UNIVERSIDAD POLITÉCNICA DE CARTAGENA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Estudio y desarrollo de una máquina Enigma-software

Autor: Carlos Alberto Pérez Martínez
Director: Josemaría Malgosa Sanahuja

Cartagena, Noviembre 2005

ÍNDICE

1. Introducción…….. Pág.5
2. Objetivos…… Pág.9
3. Historia de la Criptografía Clásica…….. Pág.11
4. Método de sustitución Monoalfabética……. Pág.27
5. Método de sustitución Polialfabética……. Pág.35
6. Método de transposición……. Pág.41
7. La Enigma……. Pág.45
8. Guión de Practicas..…… Pág.65
9. Anexo: Código C implementado…… Pág.83
10. Conclusión……..……..…… Pág.121
Bibliografía……..…… Pág.123

Capítulo 1: Introducción

Este capítulo trata de dar una visión global de la criptografía así como dar una breve explicación de los conceptos criptográficos más relevantes.

La criptografía (del griego *kryptos*, "ocultar", y *grafos*, "escribir", literalmente "escritura oculta") es el arte o ciencia de cifrar y descifrar información utilizando técnicas matemáticas que hagan posible el intercambio de mensajes de manera que sólo puedan ser leídos por las personas a quienes van dirigidos. Cuando se habla de esta área de conocimiento como ciencia se debería hablar de criptología, que engloba tanto las técnicas de cifrado (la criptografía propiamente dicha) como sus técnicas complementarias: el criptoanálisis, que estudia los métodos que se utilizan para romper textos cifrados.

Su finalidad es garantizar el secreto en la comunicación entre dos entidades (personas, organizaciones, etc.) y asegurar que la información que se envía es auténtica en un doble sentido: que el remitente sea realmente quien dice ser y que el contenido del mensaje enviado, habitualmente denominado criptograma, no haya sido modificado en su tránsito.

La operación de cifrado es una función T que se aplica sobre el mensaje m más una clave (ver figura 1):

$$c=\mathrm{T}(m,k)$$

Es decir, T es una familia de funciones sobre el mensaje. La función T tiene inversa, que es el descifrado:

$$m=\mathrm{T}^{-1}(c,k) = \mathrm{T}^{-1}(\mathrm{T}(m,k),k)$$

Usaremos esta notación: *c* para el mensaje cifrado, *m* para el mensaje en claro y *k* para la clave.

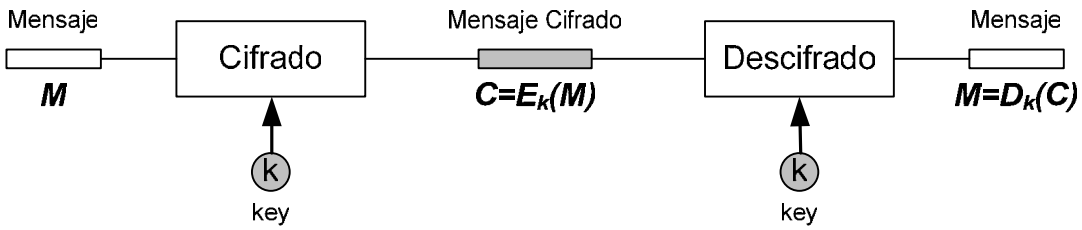


Figura 1. Diagrama de bloques del proceso de cifrado y descifrado de un mensaje.

1.1 Conceptos

En la jerga de la criptografía, la información original que debe protegerse se denomina texto plano. El cifrado es el proceso de convertir el texto plano en un galimatías ilegible, denominado texto cifrado o criptograma. Por lo general, la aplicación concreta del algoritmo de cifrado (también llamado cifra) se basa en la existencia de una clave: información secreta que adapta el algoritmo de cifrado para cada uso distinto.

Las dos técnicas más básicas de cifrado en la criptografía clásica son la sustitución (que supone el cambio de significado de los elementos básicos del mensaje -las letras, los dígitos o los símbolos-) y la transposición (que supone una reordenación de las mismas); la gran mayoría de las cifras clásicas son combinaciones de estas dos operaciones básicas.

