Dr. Ammar Haider
Assistant Professor
School of Computing

CS3002 Information Security

Modern Symmetric Ciphers

Source: Stallings CNS chap 4, 6

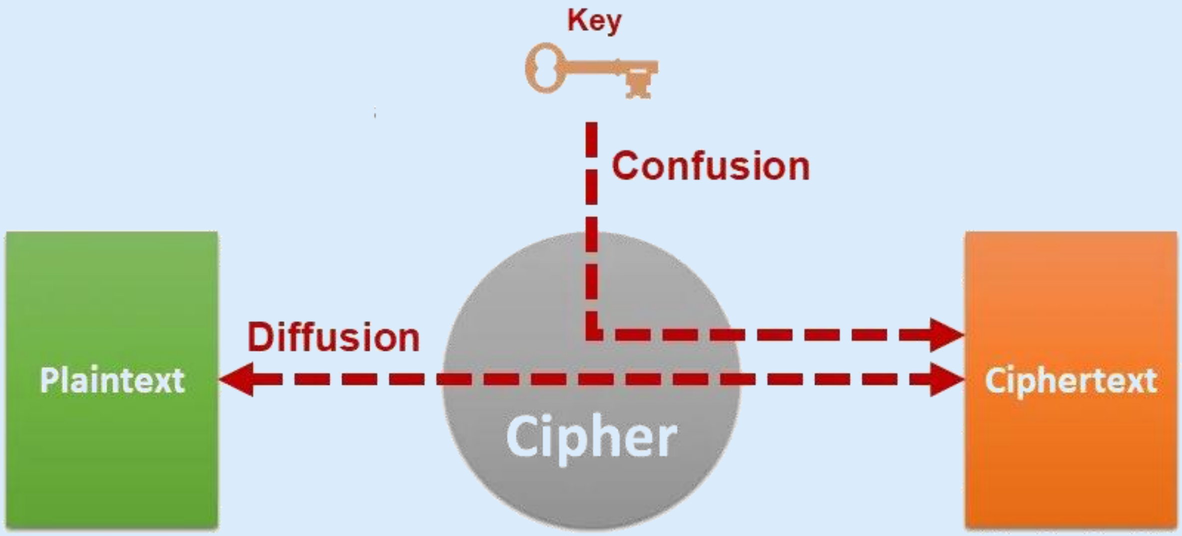
Modern Ciphers

"A modern block cipher can be designed to act as a substitution cipher or a transposition cipher. This is the same idea as is used in traditional ciphers, except that the symbols to be substituted or transposed are bits instead of characters."

Forouzan, Cryptography & Network Security

Strength of a cipher

- A cipher needs to completely obscure statistical properties of original message
- In 1945, Shannon suggested that a cipher needs to have these two properties:



Confusion

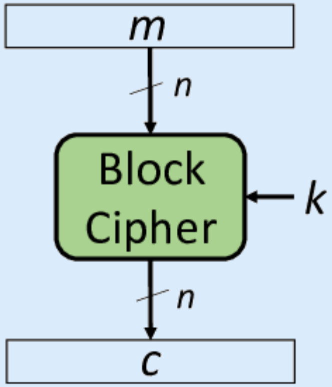
- Confusion means that each character (e.g. bit) of the ciphertext should depend on several parts of the key, obscuring the connection between the two.
- This property makes it difficult to find key from ciphertext. If a single bit in key is changed, calculation of values of most or all bits in ciphertext will be affected.
  - In classical ciphers, substitution provides confusion

Diffusion

- Diffusion means the influence of one plaintext symbol is spread over many ciphertext symbols, so that statistical properties of plaintext remain hidden.
  - In classical ciphers, permutation provides diffusion
- Ciphers that only perform confusion or only diffusion are not secure enough.
  - Except for one-time pad, which is confusion only; but as discussed, it is impractical.
- A product cipher is the one which includes both of these operations.

Block & Stream ciphers

- **Block ciphers** break messages into blocks of predetermined sizes – e.g. 64 bits or 128 bits.
  - Cipher encrypts one block of plaintext, creating a ciphertext block of same size.
  - Each character (e.g. bit) contributes to encryption of other characters in the block

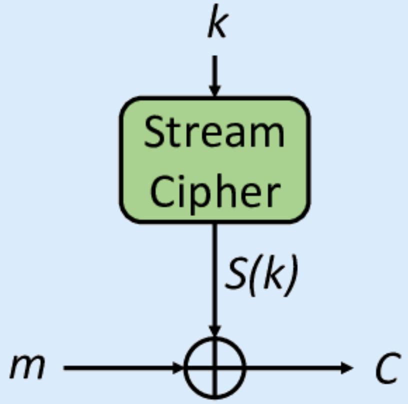


https://ieeexplore.ieee.org/document/9093670/

Block & Stream ciphers

- In **stream ciphers**, encryption is done on much smaller unit (bit, byte etc.)
  - Units in plaintext are fed into encryption algorithm as a stream (one at a time), ciphertext is similarly outputted as a stream.
  - They are more suitable when data is a continuous stream

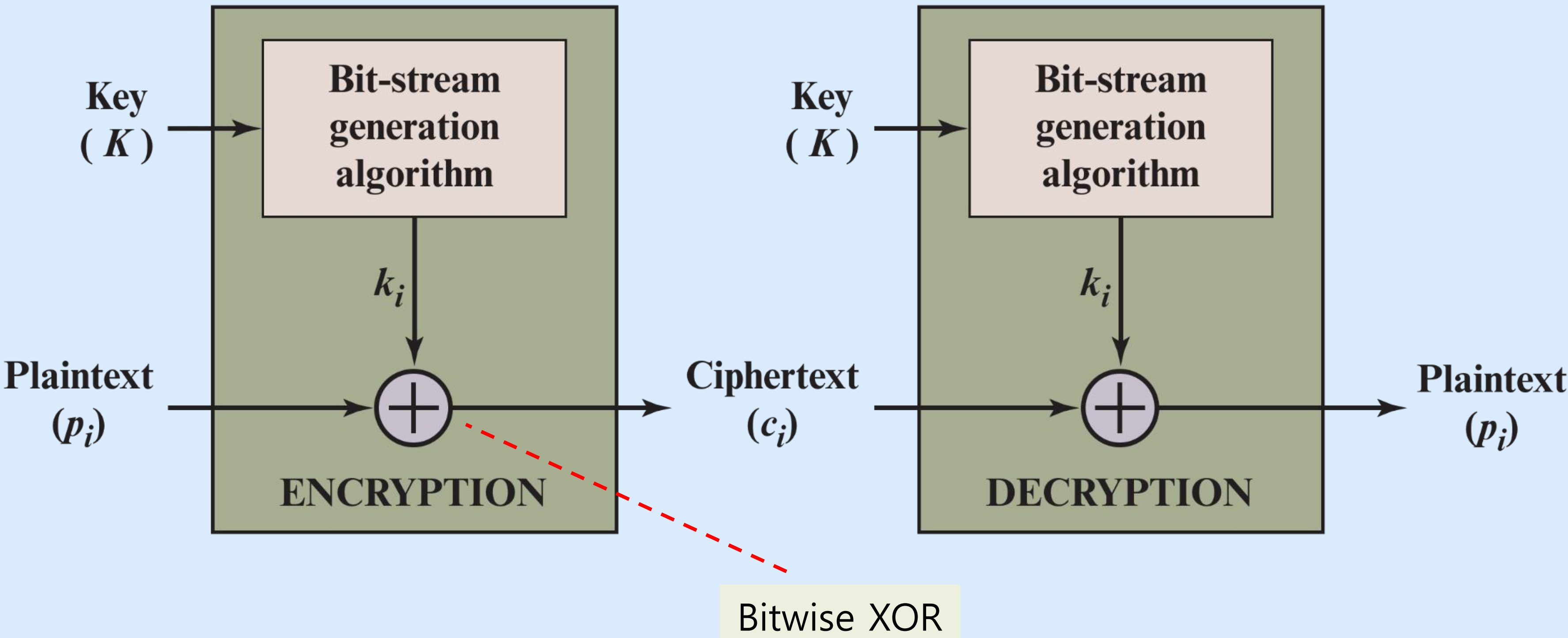
Stream ciphers *do not provide* diffusion, since there is a one-to-one mapping between units in plaintext and ciphertext.



https://ieeexplore.ieee.org/document/9093670/

Block & Stream ciphers

Stream ciphers high level structure



XOR operation: extremely useful in crypto

XOR

0	0	0
0	1	1
1	0	1
1	1	0

AND

0	0	0
0	1	0
1	0	0
1	1	$\boxed{1}$

OR

0	0	$\boxed{0}$
0	1	1
1	0	1
1	1	1

Information Leak!

- Primary reason: It does not reveal any information about plaintext (both 0 and 1 are equally likely).
- Secondary advantage: Both encryption and decryption can be done the same way, by adding the key. XOR is a self-inverse function!

$$\underbrace{010010}_{\text{plaintext}} \oplus \underbrace{101100}_{\text{key}} = \underbrace{111110}_{\text{ciphertext}}$$

$$\underbrace{111110}_{\text{ciphertext}} \oplus \underbrace{101100}_{\text{key}} = \underbrace{010010}_{\text{plaintext}}$$

