

TREBALL DE RECERCA DE BATXILLERAT

Història i actualitat de la criptografia



Nom: Sergi Magret Goy

Nom tutora: Susanna López

Curs: 2n Batxillerat C

INS Ramon Muntaner

Localitat i data de lliurament:

Figueres, 30 de setembre de 2016

Índex

Abstract.....	3
Introducció	4
Part 1. Coneixements teòrics	5
1. Història de la criptografia	5
1.1. Criptografia clàssica.....	5
1.1.1. L'escital espartà	6
1.1.2. Xifratge Cèsar	7
1.1.3. El mètode de Polibi.....	9
1.2. Criptografia medieval	10
1.3. Criptografia Europea	12
1.3.1. Els xifratges polialfabètics.....	12
1.3.1.1. Xifratge d'Alberti.....	12
1.3.1.2. Xifratge de Vigenère o taula de Vigenère	13
1.4. La criptografia fins la Segona Guerra Mundial.....	15
1.5. La màquina Enigma a la Segona Guerra Mundial.	16
1.6. Criptografia moderna	19
1.6.1. Criptografia simètrica	19
1.6.1.1. Data Encryption Standard (DES)	19
1.6.1.2. Triple Data Encryption Standard	22
1.6.1.3. Advanced Encryption Standard (AES)	23
1.6.2. Criptografia asimètrica	24
1.6.2.1. RSA (Rivest, Shamir i Adleman).....	25
1.6.3. Criptografia híbrida.....	26
Part 2. Part pràctica – Xifratge Cèsar en C++.....	27
Conclusions	33
Índex d'imatges	34
Webgrafia.....	35

Annex 1	37
Annex 2	39
Annex 3	40
Annex 4	43

Abstract

Aquest treball explica la història de la criptografia des del que feien servir a Esparta, fins a algorismes que es fan servir actualment per a guardar les nostres dades de forma segura, passant pel funcionament de màquines com l'Enigma, utilitzada pels alemanys a la Segona Guerra Mundial. A la part pràctica he desenvolupat un programa el qual en introduir-hi un missatge, es xifra o es desxifra utilitzant el mètode de Juli Cèsar.

Paraules clau: criptografia, Cèsar, c++, criptografia simètrica, criptografia asimètrica, programació, substitució, transportació.

This research paper explains the history of the cryptography from what used in Esparta, to algorithms currently used to store our data safely, through how worked machines like Enigma, used by the Germans in World War II. In the practical part I have developed a program which when entering a message, is encrypted or decrypted using the method of Julius Caesar.

Key words: cryptography, Caesar, c++, symmetric cryptography, asymmetric cryptography, programming, substitution, transportation.

Introducció

Des del principi tenia clar que el meu treball de recerca, com a mínim tindria una part de programació, el problema que tenia era, què programo?

Un dia mirant la televisió, vaig sentir que parlaven sobre uns papers de principis del segle XIX que encara no s'havien pogut desxifrar, eren "Els papers de Beale", tres textos plens de números aparentment sense sentit, que uns miners van enterrar fa temps. Buscar informació sobre això em va portar a llegir alguns articles de com es xifraven els documents i vaig pensar, per què no faig el treball sobre l'enciptació i faig un programa que xifri frases i missatges?

Després de buscar informació de com es podia fer i si seria gaire complicat, vaig pensar que si, que en seria capaç i que podia fer un programa i és el que vaig decidir de fer.

Els meus objectius són:

- Trobar la suficient informació a internet com per a aprendre un llenguatge de programació i poder fer un programa de xifratge de missatges.
- Aprendre sobre els actuals mètodes de xifratge i així saber si les dades estan guardades de forma segura.

Part 1. Coneixements teòrics

La **criptografia** és la disciplina que s'encarrega de l'estudi de codis secrets, anomenats també codis xifrats (en grec *kriptos* significa secret i *gráphos* escriptura).

La criptografia és una disciplina molt antiga, els seus orígens es remunten al naixement de la nostra civilització. Al principi, el seu únic objectiu era el de protegir la confidencialitat d'informacions militars i polítiques. Però, en l'actualitat és una ciència important no només en aquests camps sinó per a qualsevol altre que estigui interessat en la confidencialitat d'unes determinades dades.

Encara que l'objectiu original de la criptografia era mantenir en secret un missatge, en l'actualitat no es persegueix únicament la **privacitat o confidencialitat** de les dades, sinó que es busca a més garantir l'**autenticació** dels mateixos (l'emissor del missatge és qui diu ser, i no un altre), la **seva integritat** (el missatge que llegim és el que ens han enviat, i no un altre), i el **seu no repudi** (l'emissor no pugui negar el fet d'haver enviat el missatge).

1. Història de la criptografia

1.1. Criptografia clàssica

Qüestions militars, religioses i comercials van impulsar des de temps remots la utilització d'escriptures secretes. Ja els antics egipcis utilitzaven mètodes criptogràfics. Per exemple, els sacerdots egipcis van usar l'escriptura hieràtica (jeroglífica) que era clarament incomprensible per a la resta de població. Els antics babilonis també usaren mètodes criptogràfics en la seva escriptura cuneïforme.

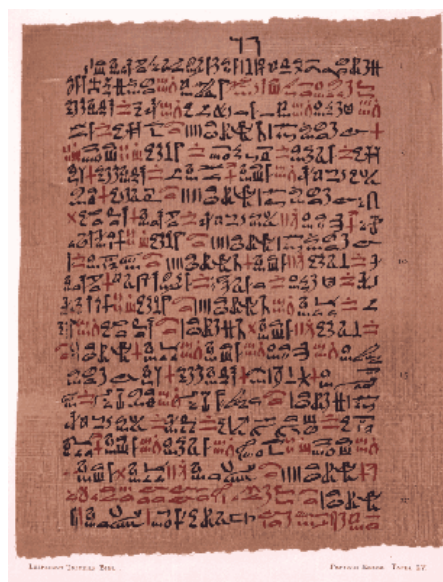


Figura 1 - Escripura hieràtica

Font: ca.wikipedia.org/wiki/Hier%C3%A0tic

1.1.1. L'escital espartà

El primer cas clar d'ús de mètodes criptogràfics va ser durant la guerra entre Atenes i Esparta. L'historiador grec Plutarc, descriu l'escital de la següent forma:

“La escitala era un palo o bastón en el cual se enrollaba en espiral una tira de cuero. Sobre esa tira se escribía el mensaje en columnas paralelas al eje del palo. La tira desenrollada mostraba un texto sin relación aparente con el texto inicial, pero que podía leerse volviendo a enrollar la tira sobre un palo del mismo diámetro que el primero”¹

Amb aquest sistema els governants d'Esparta van transmetre, amb eficàcia, les seves instruccions secretes als generals del seu exèrcit durant les campanyes militars. Lògicament, aquest procediment requeria que tant l'emissor com el receptor del missatge, disposessin d'un pal o bastó amb les mateixes característiques físiques.



Figura 2 - Escital Espartà

Font: <http://timerime.com/en/event/2800925/La+Escitala+Espartana/>

Per exemple el text “Així es xifraven els missatges” si utilitzéssim un escital de 5 columnes quedaria com “Aefn aisr mtx aeigíxvlse iesss”

¹ PLUTARC, Luci Mestri. *Vida de Lisandro*. 117 d.C

1.1.2. Xifratge Cèsar

Aquest mètode es va utilitzar durant la Roma Imperial. És anomenat així perquè és el procediment que utilitzava **Juli Cèsar** per enviar missatges secrets als seus legionaris, en aquell temps era molt difícil que els enemics aconseguissin descobrir què deia el missatge, però avui en dia és un mètode bastant senzill, ja que s'ha evolucionat molt en aquest camp.

És un **algorisme de substitució**, el xifratge original consistia a substituir la lletra del text original per la que hi ha tres llocs més endavant a l'alfabet, per tant la lletra A passaria a ser la D, la B a la E i així successivament fins que la Z passaria a convertir-se en la C.

Original	a	b	c	d	e	f	g	h	i	j	k	l	m
Xifrat	d	e	f	g	h	i	j	k	l	m	n	o	p
Original	n	o	p	q	r	s	t	u	v	w	x	y	z
Xifrat	q	r	s	t	u	v	w	x	y	z	a	b	c

En l'actualitat hi ha una variant, que és la que utilitzaré jo per a fer el treball de camp que és utilitzant tot l'alfabet, però en comptes de fer que les lletres es substitueixin per la de tres posicions més endavant, el meu usarà una **clau**, és a dir, depenent del número que tu vulguis utilitzar les lletres es substituiran per unes o per altres, i així és més segur, ja que per desxifrar el missatge necessites aquesta clau (el número) perquè si no el text no cobrarà sentit.

Aquest xifratge es pot reduir a una fórmula matemàtica bastant simple:

$$T = M + C$$

On M és la lletra del missatge original

On C és la clau a fer servir

On T és la lletra resultant de M després de sumar C

L'alfabet si el passem a números quedaria així:

A	B	C	D	E	F	G	H	I	J	K	L	M
01	02	03	04	05	06	07	08	09	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Si per exemple volguéssim xifrar “lluïta” amb una clau de 6, el que faríem seria:

1. Substituir la primera lletra del missatge per el número, en aquest cas la L equival al número 12.
2. Realitzar la operació abans indicada: $T = 12 + 6 = 18$.
3. Substituïm el número resultant per el seu lloc en l'alfabet, en aquest cas el 18 equival a la lletra R.
4. Realitzem el mateix per a cada lletra del missatge.

M	T
L	R
L	R
U	A
I	O
T	Z
A	G

Per tant el “lluïta” xifrat en clau 6 quedaria “RRAOZG”

1.1.3. El mètode de Polibi

Aquest mètode **inventat per l'historiador Polibi**, va acabar sent adoptat molt sovint com a mètode criptogràfic. El que va fer Polibi va ser posar l'alfabet en una **taula de 5x5**, el sistema de xifratge consistia a fer correspondre a cada lletra de l'alfabet un parell de lletres que indicaven la fila i la columna.