El descifrado es el proceso inverso que recupera el texto plano a partir del criptograma y la clave. El protocolo criptográfico especifica los detalles de cómo se utilizan los algoritmos y las claves (y otras operaciones primitivas) para conseguir el efecto deseado. El conjunto de protocolos, algoritmos de cifrado, procesos de gestión de claves y actuaciones de los usuarios, en su globalidad es lo que constituyen un criptosistema, que es con lo que el usuario final trabaja e interactúa.

Existen dos grandes grupos de cifras: los algoritmos que utilizan una misma clave para el proceso de cifrado y descifrado y los que utilizan una clave para cifrar mensajes y otra distinta para descifrarlos. Los primeros se denominan cifras simétricas o de clave simétrica y son la base de los algoritmos de cifrado clásico. Los segundos se denominan cifras asimétricas de clave asimétrica o de clave pública-privada y forman en núcleo de las técnicas de cifrado modernas.

En el lenguaje cotidiano, la palabra código se usa de forma indistinta con cifra. En el la jerga de la criptografía, sin embargo, el término tiene un uso técnico especializado: los códigos son un método de criptografía clásica que consiste en sustituir unidades textuales más o menos largas o complejas (habitualmente palabras o frases) para ocultar el mensaje; por ejemplo, "*cielo azul*" podría significar "*atacar al amanecer*". Por contra, las cifras clásicas normalmente sustituyen o reordenan los elementos básicos del mensaje -letras, dígitos o símbolos-; en el ejemplo anterior, "*rcnm arcteeaal aaa*" sería un criptograma obtenido por transposición. Cuando se usa una técnica de códigos, la información secreta suele recopilarse en un libro de códigos.

Con frecuencia los procesos de cifrado y descifrado se encuentran en la literatura como encriptado y desencriptado, aunque ambos son neologismos todavía sin reconocimiento académico. Hay quien hace distinción entre "cifrado/descifrado" y "encriptado/desencriptado" según esté hablando de criptografía simétrica o asimétrica, pero la mayoría de los expertos en el mundo académico prefiere evitar ambos neologismos.

Conceptos básicos de seguridad.

A continuación definimos una serie de conceptos relacionados con el campo de la criptografía y muy utilizados en el argot propio de esta ciencia.

- Confidencialidad: Asegura que la información en un sistema o en tránsito es revelada sólo a usuarios autorizados. Protege contra el análisis y manipulación de los datos.

- Autenticación de Entidad: asegura que un usuario es quién dice ser. Protege contra la suplantación de usuarios.

¿Cómo comprueba Bernardo (B) que el mensaje recibido del emisor que dice ser Alicia(A) es efectivamente de esa persona?

- Integridad: asegura que los datos no han sido modificados durante la comunicación o el almacenamiento. Protege contra la manipulación de los datos.

¿Cómo comprueba Bernardo(B) que el mensaje recibido del emisor Alicia(A) es el auténtico y no un mensaje falso?

- No Repudio: asegura la integridad y la identidad del creador de los datos, de modo que el mensaje pueda ser una prueba de ambas cosas. Protege contra el repudio.

- Control de Acceso: evita el uso no autorizado de recursos. Protege contra la manipulación y el análisis de los datos.

- Disponibilidad: asegura que los recursos estarán disponibles para los usuarios autorizados. Protege contra la denegación de servicio.

1.2 Tipos de Algoritmos de Cifrado

Todos los algoritmos de cifrado pueden englobarse en dos categorías:

- Algoritmos de sustitución: Sustitución de cada uno de los símbolos de un alfabeto por otro símbolo del mismo alfabeto, siguiendo un determinado patrón.

- Algoritmos de transposición: Permutación de los símbolos del texto en claro.

En general, los métodos consisten en aleatorizar los datos de manera que el texto cifrado no presente regularidades estadísticas. La ausencia de regularidades obliga a que la única forma de atacar sea el análisis exhaustivo, en cuyo caso la seguridad residirá en la seguridad computacional, en el caso que el espacio de claves sea lo suficientemente extenso como para hacer el ataque inviable.

