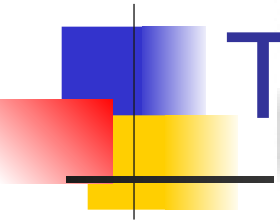
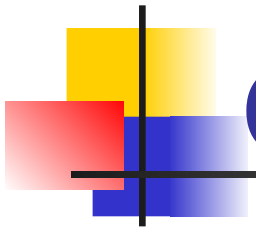


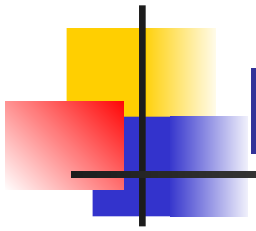
Topic 2: Cryptography

Network Security
Course 2021/2022

Maria Dolores Cano Banos
Francesc Burrull i Mestres

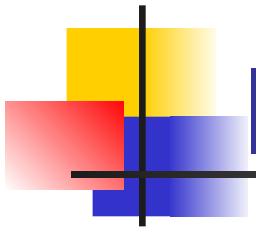


Contents



Introduction

- Cryptography (cryptos = hidden + graphy = writing): art of writing with a secret key or in an enigmatic way ⇨ set of techniques dealing with the protection of information
  - Cryptanalysis
  - Cryptology
- Cryptographic systems are classified based on:
  - Type of operation: substitution, transposition.
  - The number of keys: symmetric, asymmetric.
  - Plain text processing mode: block, stream.

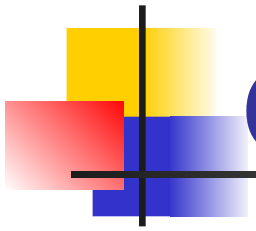


Introduction

- Notation:
  - Plain text X
  - Ciphertext Y
  - Encryption algorithm E

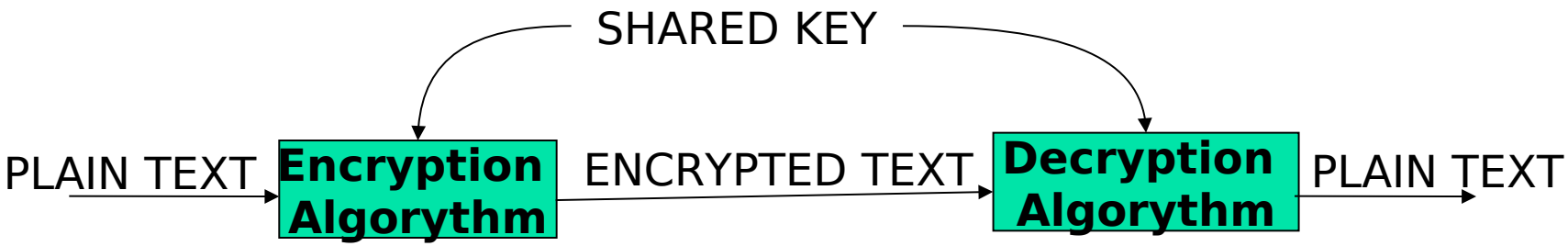
  } $Y = E_K(X)$
  - Key K
  - Decryption algorithm D ⇨ $X = D_K(Y)$

- Ex: Caesar's Cipher
- Ex: transposition technique



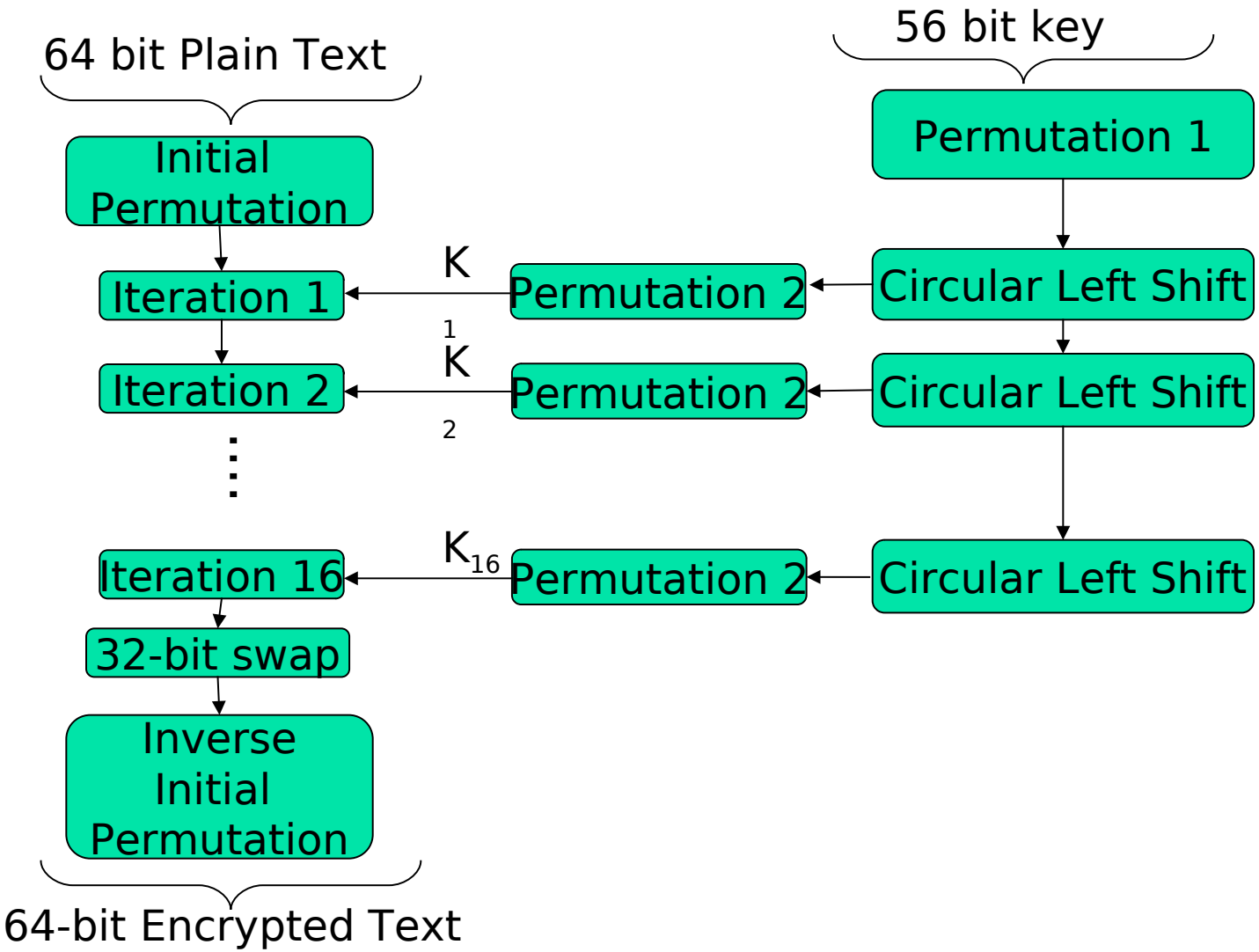
Contents

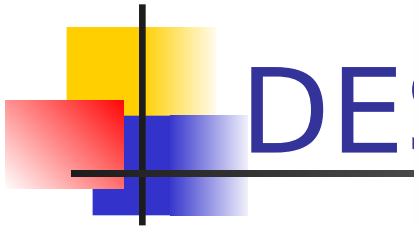
Symmetric algorithms



- Key: value independent of plain text
- Classic crypto security:
  - The encryption algorithm
  - The secret of the key
- Ex: DES, triple DES, AES, ...

DES: Encryption scheme





DE les

TABLE 2.5 Permutation Tables for DES

(a) Initial Permutation (IP)

Output bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
From input bit	58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
Output bit	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
From input bit	62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
Output bit	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
From input bit	57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
Output bit	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
From input bit	61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

Output bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
From input bit	40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
Output bit	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
From input bit	38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
Output bit	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
From input bit	36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
Output bit	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
From input bit	34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

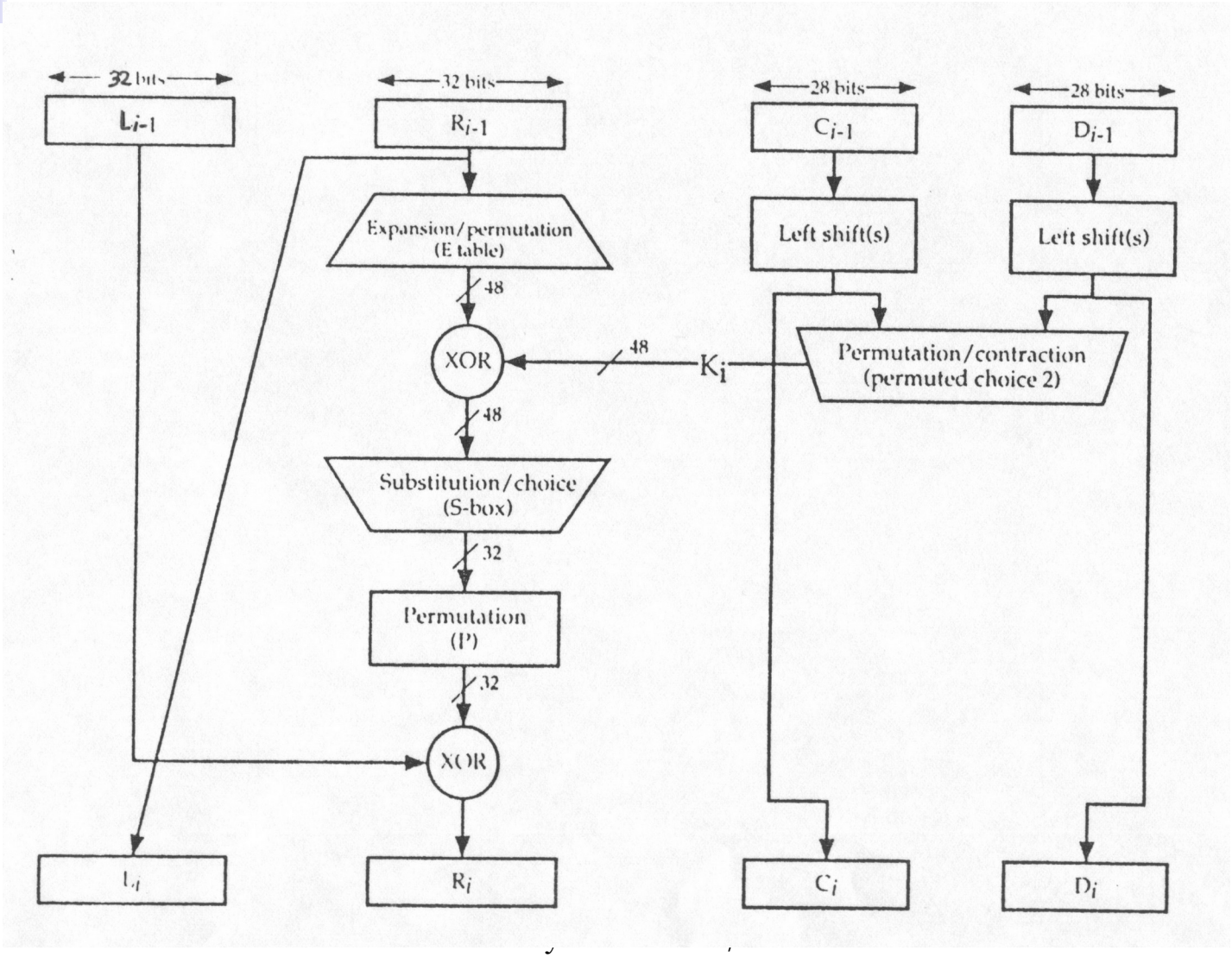
(c) Expansion Permutation (E)

Output bit	1	2	3	4	5	6	7	8	9	10	11	12
From input bit	32	1	2	3	4	5	4	5	6	7	8	9
Output bit	13	14	15	16	17	18	19	20	21	22	23	24
From input bit	8	9	10	11	12	13	12	13	14	15	16	17
Output bit	25	26	27	28	29	30	31	32	33	34	35	36
From input bit	16	17	18	19	20	21	20	21	22	23	24	25
Output bit	37	38	39	40	41	42	43	44	45	46	47	48
From input bit	24	25	26	27	28	29	28	29	30	31	32	1