	A	B	C	D	E
A	A	B	C	D	E
B	F	G	H	I,J	K
C	L	M	N	O	P
D	Q	R	S	T	U
E	V	W	X	Y	Z

La I i la J s'han de posar juntes per tal de que quedi just per fer la quadrícula de 5x5.

Així per exemple la frase “**no volem lluitar**” quedaria de la següent manera:

CCCD EACDCAAECB CACADEBDDDAADB

Actualment també hi ha una altre variable més moderna que és substituint les lletres pels números, es a dir:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

I d'aquesta manera, la mateixa frase anterior quedaria:

3334 5134311532 31314524441142

Aquests mètodes de xifratge il·lustren dos de les bases en què es fonamenta la criptografia: **la substitució i la transportació.**

1.2. Criptografia medieval

Durant moltes dècades, la criptografia es va basar en els mètodes de **substitució** i de **transportació**. La substitució va ser el procediment dominant al llarg del primer mil·lenni, per aquella època molta gent considerava indesxifrable aquest mètode.

Això va ser així fins que a la ciutat de Bagdad, el savi, filòsof i estudiós de les ciències Abū Yūsuf Ya'qūb ibn Ishāq al-Kindī, més conegut com a **Al-Kindi**, al segle IX i gràcies a l'estudi del Corà, va veure que l'àrab tenia una característica freqüència de lletres. Al-Kindi va escriure el llibre *"Manuscrito sobre el desciframiento de mensajes criptográficos"* on ell mateix en dos paràgrafs explica com "trencar" mètodes de xifratge per substitució:

"Una manera de resolver un mensaje cifrado, si sabemos en qué lengua está escrito, es encontrar un texto claro diferente escrito en la misma lengua y que sea lo suficientemente largo para llenar alrededor de una hoja, y luego contar cuántas veces aparece cada letra. A la letra que aparece con más frecuencia la llamamos "primera", a la siguiente en frecuencia la llamamos "segunda", a la siguiente "tercera", y así sucesivamente, hasta que hayamos cubierto todas las letras que aparecen en la muestra de texto claro.

*Luego observamos el texto cifrado que queremos resolver y clasificamos sus símbolos de la misma manera. Encontramos que el símbolo que aparece con más frecuencia y lo sustituimos con la forma de la letra "primera" de la muestra de texto claro, el siguiente símbolo más corriente lo sustituimos por la forma de la letra "segunda", y el siguiente en frecuencia lo cambiamos por la forma de la letra "tercera", y así sucesivamente, hasta que hayamos cubierto todos los símbolos del criptograma que queremos resolver."*²

El que Al-Kindi descriu en aquest fragment de text s'anomena **anàlisi de freqüències**, i per a poder aplicar-lo, es necessita saber quin és el percentatge d'aparició de cada lletra en els texts sense xifrar. Naturalment, aquest percentatge depèn de l'idioma en què creiem que està escrit el text. L'aplicació directa de l'anàlisi gairebé mai desxifra un missatge del tot, però sí que sol succeir que les lletres que més es repeteixen en un text xifrat solen coincidir amb les més utilitzades normalment en la llengua. Per exemple, aquestes són les taules de freqüències de les lletres que s'utilitzen més en la llengua catalana:

² AL-KINDI. *Manuscrito sobre el desciframiento de mensajes criptográficos*. s.IX

Lletres de freqüència alta		Lletres de freqüència mitjana		Lletres de freqüència baixa		Lletres de freqüència molt baixa	
Lletra	Freq. %	Lletra	Freq. %	Lletra	Freq. %	Lletra	Freq. %
E	16.01	R	5.99	M	2.79	F	0.90
A	14.47	L	5.94	P	2.39	H	0.65
I	8.08	N	5.84	V	1.25	X	0.45
S	7.43	T	5.44	Q	1.20	Y	0.35
O	6.58	U	4.84	B	1.15	J	0.25
		D	3.47	G	1.15	Ç	0.10
		C	3.19			Z	0.05
						K	0.02
						W	0.01

D'aquesta taula se'n poden extreure les següents conclusions:

- Les vocals ocuparan aproximadament un 50% del text.
- Només la E i la A es poden identificar amb relativa fiabilitat, ja que destaquen molt més sobre les altres, de fet entre les dues vocals ocuparan un 30% del text.
- Les consonants més freqüents són la S, la R i la L ocupant aproximadament un 20% del text.
- Les consonants menys freqüents són la Z, K i W ocupant només un 0.08% del text.

1.3. Criptografia Europea

1.3.1. Els xifratges polialfabètics

Va ser **Leone Battista Alberti**, músic, pintor, escriptor i arquitecte, el que va concebre per primera vegada el sistema de **substitució polialfabètica** que utilitza diversos abecedaris, saltant d'un a l'altre cada tres o quatre paraules. L'emissor i el destinatari s'han de posar d'acord per a fixar la posició relativa de dos cercles concèntrics, que determinarà la correspondència dels signes.

Basant-se en aquesta tècnica es van començar a utilitzar els discos d'Alberti.

1.3.1.1. Xifratge d'Alberti

El disc d'Alberti està format per dos discos, un d'exterior on hi ha representats 20 caràcters del llatí, els mateixos que l'alfabet en català excepte les lletres H, J, K, U, W, Y, i s'hi inclouen els números 1, 2, 3, 4 podent fer així combinacions des de l'1 fins al 4444. Al disc interior hi apareixen tots els caràcters del llatí a més del signe & i les lletres H, K, Y. En ser 24 els caràcters representats en cada disc, és possible definir fins a 24 substitucions diferents, és a dir, depenent de la posició del disc interior la quantitat màxima d'alfabets de xifratge és igual a 24. Per a xifrar un missatge es repassa lletra a lletra el text en clar del disc exterior i es substitueix cada una d'elles per la lletra corresponent al disc interior. Fixada una lletra com a índex (lletra que es fa servir com a referència del disc interior i que l'emissor i el receptor ja han d'haver pactat quina és), s'ha d'ajustar el disc intern i escriure com a primera lletra del criptograma (missatge xifrat), una lletra majúscula que serà la que coincidirà amb l'índex, una vegada fet tot això es xifren algunes lletres i si es vol canviar l'alfabet, el que s'ha de fer és escriure una nova lletra majúscula que aquesta serà la que coincidirà amb l'índex.

Per entendre-ho millor, anem a veure un exemple:

Suposem que volem xifrar el missatge **“Ataca amb la posta de sol”**

En aquest cas en el disc interior farem servir el següent alfabet, encara que es pot fer el que cadascú vulgui i després canviaran les lletres, però aquesta és la raó per la qual aquest xifratge és més segur que els explicats fins ara, l'alfabet és el següent: “egkl nprtvz&xysomqihfdbac”. Per tant els discos quedarien així:

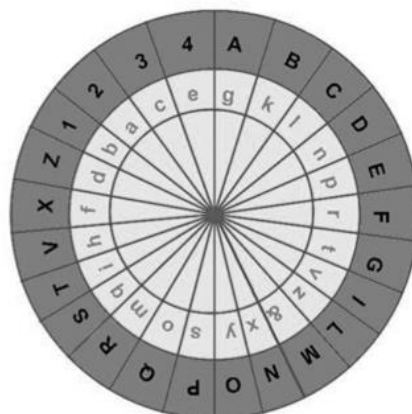


Figura 3 - Disc d'Alberti

Font: <http://quhist.com/disco-alberti-criptografia/>

Com a índex farem servir la m , per tant hem de fer coincidir la lletra m del disc interior amb la A de l'exterior, ja que és la primera lletra del missatge per després veure amb quines lletres coincideixen les altres, una vegada col·locats els discos comencem a mirar lletra per lletra i substituïm l'exterior per la interior, però imaginem que per fer-ho més segur i així no funcioni l'anàlisi de freqüències volem que "posta de sol" utilitzi un altre alfabet, docs quan arribéssim a la p hauríem de fer coincidir la m amb la P i escriure la P en majúscula. Finalment el missatge xifrat quedaria com: **"Atmim meq cm Pohfl rt hox"**

1.3.1.2. Xifratge de Vigenère o taula de Vigenère

Després del xifratge d'Alberti, al 1508 Johannes Trithemius va inventar la tabula recta, que és amb el que es basa la taula de Vigenère. Més tard l'any 1533 Giovan Battista Belasso va descriure el mètode original al llibre que va publicar anomenat *"La cifra del Sig. Giovan Battista Belasso"*, no obstant va ser incorrectament atribuït a Blaise Vigenère, ja que ell va publicar un xifratge semblant però més robust abans del 1586, fins al segle XIX encara que s'ha quedat amb el seu nom.

La taula de Vigenère es basa en una matriu quadrada de 26x26 on apareixen 26 alfabetes, on cada un està desplaçat una lletra més que l'anterior. Per a xifrar un missatge el que s'ha de fer primer de tot és pactar una paraula clau amb el destinatari, ja que sense aquesta no es podria desxifrar el missatge. Un cop fet això s'ha d'escriure la clau a sota del missatge de tal manera que a cada lletra del missatge li estigui assignada una lletra de la clau, es pot repetir tantes vegades com faci falta la clau per tal de completar el missatge. Després la lletra del missatge s'ha de buscar en l'alfabet que comenci per la lletra de la paraula clau que li toqui, i substituir la lletra del missatge per la que li correspon.

Per entendre-ho millor anem a veure un exemple xifrant el missatge **"Els soldats estan preparats"** utilitzant la paraula **"castell"** com a clau.