Modern Encryption Algorithms

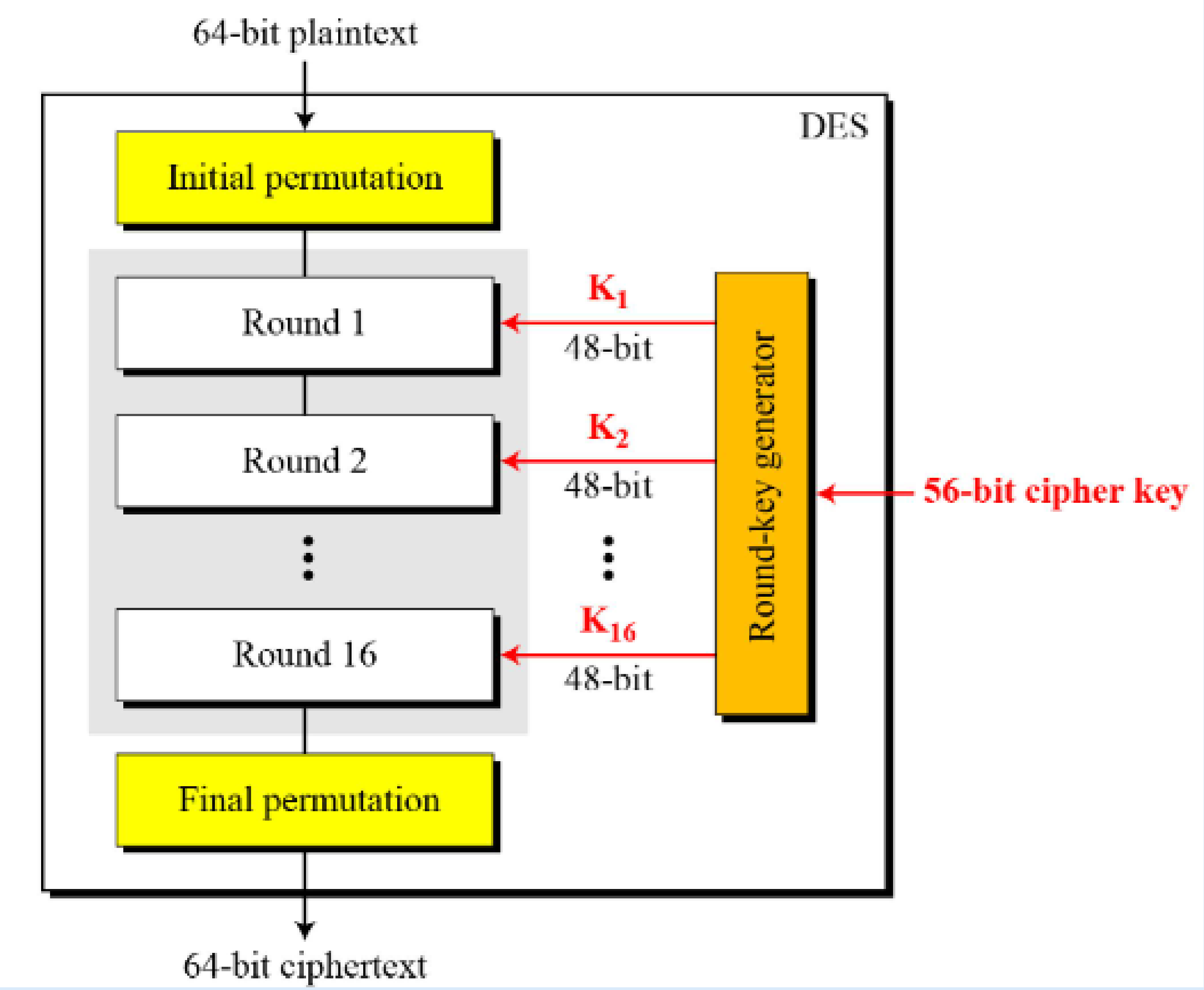
Examples

- Block Ciphers
  - DES
  - 3DES
  - AES
  - Twofish
- Stream Ciphers
  - RC4
  - ChaCha20

Data Encryption Standard

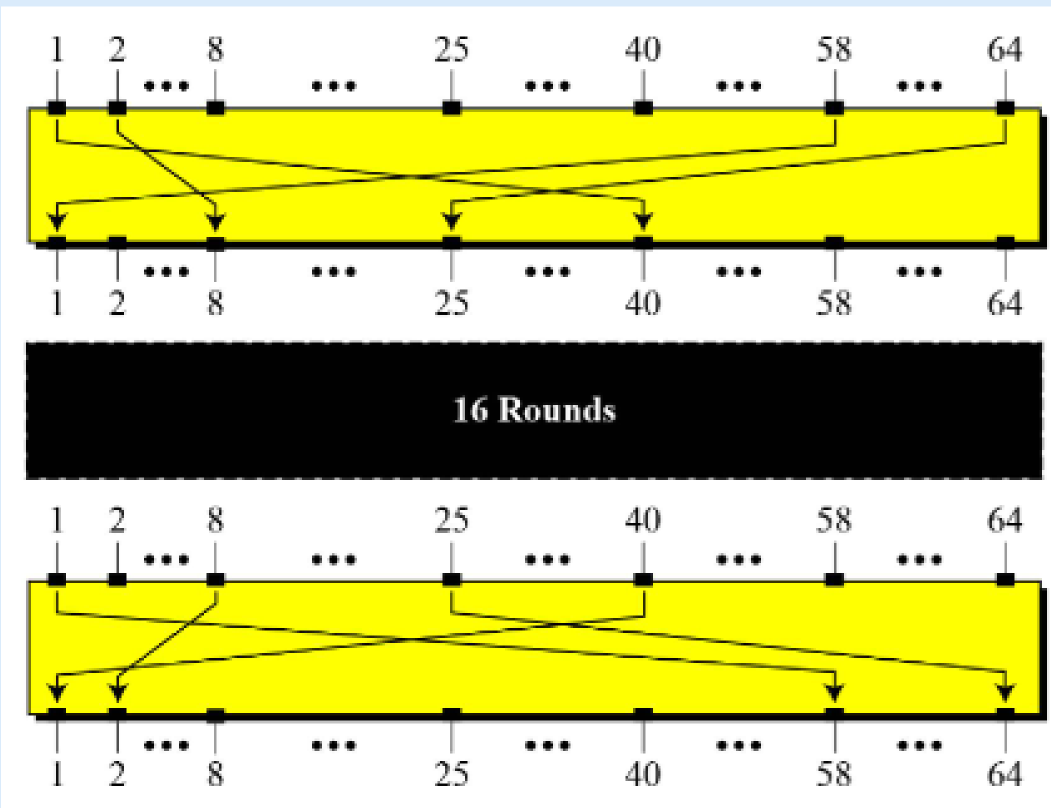
- DES was once the most widely used encryption scheme
  - developed in early 1970s at IBM, slightly tweaked by NSA
  - the first modern, public, freely available encryption algorithm
  - Standardized by National Bureau of Standards in 1977
- Uses 64 bit plaintext block and 56 bit key to produce a 64 bit cipher text block

DES Algorithm



DES Algorithm

Initial and final permutation (inverse of each other)

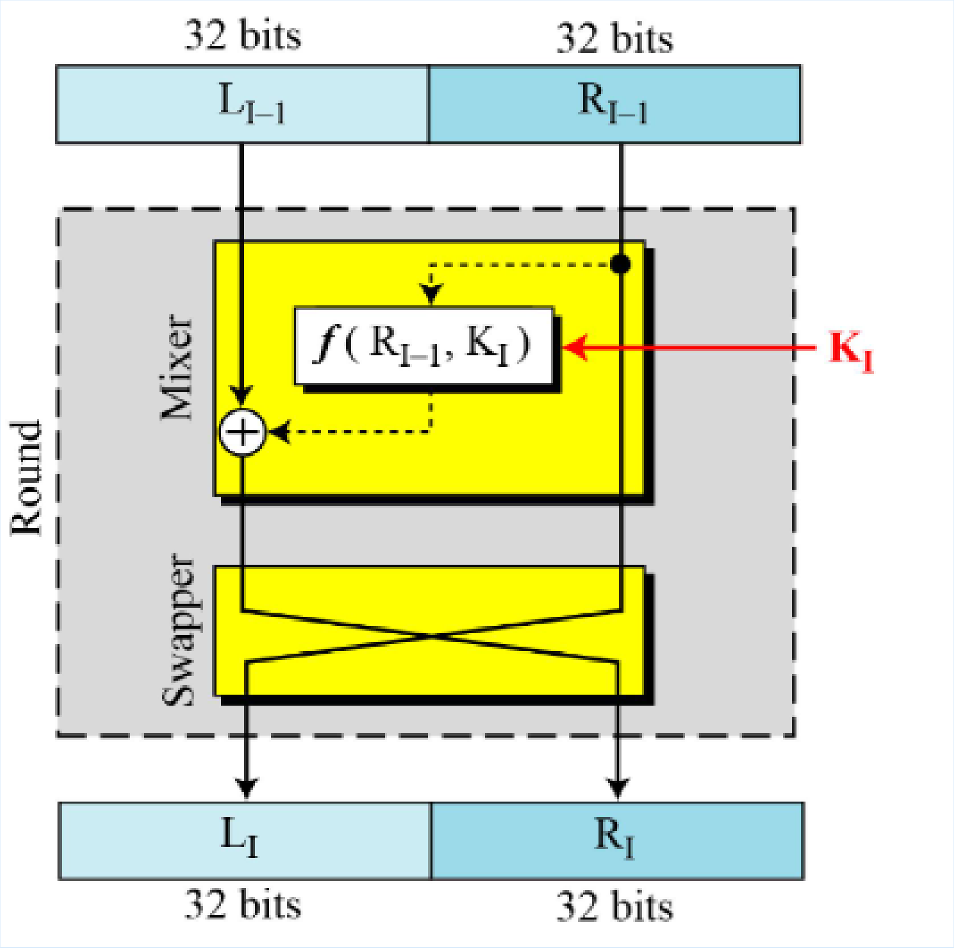


Initial Permutation							
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Final Permutation							
40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

