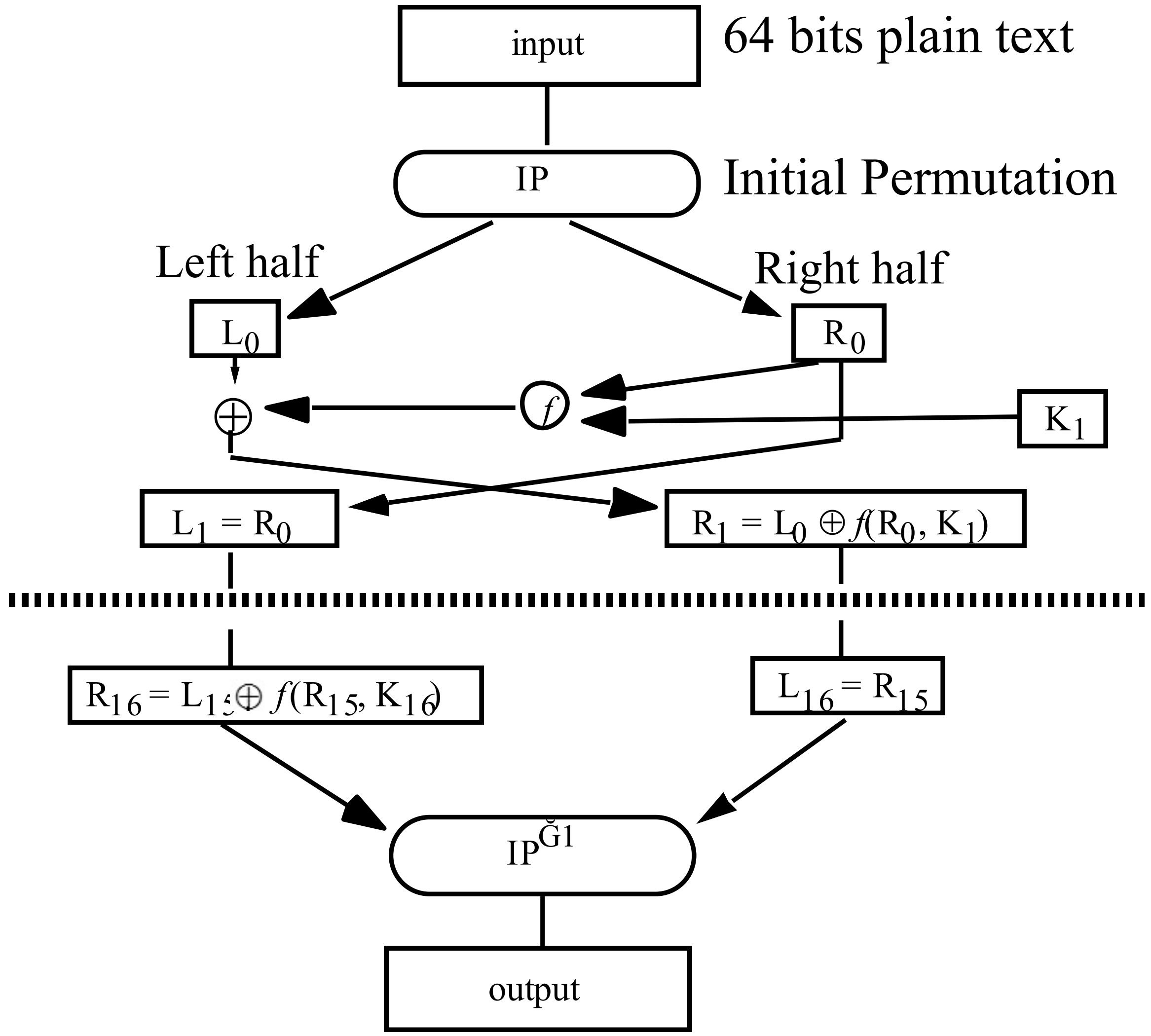
Overview of the DES

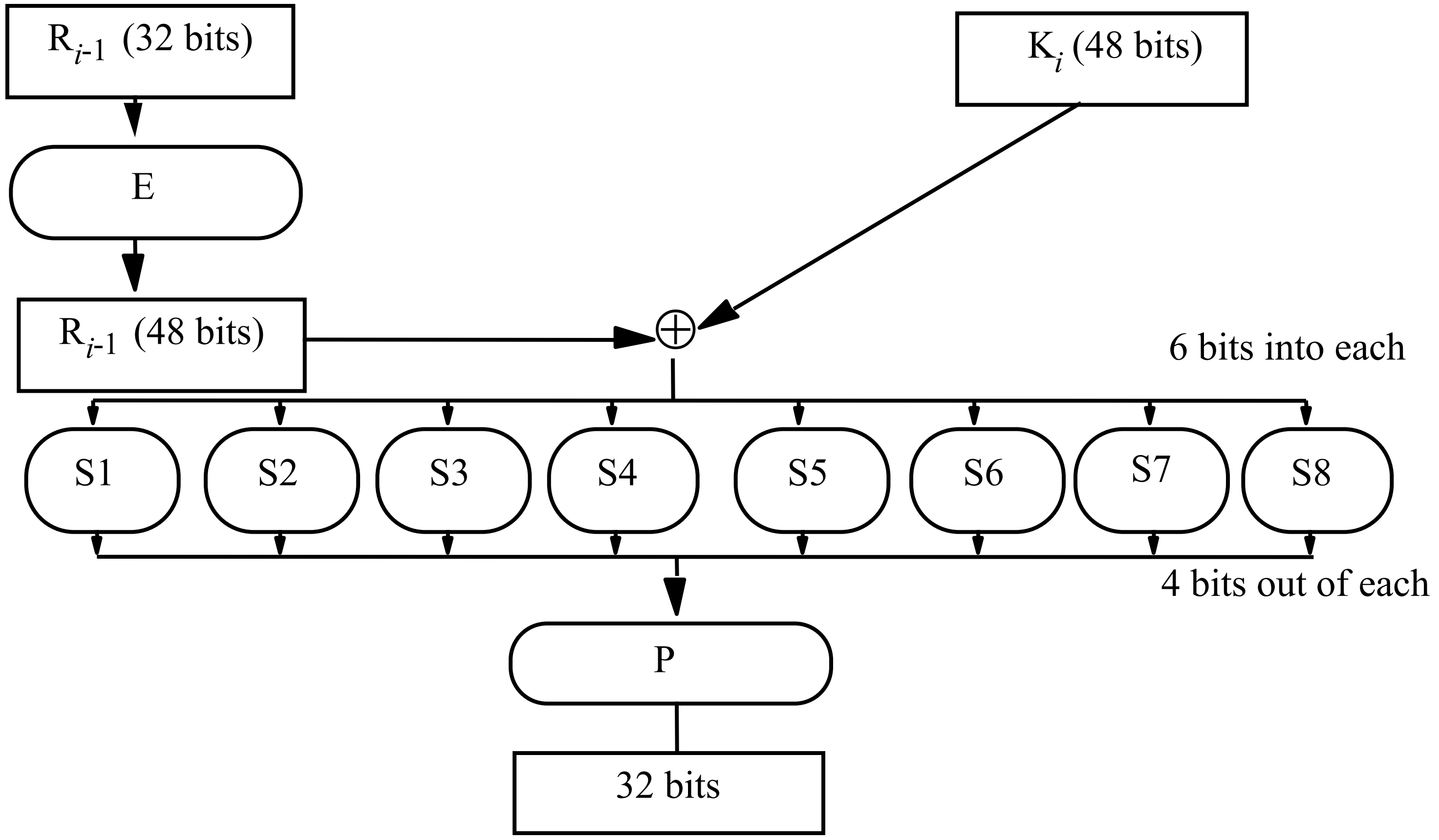
- A Symmetric Key Scheme
- A block cipher:
  - encrypts blocks of 64 bits using a 64 bit key
  - outputs 64 bits of ciphertext
  - A product cipher
    - performs both substitution and transposition (permutation) on the bits
  - basic unit is the bit
- Cipher consists of 16 rounds (iterations), each with a round key generated from the user-supplied key