Com he dit abans primer assignem a cada lletra del missatge una de la clau:

E	L	S	S	O	L	D	A	T	S	E	S	T	A	N	P	R	E	P	A	R	A	T	S
C	A	S	T	E	L	L	C	A	S	T	E	L	L	C	A	S	T	E	L	L	C	A	S

Un cop fet això busquem a la taula quina lletra hi ha en la posició de la lletra E a l'alfabet C, li correspon la lletra G, amb la L busquem a l'alfabet A, per tant és la L, la S es substitueix per la K, i així amb totes les lletres fins a completar el missatge, en aquest cas el missatge xifrat quedaria com: **"GLK LSWOCTK XWELP PJXTLCCTK"**

Taula de Vigenère

	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
A	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
B	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	A
C	C	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
D	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
E	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	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	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

1.4. La criptografia fins la Segona Guerra Mundial

Durant aquests anys la criptografia va seguir avançant, encara que no tan ràpid com fins ara.

Un dels descobriments que es va fer servir durant molt de temps i va ser bastant important va ser el codi Morse. El codi Morse va ser desenvolupat per Alfred Vail l'any 1830 mentre col·laborava amb Samuel Morse en la invenció del telègraf. El que va fer Vail va ser assignar a cada lletra una successió de ratlles i punts els quals es diferenciaven per la duració del senyal.

A	· —	U	· · —
B	— · · ·	V	· · · —
C	— · — ·	W	— · —
D	— · ·	X	· — · —
E	·	Y	· · — —
F	· · — ·	Z	— — · ·
G	— · —		
H	· · · ·		
I	· ·		
J	· — — —		
K	— · —	1	— — — —
L	· — · ·	2	· — — —
M	— —	3	· · — —
N	— ·	4	· · · —
O	— — —	5	· · · ·
P	· — — ·	6	— · · · ·
Q	— — · —	7	— — · · ·
R	· — ·	8	— — — · ·
S	· · ·	9	— — — — ·
T	—	0	— — — — —

Una altra aportació a la criptografia la va fer el lingüista i criptògraf Auguste Kerckhoffs el 1883 treballant de professor d'alemany a París va publicar l'article “*La cryptographie militaire*” a la revista “*Journal des Sciences*

Militaires” on entre altres coses hi ha els principis de Kerckhoffs. En aquests principis es diuen una sèrie de normes, les quals Kerckhoffs recomana que tots els sistemes de xifratge segueixin, i efectivament gran part de la comunitat criptogràfica ha adoptat aquestes normes. Els principis diuen el següent:

- 1- Si el sistema no és teòricament irrompible, almenys ho ha de ser a la practica.
- 2- El sistema no ha de ser secret i no ha de ser un problema que caigui en mans enemigues.
- 3- La clau del sistema ha de ser fàcil de memoritzar i comunicar als altres sense necessitat d'haver d'escriure-la, aquesta serà modificable pels interlocutors vàlids.
- 4- El sistema ha de donar resultats alfanumèrics.
- 5- El sistema no ha de requerir la intervenció de diverses persones.
- 6- El sistema ha de ser fàcil d'utilitzar, no requerirà coneixements especials ni tindrà una llarga sèrie de regles.

Tots aquests principis encara són aplicables a la criptografia actual 183 anys després.

Figura 4 - Equivalència de cada lletra i número al codi Morse

Font:

https://es.wikipedia.org/wiki/C%C3%B3digo_morse

1.5. La màquina Enigma a la Segona Guerra Mundial.

La màquina Enigma va ser inventada per l'enginyer alemany Arthur Scherbius, un expert en electromecànica que acabada la Primera Guerra Mundial va voler aplicar la tecnologia existent per a millorar els sistemes de criptografia dels exèrcits. Com que Scherbius no tenia els recursos suficients per a fabricar-la, es va associar amb Willie Korn que tenia una companyia anomenada Enigma Chiffiermaschinen AG a Berlín. Entre els dos van anar millorant el disseny fins al 1923 presentar-la a l'Exhibició Postal Internacional de Berlín per al xifratge de secrets comercials.

La màquina Enigma era un dispositiu electromecànic, és a dir, tenia una part elèctrica i una altra de mecànica. No era en realitat molt complicada, almenys per a l'operador que l'utilitzava, però d'explicar i entendre si que ho és una mica més, i ja ni en parlem d'inventar tot el conjunt de parts que la componien.

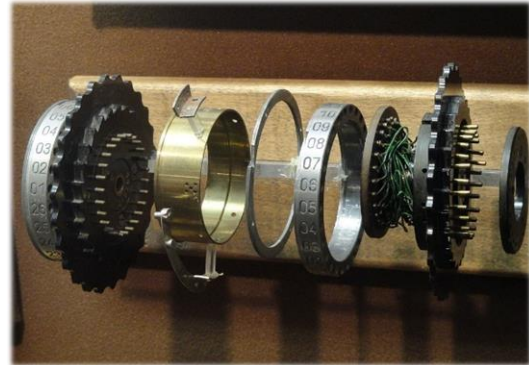


Figura 5 - Rotor de l'Enigma descompost en les diferents parts

Font: <http://www.wikiwand.com/de/Enigma-Walzen>

Bàsicament Enigma consistia en un teclat de màquines d'escriure amb les corresponents lletres de l'alfabet, un dispositiu anomenat modificador (també anomenat rotor), aquest era el que realitzava la codificació, i un tauler amb llums on s'il·luminaven les lletres del text xifrat. Aquest conjunt aconseguia que l'operador al prémer una tecla s'il·luminés la corresponent lletra xifrada, i aquesta l'operador se l'apuntava per posteriorment trametre-la per ràdio.

El rotor consistia en un cilindre amb 26 cables, connectat al teclat per 26 punts, un per a cada lletra i cada cable donava una sèrie de voltes pels rotors fins a il·luminar la lletra corresponent, sempre diferent de la que s'havia teclejat. Fins aquí sembla que Enigma era bastant dèbil i senzilla perquè només feia un xifratge de substitució monoalfabètic, essent fàcilment desxifrabable provant una per una les 26 possibles claus. Per evitar això van fer que cada vegada que es premés una tecla el rotor giraria un vint-i-sisè de volta, així s'aconseguia que en cada pulsació canviés la posició dels cables i amb això els punts d'entrada i sortida des del teclat fins al tauler amb les llums. Fent aquesta modificació ja es va convertir en un xifratge de substitució polialfabètic. Tot això vol dir que si tu polsaves dues vegades seguides la A et sortiria per exemple RT, en comptes de RR com hauria sortit si fos monoalfabètic.

Per si no n'hi hagués prou, s'hi van afegir 2 rotors més connectats entre si, de tal manera que quan el primer hagués donat la volta completa passant per les 26 posicions, el segon giraria una posició i el primer tornaria a passar per les 26 posicions i quan el segon hagués passat per les 26 posicions el tercer rotor avançaria una posició. Es poden entendre com les agulles dels segons, minuts i hores, cada una avança quan l'anterior ja ha fet una volta completa. Això donava un gran nombre de combinacions, en concret 17.576 ($26 \times 26 \times 26$). Però a més dels que s'hi posaven a la màquina, els operadors tenien una caixa amb 2 rotors més que feien un total de 5 rotors entre els quals triar, això ens dona 60 possibilitats per triar els rotors i col·locar-los de maneres diferents, ja que al principi tindràs 5 rotors entre els quals triar, després 4 i després 3, per tant, $5 \times 4 \times 3$.

Un altre element que s'hi va afegir va ser el reflector, que en realitat no gira, el que fa és rebre el senyal del tercer rotor i reenviar-la una altra vegada a través dels tres rotors però per un altre camí. Per al que servia també era a l'hora de desxifrar el missatge, el receptor ordenava els rotors igual que l'emissor i el reflector feia que si introduïes la lletra xifrada, s'il·luminés la lletra del missatge original.

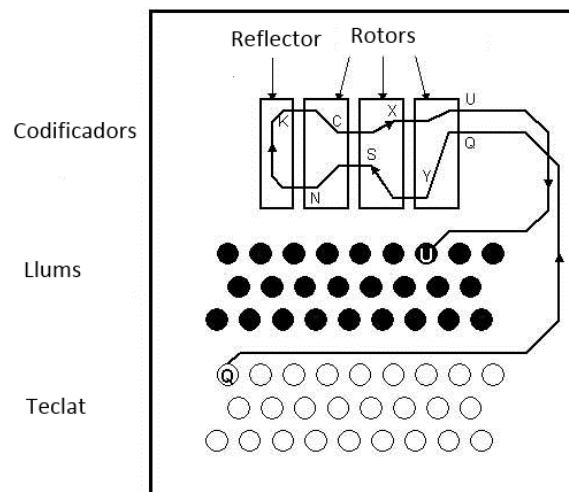


Figura 6 - Seqüència que seguiria el corrent elèctric al pulsar una tecla

Font: http://www.portierramaryaire.com/arts/enigma_2.php

Un element que van afegir més endavant a versions d'Enigma només disponibles per a l'exèrcit més modernes, va ser un tauler amb clavilles on es podien intercanviar parells de lletres abans d'entrar als rotors mitjançant cables. Això significa que si tu connectaves les lletres B i T i tu pulsaves la lletra B, en comptes de passar per tots els rotors com a B, primer passaria pel cable, es "convertiria" en la lletra T i es xifrarà com si haguessis pulsat la lletra T, i a la inversa, si pulsaves la T, es codificaria com a B. Al principi s'utilitzaven 6 cables, encara que més endavant es va augmentar el nombre fins a 10.

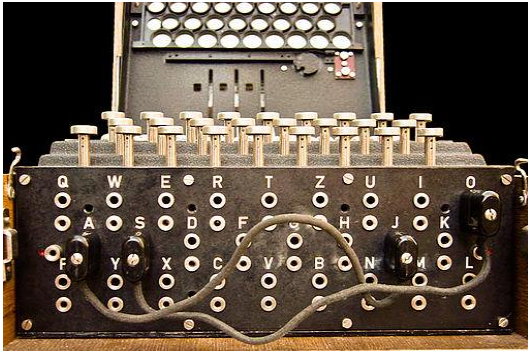


Figura 7 - Tauler amb les clavilles i cables de l'Enigma

Font: <http://tecno-info->

sr.blogspot.com.es/2015/09/maquina-enigma.html

El nombre de combinacions que es poden fer amb aquests cables és bastant gran, per començar hi ha 26 lletres i es poden combinar entre elles, això és el mateix que $26!$ ($26 \times 25 \times 24 \times 23 \times \dots \times 3 \times 2 \times 1$) però com que pots fer 10 parelles, això vol dir que en queden 6 que no

es poden fer servir, per tant es divideix per $6!$, al tenir 10 parelles no ens importa l'orde en què es facin, per tant podem dividir per $10!$, i com que

com hem vist anteriorment dues lletres es poden fer d'una part cap a l'altre i a l'inrevés, això segueix essent la mateixa parella, per tant dividim per 2 i com que són 10 parelles ho fem 10 vegades, és a dir, 2^{10} . Tot aquest paràgraf queda reduït a aquesta operació:

$$\frac{26!}{6! \cdot 10! \cdot 2^{10}}$$

El resultat d'aquesta operació és: 150.738.274.937.250.

