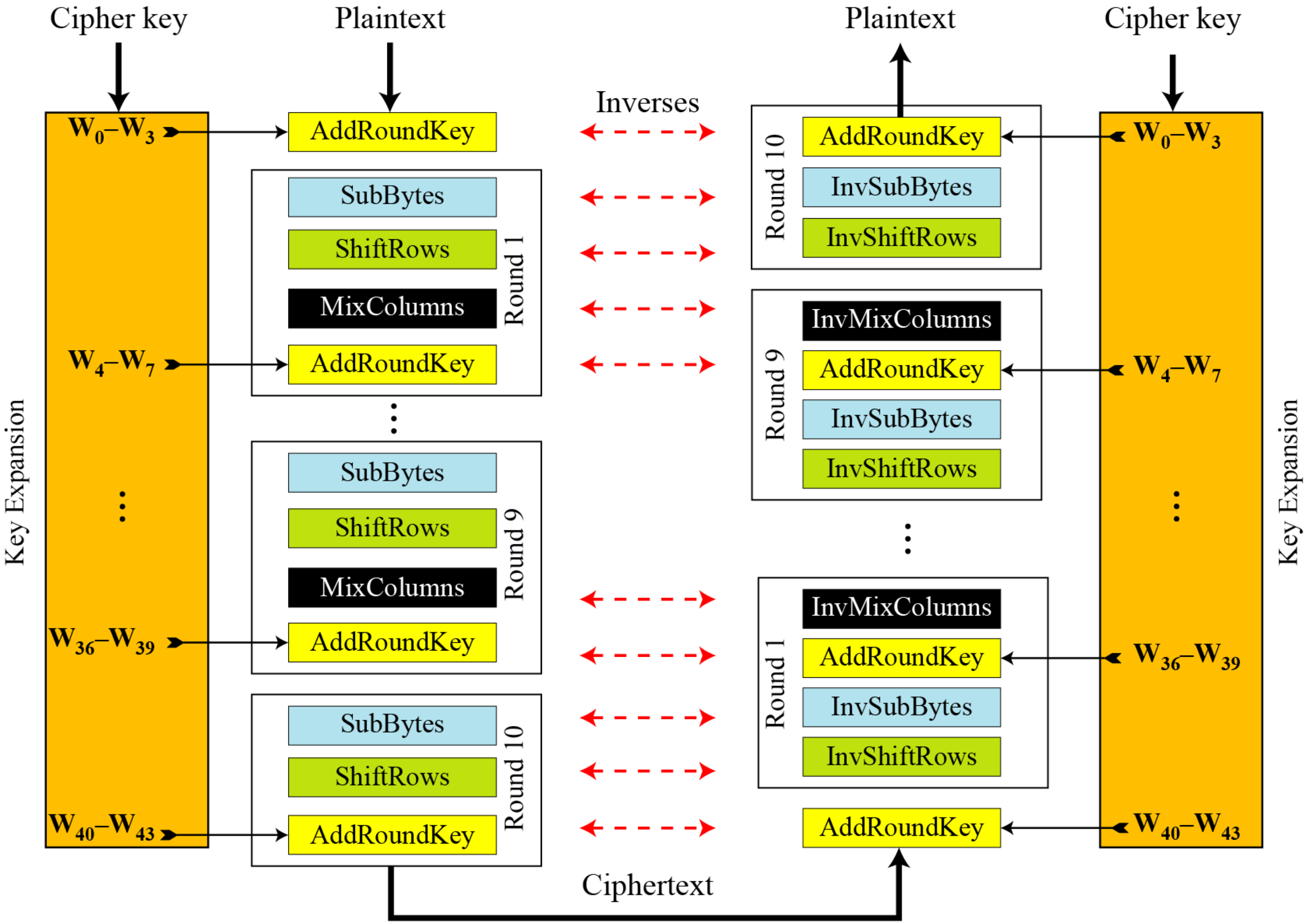
- SubBytes : non-linear substitution step
- ShiftRows : transposition step
- MixColumns : mixing operation of each column.
- AddRoundKey