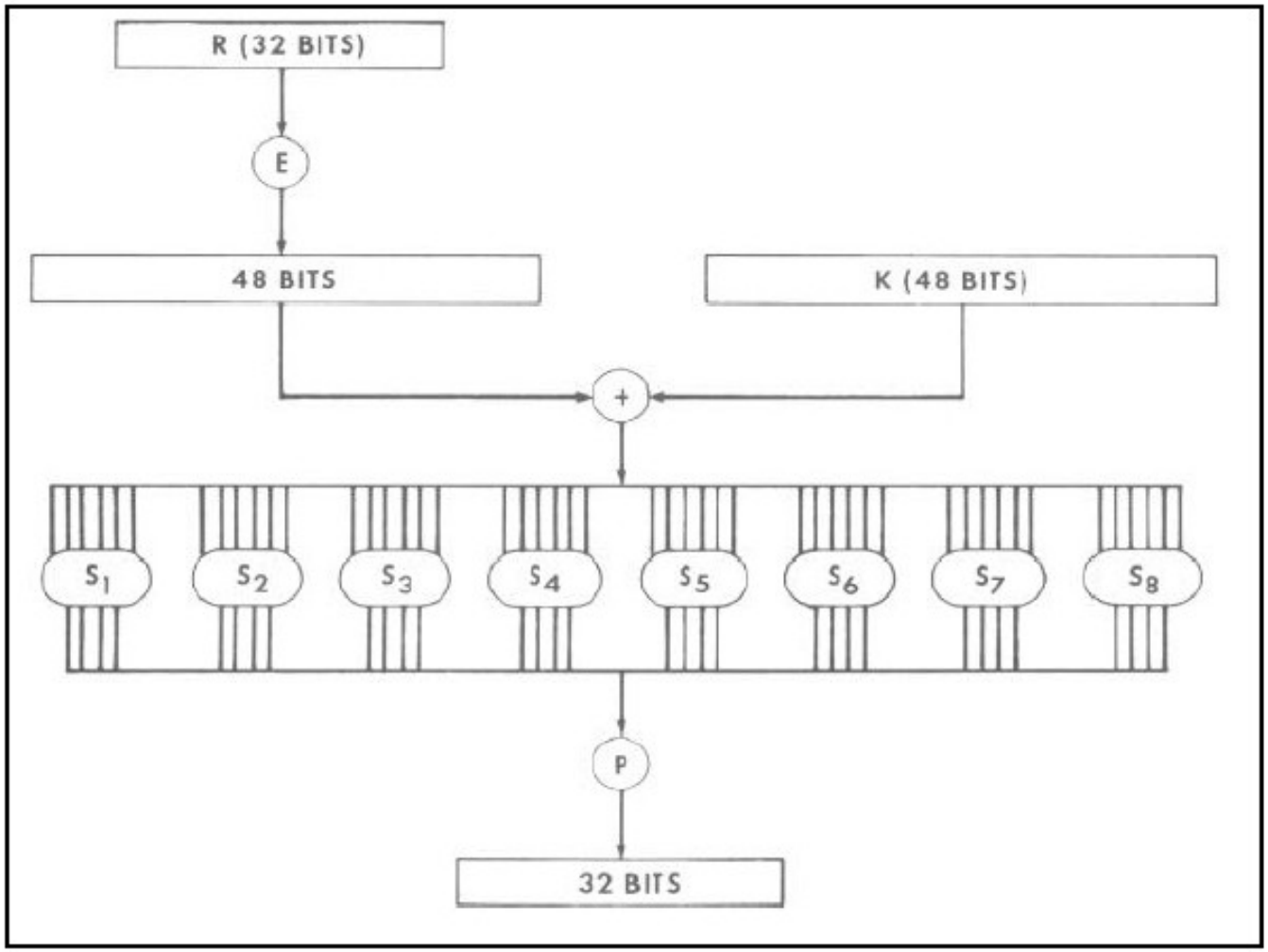
(d) Permutation Function (P)

Output bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
From input bit	16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
Output bit	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
From input bit	2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

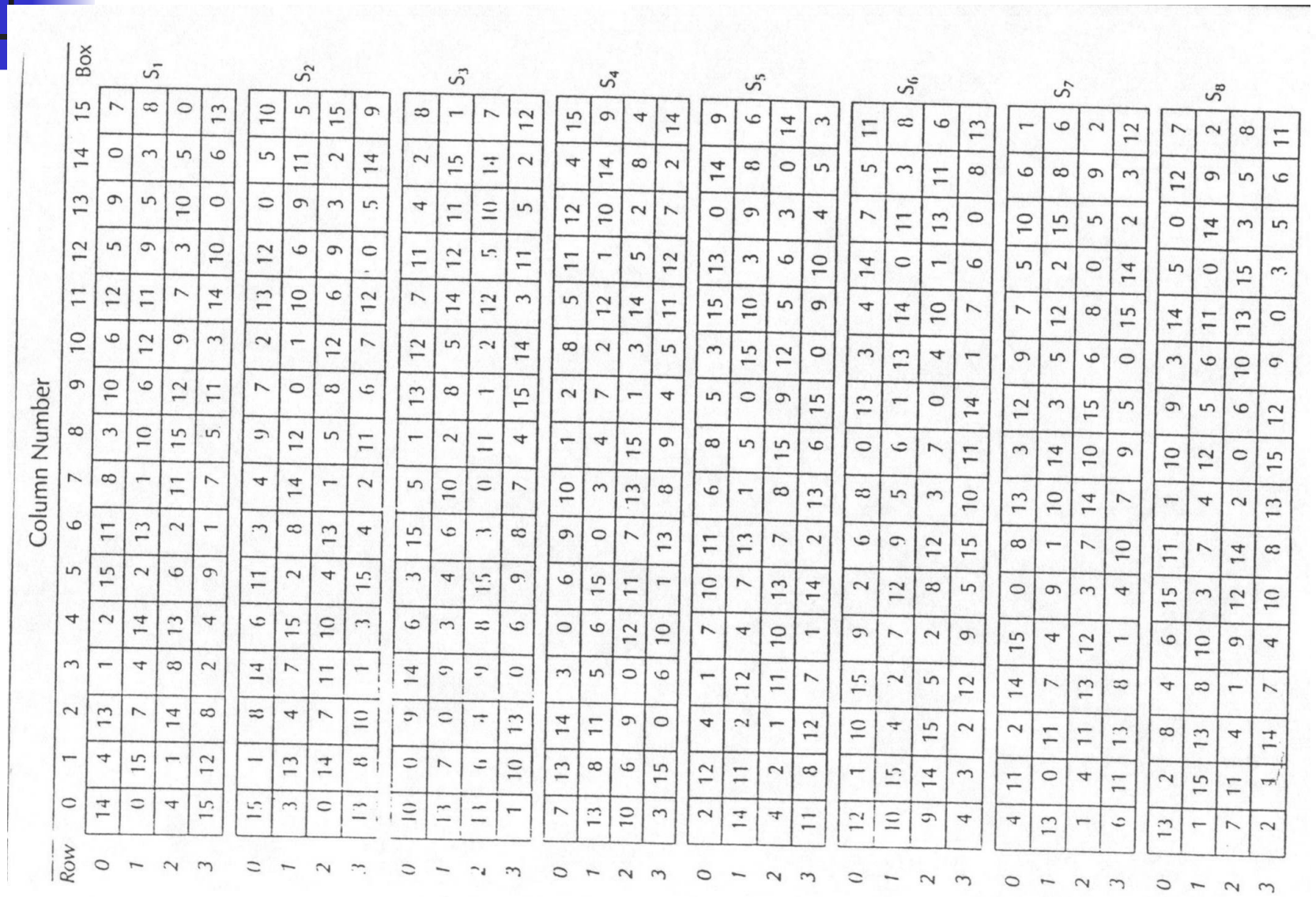
A single iteration in DES



Calculation of f(R,K)



Definition of S-boxes in DES



Box	Row	Column Number: 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tables for calculating keys in DES

(a) Permuted Choice One (PC-1)

Output bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14
From input bit	57	49	41	33	25	17	9	1	58	50	42	34	26	18
Output bit	15	16	17	18	19	20	21	22	23	24	25	26	27	28
From input bit	10	2	59	51	43	35	27	19	11	3	60	52	44	36
Output bit	29	30	31	32	33	34	35	36	37	38	39	40	41	42
From input bit	63	55	47	39	31	23	15	7	62	54	46	38	30	22
Output bit	43	44	45	46	47	48	49	50	51	52	53	54	55	56
From input bit	14	6	61	53	45	37	29	21	13	5	28	20	12	4

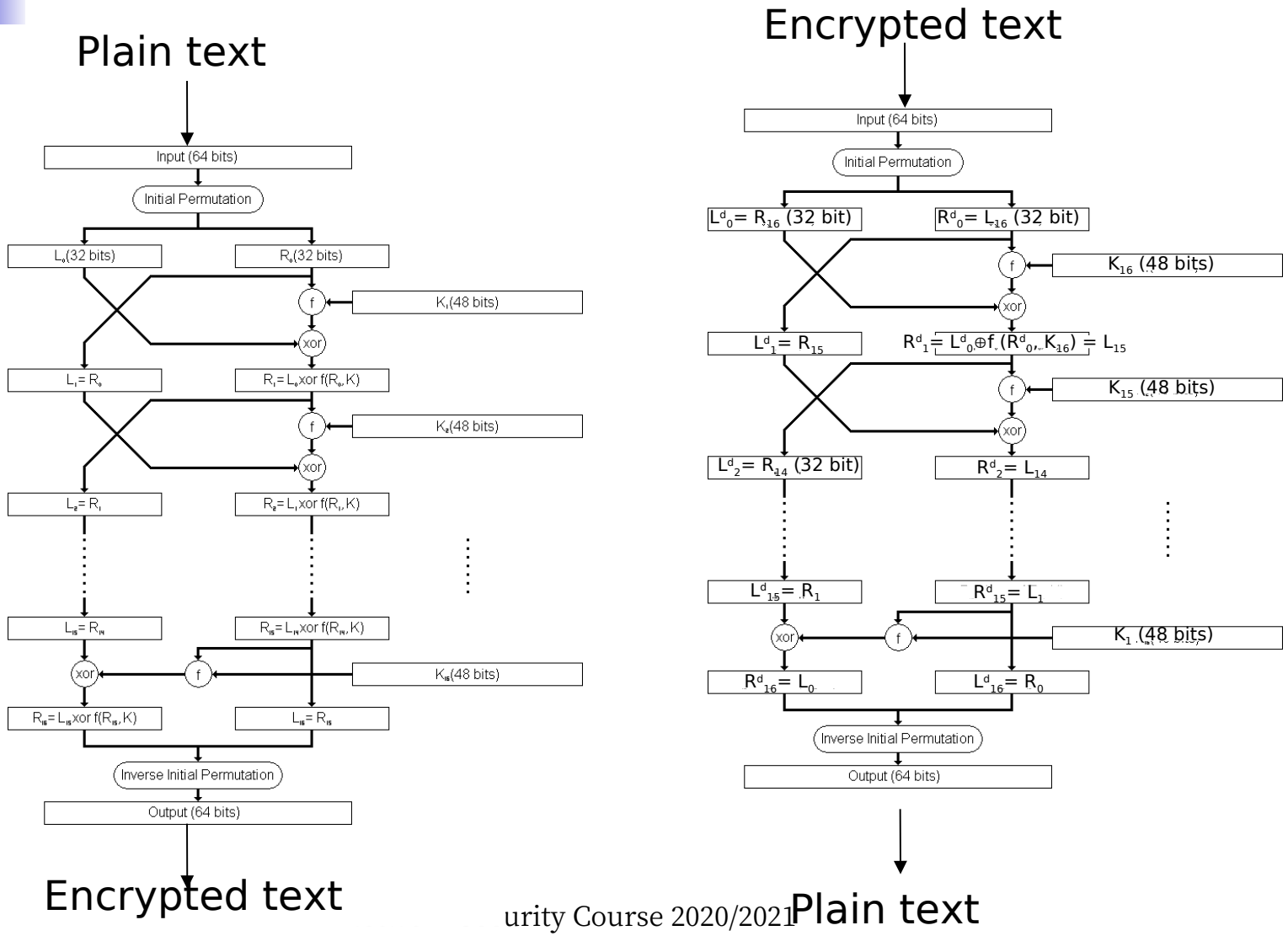
(b) Permuted Choice Two (PC-2)