Capítulo 2: Objetivos

Este trabajo trata sobre criptografía clásica y su principal fin es crear un software para la simulación de algunos de estos métodos de criptografía, que intentan ayudar a comprender la metodología y algoritmo de los métodos que simulan. La intención de estos pequeños programas es docente, es decir, pueden ser usados en unas prácticas que traten de criptografía clásica y que intente explicar de forma practica los métodos de cifrado. En definitiva, estos programas cifran la información contenida en un fichero de texto según el algoritmo clásico utilizado.

Otro objetivo es crear herramientas, afines al software antes comentado, con el fin de que aquellas personas que estén usando los programas para uso docente o particularmente, puedan analizar los ficheros cifrados con los diferentes algoritmos y proceder a su criptoanálisis.

Como último objetivo se intenta ofrecer una Guía de Practicas que intenta dar un orden y una metodología de ayuda al uso de los programas, con ejemplos de uso de éstos, incluyendo textos para poder cifrar y descifrar usando las diferentes herramientas.

Por ultimo, un posible objetivo que se pretendía alcanzar con este Proyecto era la propia formación del autor, tanto en el campo de la criptografía como de la programación en C.

Capítulo 3:
Historia de la criptografía

Los secretos existen desde que existe el hombre, por tanto, intentar ocultarlos es un arte practicado desde el principio de la Humanidad. Ya las primeras civilizaciones desarrollaron técnicas para enviar mensajes durante las campañas militares de forma que si el mensajero era interceptado, la información que portaba no corriera el peligro de caer en manos del enemigo. En este capitulo se intenta dar una visión de la criptografía en la historia.

Las guerras han sido, desde siempre, el motivo por el que mejorar mucha de la tecnología que conocemos actualmente. La necesidad de ser más letal, rápido, inteligente y mejor que el enemigo, con herramientas cada vez más sofisticadas, ha hecho que inventos a los que en un principio no se les concedió demasiada importancia, fueran desarrollados hasta el límite si se les vislumbraba una aplicación que permitiera un mínimo de superioridad con respecto al contrario. No ha sido menos con la criptografía.

La historia de la escritura secreta empieza con algunos métodos estegranograficos: consisten en la ocultación de la información ocultando el mensaje, Algunos ejemplos son:

- El historiador Herodoto cuenta en el 480a.c. que Demarato advirtió a los griegos de una posible invasión de los persas ocultando la información grabada en un par de tablillas de madera y cubriéndolas con cera.

- El mismo Herodoto cuenta un caso que para ocultar un mensaje se afeito el pelo al mensajero y se esperó a que le volviera a crecer e pelo para transmitir el mensaje

- Giovanni Porta (siglo XV) describe cómo ocultar un mensaje dentro de un huevo duro.

- Plinio el Viejo (siglo I) describe cómo la leche de la planta Thithy-mallus puede usarse para hacer tinta invisible.

La principal debilidad de estos métodos es que si se encuentra el mensaje, también se encuentra la información de éste. Por ello se desarrollo la criptología.

A continuación se muestra de manera esquematica una clasificación de métodos criptográficos clásicos:

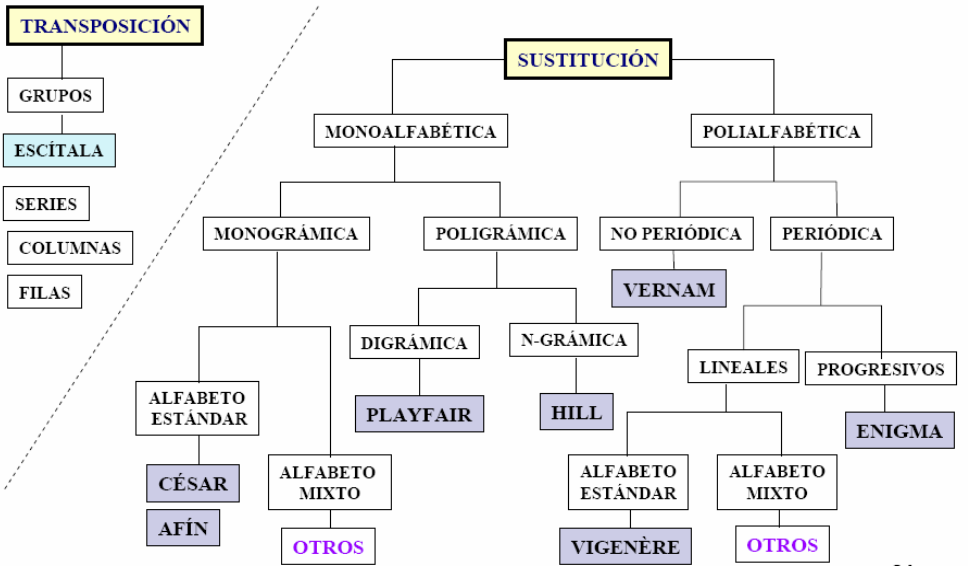


Figura 2. Clasificación de métodos criptográficos clásicos

3.1 Cifradores clásicos.

Los romanos, que sin duda presumían de una matemática muy avanzada, ya conocían ciertos "trucos" para traspasar información de un lugar a otro sin que el enemigo pudiera conocerla.

Por ejemplo, durante las guerras, para avisar a ciudades lejanas o a otros ejércitos que combatían en otras tierras de los posibles peligros o de la próxima estrategia a seguir por sus soldados, usaban a un emisario a caballo que se desplazaba entre los distintos puntos. El emisario (por supuesto los jefes no) corrían el riesgo de ser atrapados durante la marcha y que su mensaje fuese descubierto. Idearon entonces una manera de hacer que esta información fuese leída sólo por su destinatario, con el que habían acordado ya el sistema criptográfico. La más sencilla consistía en un simple intercambio de letras. Cada letra del mensaje era sustituida por la sucesiva (o la siguiente) en el orden del abecedario. Este sistema, aunque simple, impedía que la mayor parte de los soldados enemigos o propios (en su mayoría analfabetos, y si sabían a duras penas leer, pocos podían imaginar el sistema de "cifrado") pudiesen entender el mensaje. Este sistema se llamó "*de Julio Cesar*", gran estratega que se sirvió de la criptografía para ganar batallas y conquistar ciudades.

Las técnicas fueron mejorando según la civilización, dando lugar a un curioso método: el de la escítala. Fueron los griegos los que idearon este sistema, mucho más ingenioso e innovador que el de Julio Cesar. El mensaje se escribía sobre una cinta de tela, enrollada cuidadosamente sobre un palo de madera de un cierto grosor. Al desenrollar la cinta, el mensaje se perdía entre los pliegues de la tela, volviéndolo ilegible para cualquiera que no supiese cómo había sido escrito. El receptor del mensaje tenía que tomar la tela y volver a enrollarla en un palo del mismo grosor, comenzando por un lugar exacto donde se hacía una marca especial que sólo conocían entre el emisor y receptor. En este sistema, el "grosor" de la estaca se podría considerar la clave maestra que cifraba el mensaje.

La excítala

Ya en siglo V antes de J.C. los lacedemonios, un antiguo pueblo griego, usaban el método de la *escítala* para cifrar sus mensajes. El sistema consistía en una cinta que se enrollaba en un bastón y sobre el cual se escribía el mensaje en forma longitudinal como se muestra en la figura.

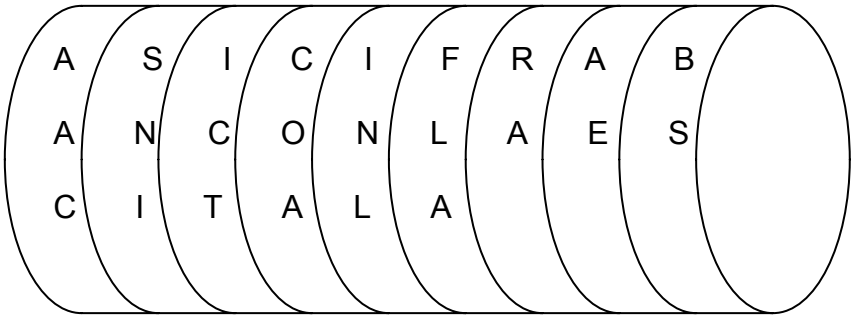


Figura 3. Cifrado mediante sistema de escítala.