Si multipliquem tots aquests resultats, tindrem el nombre de combinacions diferents totals que es podien fer amb la màquina Enigma:

$$60 \times 17.576 \times 150.738.274.937.250 = 158.962.555.217.826.360.000$$

Tot això està molt bé, encara que aquests 158 quintillons de combinacions no servien de res si el destinatari no podia llegir el missatge original. Per aconseguir això el que es feia era repartir mensualment una llista, en la qual a cada dia hi havia alguna cosa semblant a aquesta:

- 2-4-1
- 5-25-11
- A/F, Z/T, R/E, B/N, U/L, S/G, C/H, D/O, P/Q, I/V

Això vol dir que al dia que estava indicat amb aquestes instruccions, havien de posar el segon rotor al primer lloc i girar el rotor 5 vegades, el quart al segon i girar 25 vegades i el primer al tercer fent-lo girar 11 vegades. Els últims parells de lletres indicaven quines dues lletres s'havien de connectar amb les clavilles.

1.6. Criptografia moderna

L'etapa de la criptografia moderna comença realment amb Claude Shannon, considerat el pare de la criptografia matemàtica. L'any 1949 va publicar l'article *Communication Theory of Secrecy Systems*, i poc després el llibre *Mathematical Theory of Communication* conjuntament amb Warren Weaver. Aquests i altres treballs que va publicar sobre la teoria de la informació i la comunicació, van establir una sòlida base teòrica de la criptografia i la criptoanàlisi.

Actualment hi ha dos grans tipus de xifratges, la criptografia simètrica i la criptografia asimètrica.

1.6.1. Criptografia simètrica

La criptografia simètrica és el mètode pel qual es xifra i es desxifra un missatge amb la mateixa clau, és a dir, tot el que s'ha explicat fins aquest punt del treball, utilitza la criptografia simètrica, on l'emissor i el receptor s'han de posar d'acord amb una clau, i aquesta es farà servir per a xifrar i desxifrar el missatge.

Com tot, aquest tipus de criptografia té avantatges, com la facilitat a l'hora d'utilitzar-lo o la rapidesa i el fet de necessitar menys recursos informàtics que d'altres formes de xifratge per a poder-se realitzar, i també hi ha desavantatges, com la necessitat de comunicar la clau al receptor i la inseguretat que aquest pas comporta.

1.6.1.1. Data Encryption Standard (DES)

El DES és un algorisme de xifratge per blocs. Els seus orígens es remunten al 1972, quan un estudi sobre les necessitats del govern en l'àmbit de la seguretat informàtica, va concloure que es requeria un estàndard a escala governamental per xifrar la informació confidencial. El seu funcionament és el següent: divideix el text en blocs de 64 bits i aquests els posa en una matriu de 8x8, a aquesta matriu se li aplica la següent permutació inicial (PI) que ja ve definida a l'algorisme.

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

Això vol dir que el bit que estigui a la posició 58 es posarà a la posició 1 i el de la 50 a la 2 i així amb la resta del bloc de 64 bits. Una vegada fet això es divideix aquest bloc en dos de 32 bits cadascun, anomenats *Left* i *Right* (esquerra i dreta), és a dir que quedaria de la següent manera:

E_0							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8

D_0							
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

A partir d'aquest punt es tracten els dos blocs per separat fins al final i fan les següents operacions 16 vegades idènticament, aquest procés s'anomenen rondes. A cada ronda D_n passa a ser E_{n+1} , i a D_n se li aplica la funció de Feistel³ (la qual aplica un seguit de permutacions i substitucions) juntament amb la clau K_n ⁴, seguidament passa per la porta lògica XOR juntament amb L_n per acabar sent D_{n+1} . Una vegada fetes aquestes 16 rondes es tornen a ajuntar els dos blocs E i D i s'aplica la permutació inversa a PI, anomenada permutació final (PF o PI^{-1}). Per a xifrar el missatge s'apliquen les claus d'1 a 16 i per a desxifrar s'apliquen les claus en l'ordre invers, de 16 a 1.

A la següent pàgina hi ha un diagrama de flux de l'orde que segueix l'algorisme on F és la funció Feistel i \oplus és la porta lògica XOR.

³ Funció Feistel explicada a l'annex 1.

⁴ Obtenció de la clau K_n explicada a l'annex 2.

1.6.1.2. Triple Data Encryption Standard

El Triple DES, TDES o 3DES va ser desenvolupat per IBM l'any 1998 per a reforçar la seguretat de l'algorisme DES. Principalment el que fa 3DES és aplicar tres vegades seguides el DES utilitzant tres claus diferents per a encriptar i desenscriptar en el següent ordre:

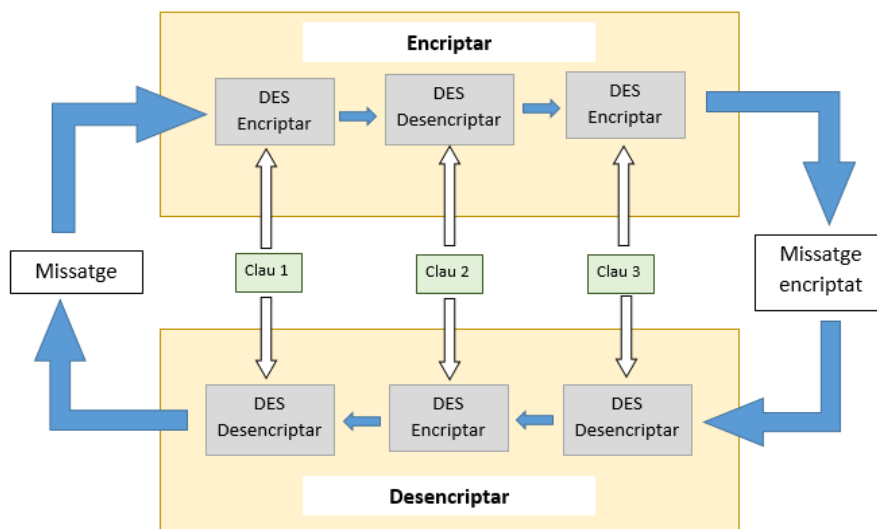
- 1- Es xifra el missatge original amb la primera clau.
- 2- Es desxifra el resultat del primer pas amb la segona clau.
- 3- Es xifra el resultat del segon pas amb la tercera clau.

Per a desxifrar el missatge una vegada arriba al destinatari el que s'ha de fer és l'invers al que s'ha fet per a xifrar-lo, és a dir:

- 1- Es desxifra el missatge rebut amb la tercera clau.
- 2- Es xifra el resultat del primer pas amb la segona clau.
- 3- Es desxifra el resultat del segon pas amb la primera clau.

Diagrama 2 - Diagrama de flux de 3DES

Font: Creació pròpia



1.6.1.3. *Advanced Encryption Standard (AES)*

AES, a l'igual que DES és un algorisme de xifratge per blocs, també és conegut com a Rijndael. Va ser desenvolupat pels dos criptògrafs belgues Joan Daemen i Vincent Rijmen quan el govern dels Estats Units, l'any 1997, va decidir realitzar un concurs per escollir un nou algorisme de xifratge capaç de protegir la informació durant el segle XXI, es va transformar en un estàndard efectiu el 26 de maig de 2002 i des del 2006, és un dels algorismes més populars utilitzats en criptografia simètrica.

A diferència del DES, l'AES pot tenir claus de 128, 192 i 256 bits, el que significa molta més seguretat. I en aquest cas el nombre de rondes depèn de la longitud de la clau:

Longitud de la clau	Nº de rondes
128	10
192	12
256	14

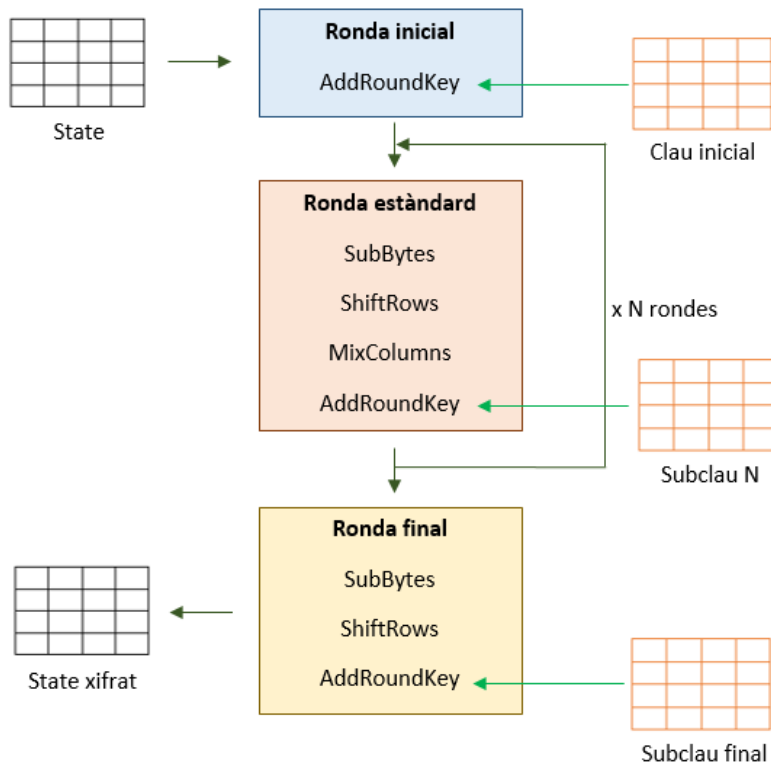
AES comença dividint l'entrada en matrius, anomenades states, de 16 bytes on cada byte equival a 8 bits. A la ronda inicial aplica la porta XOR amb l'state i la clau K. Després comencen les anomenades rondes estàndard, que el procés que es fa aquí és el que s'anirà repetint.