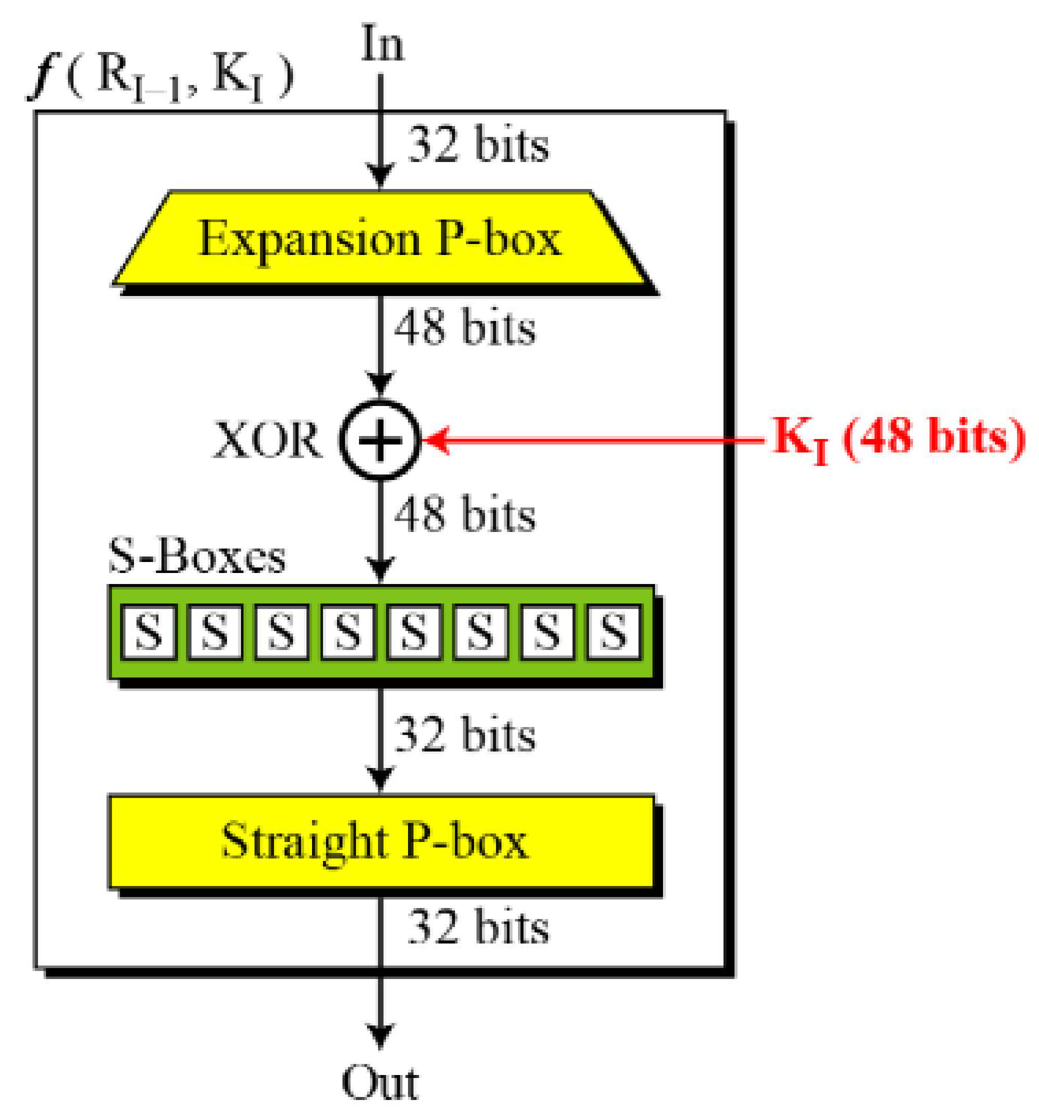
DES Round

One round in DES (Feistel structure)



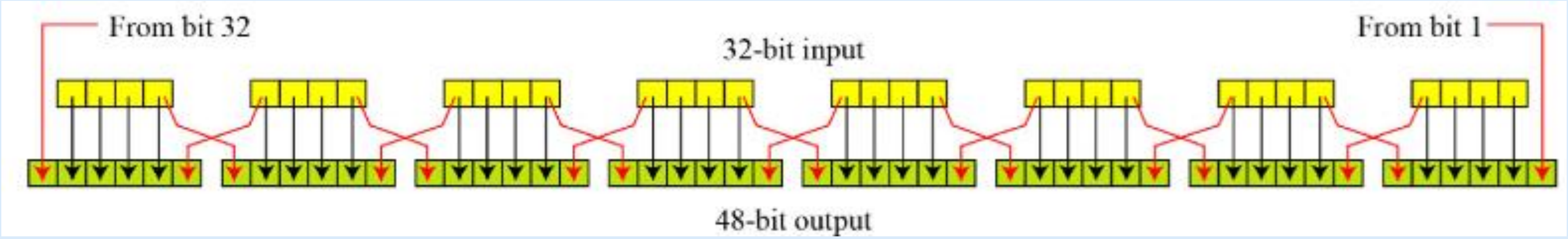
f(…) is the DES function

DES Function



DES Function: Expansion

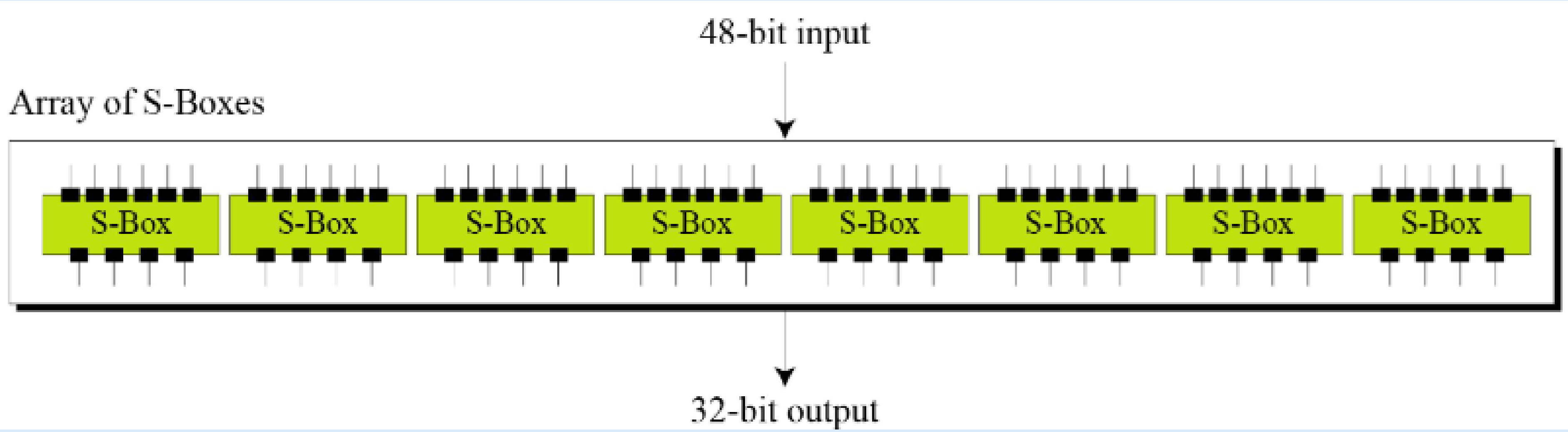
Expansion Mechanism



Expansion P-box table

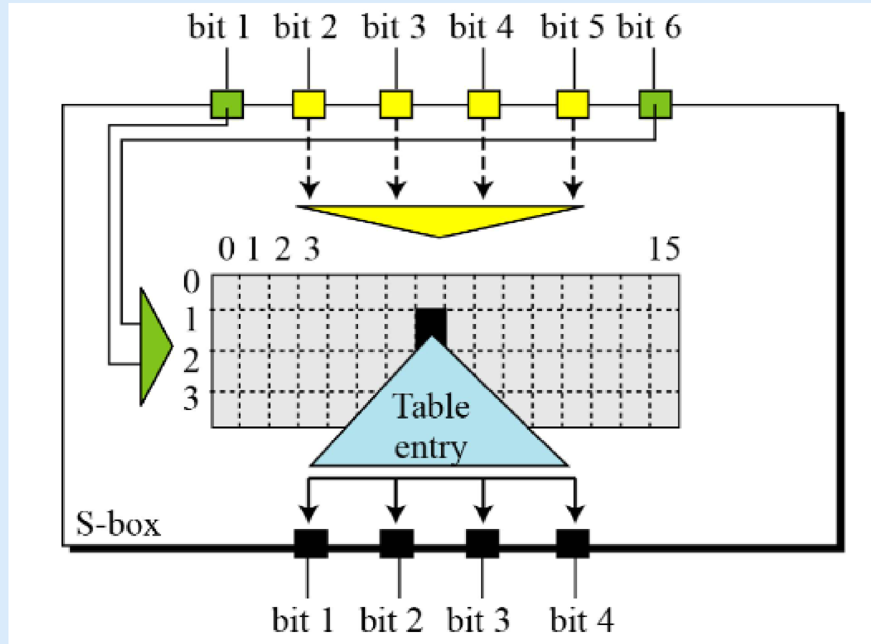
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

DES Function: S-Boxes



All 8 S-Boxes have same working principle, but different substitution tables

DES Function: S-Box-1



Example: If input is 101100_2
We check row 2 (10_2) and column 6 (0110_2) to get output 2, which is 0010_2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Find the output if S-Box-1 input is 100011

DES Function: S-Boxes

S-Box-2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

S-Box-3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

DES Function: Straight P-Box

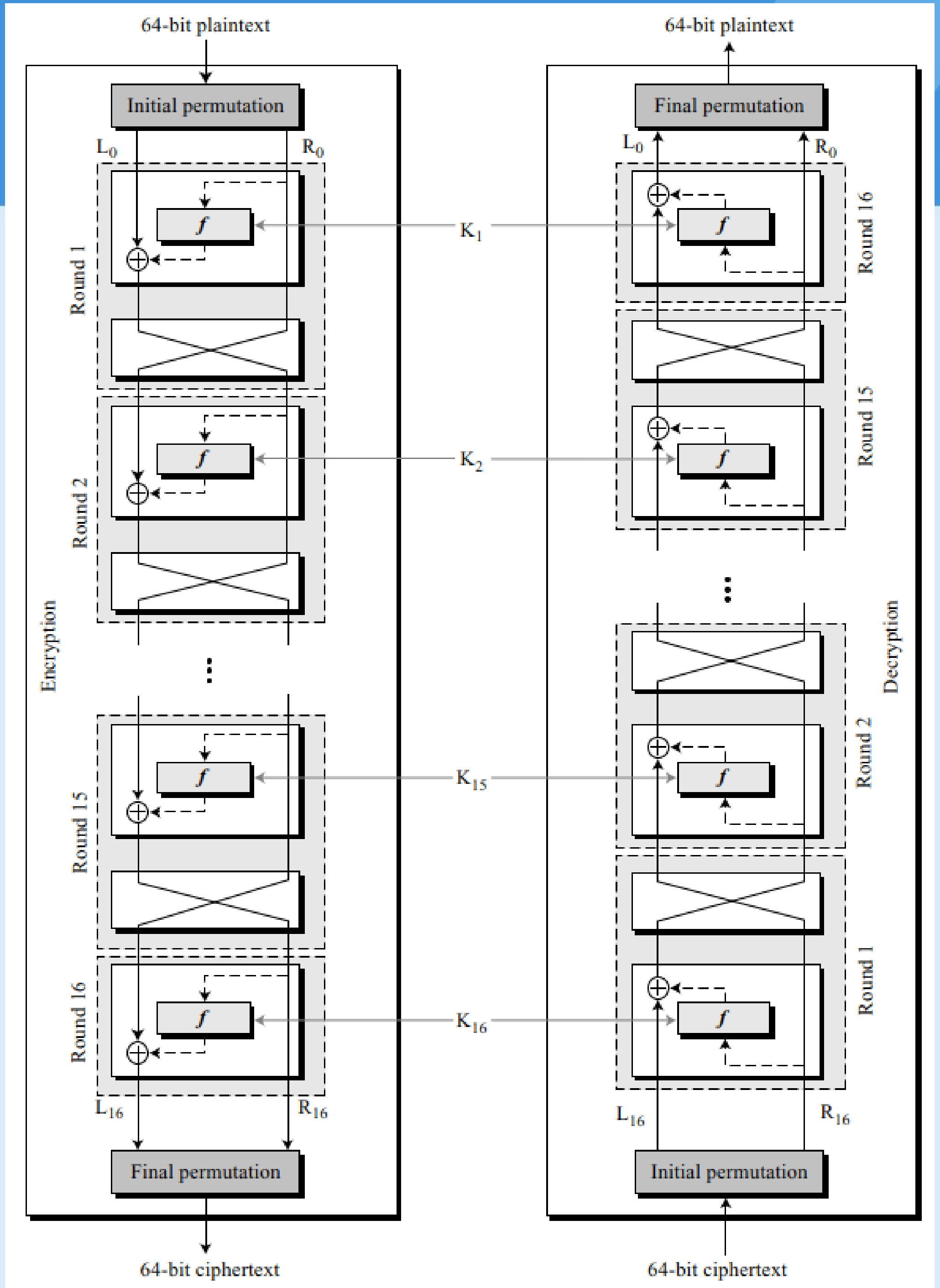
16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

32-bit input to 32-bit output

16th bit of input becomes 1st bit of output, and so on.