Encipherment Illustration

Round 1



The *f* Function



DES Modes

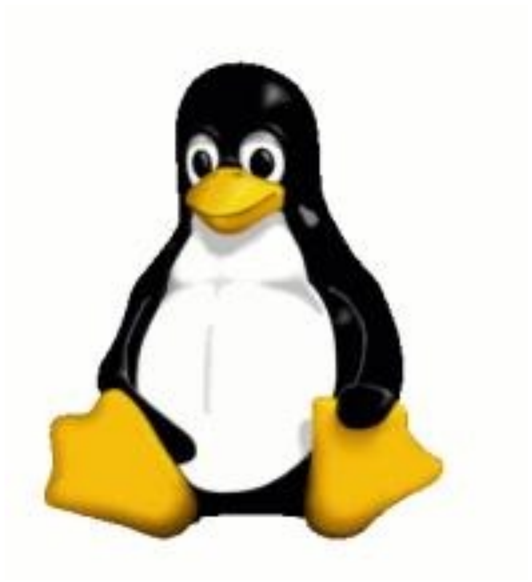
- Electronic Code Book Mode (ECB)
  - Encipher each block independently

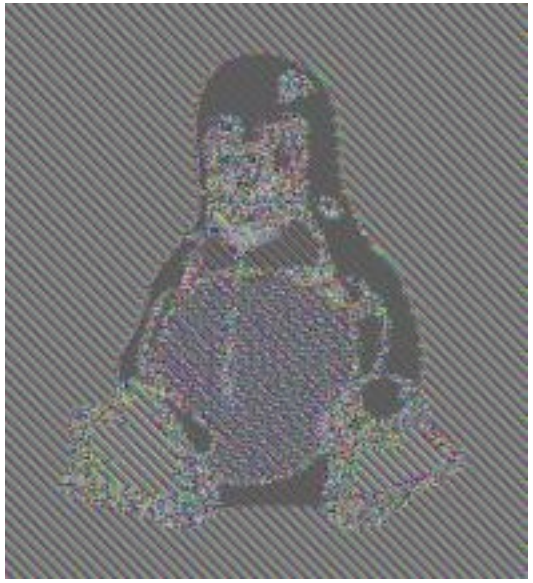
ECB Problem

- Problem: identical plaintext blocks produce identical ciphertext blocks
  - Example: two database records
    - `MEMBER: HOLLY INCOME $100,000`
    - `MEMBER: HEIDI INCOME $100,000`
  - Encipherment:
    - `ABCQZRME GHQMRSIB CTXUVYSS RMGRPFQN`
    - `ABCQZRME ORMPABRZ CTXUVYSS RMGRPFQN`
  - Fails to hide patterns in plaintext