Final Round

- SubBytes
- ShiftRows
- AddRoundKey

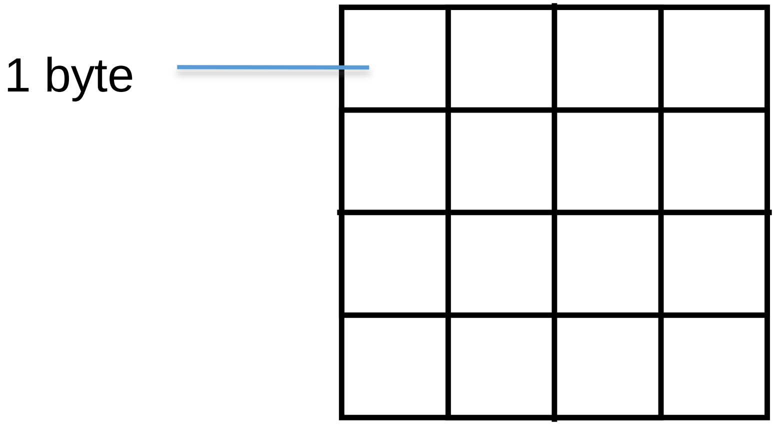
No MixColumns

Overall Structure

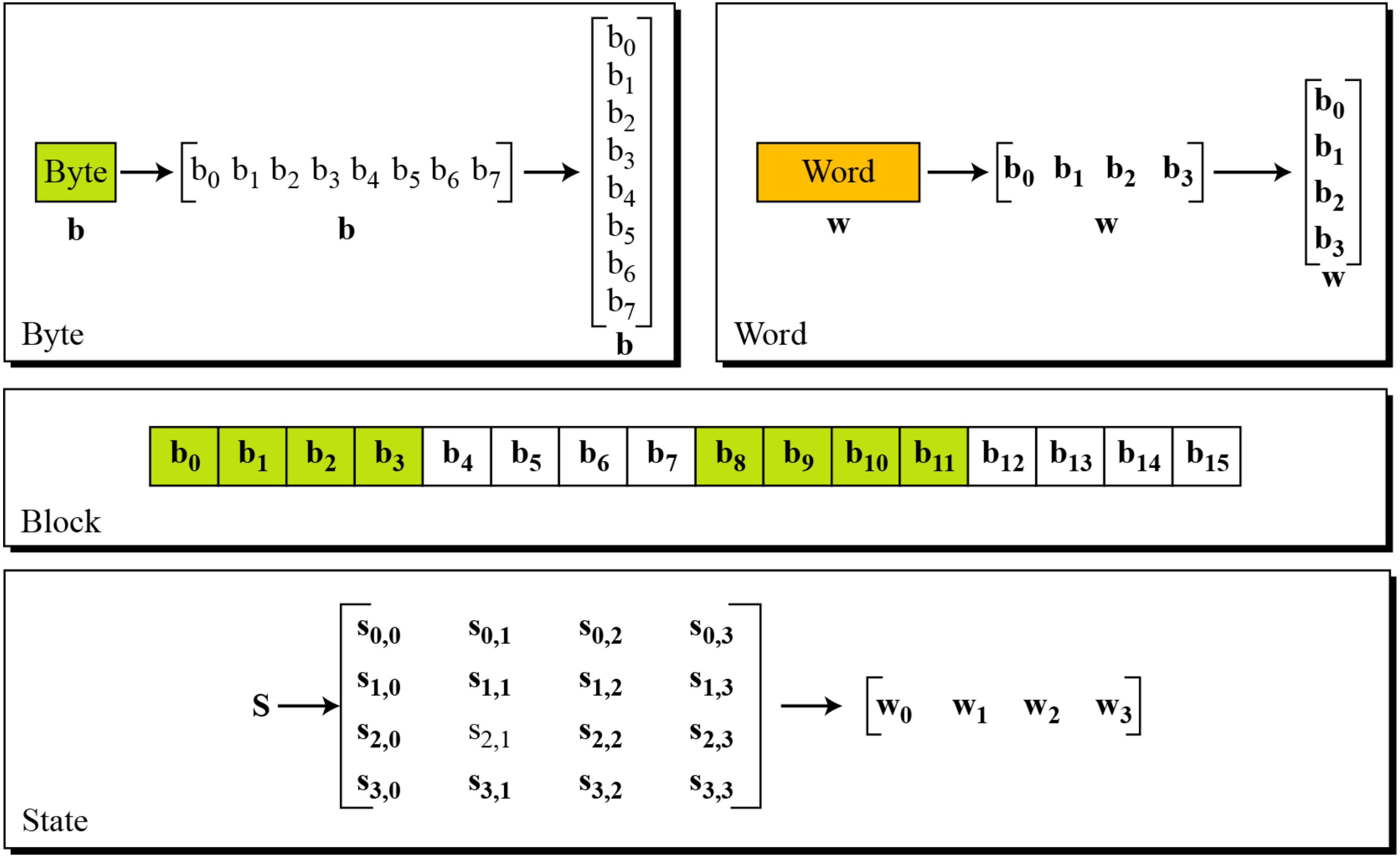


128-bit values

- Data block viewed as 4-by-4 table of bytes
- Represented as 4 by 4 matrix of 8-bit bytes.
- Key is expanded to array of 32 bits words