DES Decryption

- During encryption, skip the swapper in the last round.
- For decryption, go through the same procedure as encryption (i.e. initial permutation, 16 rounds, final permutation), just use the round keys in reverse order
  - this methodology is common in all Fiestel ciphers



DES Strength Analysis

- DES is probably the most-scrutinized encryption algorithm, and to date no substantial mathematical weakness has been discovered.

- BUT… it is not resistant to brute force attacks.

DES Strength Analysis

Short key length

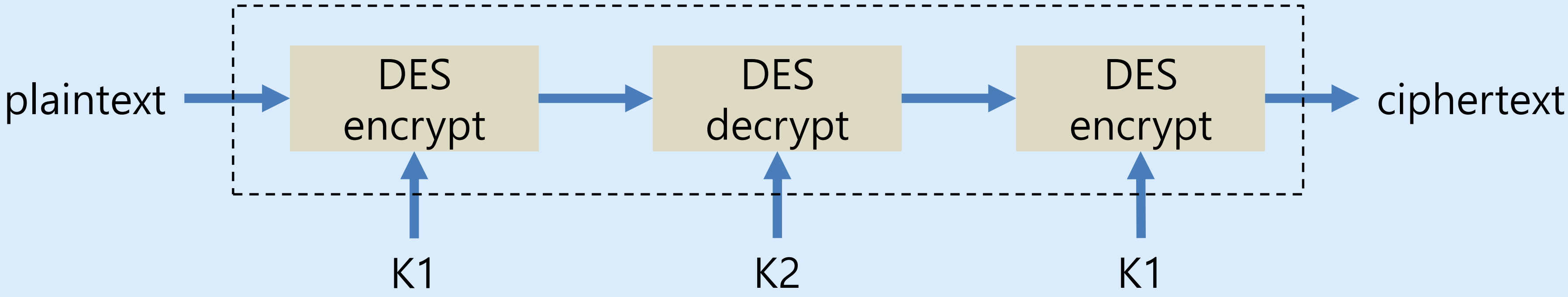
- With 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16} keys, which can be broken in relatively short time by brute forcing.

Chronology of DES Cracking	
Broken for the first time	1997
Broken in 56 hours	1998
Broken in 22 hours and 15 minutes	1999
Capable of broken in 5 minutes	2021

- More powerful and parallelized hardware yields quicker results
- Within 26 hours when using a specialized hardware: https://crack.sh

Triple DES (3DES)

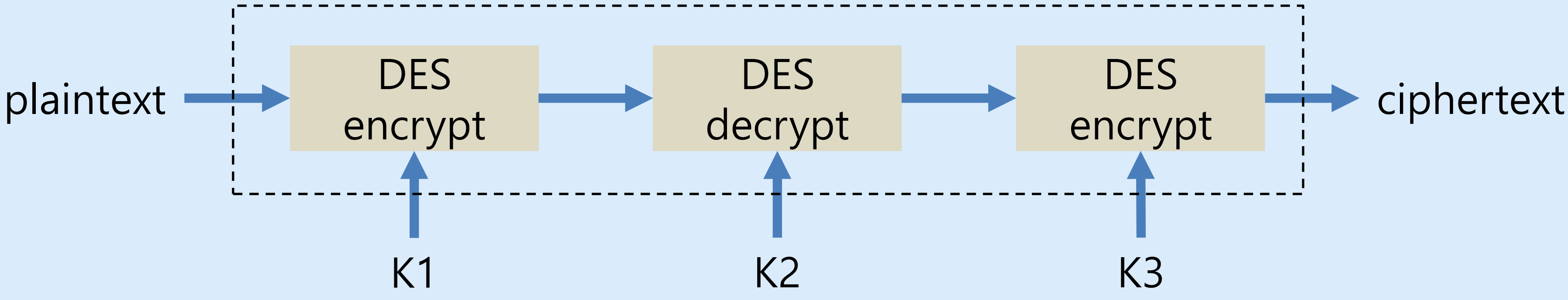
- Repeats basic DES algorithm three times
- Using either two or three unique keys
  - key size of 112 or 168 bits.
- much more secure but also much slower



With **two** keys, Effective key length = 112 bits

Triple DES (3DES)

- Repeats basic DES algorithm three times
- Using either two or three unique keys
  - key size of 112 or 168 bits.
- much more secure but also much slower



With **three** keys, Effective key length = 168 bits

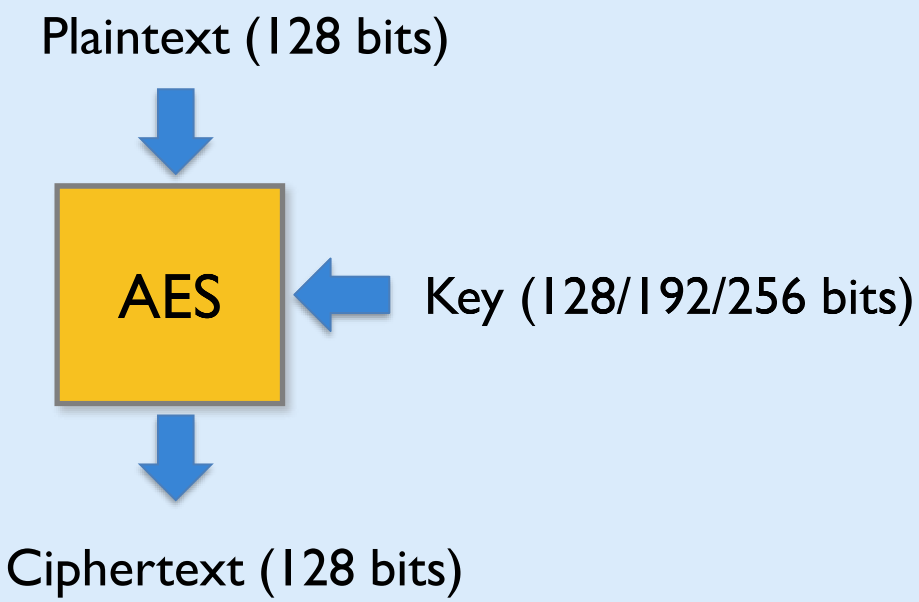
Advanced Encryption Standard

- NIST called for proposals in 1997
  - goals: efficiency, security, HW/SW suitability
  - key length 128, 192, 256 bits
  - Selected Rijndael in Nov 2001 and declared as ‘AES’

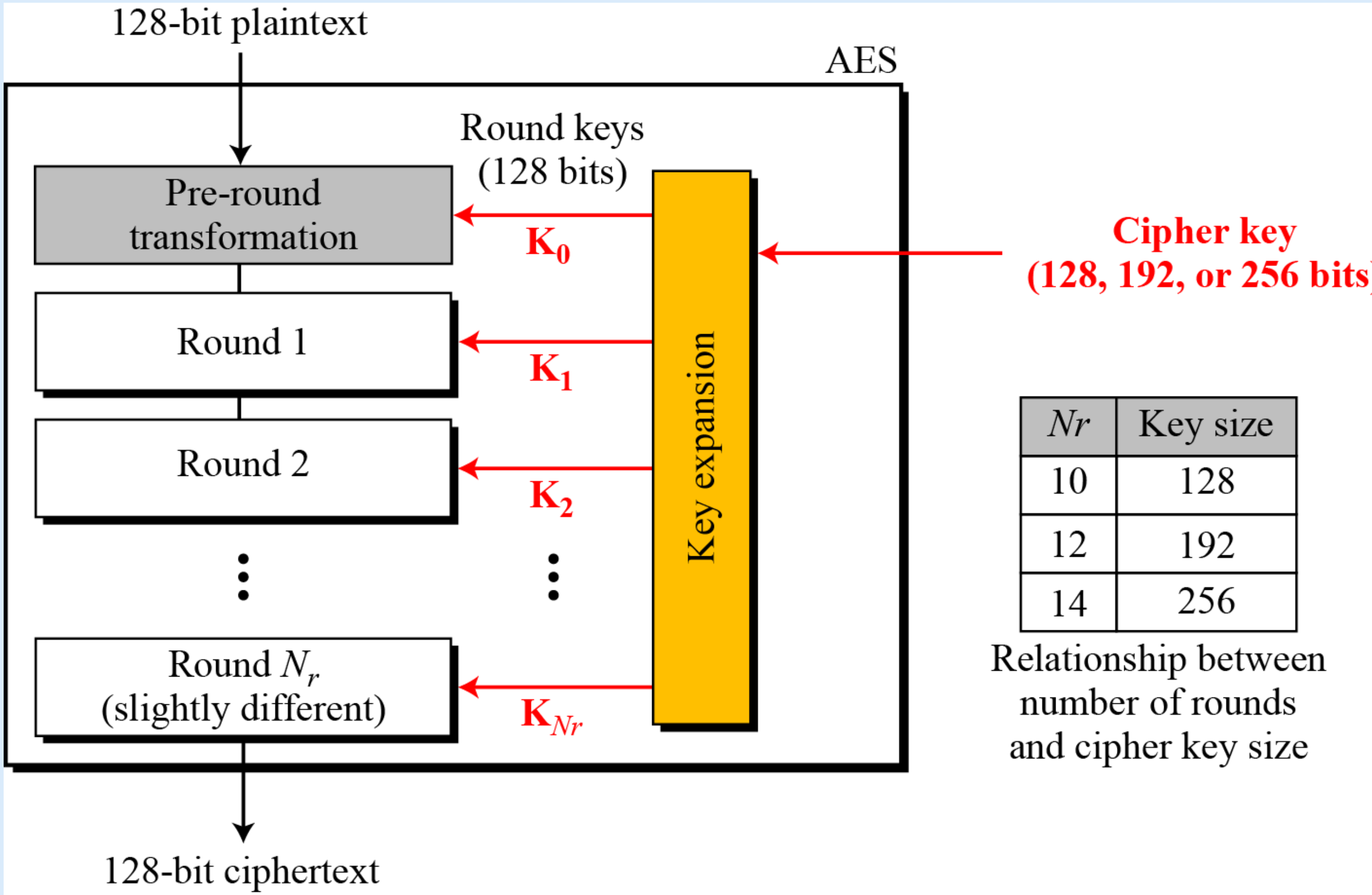
- Symmetric block cipher
- Uses 128 bit data block & 128/192/256 bit keys
- Now widely available commercially