Per començar, a la ronda estàndard el que fa és passar per la porta XOR el resultat de la ronda inicial juntament amb una S-box, i seguidament fer un procés anomenat SubBytes, el qual serveix per trobar la subclau que s'aplicarà posteriorment. Després s'aplica ShiftRows, que desplaça les files un seguit determinat de vegades, a continuació es fa el MixColumns que cada columna li aplica una transformació lineal i finalment es fa AddRoundKey que aplica la porta XOR amb el resultat dels processos anteriors i la clau trobada al SubBytes. A la ronda final es fa el mateix que a les rondes estàndard però sense aplicar el MixColumns, ja que això no incrementaria la seguretat i només realitzaria el procés del xifrat.⁵

⁵ Explicació detallada de SubBytes, ShiftRows i MixColumns l'annex 3.

Diagrama 3 - Diagrama de flux de AES

Font: Creació pròpia



1.6.2. Criptografia asimètrica

La criptografia asimètrica pel contrari, utilitza dos claus que pertanyen al mateix usuari, una de *pública* que pot saber tothom i una de *privada* que l'usuari s'ha de guardar en secret per a ell. Per explicar com funciona aquest mètode, utilitzaré un exemple en el qual dues persones, en Marc i la Laia, es volen enviar un missatge per Internet. Cadascú d'ells té un parell de claus (la pública i la privada), si en Marc vol enviar un missatge a la Laia el que hauria de fer seria xifrar el seu missatge amb la clau pública de la Laia, enviar aquest missatge xifrat a la Laia, i després la Laia mitjançant la seva clau privada, podria desxifrar el missatge (ja que aquestes claus estan associades) i viceversa, si la Laia en volgués enviar un a en Marc.

Aquest mètode també es pot utilitzar a la inversa, és a dir, si la Laia es vol assegurar que qui ha enviat el missatge de veritat és en Marc (i no una altra persona, per una suplantació de la identitat) el que hauria de fer en Marc és xifrar el missatge amb la seva clau privada i així qualsevol persona amb la seva clau pública podria desxifrar el missatge. En això, és amb el qual es basa la firma electrònica, ja que una clau privada només pot correspondre a un usuari.

Però està clar que ningú a l'hora d'enviar missatges fa tots aquests processos, ja que es fan tots automàticament gràcies als algorismes que s'han programat i creat específicament per a això.

Un dels grans avantatges, és que la distribució de la clau és més senzilla al poder donar tranquil·lament la pública, i a la vegada, més segur al poder quedar-te la privada només per a tu, un dels grans desavantatges és que es necessita més temps de procés i que les claus han de ser molt més grans que les que s'utilitzen a la simètrica.

Els dos principals desavantatges es poden resoldre amb un altre tipus de criptografia, la criptografia híbrida.

1.6.2.1. RSA (Rivest, Shamir i Adleman)

Aquest sistema és un sistema de criptografia asimètrica desenvolupat per Ron Rivest, Adi Shamir i Leonard Adleman el 1977. La seguretat d'aquest algorisme recau en la capacitat de càlcul dels ordinadors, més concretament en el fet que actualment no es poden factoritzar dos nombres enters molt grans d'una forma relativament ràpida. Es creu que aquest sistema serà segur fins que no es coneguin formes ràpides de factoritzar els nombres.

El que fa aquest algorisme és utilitzar la potenciació modular per a xifrar i desxifrar els missatges. Comença convertint tot el text a números i després per xifrar fa la següent operació:

$$c = m^e \pmod{n}$$

On c és el missatge xifrat,
 m és el missatge original en forma numèrica,
 e és la clau pública del destinatari

i la següent operació per a desxifrar el missatge xifrat:

$$m = c^d \pmod{n}$$

on m és el missatge xifrat en forma numèrica,
 c és el missatge xifrat,
 d és la clau privada del destinatari

En ambdós casos n és la multiplicació dels dos nombres primers escollits aleatòriament.⁶

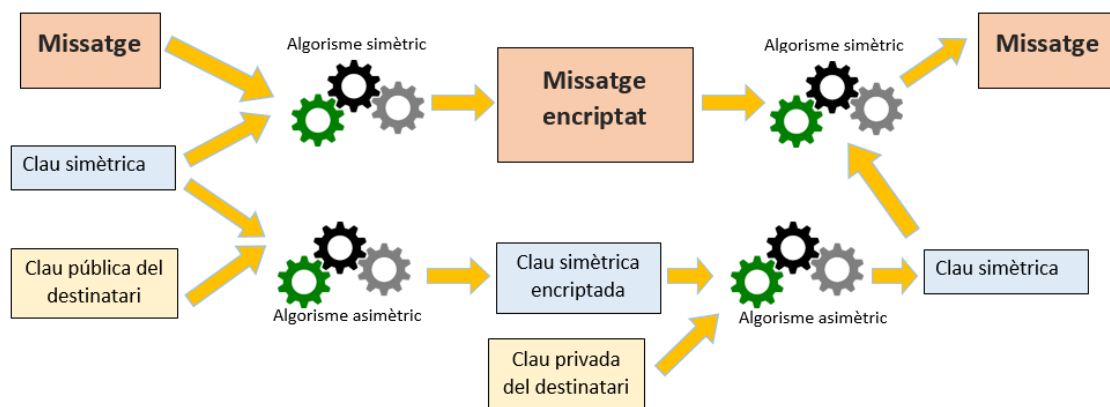
⁶ Explicació de l'obtenció de les claus i n a l'annex 4.

1.6.3. Criptografia híbrida

Aquest tipus és la combinació dels dos mètodes anteriors, i així es resol el problema de la inseguretat de la simètrica i de la lentitud en el cas de l'asimètrica. La criptografia híbrida funciona de la següent manera, utilitzant la clau simètrica (la que els dos teniu igual), xifres el missatge, i després amb la clau pública del destinatari, xifres la clau simètrica, envies les dues coses, i una vegada li arriba al destinatari, aquest utilitza la seva clau privada per a aconseguir la clau simètrica i amb aquesta es desxifra el missatge original.

Diagrama 4 - Explicació visual de la criptografia híbrida

Font: Creació pròpia



Part 2. Procés de producció – Xifratge Cèsar en C++

Com he comentat anteriorment a la introducció com a part pràctica he fet un programa el qual xifra els missatges que l'usuari introdueix. El sistema per a xifrar que he fet servir ha estat el Cèsar. Aquest programa l'he desenvolupat amb el llenguatge de programació C++, encara que inicialment el vaig fer amb Python perquè el dominava una mica més i així seria més senzill, però al ser Python un llenguatge interpretat doncs no es pot executar a un ordinador a no ser que aquest tingui instal·lat l'interpret d'aquest llenguatge, i com que al sistema operatiu Windows no hi ve preinstal·lat, no era possible que el programa s'executés a tots els ordinadors sense problema. Per solucionar això vaig haver d'aprendre a programar en C++ des del principi.

```
1 #include <iostream>
2 #include <string.h>
3 #include <conio.h>
4
5 using namespace std;
```

A les tres primeres línies s'invoca a les llibreries "iostream", "string.h" i "conio.h" de la biblioteca estàndard (STL) de C++.

- Iostream és l'abreviatura d'Input/Output Stream i conté les indicacions per a introduir dades al programa i mostrar missatges a la pantalla.
- String.h conté les indicacions per a fer operacions amb cadenes de text.
- Conio.h conté funcions i constants per a manipular la consola MS-DOS.

A la cinquena línia per evitar que alguna variable creada per mi i les que estan a les llibreries tinguin el mateix nom i es confonguin fem servir "l'espai de noms" estàndard (std).

```

7 //Funció que codifica el missatge
8 void codifica(int n, string & cadena) {
9     for (int j = 0; j < cadena.length(); j++) {
10         if (cadena[j] >= 'A' && cadena[j] <= 'Z') {
11             if (cadena[j] + n > 'Z') {
12                 cadena[j] = 'A' - 'Z' + cadena[j] + (n - 1);
13             } else if (cadena[j] + n < 'A') {
14                 cadena[j] = 'Z' - 'A' + cadena[j] + (n + 1);
15             } else {
16                 cadena[j] += n;
17             }
18         } else if (cadena[j] >= 'a' && cadena[j] <= 'z') {
19             if (cadena[j] + n > 'z') {
20                 cadena[j] = 'a' - 'z' + cadena[j] + (n - 1);
21             } else if (cadena[j] + n < 'a') {
22                 cadena[j] = 'z' - 'a' + cadena[j] + (n + 1);
23             } else {
24                 cadena[j] += n;
25             }
26         }
27     }
28 }

```

De les línies 8 a la 28 hi ha la funció que el codifica el missatge. Amb *void* es defineix la funció i entre parèntesis hi ha els paràmetres que s'utilitza. A la línia 9 comença un bucle el qual et va recorrent tot el missatge que s'hagi introduït i et va comprovant les condicions següents. De les línies 10 a la 17 i de la 18 a la 25 es fa el mateix procés, el canvi és que un es farà per a les lletres majúscules i l'altre per a les lletres minúscules, per tant a les línies 10 i 18 et comprova si la lletra és minúscula o majúscula, a continuació si la lletra més el número de desplaçament és més gran que la lletra 'z', es restarà la lletra 'a' menys la lletra 'z', al resultat se li sumarà la lletra del missatge original més el nombre de desplaçament menys 1, és a dir, $c = 'a' - 'z' + m + (n - 1)$, on 'c' és la lletra xifrada, 'm' és la lletra del missatge original i 'n' és el número de desplaçament.