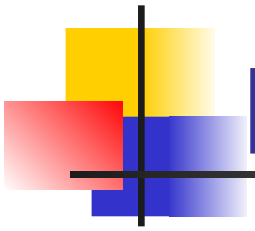
Output bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
From input bit	14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4
Output bit	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
From input bit	26	8	16	7	27	20	13	2	41	52	31	37	47	55	30	40
Output bit	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
From input bit	51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32

(c) Schedule of Left Shifts

Iteration number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES encryption and decryption

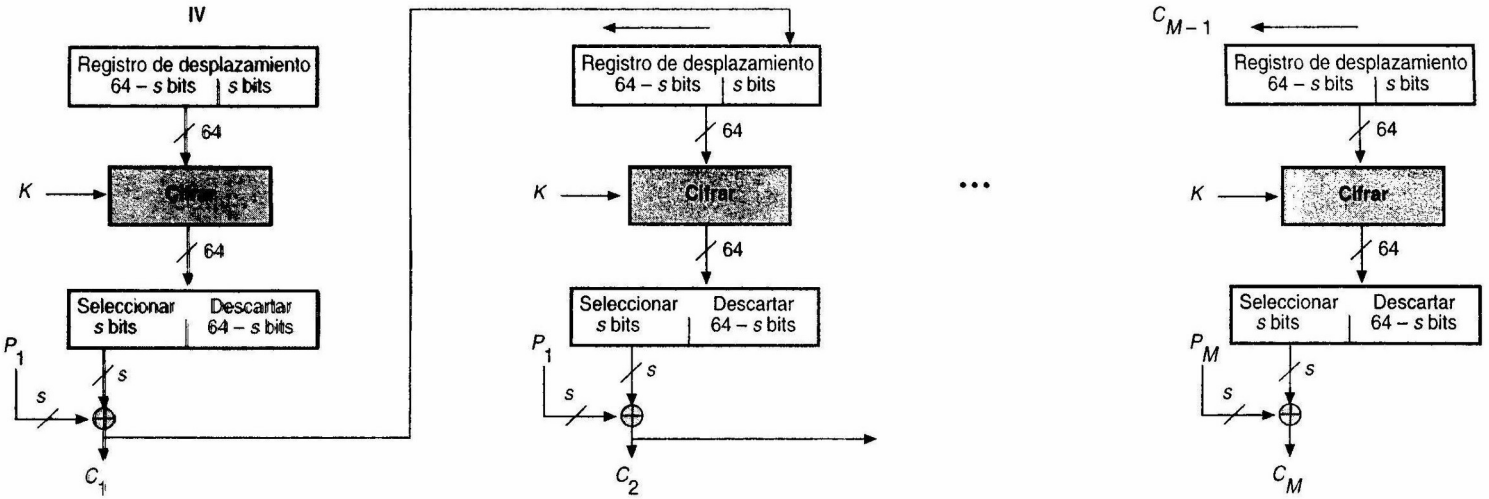




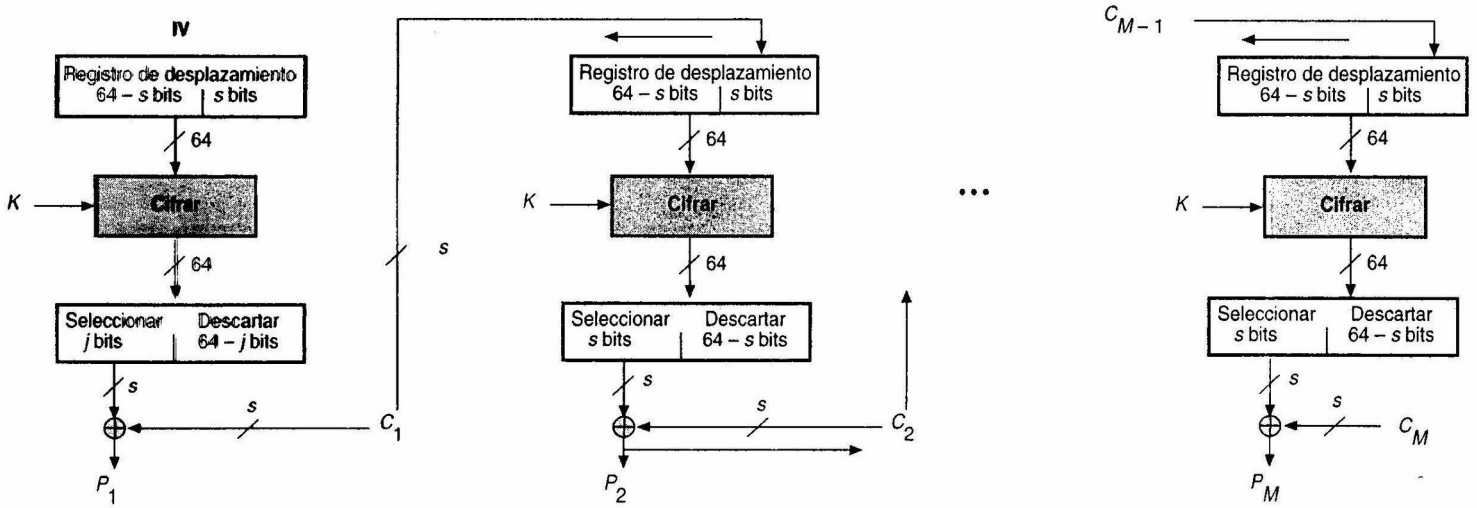
DES operating modes

- ECB (Electronic CodeBook) mode
  - Each 64-bit block is independently encrypted
  - Same key
- CBC (Cipher Block Chaining) mode
  - The input to the algorithm is an XOR of the plaintext and previous ciphertext
  - Same key
- CFB (Cipher FeedBack) mode
  - The input of the algorythm is the previous ciphertext
  - XOR output with plain text
  - Same key
- OFB (Output FeedBack) mode
  - The input to the algorithm is the previous DES output
  - Same key

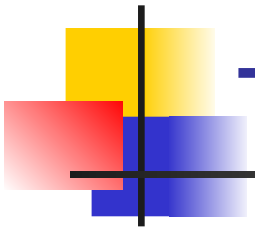
DES operating modes



(a) Cifrado



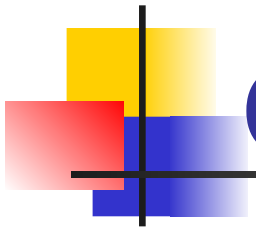
(b) Descifrado



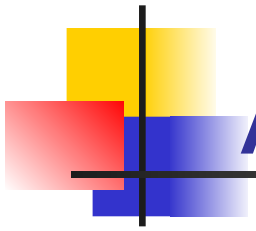
Triple DES

- $Y = E_{K1}[D_{K2}[E_{K1}(X)]]$
- $X = D_{K1}[E_{K2}[D_{K1}(Y)]]$

- Key: 128 bit
- Block: 64 bit

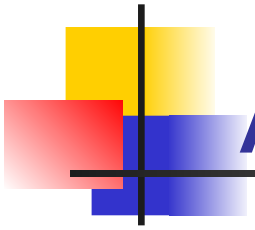


Contents



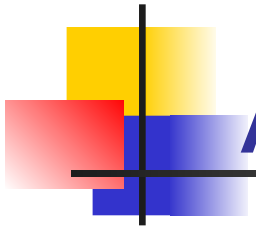
AES

- One byte is one finite field element in GF (2^8)
  - $b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0 \Rightarrow b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0$
- Operation sum between two polynomials in GF is defined as an XOR of the coefficients with the same degree of both polynomials.
- Multiplication operation in GF (2^8) corresponds to the multiplication of polynomials with modulus of an irreducible polynomial of degree 8
  - AES: $m(x) = x^8 + x^4 + x^3 + x + 1$



AES

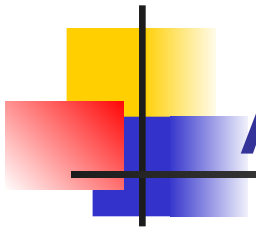
- For any nonzero polynomial b(x) of degree less than 8, the multiplicative inverse of b(x), $b^{-1}(x)$, can be obtaiend:
  - Euclid's algorithm b(x)a(x) + m(x)c(x) = 1
  - a(x)b(x) mod m(x) = 1
  - $b^{-1}(x)$ = a(x) mod m(x)



AES

- October 2000, Rijndael algorithm
- Block encryption algorithm: 128, 192 or 256 bits
- Key: 128, 192 or 256 bits
- Byte-level operations

- AES applies a specified number of rounds to an intermediate value called *state*
  - State ≡ Rectangular matrix ($4xN_b$)
- The key is stored in an array
  - Key ≡ Rectangular matrix ($4xN_k$)

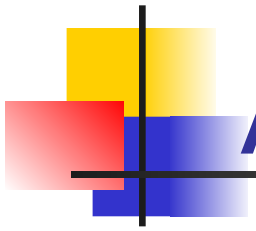


AES

- The plain text block is passed into an array the same size as the state array ($4xN_b$)

ex: $N_b=4$ $\quad X = \begin{pmatrix} b0 & b4 & b8 & b12 \\ b1 & b5 & b9 & b13 \\ b2 & b6 & b10 & b14 \\ b3 & b7 & b11 & b15 \end{pmatrix}$

- **<u>AES algorithm</u>**: Where X is the block that we want to encrypt and S is the state matrix:
  - Calculate K_0, K_{one}, K_{two}, ... K_n
  - $S = X \oplus K_0$
  - For i = 1 to n
    - Apply i-th round of the algorithm with subkey K_i



AES: Subkeys Calculation

- Subkeys are obtained by applying two functions: expansion and selection
  - The expansion function allows to obtain a sequence of $(n + 1) * 4 * N_b$ bytes
  - The selection consecutively takes blocks of the same size as the state matrix and assigns them to each K_i
- Let K (i) be a vector of bytes of size $4 * N_k$, and let W (i) be a vector of $N_b * (n + 1)$ 4-byte registers, the expansion function has two versions depending on the value of N_k:

AES: Subkeys Calculation

a. If $N_k \leq 6$

1. For i = 0 to N_{k-1}
2. W(i) = [K(4*i), K(4*i+1),K(4*i+2),K(4*i+3)]
3. For i = N_k up to $N_b(n + 1)$
4. tmp = W(i-1)
5. If i mod N_k = 0
6. tmp = Sub(Rot(tmp))$\oplus$R((i/N_k)-1)
7. W(i) = W(i-N_k) $\oplus$ tmp

AES: Subkeys Calculation

a. If $N_k > 6$

1. For i=0 to N_{k-1}
2. W(i) = [K(4*i),K(4*i+1),K(4*i+2),K(4*i+3)]
3. For i = N_k up to $N_b(n + 1)$
4. tmp = W(i-1)
5. If i mod N_k = 0
6. tmp = Sub(Rot(tmp))$\oplus$R ((i/N_k)-1)
7. If i mod N_k = 4
8. tmp=Sub (tmp)
9. W (i) = W(i-N_k) $\oplus$ tmp

AES: Subkeys Calculation

Sub () function

	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
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

AES: Subkeys Calculation

- Rot function: circular shift to the left of one position
- Function $R_c((i/N_k)-1)$: Each $R((i/N_k)-1)$ is the element of GF (2^8) corresponding to the value $x^{((i / Nk) -1)}$
  - R (0) = 01000000
  - R (1) = 02000000
  - R (2) = 04000000
  - R (3) = 08000000
  - R (4) = 10000000
  - R (5) = 20000000
  - R (6) = 40000000
  - R (7) = 80000000
  - R (8) = 1B000000
  - R (9) = 36000000
  - R (10) = 6C000000
  - R (11) = D8000000
  - R (12) = AB000000
  - R (13) = 4D000000
  - R (14) = 9A000000

AES: Rounds

- The number of rounds depends on the size of the block and the size of the key

Value of n	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k=4$	10	12	14
$N_k=6$	12	12	14
$N_k=8$	14	14	14

- Where S is the state matrix and K_i the subkey corresponding to the i-th round, each round:
  - 1. S = Sub (S)
  - 2. S = Move_row (S)
  - 3. S = Mix_columns (S)
  - 4. S = $K_i \oplus S$

NOTE: The last round is the same as the first but eliminating step 3

AES: Rounds

- Sub() function: non-linear substitution that is applied to each byte of the state matrix S using an invertible 8x8-bit S-box. (See slide 25)

- Row_Shift() function: cyclical left shift of the rows of the state array

N_b	r_1	r_2	r_3
4	1	2	3
6	1	2	3
8	1	3	4

  - Row 0 remains unchanged

AES: Rounds

- Mix_columns () function: matrix multiplication where each column is a state vector that is considered a polynomial whose coefficients belong to GF (2^8).