The AES Cipher - Rijndael

- 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



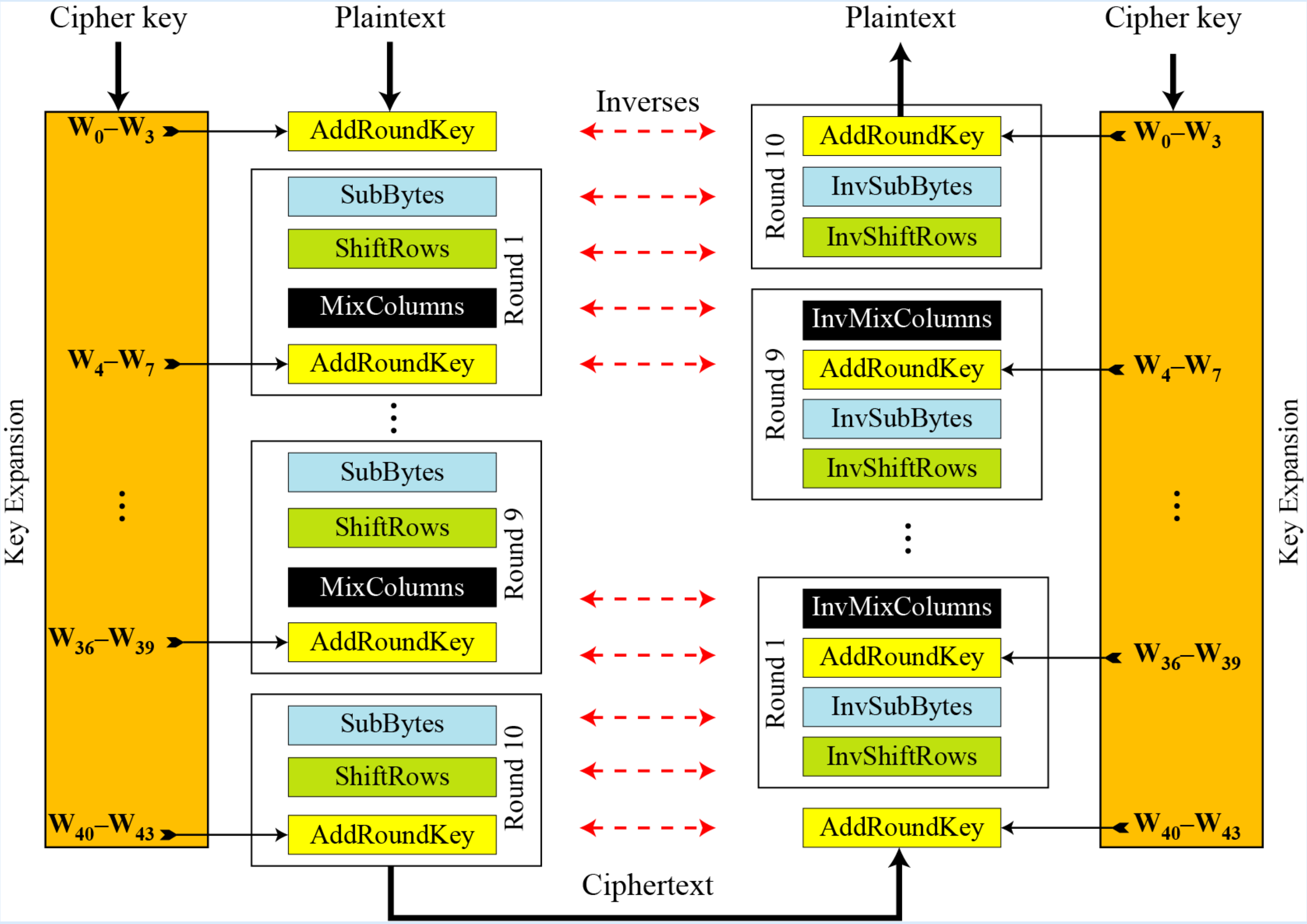
Multiple rounds



Nr	Key size
10	128
12	192
14	256

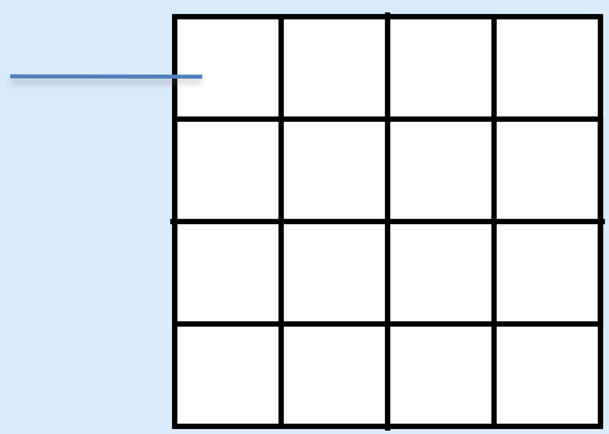
Relationship between number of rounds and cipher key size

Overall Structure



Structure of 128-bit block

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



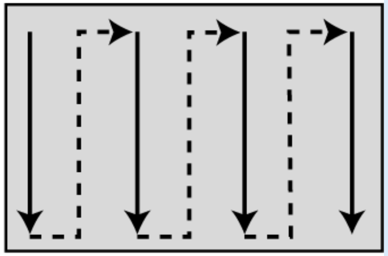
1 byte

Data block represented as ‘State’

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 (bytes)