Data Unit



Byte

Byte $\rightarrow [b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7] \rightarrow \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$

b **b** **b**

Word

Word $\rightarrow [\mathbf{b_0}\ \mathbf{b_1}\ \mathbf{b_2}\ \mathbf{b_3}] \rightarrow \begin{bmatrix} \mathbf{b_0} \\ \mathbf{b_1} \\ \mathbf{b_2} \\ \mathbf{b_3} \end{bmatrix}$

w **w** **w**

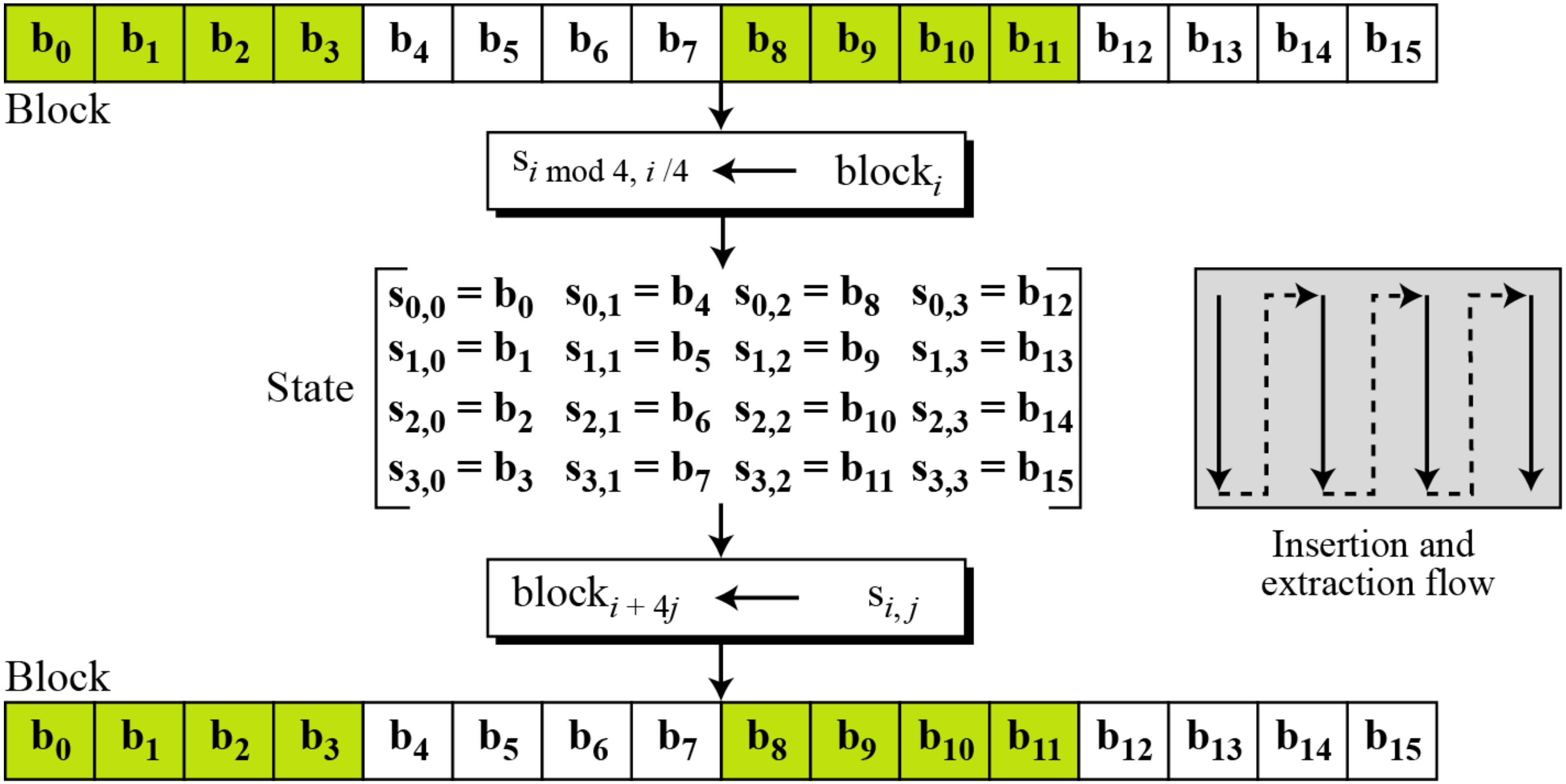
Block

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

State

$$\mathbf{S} \rightarrow \begin{bmatrix} \mathbf{s_{0,0}} & \mathbf{s_{0,1}} & \mathbf{s_{0,2}} & \mathbf{s_{0,3}} \\ \mathbf{s_{1,0}} & \mathbf{s_{1,1}} & \mathbf{s_{1,2}} & \mathbf{s_{1,3}} \\ \mathbf{s_{2,0}} & s_{2,1} & \mathbf{s_{2,2}} & \mathbf{s_{2,3}} \\ \mathbf{s_{3,0}} & \mathbf{s_{3,1}} & \mathbf{s_{3,2}} & \mathbf{s_{3,3}} \end{bmatrix} \rightarrow [\mathbf{w_0}\ \ \mathbf{w_1}\ \ \mathbf{w_2}\ \ \mathbf{w_3}]$$