Multiplication matrix

$$S = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \qquad M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

$$s_0 = s_0 * 02 \oplus s_1 * 03 \oplus s_2 * 01 \oplus s_3 * 01$$

$$s_1 = s_0 * 01 \oplus s_1 * 02 \oplus s_2 * 03 \oplus s_3 * 01$$

$$s_2 = s_0 * 01 \oplus s_1 * 01 \oplus s_2 * 02 \oplus s_3 * 03$$

....

These multiplications are in GF (2^8) ⇨ for simplicity use tables E and L

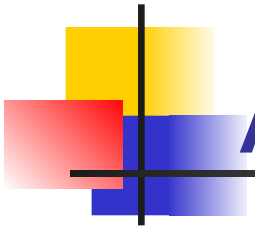
AES: Rounds

L Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	23	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	07

E Table

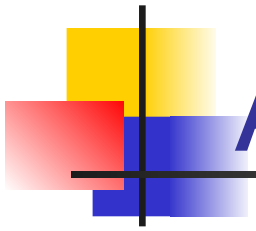
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01



AES

- Decryption:
  - Apply the inverses of each of the functions in the opposite order and use the same K_i but in reverse order.
  - Inverse of Sub() on next slide.
  - Inverse of Shift_Row(): circular shift to the right.
  - Inverse of Mix_column(): the multiplication matrix is

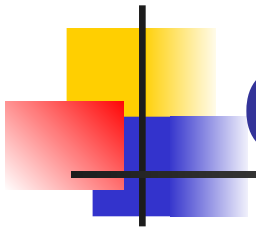
$$M = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix}$$



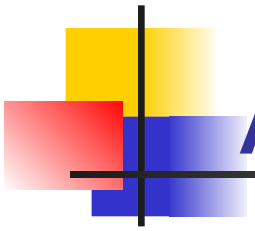
AES

Inverse Sub() function

	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
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



Contents

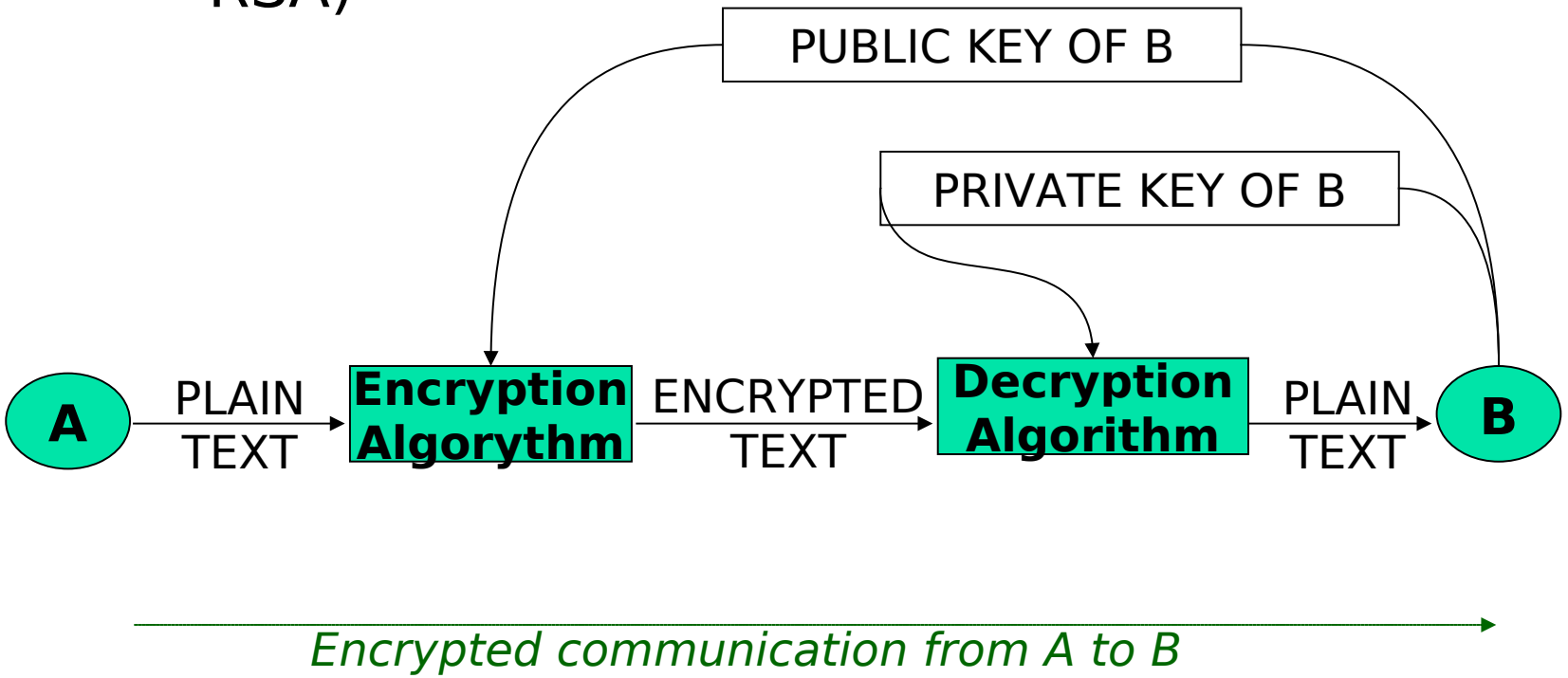


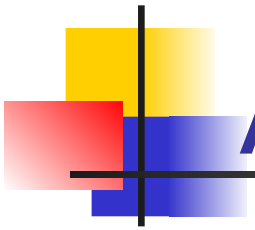
Asymmetric algorithms

- Asymmetric algorithms
  - Based on mathematical functions
  - Two keys
  - Keys of considerable length
  - Slower
  - They are basically used to encrypt the symmetric session key of each particular message or transaction
  - Ex: RSA
- Problems to solve in symmetric cryptography
  - Key distribution
  - Digital signature

Asymmetric algorithms

- In asymmetric cryptography: one key to encrypt and another to decrypt (special case RSA)





Asymmetric algorithms

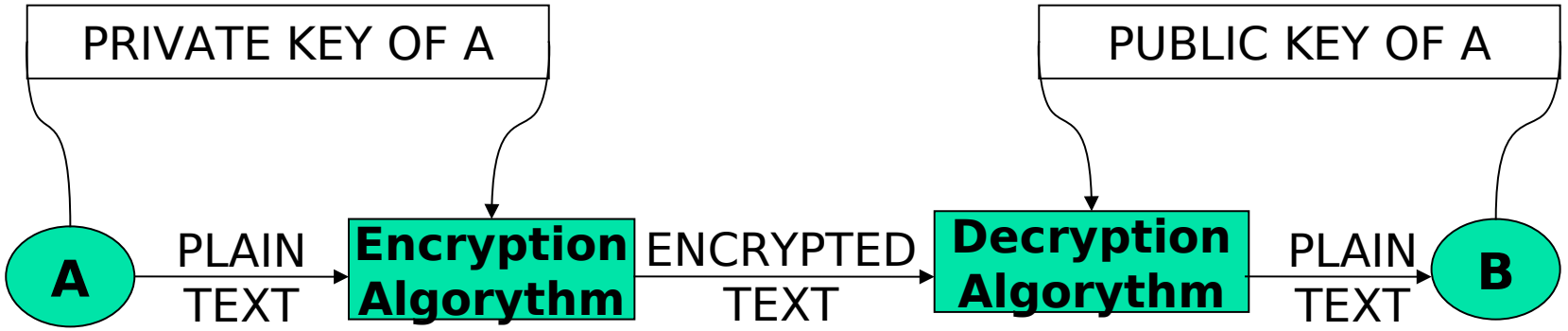
NOTATION:

- KP_A: Public key of A
- Kp_A: Private key of A

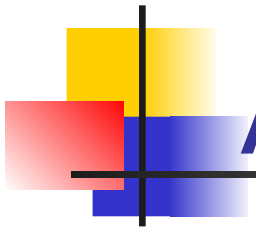
- According to the example user A would do $Y=E_{KPB}(X)$
- According to the example user B would do $X=D_{KpB}(Y)$

Asymmetric algorithms

- Use of asymmetric algorithms for authentication

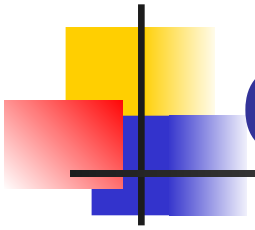


Communication with issuer authentication and data integrity

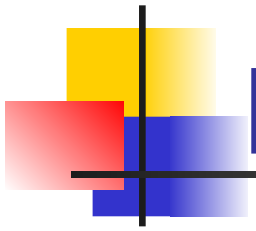


Asymmetric algorithms

- Applications
  - Encryption / decryption
  - Digital signature
  - Key exchange



Contents



RSA algorithm

- RSA = Rivest, Shamir, Adleman in 1977
- Keys indistinctly to encrypt, decrypt and / or authenticate

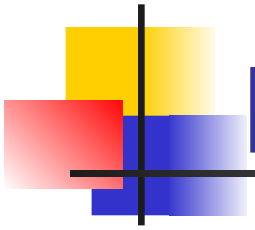
$$Y = X^e \bmod n$$

$$X = Y^d \bmod n$$

- Keys:

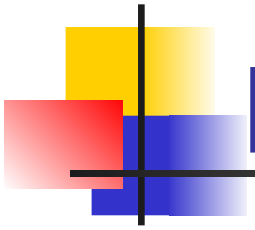
  Public key KP={e,n}

  Private key Kp={d,n}



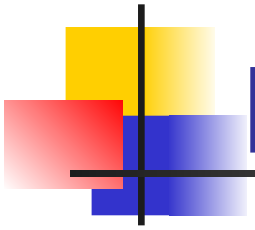
RSA algorithm

- Requirements:
  - Possible to find values e, d and n such that X^{ed} be equal to X mod n
  - Relatively easy to calculate X^{e} and Y^{d} for all values of X <n
  - Impossible to determine d even though e and n are known



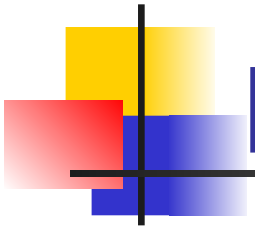
RSA algorithm

- Ingredients:
  - p and q two prime numbers
  - $n = p * q$
  - $d \mid \gcd(\Phi(n),d) = 1$ where $1 < d < \Phi(n)$
  - $e \mid e = d^{-1} \bmod \Phi(n)$



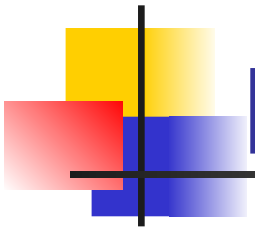
RSA algorithm

- Simple example



RSA algorithm

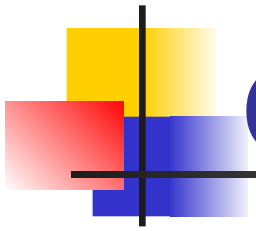
- Encryption example



RSA algorithm

- In practice it is necessary to choose values of p and q with approximately 154 digits (it is recommended that n~1024 bits ⇨ 308 digits)
- To obtain the private key from the public one, the attacker must know p and q ⇨ computationally intractable ⇨ factoring problem
- Attacks on RSA
  - Brute force
  - If $X = X^e \bmod n$ ⇨

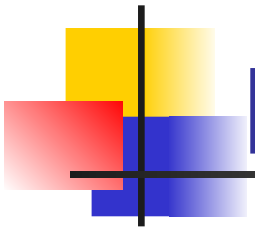
    $\theta_n = [1 + \gcd(e-1, p-1) * [1 + \gcd(e-1, q-1)]$
  - Middleman attack ⇨ trust rings



Contents

Diffie-Hellman algorithm

- 1976, first publication on public key encryption ⇨ key exchange technique
  - Public keys are not strictly necessary
- Effectiveness lies in the difficulty of calculating discrete logarithms:
  - Primitive root of a prime number p is a value α whose powers generate different integers between 1 and p-1.
  - For any integer b and a primitive root α of a prime number p, it is possible to find a single exponent i such that $b = \alpha^i \bmod p$



Diffie-Hellman algorithm

- Two public values:
  - a prime number q
  - An integer α which is primitive root of q
- If A and B want to exchange a key:

1) A selects random number x_A (x_A <q) and calculates

$$y_A = \alpha^{x_A} \bmod q$$

Diffie-Hellman algorithm

2) In the same way, B selects random number x_B ($x_B<q$) and calculates

$$y_B = \alpha^{x_B} \bmod q$$

3) Each one saves x (x_A and x_B) secretly and publish y (y_A and y_B) to the other partner

4) User A calculates his secret key as

$$K_A = (y_B)^{x_A} \bmod q$$

5) User B calculates his secret key as

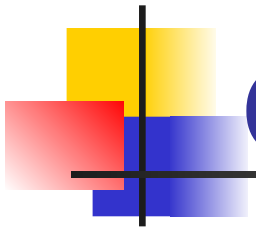
$$K_B = (y_A)^{x_B} \bmod q$$

It can be shown that K_A and K_B they are the same value.