Example of ECB failure

- Pixelmap image of Tux encoded by ECB, and not


Tux


ECB encoding

Non-ECB encoding

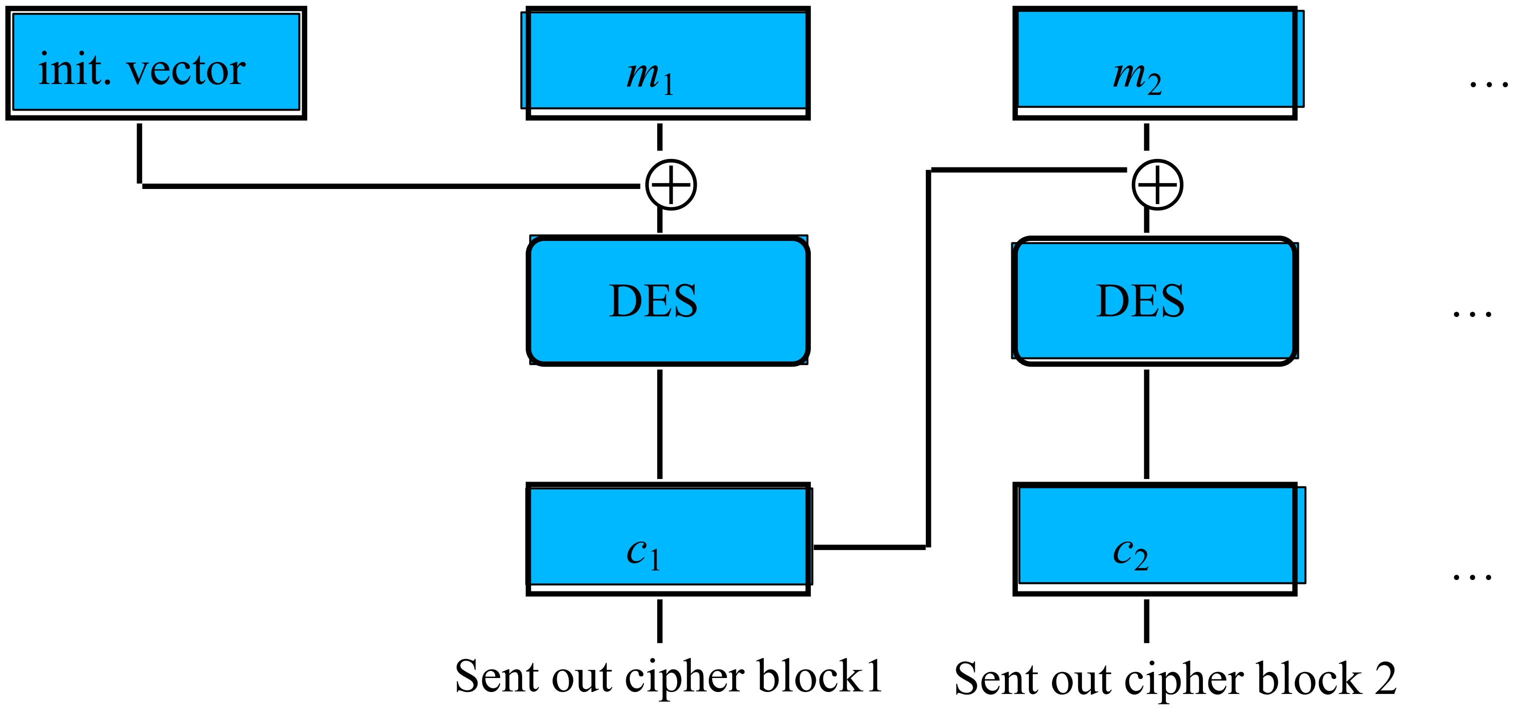
DES Modes

- Electronic Code Book Mode (ECB)
  - Encipher each block independently
- Cipher Block Chaining Mode (CBC)
  - Xor each block with previous ciphertext block
  - Requires an initialization vector for the first one