Unit Transformation



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

$s_{i \bmod 4,\, i/4} \leftarrow \text{block}_i$

State

$$\begin{bmatrix} s_{0,0} = b_0 & s_{0,1} = b_4 & s_{0,2} = b_8 & s_{0,3} = b_{12} \\ s_{1,0} = b_1 & s_{1,1} = b_5 & s_{1,2} = b_9 & s_{1,3} = b_{13} \\ s_{2,0} = b_2 & s_{2,1} = b_6 & s_{2,2} = b_{10} & s_{2,3} = b_{14} \\ s_{3,0} = b_3 & s_{3,1} = b_7 & s_{3,2} = b_{11} & s_{3,3} = b_{15} \end{bmatrix}$$

$\text{block}_{i+4j} \leftarrow s_{i,j}$

Block

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}

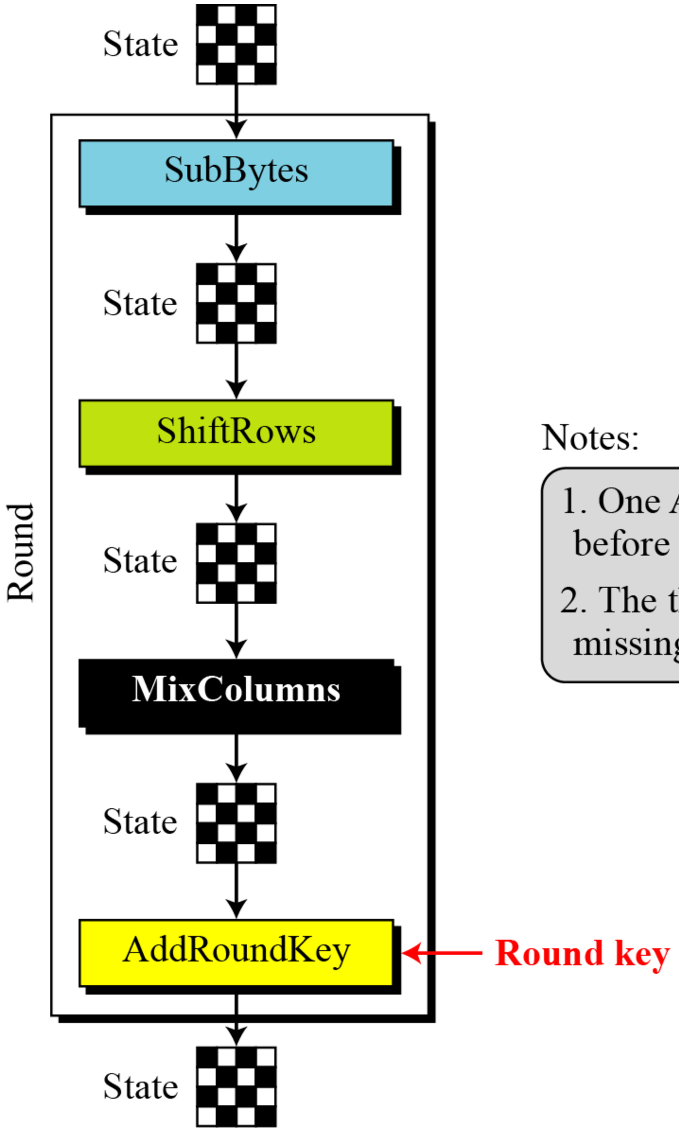
Insertion and extraction flow

Changing Plaintext to State

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

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

Details of Each Round



Notes:

1. One AddRoundKey is applied before the first round.

2. The third transformation is missing in the last round.

SubBytes: Byte Substitution

- A simple substitution of each byte
  - provide a confusion

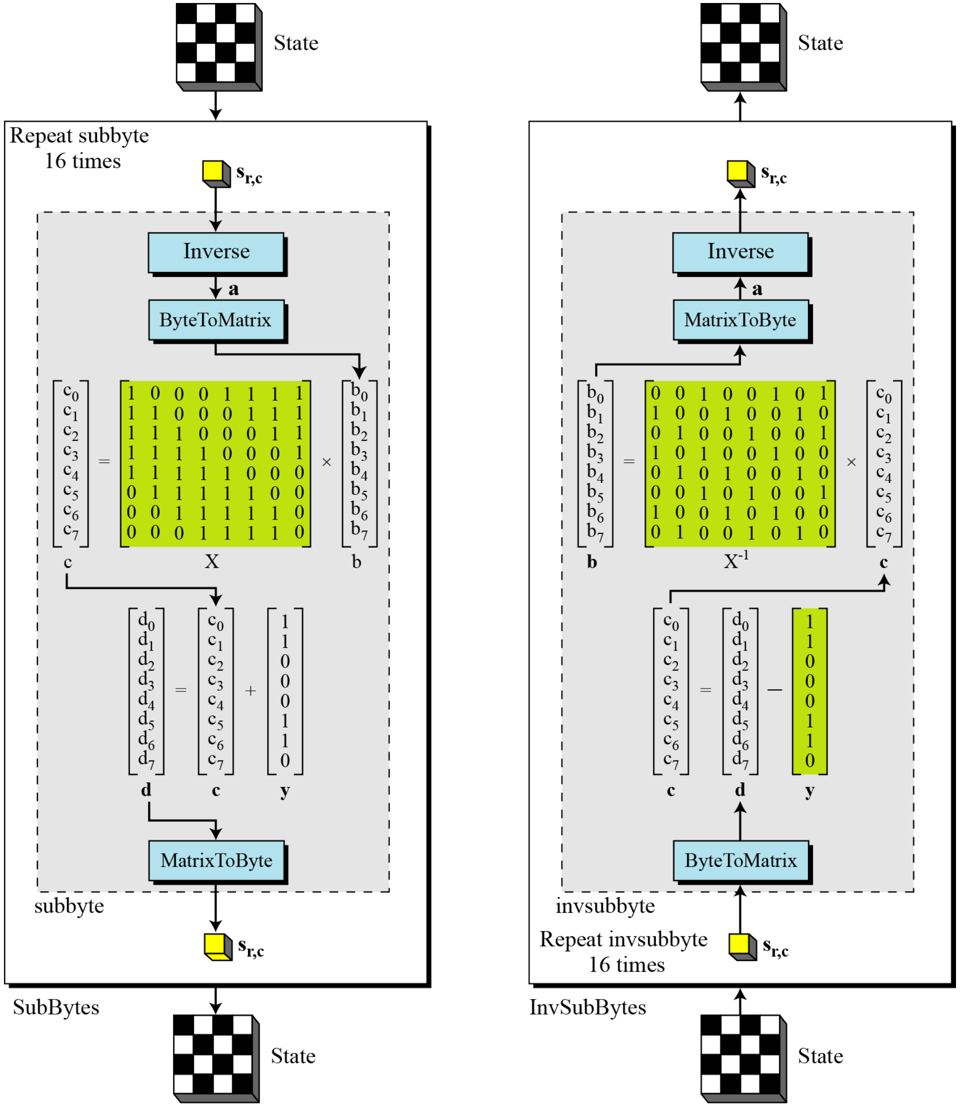
- Uses one S-box of 16x16 bytes containing a permutation of all 256 8-bit values

- Each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}

- S-box constructed using defined transformation of values in Galois Field-GF(2^8)

Galois : pronounce “Gal-Wa”