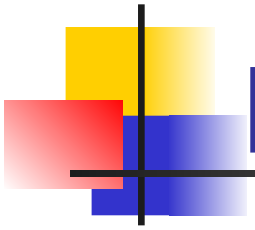
Diffie-Hellman algorithm

- Both partners exchange a secret key securely
  - If the transmitted messages are captured (q, α, y_A, y_B) we would have to calculate discrete logarithm
  - Example

$$x_B = \ln d_{\alpha,q}(y_B)$$



Contents

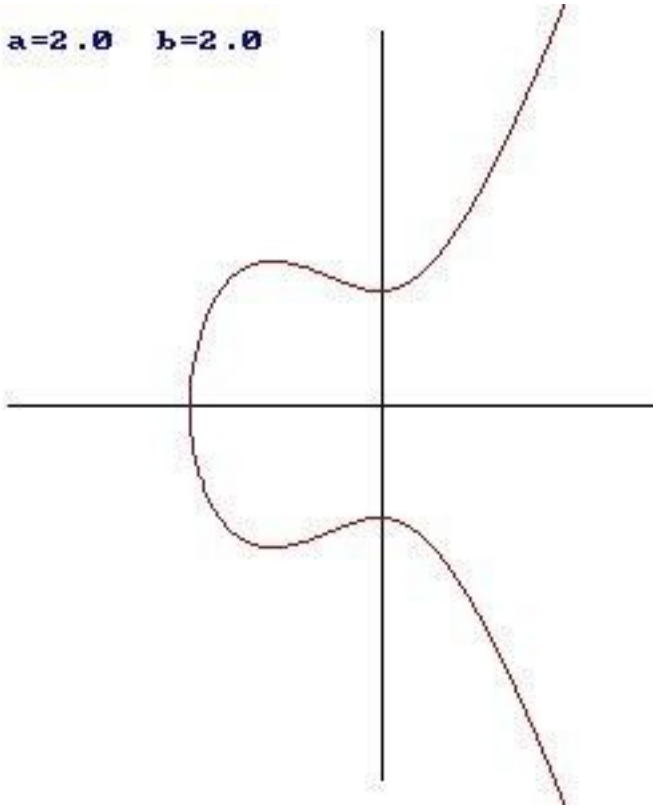


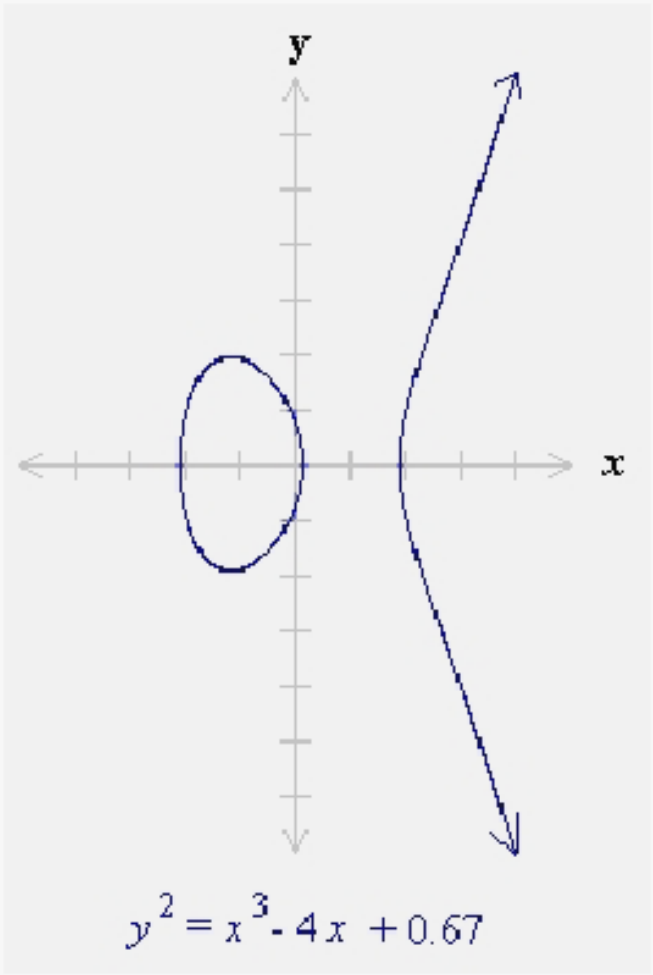
Elliptical curves

- RSA uses increasingly long keys $\Rightarrow \uparrow t_{processing}$
- New competitive system: Elliptic Curve Cryptography (ECC):
  - RFC 3278, IPSec, etc ...
  - Advantage of ECC over RSA: offers equal security for much shorter key lengths
  - Recent Cryptanalysis: www.certicom.com

Elliptical curves

- Elliptical curves are not ellipses:





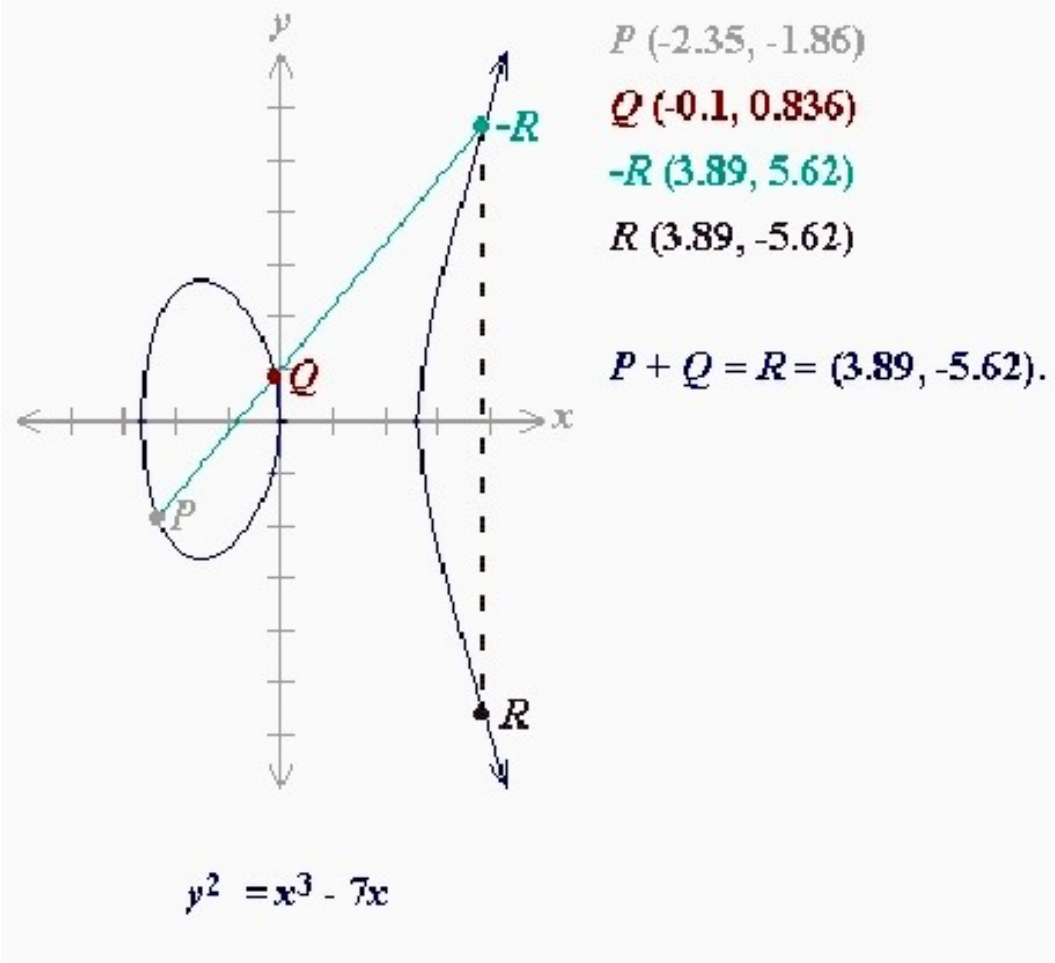
Elliptical curves

- Equation of an elliptical curve:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

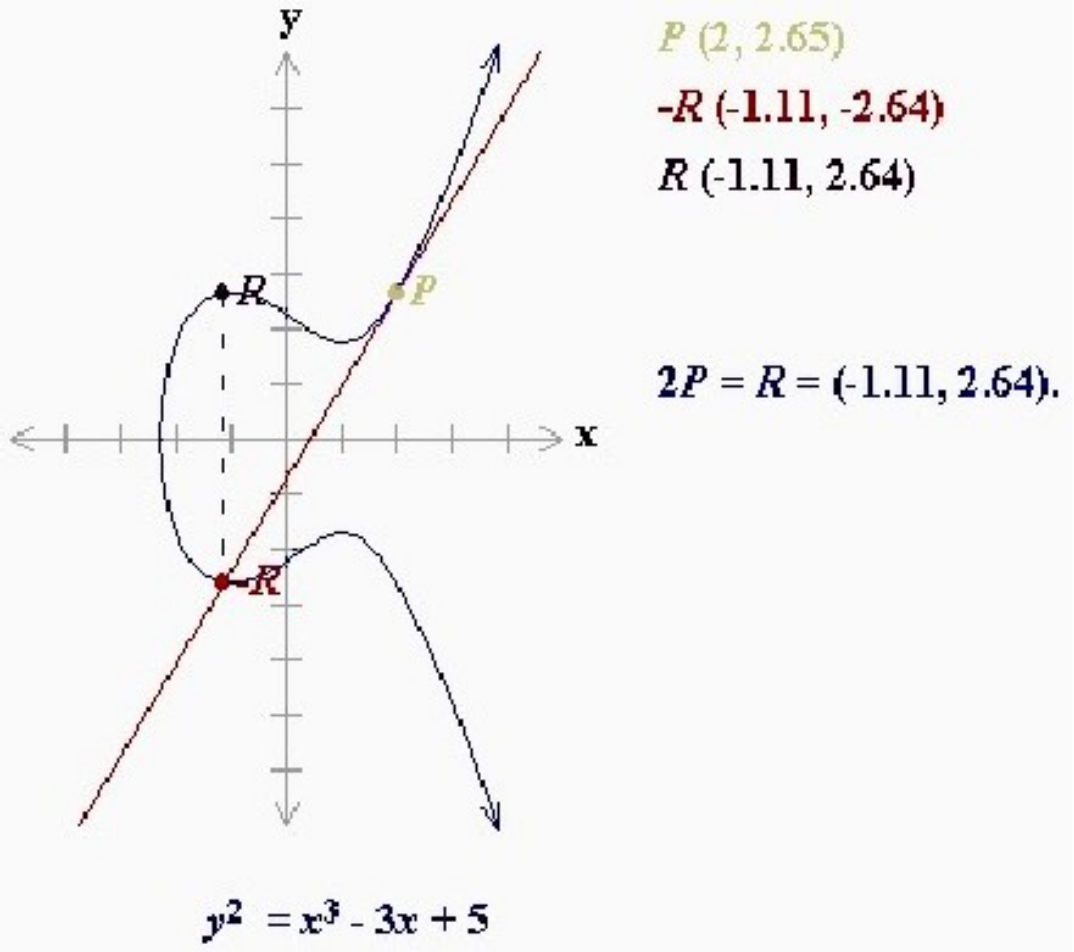
- Sum operation: ”if three points of an elliptical curve fall in a straight line, then their sum is *0*"
  - *0* ≡ point at infinity
- Sum rules of an elliptic curve:
  - 1) To add two points P and Q with different x coordinate, you have to draw a line between them, find the third point of intersection R ⇒ Q + P + R =*0* ⇒ Q + P = -R