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

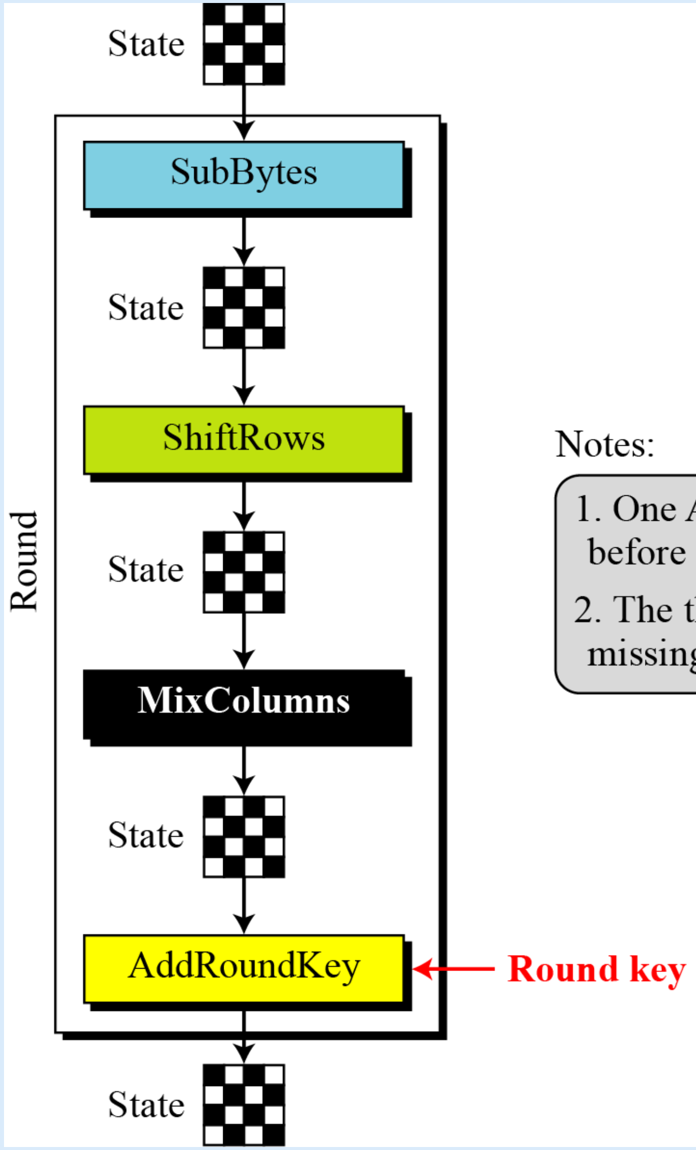


Example - changing plaintext to State

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

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

Details of each round

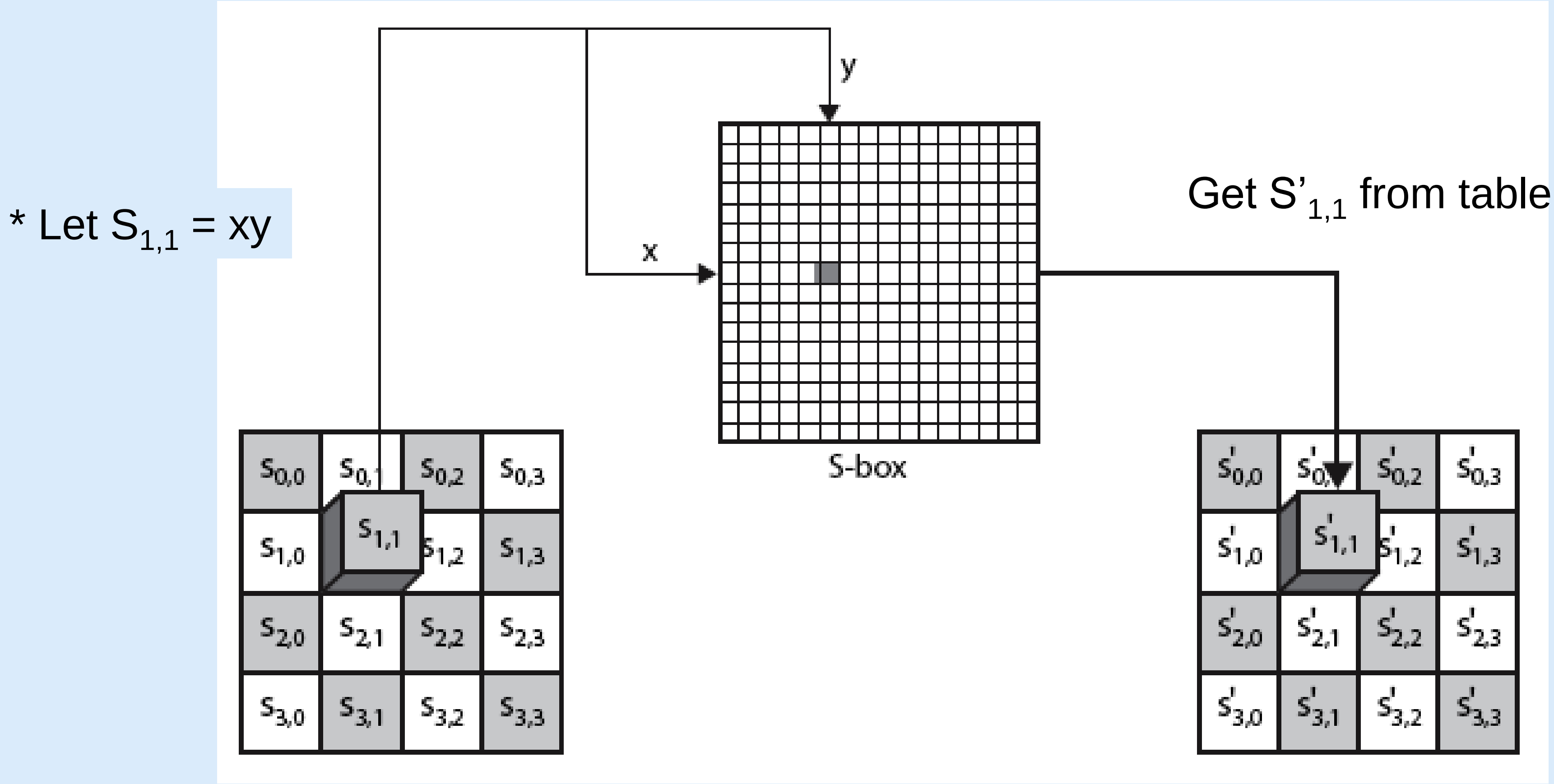


Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

SubBytes Operation

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



*** Interpret a byte as two hexadecimal digits *xy***

SubBytes Table

Implemented by table lookup

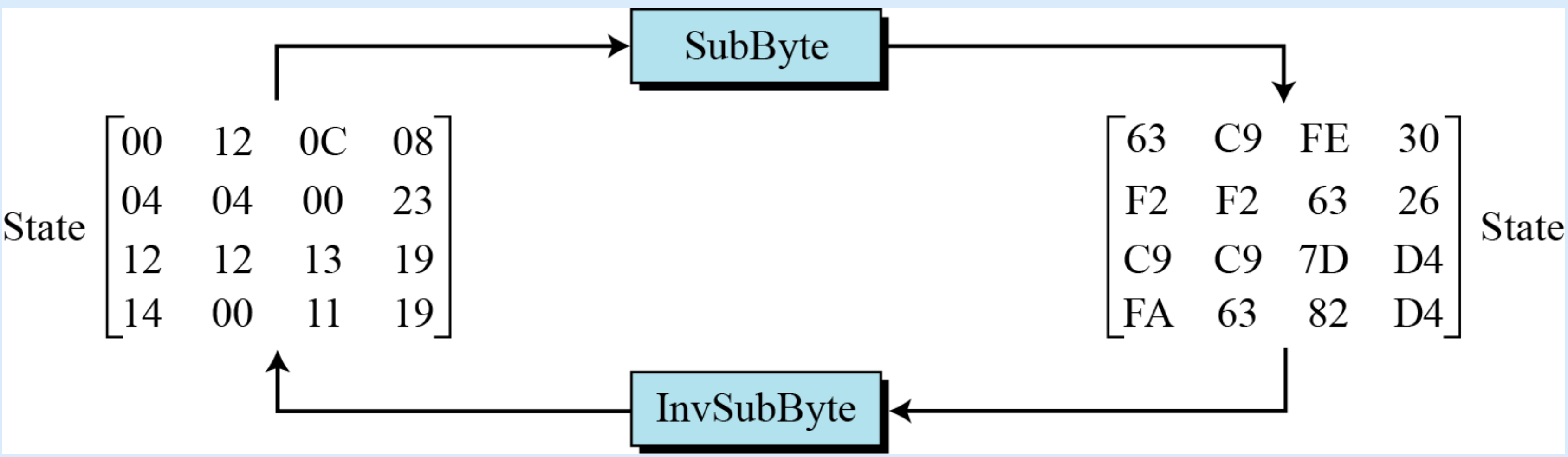
		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2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

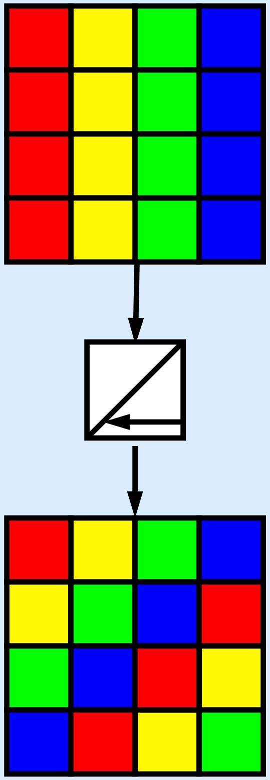
SubByte Example

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

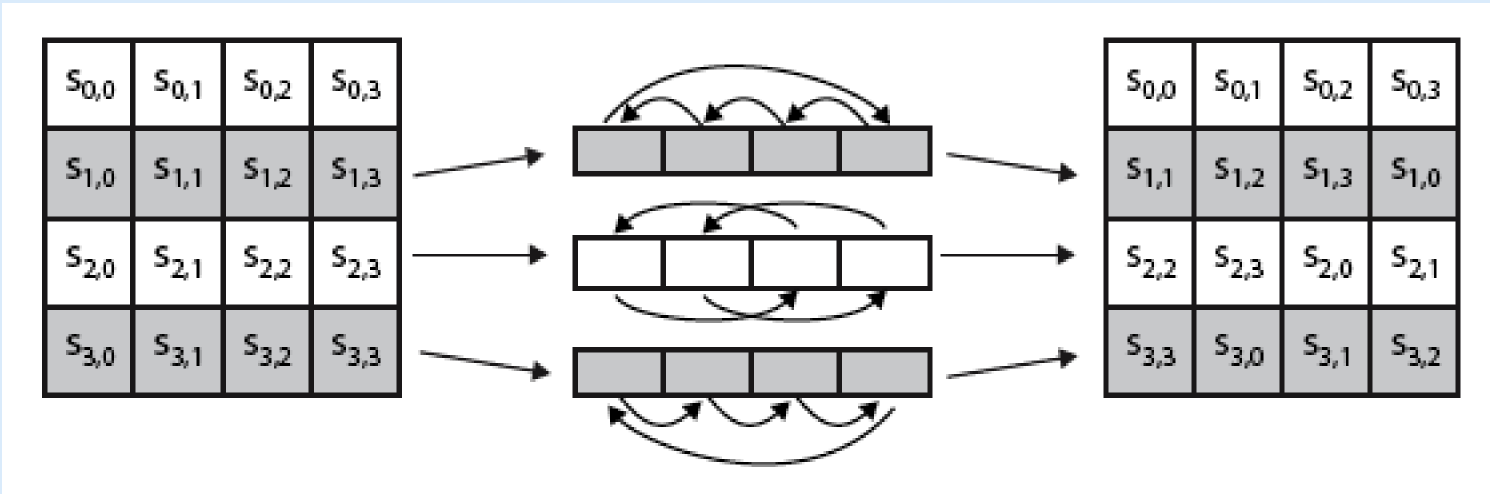


ShiftRows

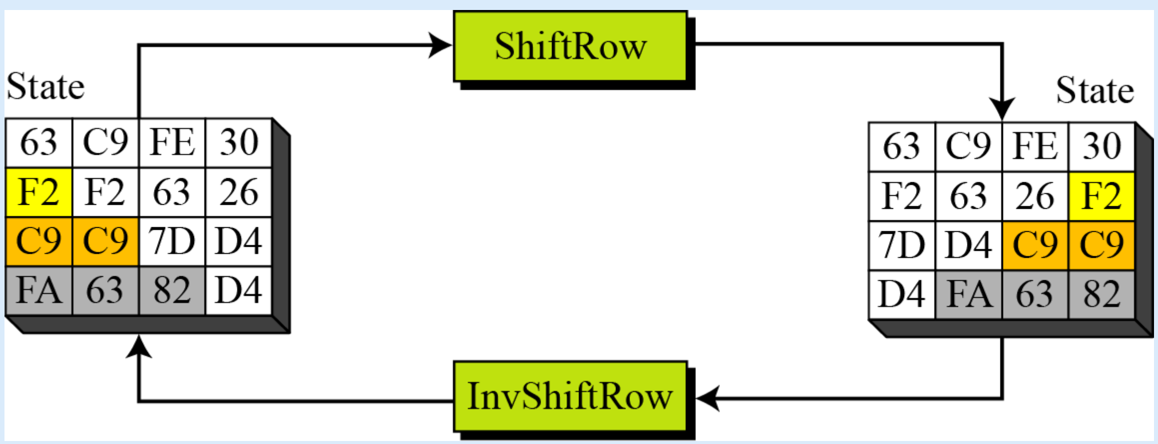
- Shifting permutes the bytes.
- Do a circular byte shift in each row
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- In the encryption, the transformation is called ShiftRows
- In the decryption, the transformation is called InvShiftRows and the shifting is to the right