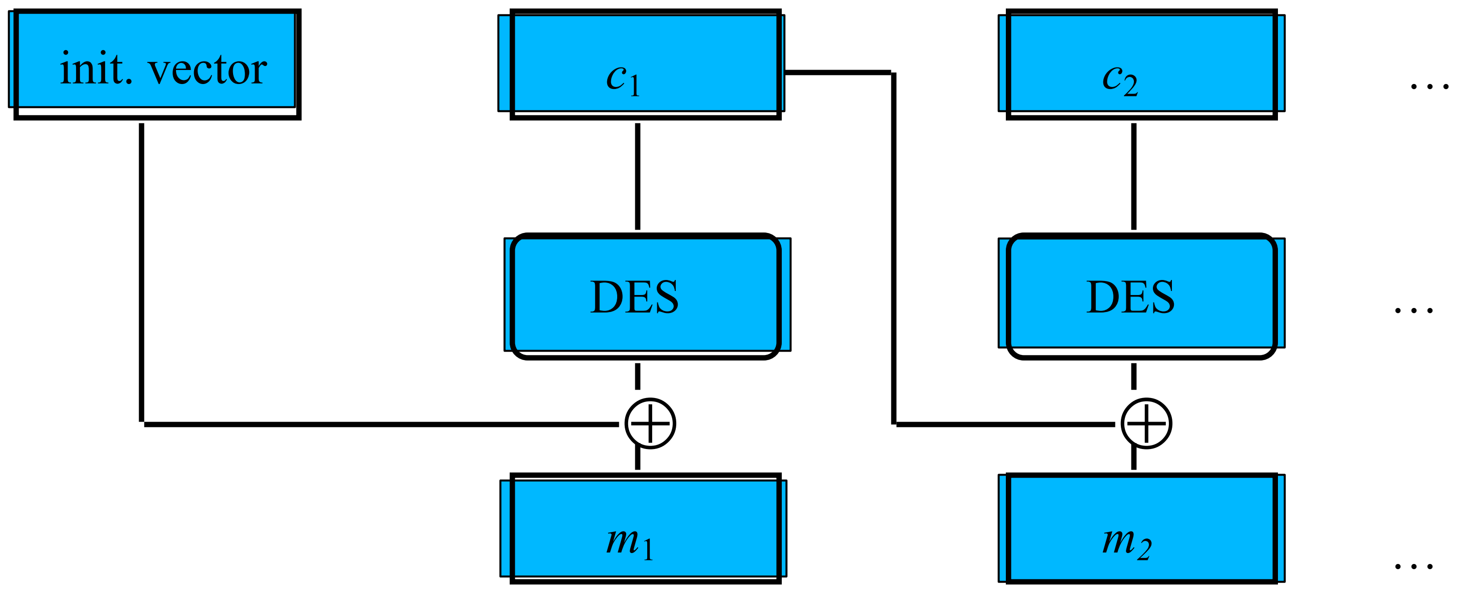
CBC Mode Encryption

Cipher Block Chaining Mode (CBC)

- Xor each block with previous ciphertext block
- Requires an initialization vector for the first one



CBC Mode Decryption



CBC mode Self-Healing Property

- Initial message
  - 3231343336353837 3231343336353837
    3231343336353837 3231343336353837
- Received Ciphertext as (underlined 4c should be 4b)
  - ef7c**4c**b2b4ce6f3b f6266e3a97af0e2c
    746ab9a6308f4256 33e60b451b09603d
- Which decrypts to
  - efca61e19f4836f1 3231333336353837
    3231343336353837 3231343336353837
  - Incorrect bytes underlined
  - Plaintext “heals” after 2 blocks

How does self-healing work?

- Suppose c_i becomes corrupted in transmission (e.g. after encryption) Note that

$$\begin{aligned} m_i &= D_k(c_i) \oplus c_{i-1} \\ m_{i+1} &= D_k(c_{i+1}) \oplus c_i \\ m_{i+2} &= D_k(c_{i+2}) \oplus c_{i+1} \end{aligned}$$

- so the ith and (i+1)st message blocks are corrupted. The (i+2)nd block is free from the corrupted ciphertext
- What about c_i is corrupted during the encryption, before c_{i+1} is calculated?

DES Modes

- Electronic Code Book Mode (ECB)
  - Encipher each block independently
- Cipher Block Chaining Mode (CBC)
  - Xor each block with previous ciphertext block
  - Requires an initialization vector for the first one
- Encrypt-Decrypt-Encrypt Mode (2 keys: k, k')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}^{-1}(\mathrm{DES}_k(m)))$
- Encrypt-Encrypt-Encrypt Mode (3 keys: k, k', k'')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}(\mathrm{DES}_{k''}(m)))$

Advanced Encryption Standard (AES) Background

- Clearly a replacement for DES was needed
- US NIST issued call for ciphers in 1997
  - 15 candidates accepted in Jun 98
  - 5 were short-listed in Aug-99
- Rijndael was selected as AES in Oct-2000
  - issued as FIPS PUB 197 standard in Nov-2001
  - http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

AES 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
- NIST have released all submissions & unclassified analyses

AES Evaluation Criteria

- Final criteria
  - general security
  - software & hardware implementation ease
  - defence against attacks
  - flexibility

AES Shortlist

- Shortlist August-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
- Saw contrast between algorithms with
  - few complex rounds verses many simple rounds
  - refined existing ciphers verses new proposals

The AES Cipher - Rijndael

- Designed by Rijmen-Daemenin Belgium
  - Has 128/192/256 bit keys, 128 bit data
- An **iterative** cipher
  - treats data in 4 groups of 4 bytes
  - 4x4 matrix in column major order
  - operates an entire block in every round

AES Overview

- 128 bit block represented by a 4x4 byte matrix
- Processing on each block comprised of several rounds
  - 10 for 128-bit key, 12 for 192-bit key, 14 for 256-bit key