Elliptical curves

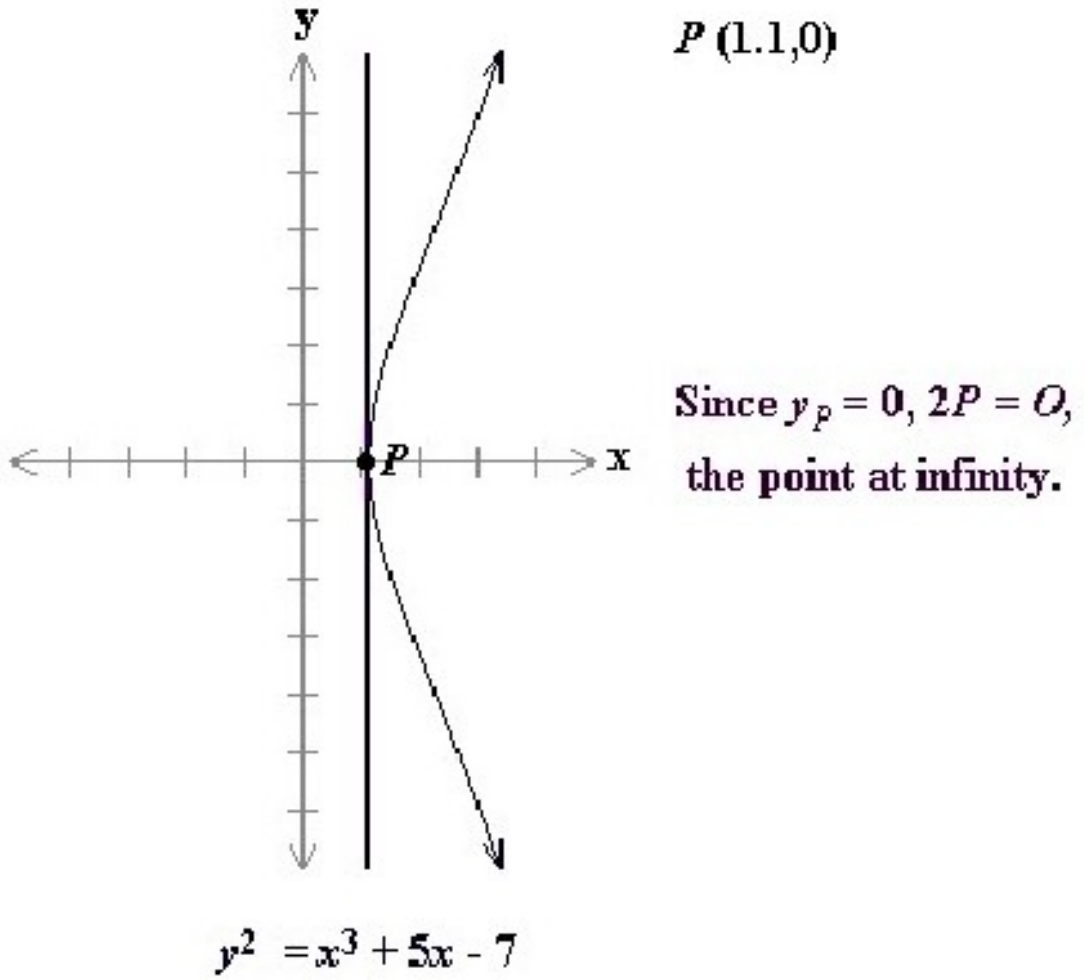


Elliptical curves

- 2) To double a point P, draw the tangent and find the other point of intersection R. Draw the vertical parallel to the axis through R ⇒ P + P = 2P = R

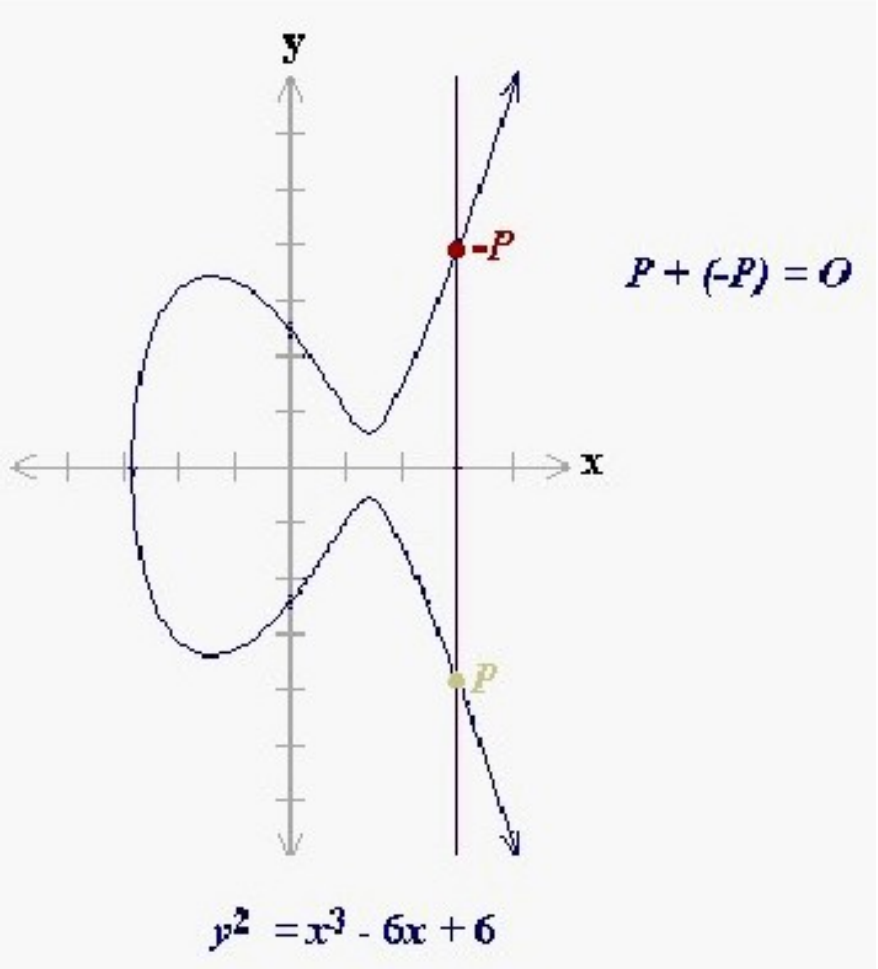


Elliptical curves



Elliptical curves

- 3) Given a vertical line that cuts the elliptical curve at two points with the same x-coordinate, it also cuts the curve at the point at infinity $\Rightarrow P_1+P_2=0 \Rightarrow P_1=-P_2$



Elliptical curves

- 4) *0* is the additive identity ⇒ *0* = *-0* ⇒ P + *0* = P

- In any case, the associative and commutative property hold

Elliptical curves

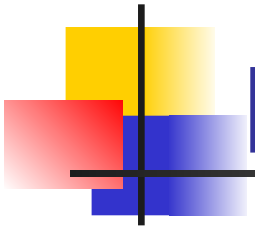
- Elliptic curves over finite fields: elliptical group in modulus p (p is a prime number)
- We choose two non-negative integers (a and b) where a, b <p such that:

$$4a^3+27b^2 \pmod{p} \neq 0$$

- $E_p(a, b)$ **elliptical group** mod p whose elements (x, y) are non-negative integer pairs less than p that satisfy:

$$y^2 \equiv x^3+ax+b \pmod{p}$$

along with the point at infinity



Elliptical curves

- Create list of points that belong to $E_p(a,b)$:
  - For each x / $0 \leq x < p$ calculate $x^3+ax+b \pmod p$
  - For each previous result determine if it has a square root mod p.
    - If it does not have then there are no points in $E_p(a,b)$ with this value of x.
    - If it does then there will be two values of y that satisfy the root.

Elliptical curves

- Sum rules in $E_p(a,b)$: for all points $P,Q \in E_p(a,b)$:

1. $P+\mathit{0}=P$
2. If $P=(x,y) \Rightarrow P+(x,-y)=\mathit{0}(x,-y)=-P$
3. If $P=(x_1,y_1)$ and $Q=(x_2,y_2)$ with $P \neq -Q \Rightarrow P+Q = (x_3,y_3)$

$$x_3=\lambda^2-x_1-x_2 \pmod p \qquad y_3=\lambda(x_1-x_2)-y_1 \pmod p$$

$$\lambda=\begin{cases}\dfrac{y_2-y_1}{x_2-x_1} \pmod p \ if \ P\neq Q \\ \dfrac{3x_1^2+a}{2y_1} \pmod p \ if \ P=Q\end{cases}$$

Elliptical curves

- We need to find a difficult problem to solve ...
  - $Q = k \cdot P$ where $Q, P \in E_p(a, b)$ and $k < p$
- **Diffie-Hellman key exchange with ECC**
  - We choose prime number p (~180 bits) and parameters a and b
  - We select generator point G (x_1, y_1) in $E_p(a, b)$ such that $n \cdot G = 0$ for a very large n (n prime number)
  - A selects integer $n_A < n$ and calculates $P_A = n_A \cdot G$ (n_A secret)
  - B selects integer $n_B < n$ and calculates $P_B = n_B \cdot G$ (n_B secret)
  - A and B exchange P_A, P_B
  - A calculates its secret key as $K = n_A \cdot P_B$
  - B calculates its secret key as $K = n_B \cdot P_A$

Elliptical curves

- **Encryption / Decryption with ECC**
  - Various possibilities: (ElGamal, DSA, etc.) with ECC
  - Encode message in points $P_m(x,y)$
  - Generating point G and elliptical group $E_p(a,b)$
  - Each user chooses a secret value n_x and generates a public key $P_x = n_x \cdot G$
- Encrypt (message from A to B):
  - A picks random positive integer K ⇨ encrypted message $C_m = \{K \cdot G, P_m + K \cdot P_B\} = \{P_1, P_2\}$
- Decrypt:
  - B multiplies the first point of the pair (P_1) with its secret value n_B and subtracts the result from the second point of the pair (P_2)

Elliptical curves

- Attacks with the Pollard's Rho method
- Comparison of security levels:

Nivel de Seguridad	Esquema Simétrico (tamaño de clave)	Esquema basado en ECC (tamaño de n)	DSA/RSA (tamaño del modulo)
56	56	112	512
80	80	160	1024
112	112	224	2048
128	128	256	3072
192	192	384	7680
256	256	512	15360

COMPARABLE KEY SIZES

	Parámetros del sistema	Clave Pública	Clave Privada
RSA	n/a	1088	2048
DSA	2208	1024	160
ECC	481	161	160

SYSTEM PARAMETER SIZES AND KEY PAIR (bits)

	Tamaño mensaje encriptado
RSA	1024
ElGamal	2048
ECC	321

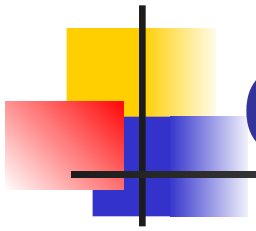
SIZE OF ENCRYPTED MESSAGES

	Tamaño de firma
RSA	1024
DSA	320
ECC	320

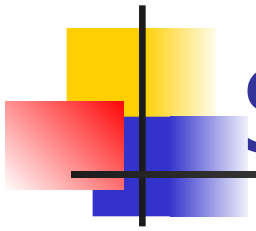
SIGNATURE SIZE (bits)

ECC	
Key size	MIPS-Years
150	$3.8\ 10^{10}$
205	$7.1\ 10^{18}$

RSA	
Key size	MIPS-Years
512	$3\ 10^{4}$
1024	$3\ 10^{11}$

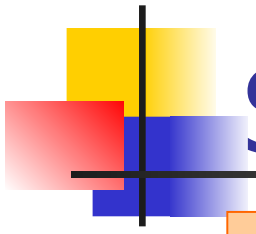


Contents



Stream Encryption

- The message to be encrypted is NOT divided into blocks
- Stream encryption encrypts in real time
- 1917 Mauborgne and Vernam invented the first cryptosystem for stream encryption:
  - Combine character by character plain text with a random sequence of equal length using a simple and reversible function (ex. XOR)
  - Sent only once
  - Problem: the password is as long as the message itself and how to send the password?