Si la condició anterior no es compleix però es compleix que la lletra més el número de desplaçament és més petit que la lletra 'a', es restarà la lletra 'z' menys la lletra 'a' i al resultat se li sumarà la lletra del missatge original més el número de desplaçament més 1, és a dir, $c = 'z' - 'a' + (n + 1)$.

Si cap de les dues condicions es compleix simplement se sumaran la lletra del missatge original més el número de desplaçament.

```
31 //Funció que descodifica el missatge
32 void descodifica(int n, string & cadena) {
33     codifica(-n, cadena);
34 }
```

Aquesta és la funció que descodifica el missatge, el que es fa en aquesta funció és invocar a la funció de codificar però amb el signe del número de desplaçament canviat de signe, per tant en comptes de sumar-se a la lletra del missatge original se li restarà.

```
37 //Programa principal
38 int main() {
39     string cadena;
40     string orden;
41     int n;
42
43     cout << "Introduce el mensaje a codificar/descodificar: ";
44     getline(cin, cadena);
45
46     while (1<100){
47         cout << "\n" << "Introduce el numero de desplazamiento deseado: ";
48         cin >> n;
49         if (1 < n && n < 26){
50             break;
51         } else {
52             cout << "\n" << "Introduce un numero menor a 26";
53         }
54     }
```

A la línia 38 s'inicia la funció que a la llibreria estàndard ve definida com a principal, la *main*. A les línies 39, 40 i 41 creo les variables que necessitaré més endavant. A les que hi posa *string* (cadena en anglès), és perquè en aquelles s'hi guardarà només informació en forma de text i a la que hi posa *int* (enter de l'anglès integer), s'hi guardarà només informació numèrica.

A la línia 43 imprimeixo en pantalla el missatge que hi ha entre cometes mitjançant l'ordre *cout*, definida a la llibreria *iostream* importada a l'inici, i a la 44 guardo el que hagi introduït l'usuari a la variable 'cadena', mitjançant l'orde *getline*, també definida a la llibreria *iostream*.

A la línia 46 s'inicia un bucle infinit, ja que 1 sempre serà més petit que 100, que anirà imprimint en pantalla "Introduce el numero de desplazamiento deseado: " fins que l'usuari hagi escrit un nombre més petit de 26. A la 49 hi ha la condició que fa que el bucle s'aturi si es compleix la condició esmentada anteriorment i si no es compleix informa l'usuari del fet que ha d'escriure un nombre menor a 26. "\n" és un salt de línia per tal que no quedi tot el text junt.

```
55
56
57 while (1<100){
58     cout << "\n" << "Desea codificar o descodificar el mensaje?" << endl;
59     cin >> orden;
60     if(orden == "codificar" || orden == "Codificar" || orden == "CODIFICAR" ||
61        orden == "descodificar" || orden == "Descodificar" || orden == "DESCODIFICAR"){
62         if (orden == "codificar" || orden == "Codificar" || orden == "CODIFICAR"){
63             codifica(n, cadena);
64             cout << "\n " << cadena << "\n" << endl;
65             cout << "Pulse cualquier tecla para salir... ";
66         } else if (orden == "descodificar" || orden == "Descodificar" || orden == "DESCODIFICAR"){
67             descodifica(n, cadena);
68             cout << "\n " << cadena << "\n" << endl;
69             cout << "Pulse cualquier tecla para salir... ";
70         }
71         break;
72     }
73 }
74
75 char cad;
76 cad = (char)getch();
77 }
```

A la línia 57 comença un altre bucle infinit que anirà imprimint en pantalla “Desea codificar o descodificar el mensaje?” fins que se li introdueixi ‘Codificar’ o ‘Descodificar’. A la 59 es desa el que hagi introduït l’usuari a la variable ‘orden’.

A les línies 60 i 61 hi tenim la condició acabada d’esmentar, a la 62 hi ha la condició per a codificar, si el valor de la variable ‘orden’ és “codificar” s’invocarà la funció ‘codifica’ amb els paràmetres que hi hagi a les variables ‘n’ i ‘cadena’, i després et mostrarà el resultat de la funció ‘cadena’ seguit del missatge “Pulse cualquier tecla para salir...”.

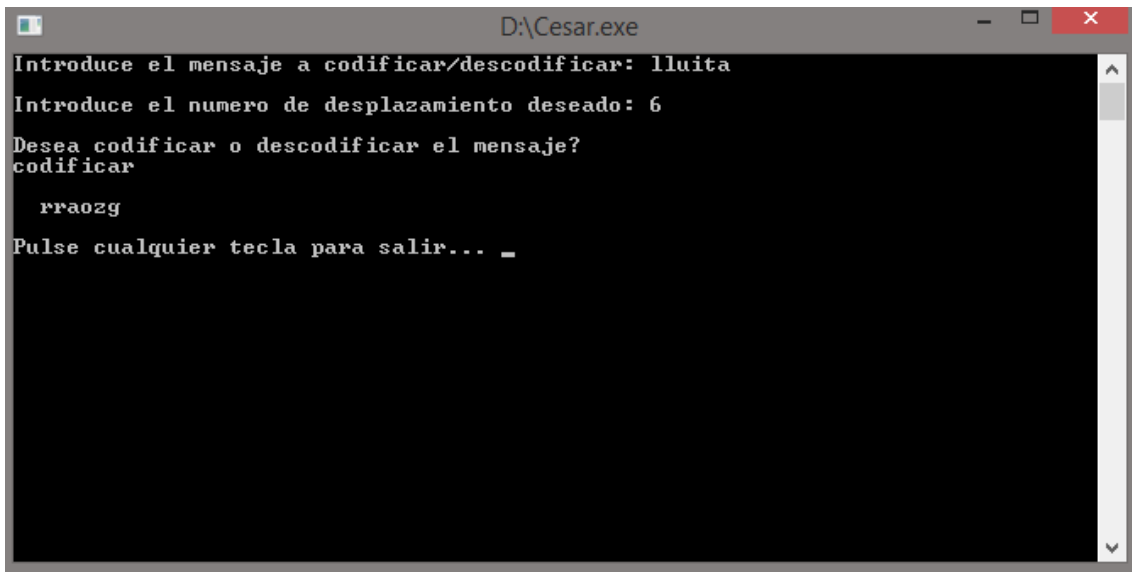
De la 66 a la 70 es fa el mateix que a les línies 62 a 65, però aquesta vegada per a desxifrar el missatge amb la funció ‘descodifica’.

A la línia 71 la paraula *break* (trençar en anglès), és una altra paraula reservada a *iostream* que serveix per a trençar el bucle.

Finalment a les últimes tres línies es crea una variable *char* (caràcter de l’anglès character), la qual guarda només caràcters de forma individual, a la línia 76 el que es fa en “mantenir activa” la variable, com si estigués esperant a alguna cosa, es fa servir mitjançant l’ordre *getch()* que està definida a la llibreria *conio.h* també importada a l’inici. L’última clau és la que tanca la funció *main*.

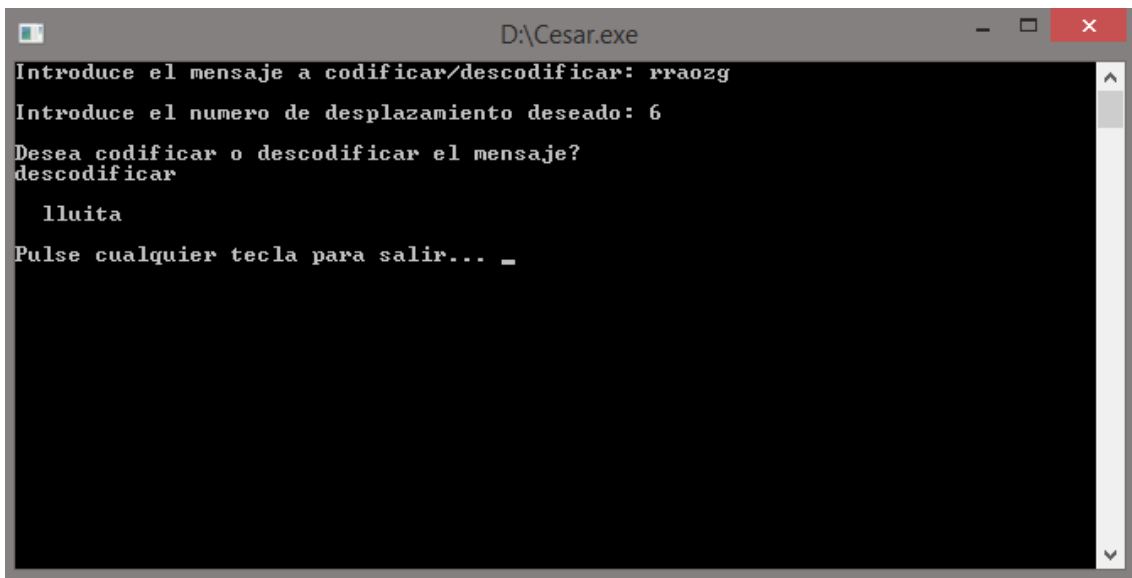
A continuació alguns exemples del programa en funcionament:

Prova 1 – Xifrant i desxifrant la paraula feta servir a l'inici del treball



A screenshot of a Windows command prompt window titled "D:\Cesar.exe". The text inside the window is as follows:

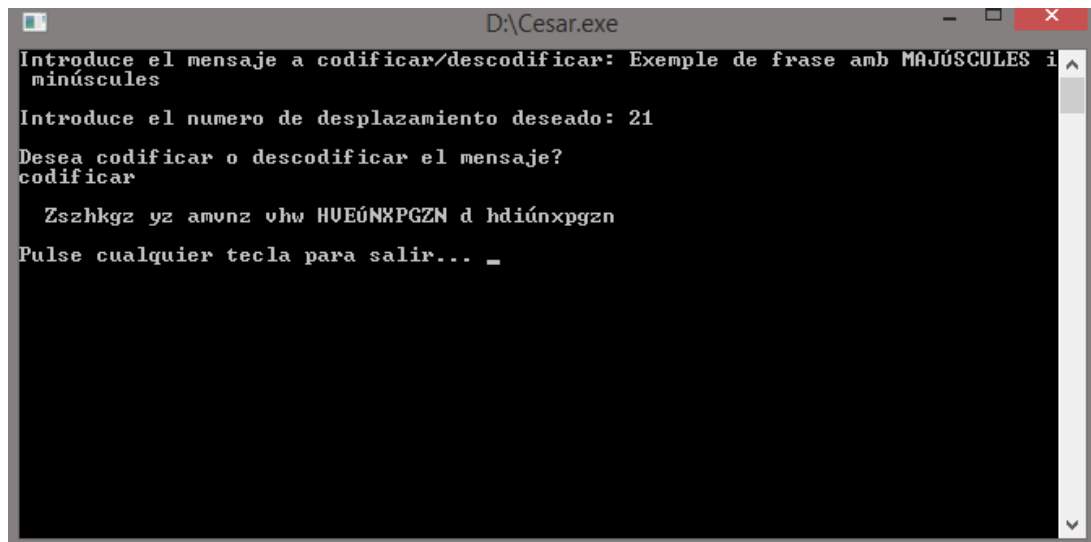
```
Introduce el mensaje a codificar/descodificar: lluita
Introduce el numero de desplazamiento deseado: 6
Desea codificar o descodificar el mensaje?
codificar
    rraozg
Pulse cualquier tecla para salir... _
```



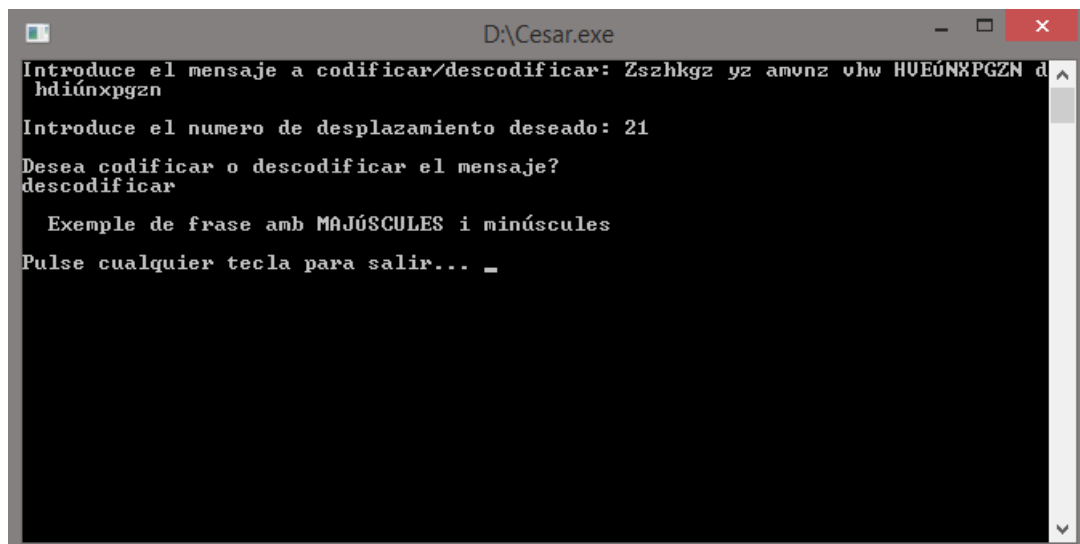
A screenshot of a Windows command prompt window titled "D:\Cesar.exe". The text inside the window is as follows:

```
Introduce el mensaje a codificar/descodificar: rraozg
Introduce el numero de desplazamiento deseado: 6
Desea codificar o descodificar el mensaje?
descodificar
    lluita
Pulse cualquier tecla para salir... _
```


Prova 2 - Provant amb una frase que conté majúscules i minúscules



```
D:\Cesar.exe
Introduce el mensaje a codificar/descodificar: Exemple de frase amb MAJÚSCULES i minúscules
Introduce el numero de desplazamiento deseado: 21
Desea codificar o descodificar el mensaje?
codificar
    Zszhkgz yz amvnz vhw HVEÚNXPGZN d hdiúnxpgzn
Pulse cualquier tecla para salir... _
```



```
D:\Cesar.exe
Introduce el mensaje a codificar/descodificar: Zszhkgz yz amvnz vhw HVEÚNXPGZN d hdiúnxpgzn
Introduce el numero de desplazamiento deseado: 21
Desea codificar o descodificar el mensaje?
descodificar
    Exemple de frase amb MAJÚSCULES i minúscules
Pulse cualquier tecla para salir... _
```

Conclusions

Aquest treball de recerca sobre la criptografia i la seva història des dels inicis fins a l'actualitat, per mi, ha sigut una gran experiència. Ha suposat un gran esforç i dedicació constant, però he gaudit realitzant-lo.

Reprenent els objectius de l'inici del treball, crec que he assolit el primer, ja que he aconseguit aprendre el llenguatge C++, només mirant vídeos i llegint arxius d'internet. Ha sigut bastant complicat fer que tot funcionés, però amb bastant constància a l'estiu ho he aconseguit.

Referent a la seguretat de les dades, després d'haver llegit moltes pàgines i haver après bastant, puc dir que les nostres dades estan guardades de forma segura, si, però tard o d'hora hi haurà alguna manera d'aconseguir-les, perquè cap d'aquests mètodes serà infal·lible sempre. Per tant com més innovem en aquest camp més segures estaran les nostres dades.

Finalment dir que aquest treball, entre altres coses, ha reafirmat la idea que tenia sobre estudiar una enginyeria informàtica, ja que he vist que com més apreng sobre aquests temes més m'agraden.

Índex d'imatges

Figura 1 – Escriptura hieràtica. *Pàgina 5*. Extreta de ca.wikipedia.org/wiki/Hier%C3%A0tic

Figura 2 – Escitala espartà. *Pàgina 6*. Extreta de <http://timerime.com/en/event/2800925/La+Escitala+Espartana/>

Figura 3 – Disc d'Alberti. *Pàgina 12*. Extreta de <http://quhist.com/disco-alberti-criptografia/>

Figura 4 – Equivalència de cada lletra i número al codi Morse. *Pàgina 15*. Extreta de https://es.wikipedia.org/wiki/C%C3%B3digo_morse

Figura 5 – Rotor de l'Enigma descompost en les diferents parts. *Pàgina 16*. Extreta de <http://www.wikiwand.com/de/Enigma-Walzen>

Figura 6 – Seqüència que seguiria el corrent elèctric al polsar una tecla. *Pàgina 117*. Extreta de http://www.portierramaryaire.com/arts/enigma_2.php

Figura 7 – Tauler amb les clavilles i cables de l'Enigma. *Pàgina 18*. Extreta de <http://tecno-info-sr.blogspot.com.es/2015/09/maquina-enigma.html>

Diagrama 1 – Diagrama de flux de DES. *Pàgina 21*. Creació pròpia.

Diagrama 2 – Diagrama de flux de 3DES. *Pàgina 22*. Creació pròpia.

Diagrama 3 – Diagrama de flux de AES. *Pàgina 24*. Creació pròpia.

Diagrama 4 – Explicació visual de la criptografia híbrida. *Pàgina 26*. Creació pròpia.

Webgrafia

Juan Jesús Velasco. (20/05/2014). *Breve historia de la criptografia*. Visites varies el novembre del 2015, des de www.eldiario.es/turing/criptografia/Breve-historia-criptografia_0_261773822.html

Wikipedia. (02/01/2016). *Escítala*. Visitat el 7 de gener de 2016, des de es.wikipedia.org/wiki/Esc%C3%ADtala

Andoni Talavera Préstamo. (05/07/2010). *La necesidad de ocultar: desde la escitala a la criptografía cuántica*. Visitat el 7 de gener de 2016, des de www.teknoplof.com/2010/07/05/la-necesidad-de-ocultar-desde-la-escitala-a-la-criptografia-cuantica/

Wikipedia. (03/02/2016). *Cifrado César*. Visitat el 13 de febrer de 2016, des de es.wikipedia.org/wiki/Cifrado_C%C3%A9sar

Jose Luis Tabara Carbajo. (20/09/2015). *El analisis de frecuencias*. Visitat el 13 de febrer de 2016, des de <https://joseluistabaracarabajo.gitbooks.io/criptografia-clasica/content/Cripto07.html>

Tania Vargas. (19/04/2013). *Cifrado de Alberti*. Visitat el 12 de març de 2016, des de es.calameo.com/books/0023432273c459385ed6e

John Fernando Mandón Quintero. (16/05/2010). *Criptografia*. Visitat el 12 de març de 2016, des de www.slideshare.net/johfermq/criptografia-by-fernandao

Mig23. *Enigma*. Visitat el 6 de maig de 2016, des de www.portierramaryaire.com/arts/enigma_1.php

Wikipedia. (18/07/2016). *Historia de la criptografia*. Visitat el 13 d'agost de 2016, des de es.wikipedia.org/wiki/Historia_de_la_criptograf%C3%ADa

Universidad Politécnica de Pachuca. *Criptografía simétrica y asimétrica*. Visitat el 23 d'agost de 2016, des de www.maytics.web44.net/web_documents/criptograf_a_sim_trica_y_asim_trica.pdf

Eneko Amieva. (13/12/2015). *Criptografía: simétrica, asimétrica e híbrida*. Visitat el 23 d'agost de 2016, des de enekoamieva.com/criptografia-simetrica-asimetrica-e-hibrida/

Sergio De Luz. (04/10/2010). *Criptografía: Algoritmos de cifrado de clave simétrica*. Visitat el 23 d'agost de 2016, des de www.redeszone.net/2010/11/04/criptografia-algoritmos-de-cifrado-de-clave-simetrica/

CCM. (-/07/2016). *Introducción al cifrado mediante DES*. Visitat l'11 de setembre de 2016, des de es.ccm.net/contents/130-introduccion-al-cifrado-mediante-des

Elvis Vinda. (12/10/2013). *Sencilla explicación sobre AES*. Visitat el 17 de setembre de 2016, des de www.slideshare.net/elvisvinda/sencilla-explicacin-sobre-aes

Departamento de Matemática Aplicada de la Universidad Politécnica de Madrid. *Criptosistemas de clave pública. El cifrado RSA*. Visitat el 18 de setembre de 2016, des de www.dma.fi.upm.es/recursos/aplicaciones/matematica_discreta/web/aritmetica_modular/rsa.html

Wikipedia. (25/04/2016). *Función φ de Euler*. Visitat el 18 de setembre de 2016, des de es.wikipedia.org/wiki/Funci%C3%B3n_%CF%86_de_Euler

Annex 1

Funció Feistel

La funció Feistel comença aplicant una taula d'expansió que ja ve determinada a l'algorisme que el que fa és duplicar alguns dels bits per passar de 32 a 48 bits per així amb la matriu resultant aplicar una porta lògica XOR juntament amb la clau K_n . Obtindrem una altra matriu de 8x6 bits, anomenem-la B. B es divideix en 8 blocs de 6 bits cadascun.