AES: State array

input bytes

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}

output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

input bytes

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}

→

State array

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

→

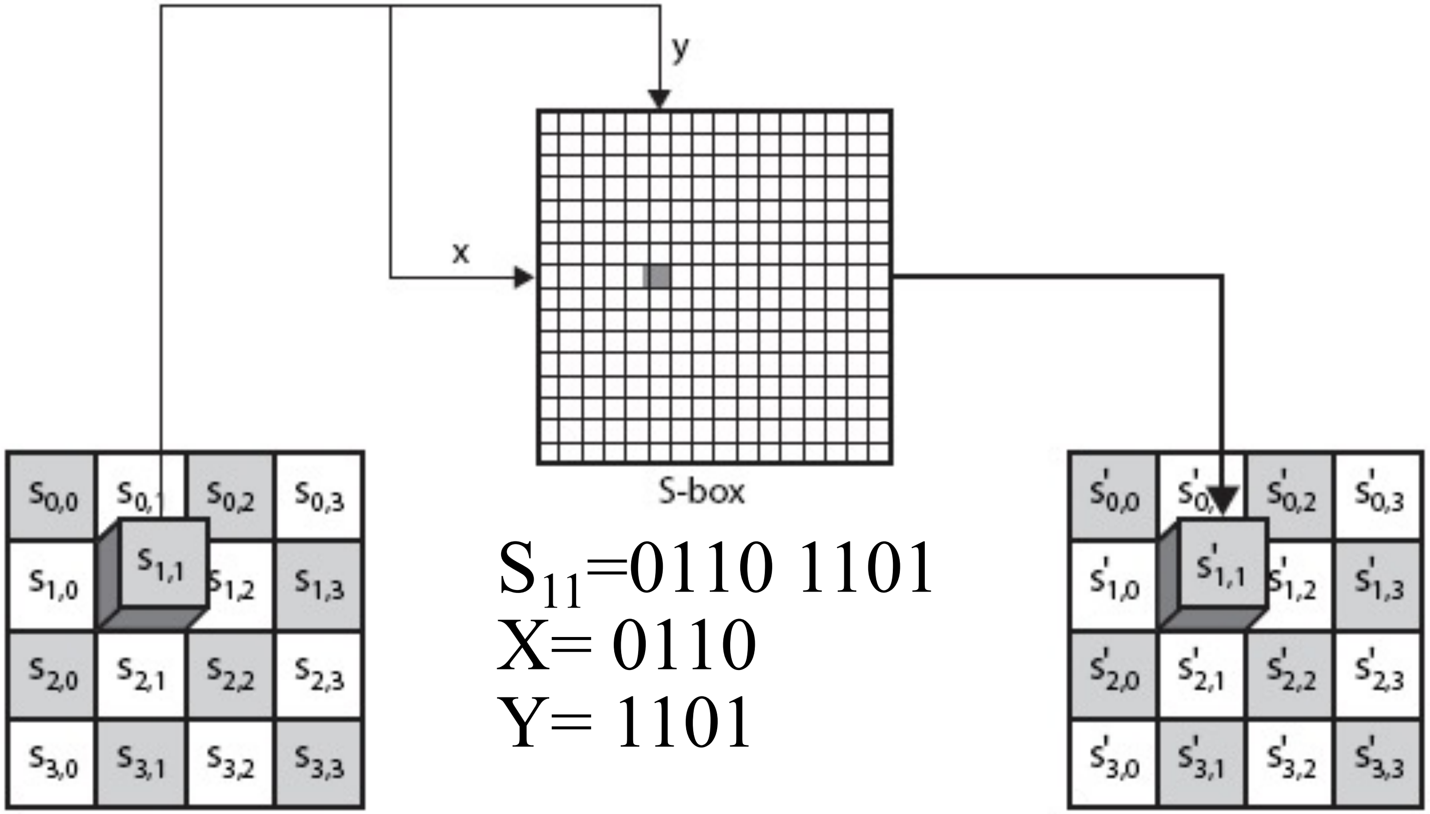
output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Four Steps in Each Round

- Substitute Bytes – single byte based
- Shift Rows – row-wise permutation
- Mix Columns – column-wise mixing
- Add Round Keys