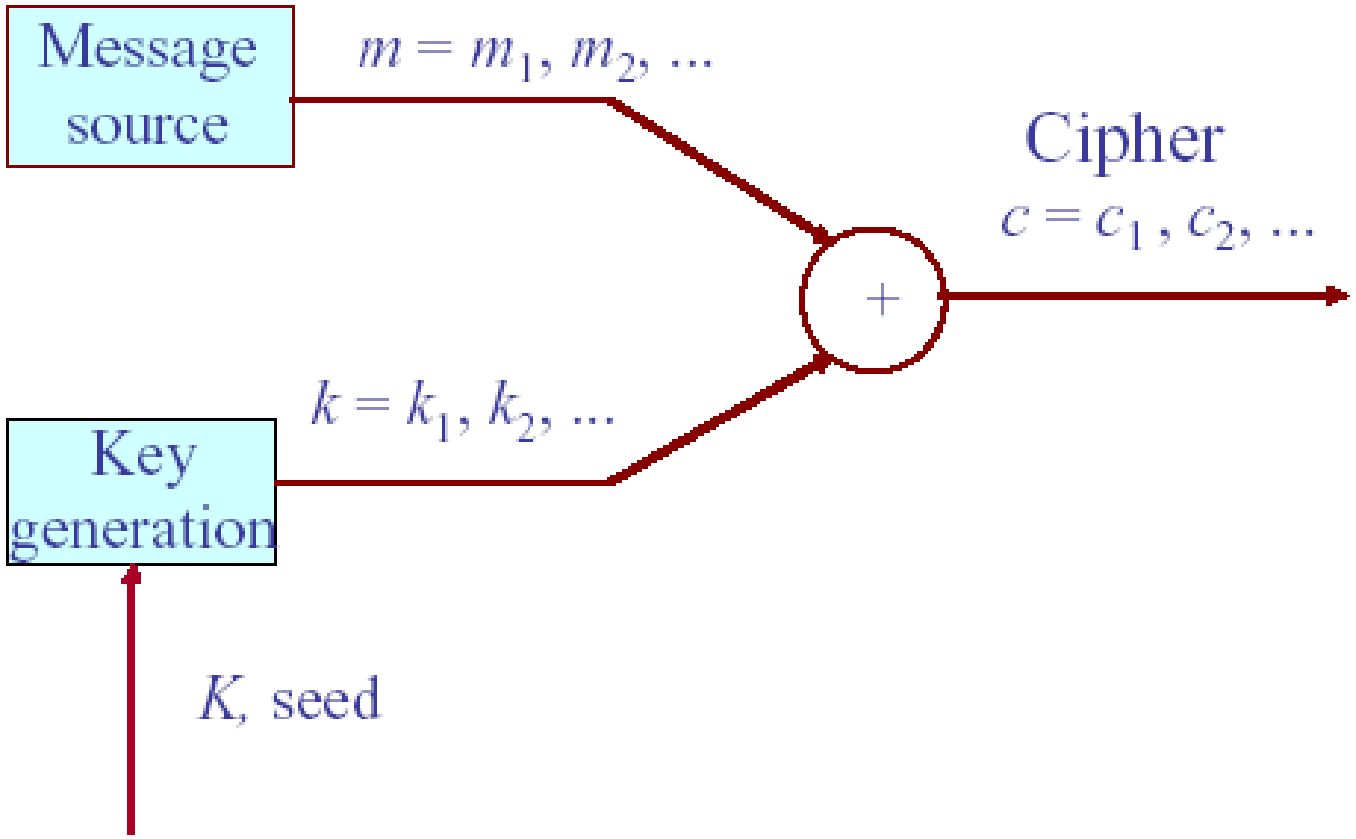


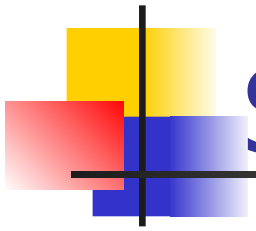
Stream Encryption

Pseudo-random generator ⇨ cryptographically random sequences
+
Pseudo-random generator seed = K key

Encrypt: XOR pseudo-random sequence with plain text
Decrypt: From the seed, reconstruct pseudo-random sequence and XOR with encrypted message

Stream Encryption





Stream Encryption

- We will see private key cryptosystems
  - Specifications of a pseudo-random generator
  - Combination using the XOR function
  - Byte-by-byte operations

- Types of generators:
  - Synchronous
  - Asynchronous

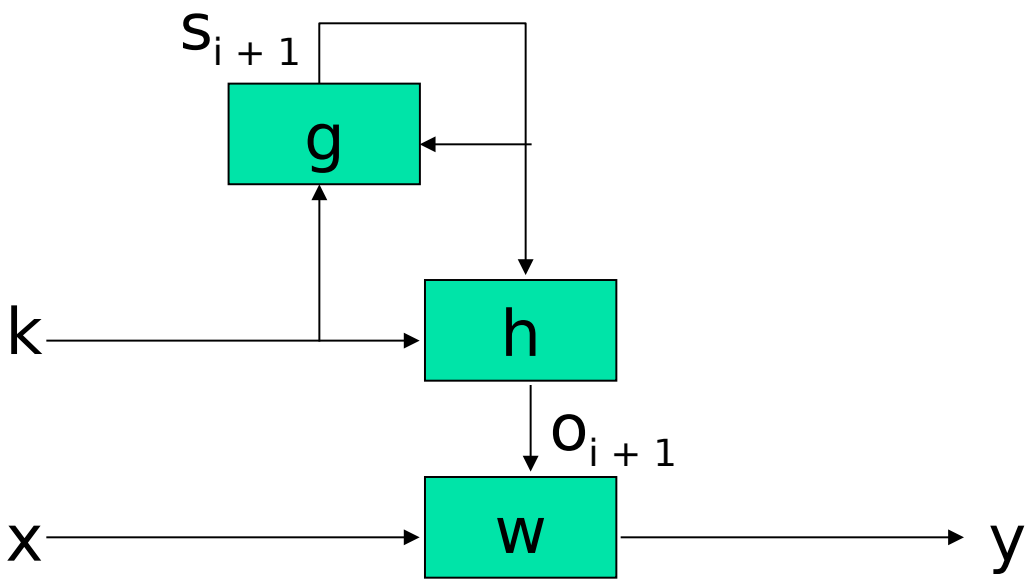
Stream Encryption

- **Synchronous generator**: The sequence is calculated independently of both plaintext and encryption.

$s_{i+1} = g(s_i, k)$

$o_i = h(s_i, k)$

$y_i = w(x_i, or_i)$



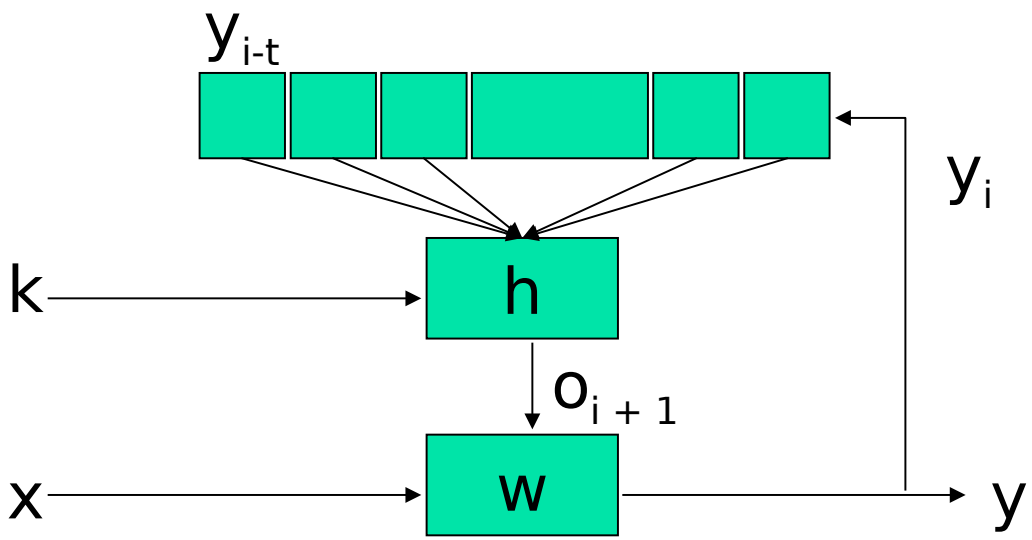
- Transmitter and receiver must be synchronized (verification and synchronization restoration techniques)

Stream Encryption

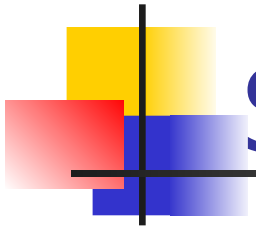
- **Asynchronous generator**: The generated sequence is a function of a seed plus a fixed amount of the previous bits of the sequence itself.

$o_i = h\ (K, y_{i-t}, y_{i-t+1}, y_{i-t+2}, \ldots, y_{i-1})$

$y_i = w\ (o_i, x_i)$

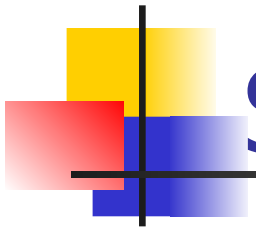


- Bit loss or insertion resistant
- Sensitive to tampering with encrypted messages (verification techniques)



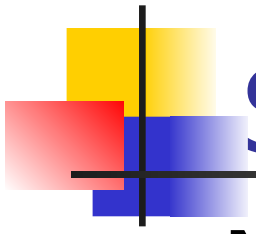
Stream Encryption

- Sequence generators for streaming encryption based on Feedback Shift Register:
  - Linear
  - Non linear



Stream Encryption

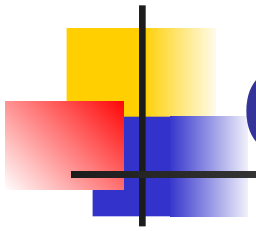
- **Linear Feedback Displacement Registers** (Linear Feedback Shift Register, LFSR)
  - Set of L states $\{s_0, s_1, ..., s_{L-1}\}$, where each state stores 1 bit
  - Clock controls variation of states
  - Each time unit:
    - s_0 is the register output
    - Content of s_i shifts to s_{i-1} $(1 \leq i \leq L-1)$
    - the content of s_{L-1} is calculated as the modulo 2 sum of the values of a preset subset of the register.



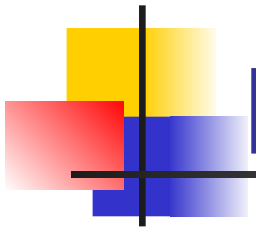
Stream Encryption

- **Nonlinear Feedback Displacement Records** (Non Linear Feedback Shift Register, NLFSR)
  - Set of L states $\{s_0, s_1, ..., s_{L-1}\}$, where each state stores 1 bit
  - Clock controls variation of states
  - Each time unit:
    - s_0 is the register output
    - content of s_i shifts to s_{i-1} $(1 \leq i \leq L-1)$
    - Content of s_{L-1} is calculated as a boolean function

    $f(s_{j-1}, s_{j-2}, ..., s_{jL})$
- In general they are used *n* linear generators and a nonlinear function f to combine their outputs: f $(R_1, R_2, ..., R_n)$

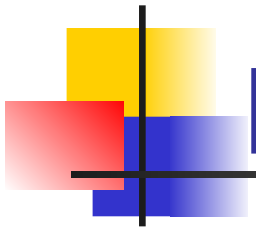


Contents



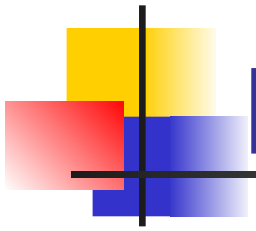
RC4

- Private key stream encryption algorithm designed by Ron Rivest (1987)
- Proprietary algorithm
- Software implementation
- Included in protocols and standards such as WEP (Wired Equivalent Privacy) or SSL (Secure Socket Layer)
- Key up to 256 bits (typically 40-256 bits)



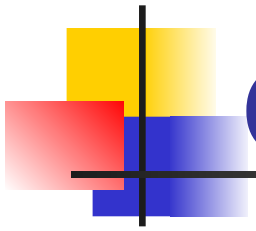
RC4

- Byte-by-byte encryption
- Encryption / decryption operations: uses 256 bytes of memory (s[0] to s[255]) and 3 variables i, j, k.
- Initial state:
  - 1) s[i]=i $\forall$i 0 $\leq$i $\leq$ 255
  - 2) j=0
  - 3) For i=0 to 255 do:
    - j=(j + s[i]+key[i mod key_length]) mod 256
    - Swap s[i] and s[j]

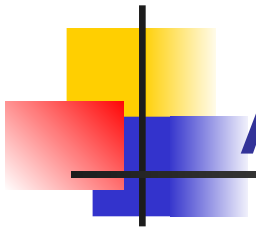


RC4

- Encrypt / Decrypt:
  - 1) i = 0; j = 0
  - 2) for each byte to be encrypted
    - i = (i + 1) mod 256
    - j = (j + s [i]) mod 256
    - Swap s [i] and s [j]
    - k = (s [i] + s [j]) mod 256
    - XOR of s [k] with next input byte
- Discard the first few bytes of generator output and do not use them for encryption
- Other applications: Lotus Notes, Windows key encryption, MS Access, Adobe Acrobat, Oracle Secure Server, etc.