B						
1	1	0	1	1	0	B[1]
0	0	1	0	1	0	B[2]
1	1	0	1	1	0	B[3]
0	1	0	1	0	0	B[4]
0	0	0	1	0	0	B[5]
1	0	0	1	1	0	B[6]
1	0	1	0	0	1	B[7]
0	1	0	0	1	1	B[8]

A cada $B[n]$ se li aplica una S-box, una “caixa de substitució” que és una matriu de 4x16 on hi ha números del 0 al 15 per poder així representar cada número amb 4 bits i la matriu final torni a quedar de 32 bits. El que es fa per saber quin número li toca a cada $B[n]$ és el següent:

- 1- Agafes el primer i l'últim bit de $B[n]$ i passes els dos conjuntament a decimal, el número que surti indicarà la fila. Agafes els 4 bits centrals i ho passes a decimal i el número que surti indicarà la columna.
- 2- Busques a la S-box quin número correspon a la fila i la columna que t'han sortit en el pas anterior.
- 3- El número que li correspongui el passes a binari i substitueixes el $B[n]$ original pel nou número passat a binari.
- 4- Es fa el mateix amb tots els $B[n]$ amb la seva S-box corresponent.

Per entendre-ho faré el procediment anterior amb B[1], però es fa el mateix amb cada B[n].

- 1- El primer i últim bit de B[1] són 1 i 0, és a dir, 10 que passat a decimal és el número 2 (fila) i els 4 bits centrals són 1011 que passat a decimal és l'11 (columna).
- 2- La S-box que es fa servir sempre per a B[1] és la següent:

S-box 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	15	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	3

Per tant en aquesta taula busquem el número que correspon a la fila 2, columna 11, veiem que és el 7.

- 3- El 7 el passem a binari que queda com 0111. Substituïm els bits originals que hi havia per aquests, es a dir que B[1] passa de ser 110110 a ser 0111.

Després d'aplicar totes les S-box a tots els B[n] queda el resultat següent:

B				
0	1	1	1	B[1]
1	0	1	1	B[2]
1	1	0	0	B[3]
1	0	0	0	B[4]
0	1	0	0	B[5]
0	1	0	1	B[6]
0	0	0	1	B[7]
0	1	0	1	B[8]

El resultat R de tot aquest procediment serà la concatenació de tots els B[n] en una matriu de 32 bits. Finalment a R se li aplica una altra permutació, també fixada a l'algorisme, i el resultat d'aquesta permutació serà el resultat de la funció Feistel.

Annex 2

Al ser DES un algorisme que fa servir el sistema simètric l'emissor i el receptor tenen la mateixa clau, en aquest cas suposem que és la següent:

K							
1	1	1	0	0	0	0	1
1	0	1	0	0	1	0	0
1	0	1	1	1	0	1	1
1	0	0	1	1	0	1	1
1	1	0	0	0	0	1	0
1	1	1	0	0	0	1	1
0	0	1	0	0	1	0	0
1	0	1	1	1	0	1	1

El que es fa primer és treure un bit cada set bits, és a dir que s'acaba traient l'última columna perquè aquests bits serveixen com a bit de paritat, després se li aplica una permutació a la matriu de 8x7 i es torna a dividir en Esquerra i Dreta, a partir de la següent taula es fan rotar les columnes de bits cap a l'esquerra depenent de la clau K (de la 1 fins a la 16) que es vulgui extreure.

Número de ronda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# bits a rotar	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

És a dir que per treure K_1 es rotarien totes les columnes una cap a l'esquerra, per treure K_2 es tornaria a rotar cap a l'esquerra, K_3 rotaria dos més, i així successivament fins a tenir les 16 claus. Després de fer això es tornen a ajuntar Dreta i Esquerra i finalment s'aplica la mateixa permutació a cada K_n .

Annex 3

Imaginem que tenim l'estat E següent:

AE	1E	21	1C
03	3F	04	33
1F	01	CF	11
2A	7A	7A	27

I la clau K següent:

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Estan representats els bits en forma hexadecimal, cada parell de números, de lletres o la combinació d'ambdós representen 8 bits.

SubBytes

Es fa un procés anomenat Rotword que et passa el primer byte de la columna a l'última posició:

09	→	CF
CF		4F
4F		3C
3C		09

Posteriorment es passa per una S-box on els 4 primers bits indiquen la fila i els 4 últims la columna:

CF	→	8A
4F		84
3C		EB
09		01

S-box	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	B8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	Df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

A continuació es fa XOR amb una “Constant de Ronda” (RCON), diferent per a cada ronda.

La operació següent és el resultat per a la primera columna de la següent clau:

Primera columna de K		Columna després de passar per Rotword i S-box		Primera columna de RCON	
2B		8A		01	
7E	XOR	84	XOR	00	=
15		EB		00	
16		01		00	
		Resultat			
		A0			
		FA			
		FE			
		17			

Per aconseguir les altres columnes es fa XOR amb la columna K i la que ens acaba de sortir:

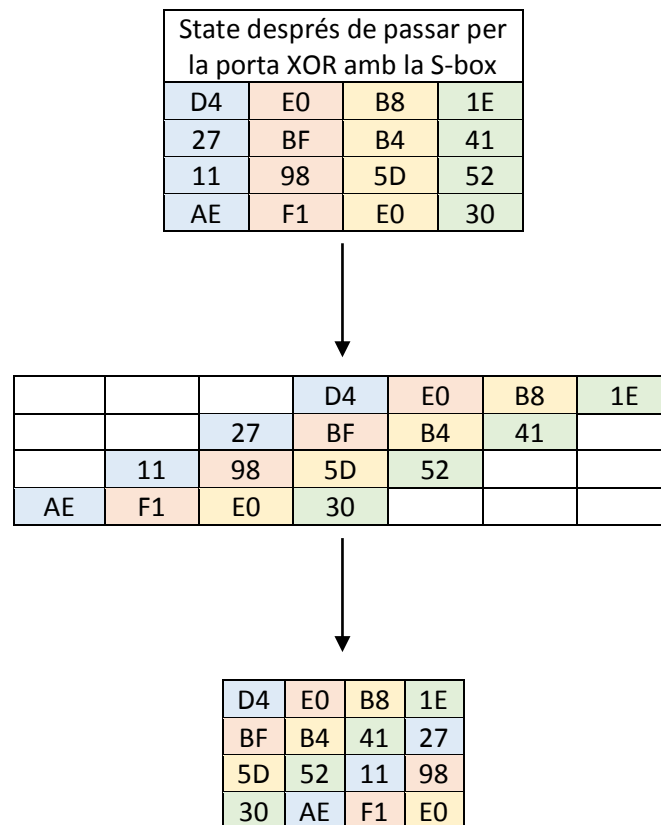
Columna de K		Resultat anterior		Resultat per a columna 2
28		A0		88
AE	XOR	FA	=	54
D2		FE		2C
A6		17		B1

Si fem el mateix amb totes les columnes, ens donarà que la subclau 1 és:

A0	88	23	2A
FA	2C	A3	76
FE	2C	39	76
17	B1	39	05

ShiftRows

El que es fa aquí és desplaçar cada fila una posició cap a l'esquerra més que l'anterior fila:



MixColumns

A cada columna de l'state se li aplica una transformació lineal, és a dir, que cada columna és multiplicada per una matriu determinada en el camp de Galois.

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

 \cdot

D4
BF
5D
30

 $=$

04
66
81
E5

04	E0	48	28
66	CB	F8	06
81	19	D3	26
E5	9A	7A	4C

Annex 4

Per a obtenir les claus privada i pública primer s'escullen aleatòriament dos nombres primers, p i q , diferents entre ells i molt grans (mínim solen ésser de 10^{100}). Una vegada tenim els dos nombres se'ls multiplica per a aconseguir n i saber en quin mòdul treballar.

Posteriorment es calcula utilitzant la funció φ d'Euler $\varphi(n) = (p-1)(q-1)$ i s'escull un nombre positiu e menor que $\varphi(n)$ que sigui coprimer amb $\varphi(n)$, és a dir, que el seu màxim comú divisor sigui 1, $\text{mcd}(e, \varphi(n)) = 1$. e es guardarà com a l'exponent de clau pública.

Mitjançant aritmètica modular es determina d tal que $e \cdot d \equiv 1 \pmod{\varphi(n)}$, que es pot calcular fent $d = \frac{y \cdot \varphi(n) + 1}{e}$ on $y=1, 2, 3, \dots$ Fins a trobar un d enter. d es guardarà com a l'exponent de clau privada.

Finalment el parell de números (e, n) seran la clau pública i el parell (d, n) seran la clau privada.