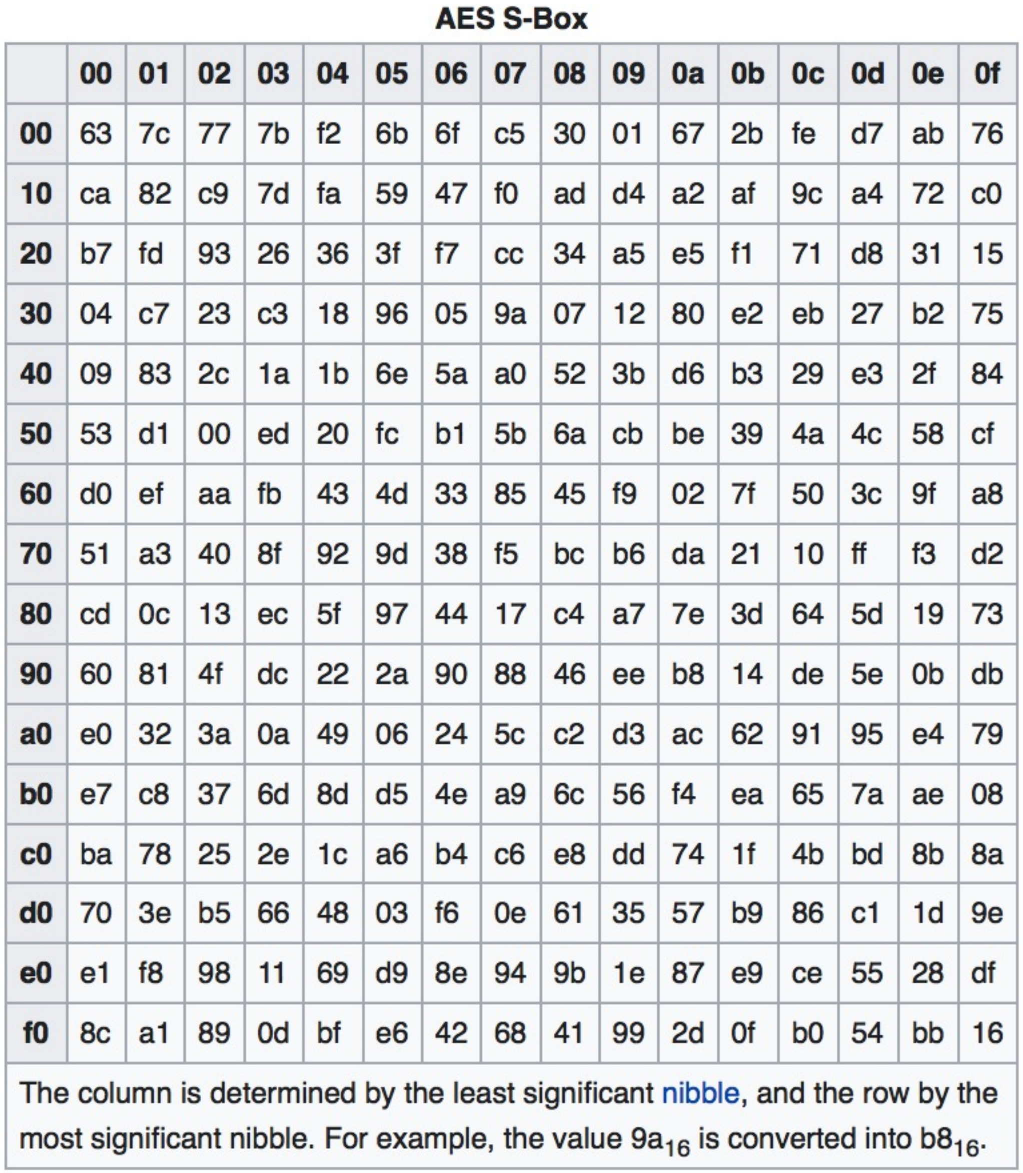
AES: SubBytes() (S-Box)



S_{11}=0110 1101
X= 0110
Y= 1101

- A simple substitution of each byte
- Uses one table of 16x16 bytes containing a permutation of all 256 8-bit values

itution

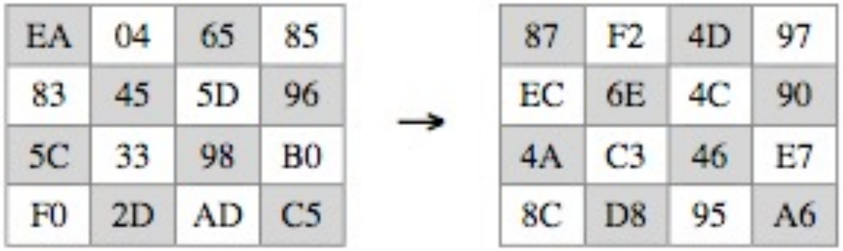


AES S-Box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value $9a_{16}$ is converted into $b8_{16}$.