SubBytes and InvSubBytes



$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

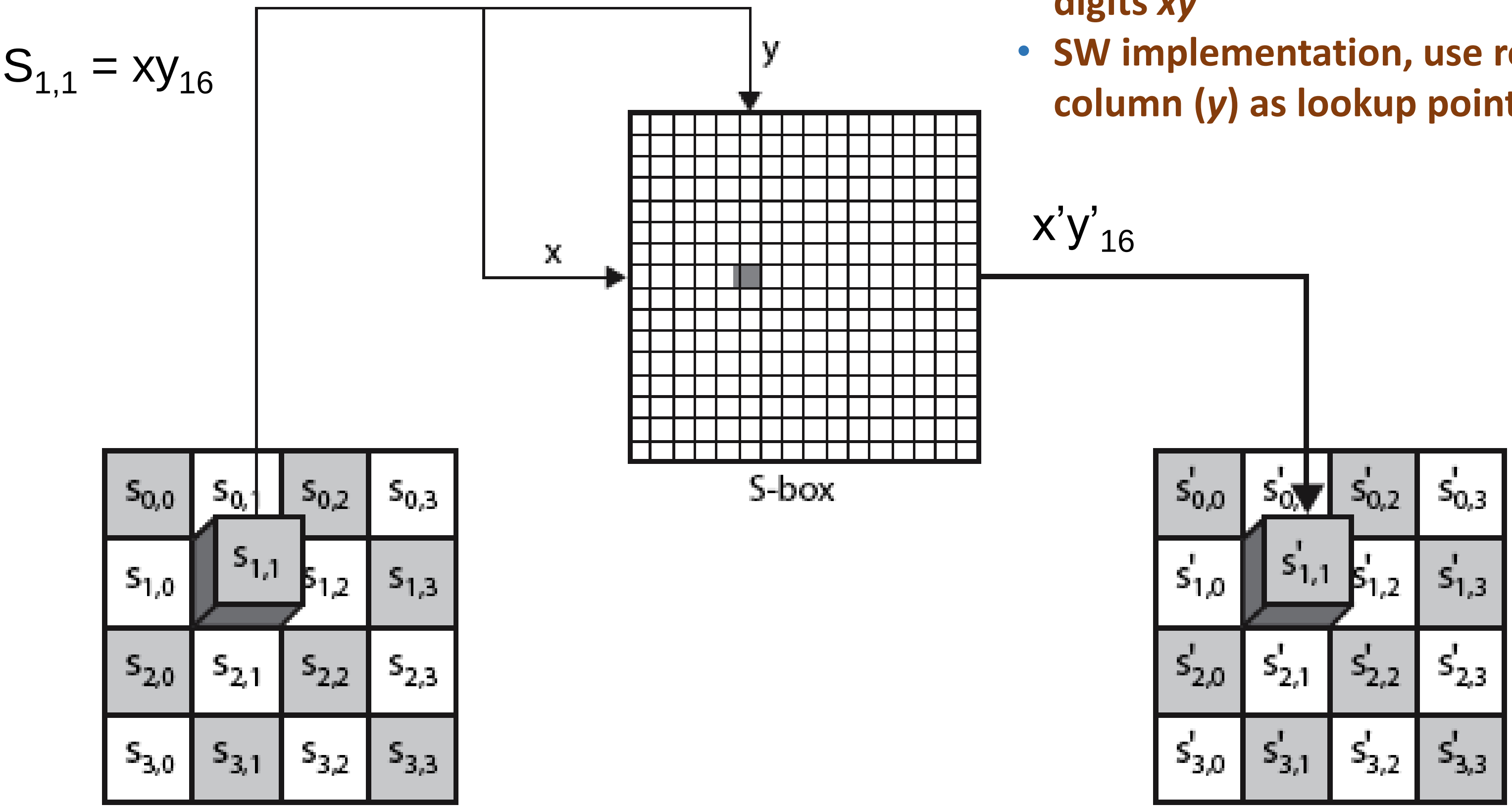
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