Contents

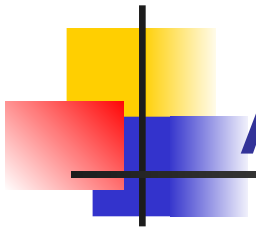


A5

- GSM developed by the European Institute of Telecommunications Standards ⇨ cryptographic protocols for confidentiality and authentication
- A3 authentication algorithm
- A5 voice encryption algorithm
- A8 key generator algorithm
- COMP128 algorithm to run A3 and A8

A5

A8; K_i SIM
Random data?
D_1= random data
A3 (K_i+ D_1) -> $result_1$
$result_1$
A3 (K_i from BS + D_1) -> $result_2$
$result_1$= $result_2$? YES
free communication
A8; K_i SIM
Random data?
D_2= random data
A8 (K_i+D_2)=K_c
A8 (K_i+ D_2) = K_c

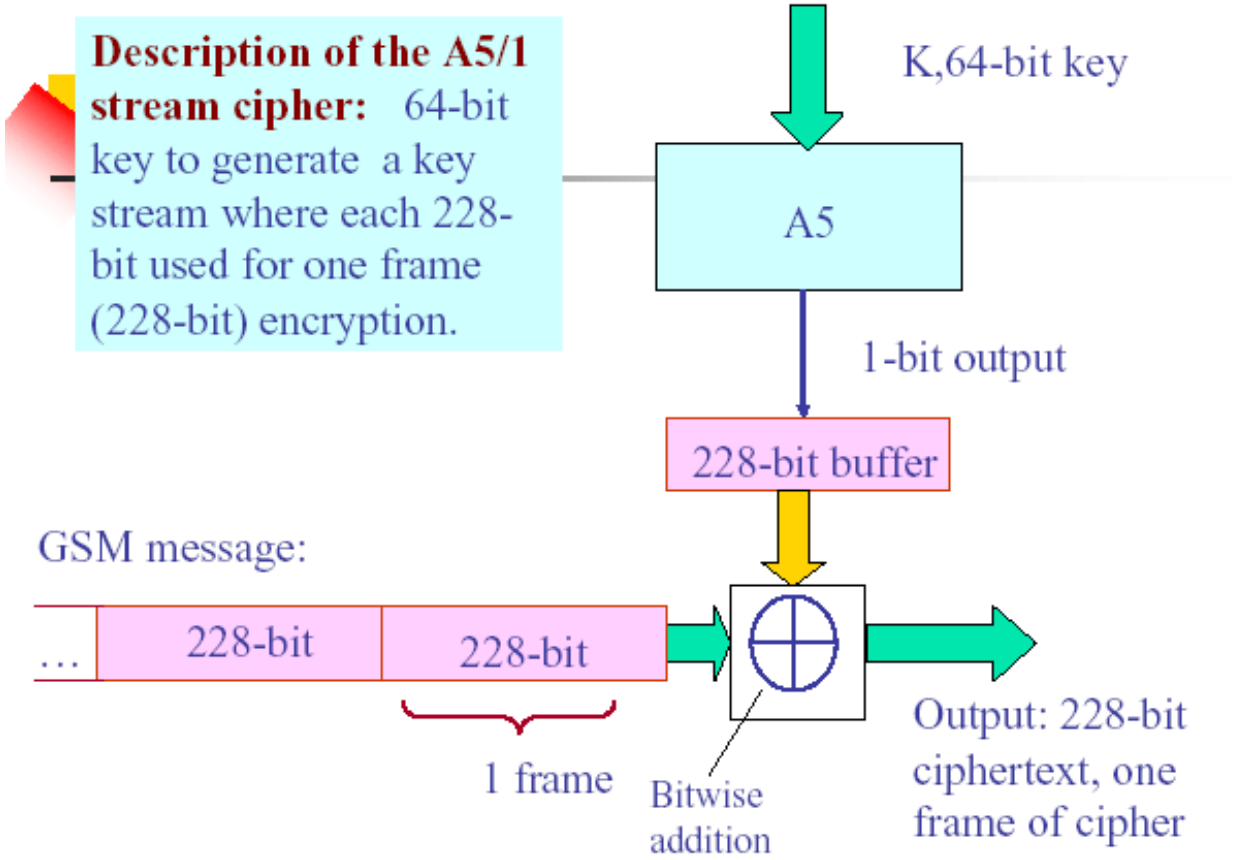


A5

- Provides privacy for GSM communications on the radio interface (GSM 1 frame every 4.6 ms; 1 frame 228 bits)
- Two versions A5/1 and A5/2
- Both are a combination of three linear feedback shift registers with irregular clock signals and a non-linear combiner

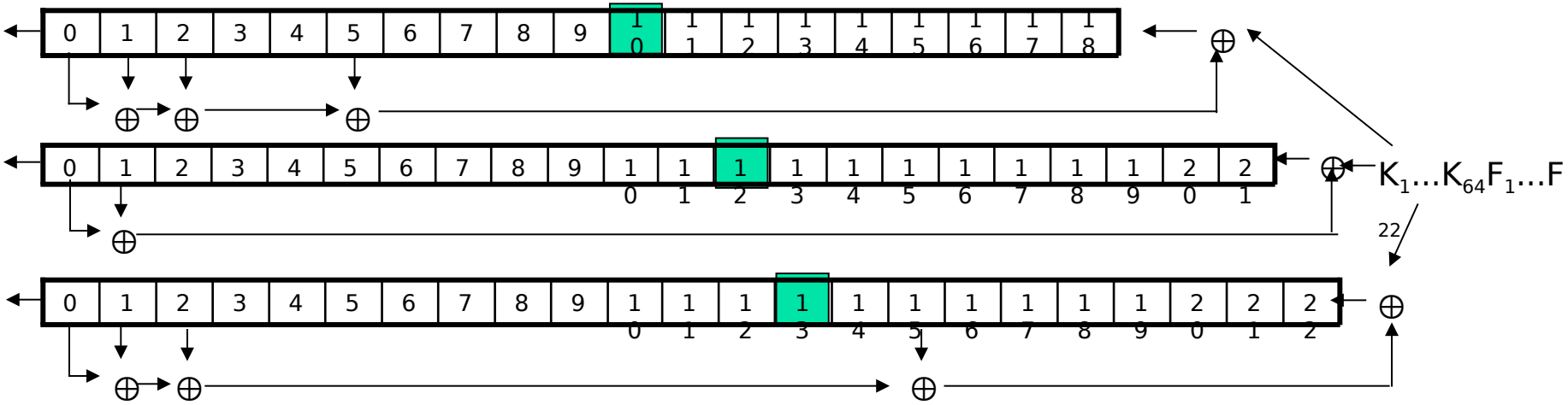
A5

- **A5/1** uses 64-bit secret key and generates sequence of bits (every 228 bits a frame is encrypted)



A5

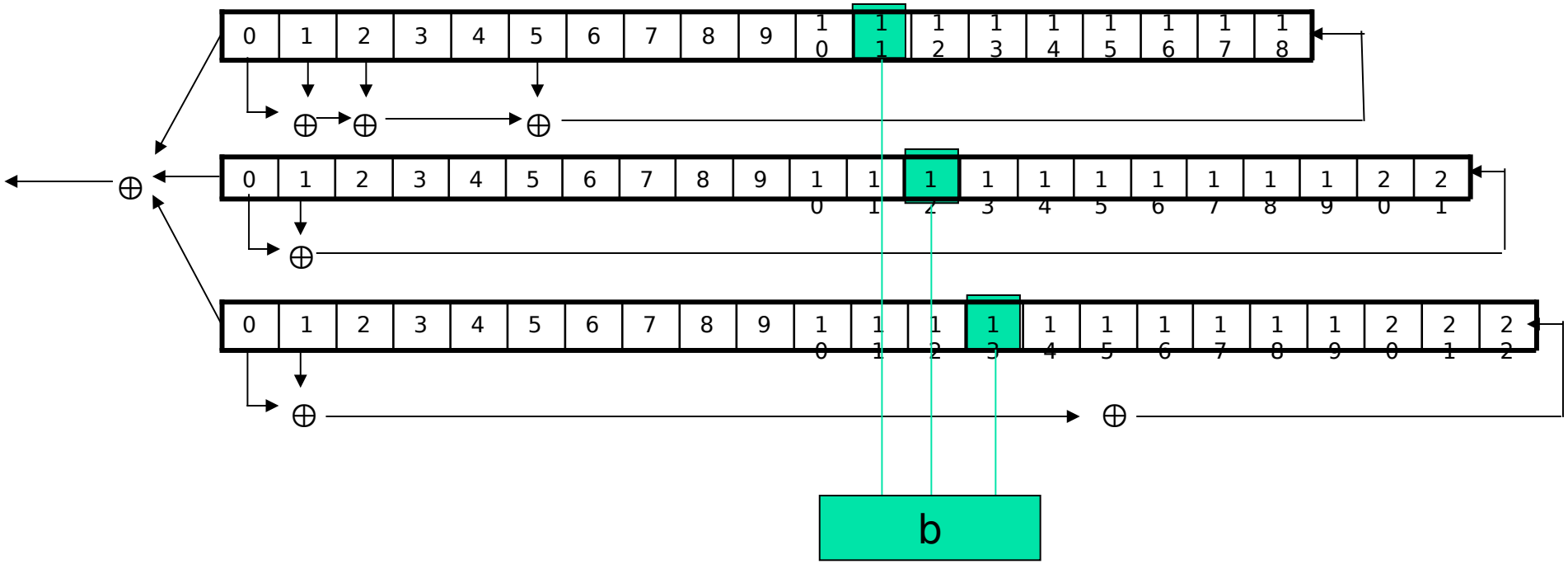
- 3 LFSR of lengths 19, 22 and 23 bits ($x^{19}+ x^5+ x^2+ x + 1$; $x^{22}+ x + 1$; $x^{23}+ x^{15}+ x^2+ x + 1$)
- Initialization of each frame: from t = 1 to t = 64 K is used, from t = 65 to t = 89 the bit (t-64) of frame number F is used

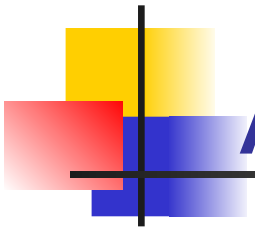


INITIALIZATION PROCESS

A5

- Each LFSR has a *tap* clock
- The majority value (b) of the three is calculated *taps* each unit of time
- LFSR receives clock signal only if its *tap* matches b value





A5

- Attacks:
  - By brute force
  - Goldberg, Wagner, Briceno => "of the 64 bits of the key ten of them are always zero"
  - Briceno => reverse engineering in December 1999
  - Biryukov, Shamir => "Real time cryptoanalysis of A5 / 1 on a PC", 1PC with 128 Mb RAM, 2-4 hard drives of 73 Gb each, digital scanner.
- 3G
  - Key generation: MILENAGE
  - Confidentiality and integrity in radio interface: KASUMI