Example:



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

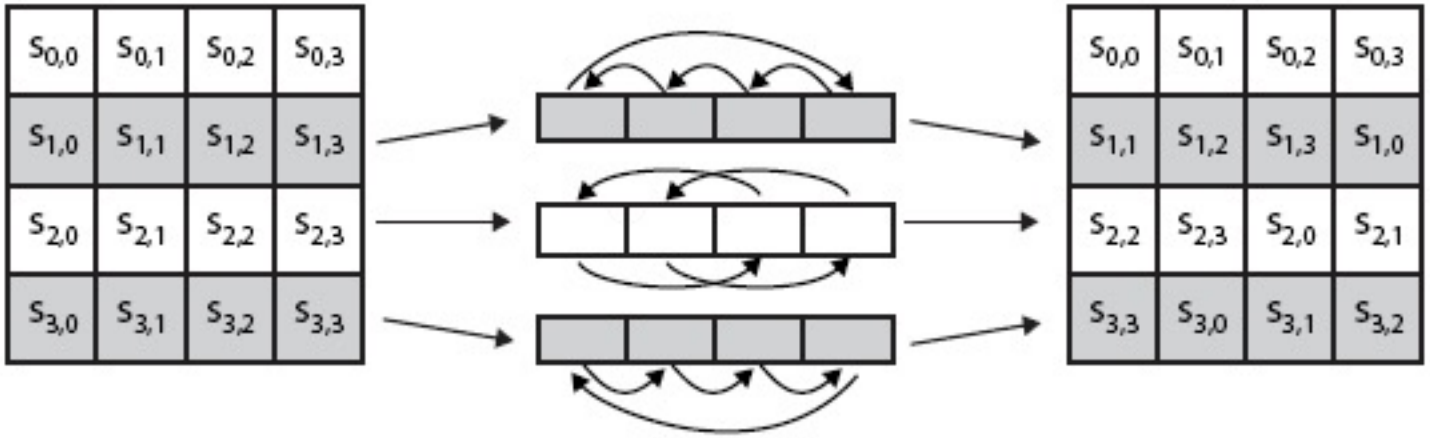
→

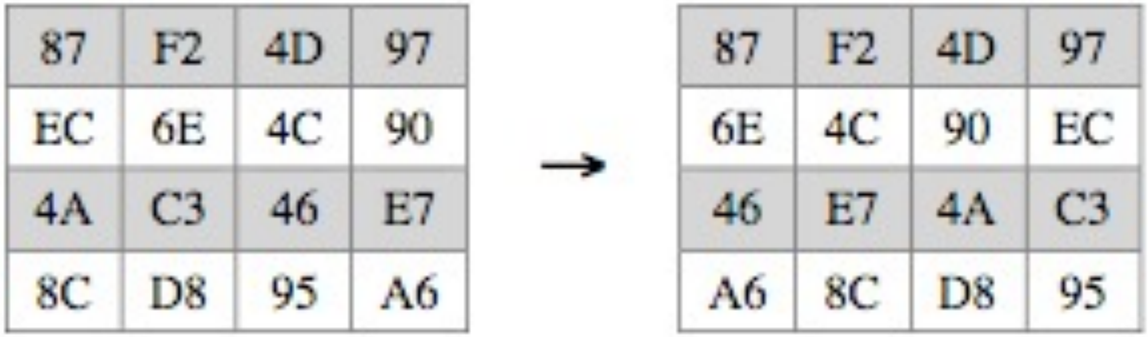
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Shift Rows

- 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
- Decrypt does shifts to right

AES: ShiftRows()


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



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