SubBytes Operation

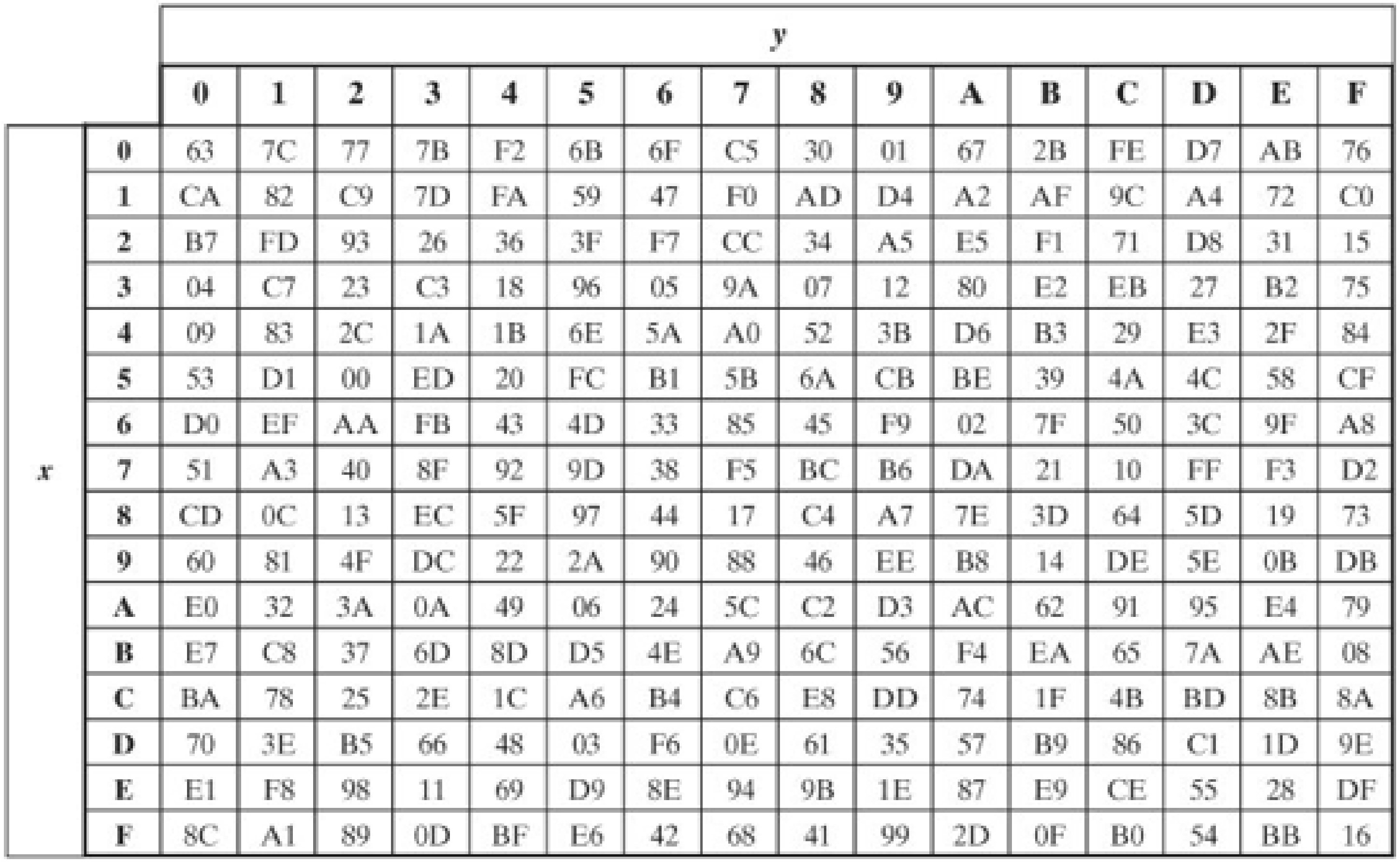
- The SubBytes operation involves 16 independent byte-to-byte transformations.

- **Interpret the byte as two hexadecimal digits *xy***
- **SW implementation, use row (*x*) and column (*y*) as lookup pointer**