ShiftRows Scheme



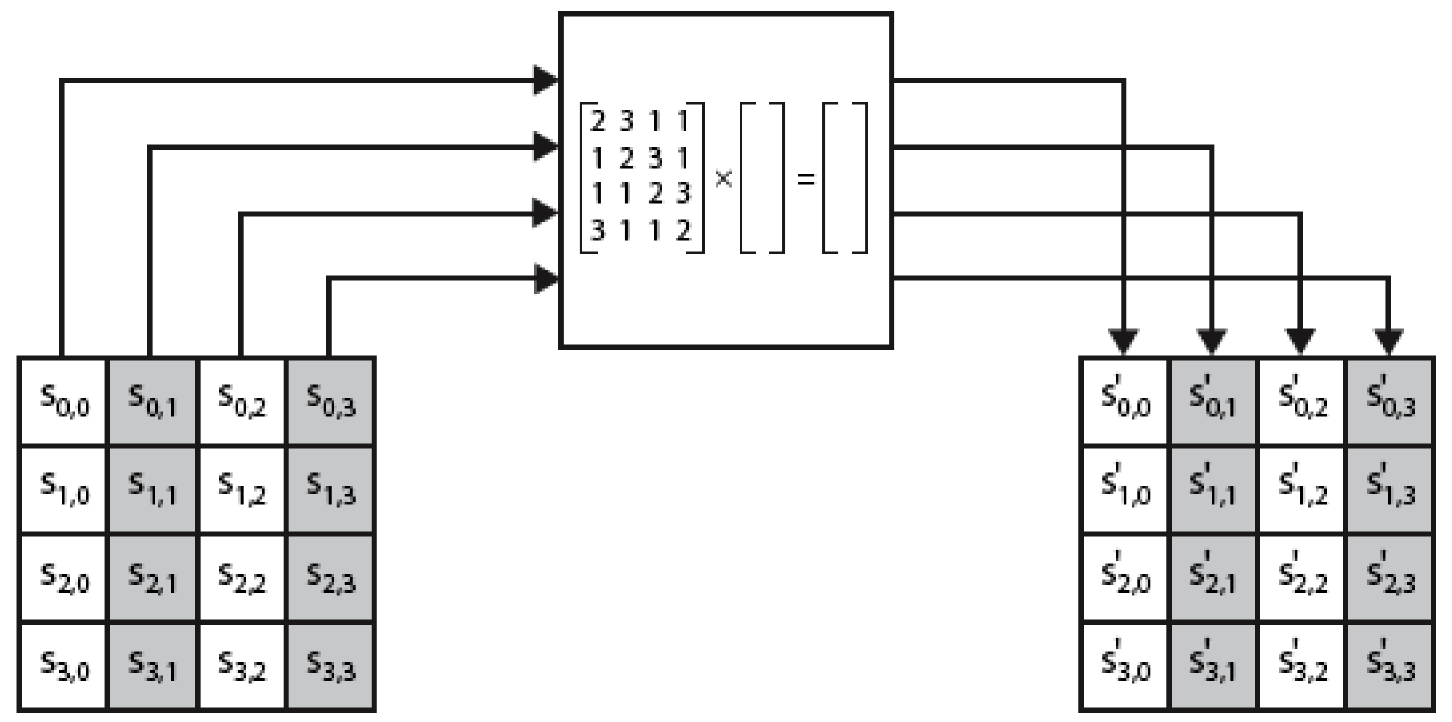
Example



MixColumns

- ShiftRows and MixColumns provide diffusion to the cipher
- In MixColumns, 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 finite field $GF(2^8)$ using prime polynomial $x^8+x^4+x^3+x+1$

MixColumns Scheme



The MixColumns transformation operates at the column level; it transforms each column of the state to a new column.

MixColumn and InvMixColumn

During decryption, inverse mixing matrix is used

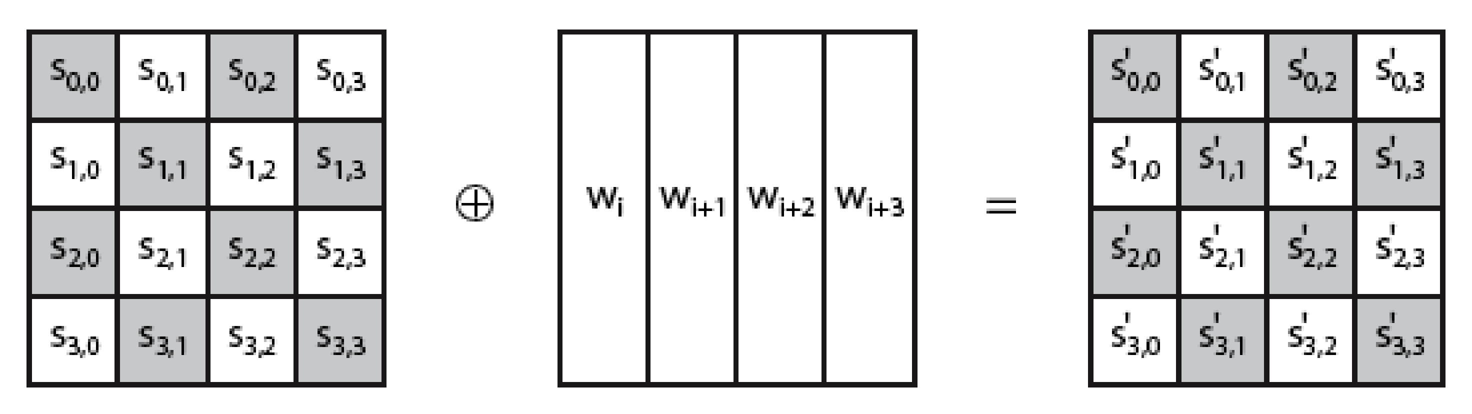
$$\underset{C}{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}} \xleftrightarrow{\text{Inverse}} \underset{C^{-1}}{\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}}$$

AddRoundKey

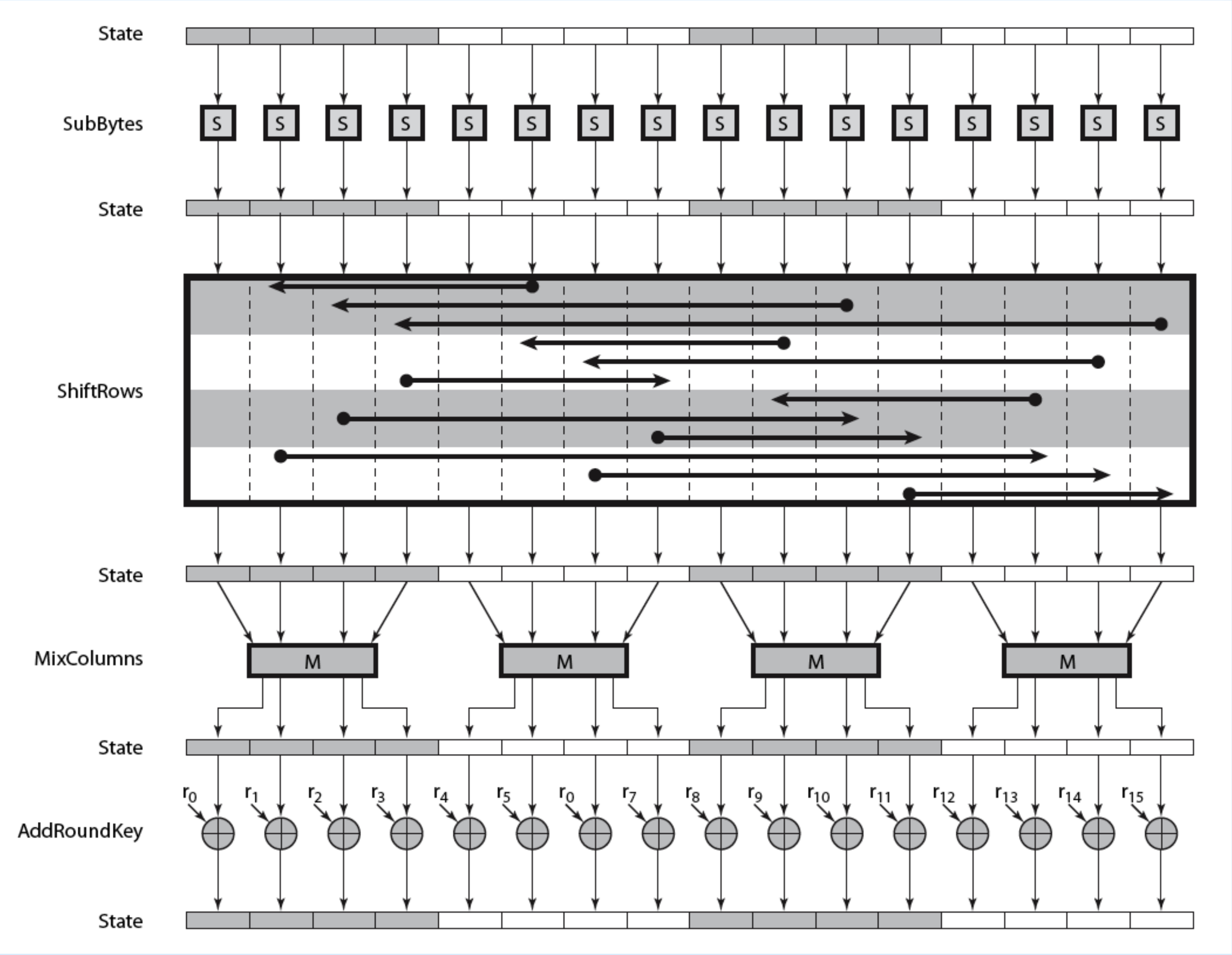
XOR state with 128-bits of the round key

- AddRoundKey proceeds one column at a time.
  - adds a round key word with each state column matrix
  - the operation is matrix addition
- Inverse for decryption is identical
  - since XOR is its own inverse, with same keys

AddRoundKey Scheme



AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State

AES Key Scheduling (generating round keys)

- takes 128-bits (16-bytes) key and expands into array of 44 words (32-bit each)

Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
...	...			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

AES Security

- AES was designed after DES.
  - Most of the known attacks on DES were already tested on AES.
- Brute-Force Attack
  - AES is definitely more secure than DES due to the larger-size key.
- Statistical Attacks
  - Numerous tests have failed to do statistical analysis of the ciphertext
- Differential and Linear Attacks
  - There are no differential and linear attacks on AES as yet.

Implementation Aspects

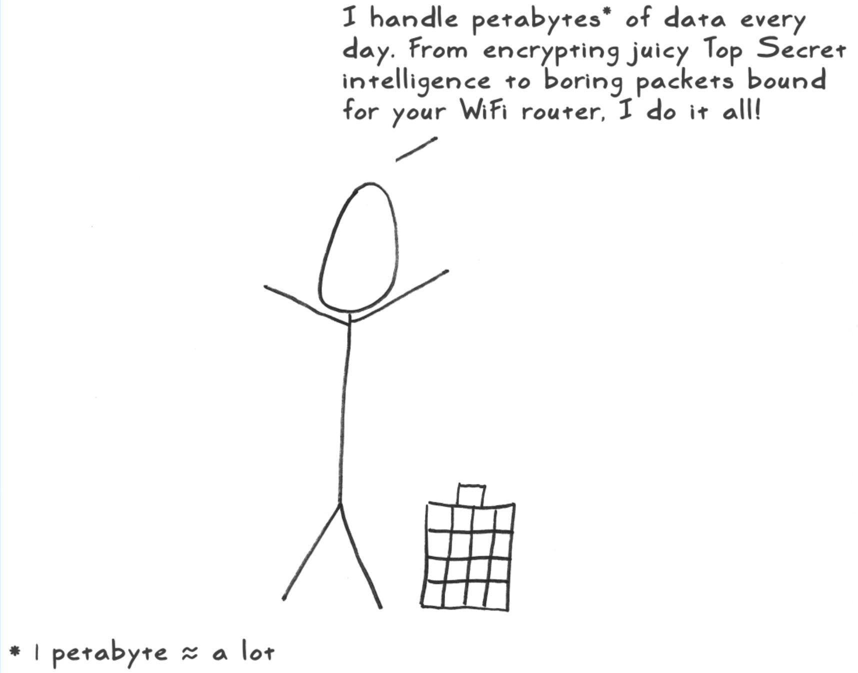
- The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.
  - Very efficient
  - Implementation was a key factor in its selection as the AES cipher
- Several modern CPU architectures include AES instructions