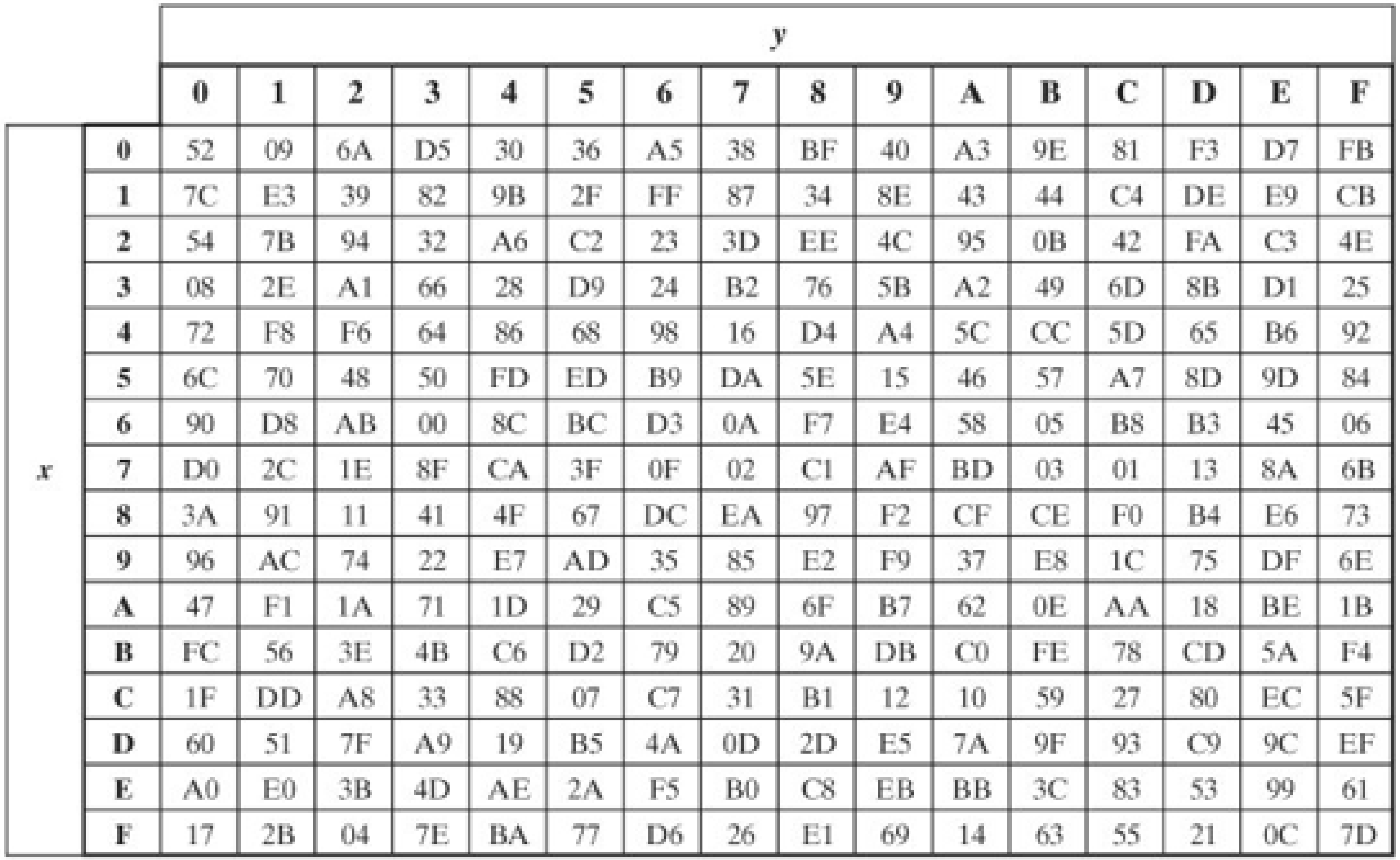
SubBytes Table

- Implement by Table Lookup



		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

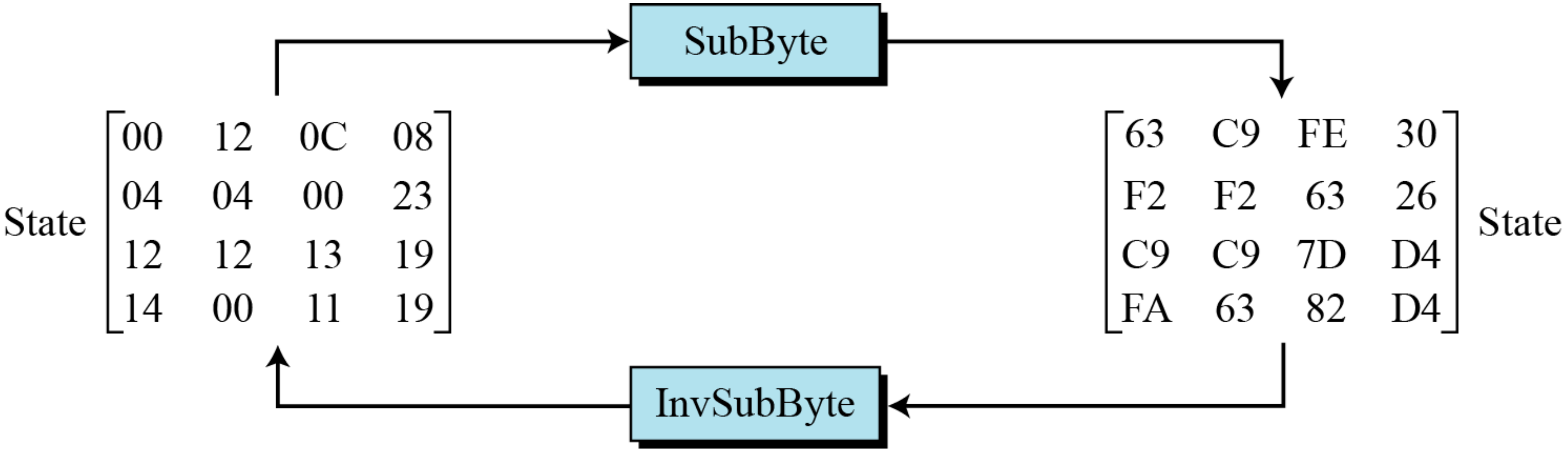
InvSubBytes Table



		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

Sample SubByte Transformation

- The SubBytes and InvSubBytes transformations are inverses of each other.



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

$$\begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & F2 & 63 & 26 \\ C9 & C9 & 7D & D4 \\ FA & 63 & 82 & D4 \end{bmatrix}\text{State}$$