https://en.wikipedia.org/wiki/AES_instruction_set

AES illustrated

A stick figure guide

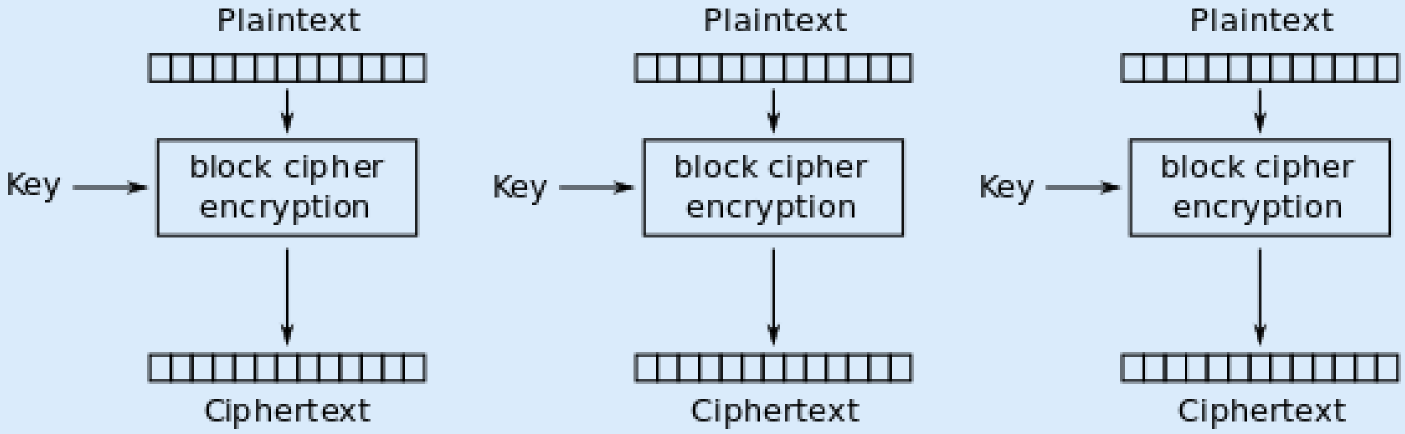
http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html



Block Cipher Modes of Operation

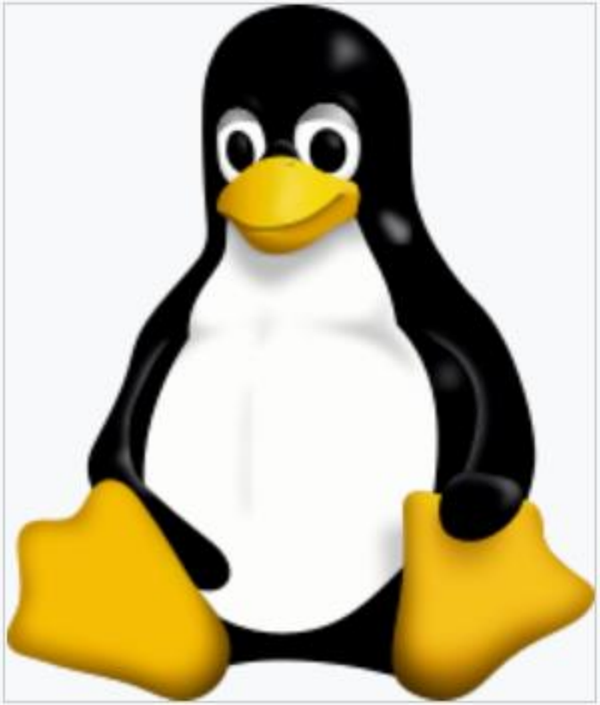
ECB mode

- Block ciphers operate only on small fixed size chunks like 64 bits (DES), 128 bits (AES) etc.
- To encrypt large data, one (lazy) option is to simply divide the whole data in blocks and encrypt them separately with same key. This is ECB mode.



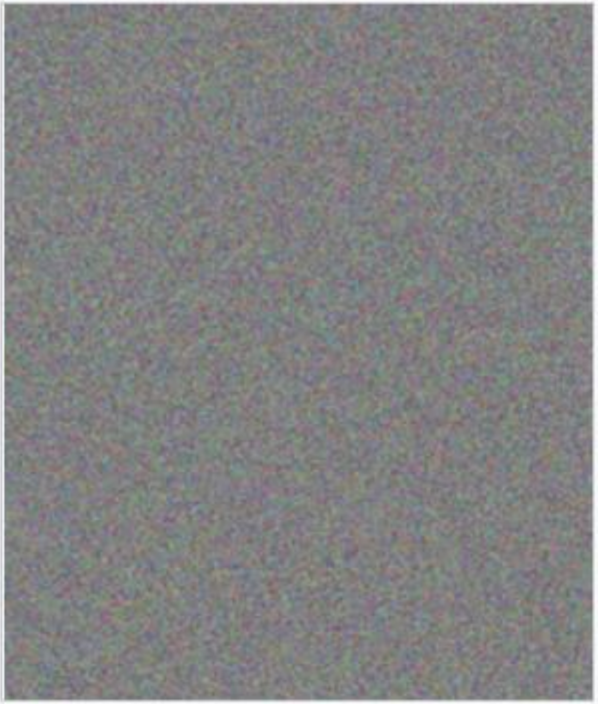
Electronic Codebook (ECB) mode encryption

ECB mode: problem



Original image

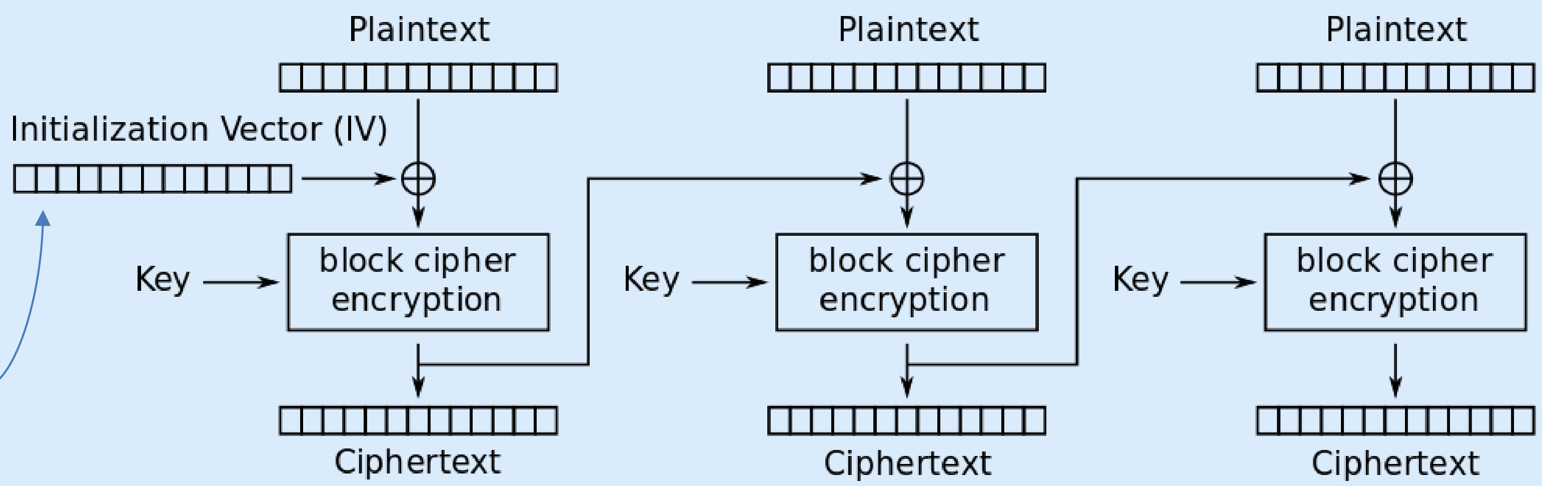
Using ECB allows patterns to be easily discerned



Modes other than ECB result in pseudo-randomness

CBC mode

- More secure modes are available, such as Cipher Block Chaining (CBC)
- Each ciphertext block depends on all plaintext blocks processed up to that point

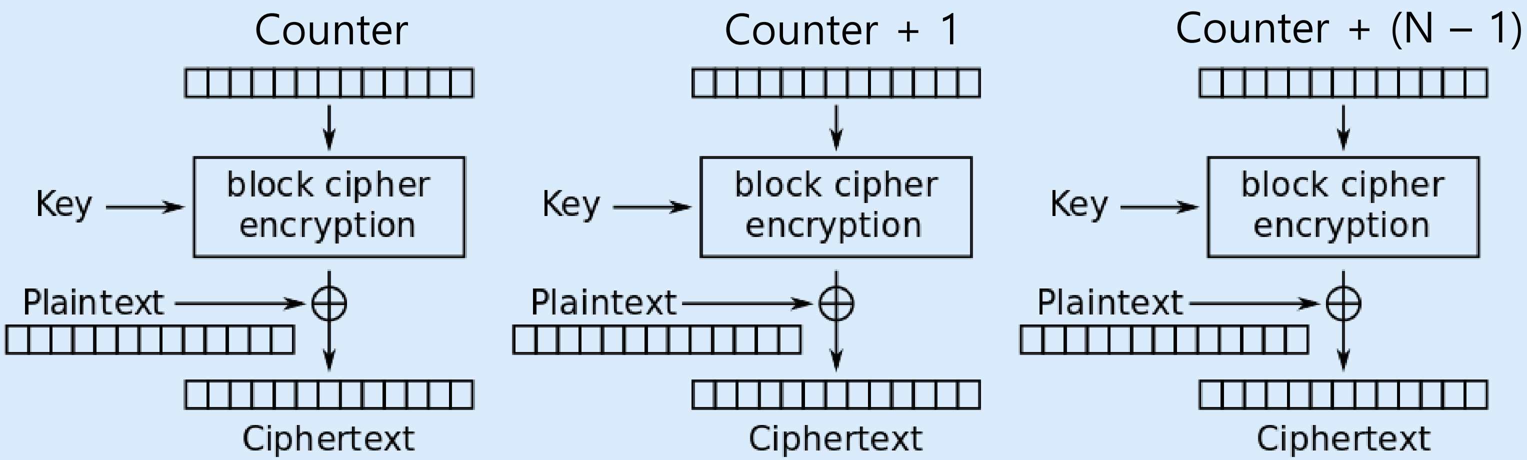


Cipher Block Chaining (CBC) mode encryption

Any fixed (non-secret) value to start with

CTR mode

- Another one is Counter mode
- Start with any pre-defined counter value and then keep incrementing.
- Can encrypt blocks in parallel (unlike CBC mode)



Counter (CTR) mode encryption