Una vez escrito el mensaje, la cinta se desenrollaba y era entregada al mensajero; si éste era interceptado por cualquier enemigo, lo único que se conseguía era un conjunto de caracteres o letras distribuidas al parecer de forma aleatoria en dicha cinta. Incluso si el enemigo intentaba enrollar la cinta en un bastón con diámetro diferente, el resultado obtenido era un conjunto de letras escritas una a continuación de otra sin sentido alguno. Por ejemplo, en el caso de la figura 2, la cinta llevará el mensaje m = ASI CIFRABAN CON LA ESCITALA si bien en ella sólo podrá leerse el criptograma c = *AACSNIICTCOAINLFLARAAEBS*. Para enmascarar completamente la escritura, es obvio que la cinta en cuestión debe tener caracteres en todo su contorno. Como es de esperar, la clave del sistema residía precisamente en el diámetro de aquel bastón, de forma que solamente el receptor autorizado tenía una copia exacta del mismo bastón en el que enrollaba el mensaje recibido y, por tanto, podía leer el texto en claro. En este sistema no existe modificación alguna del mensaje; es decir, éste va *en claro* desde el transmisor hacia el receptor, por lo que como veremos más adelante se tratará de un cifrador por transposición.

De esta forma se lograba el objetivo de la confidencialidad, en tanto que la integridad estaba en entredicho y dependía de lo *aguerrido* y *fiel* que fuese nuestro mensajero. Si la cinta era robada y se cambiaban los caracteres, podría llegar al receptor un mensaje sin sentido y, lo que es peor, con un duplicado del bastón original podía enviarse un mensaje con sentido completamente distinto al encomendado al mensajero. Haga un viaje mental al pasado e imagínese lo que significaría en aquellos tiempos que el destinatario recibiera el mensaje falso m_{falso} = *RENDICIÓN TOTAL* en vez del verdadero mensaje $m_{verdadero}$ = *ATACAMOS MAÑANA*, ambos de 14 caracteres.

Para terminar, un apunte curioso y de cultura general. De estos tiempos tan remotos se debe la famosa frase de ostentar el "*bastón de mando*" –tan popular entre nuestros queridos políticos y en particular alcaldes- y que, como es de suponer, en aquella época no se soltaba por ningún motivo puesto que en él residía la seguridad del sistema de información y la vida política de este pueblo de la antigua Grecia.

El cifrador de Polybios.

A mediados del siglo II antes de J.C., encontramos el cifrador por sustitución de caracteres más antiguo que se conoce. Atribuido al historiador griego *Polybios*, el sistema de cifra consistía en hacer corresponder a cada letra del alfabeto un par de letras que indicaban la fila y la columna en la cual aquella se encontraba, en un recuadro de 5 x 5 = 25 caracteres, transmitiéndose por tanto en este caso el mensaje como un criptograma. En la figura se muestra una tabla de cifrar de Polybios adaptada al inglés, con un alfabeto de cifrado consistente en el conjunto de letras A, B, C, D y E.

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

Figura 4. Tablas de cifrar de Polybios.

Acorde con este método, la letra A se cifrará como AA, la H como BC, etc. Esto significa que aplicamos una sustitución al alfabeto {A, B, C, .., X, Y, Z} de 26 letras convirtiéndolo en un alfabeto de cifrado {AA, AB, AC, ..., EC, ED, EE} de 25 caracteres, si bien sólo existen 5 símbolos diferentes {A, B, C, D, E}. Este tipo de tabla o matriz de cifrado será muy parecida a la que en el siglo XIX se utilizará en el criptosistema conocido como cifrador de Playfair y que será tratado más adelante en el apartado de cifradores poligrámicos, salvo que en este último la operación de cifra no se realiza por monogramas como en el de Polybios sino por diagramas, conjunto de dos caracteres del texto en claro.

Ejemplo 1: *Usando* la Tabla del cifrador de Polybios, cifre el mensaje:

m = QUE BUENA IDEA LA DEL GRIEGO.

Solución:

c = DADEAE ABDEAECCAA BDADAEAA CAAA ADAECA BBDBBDAEBBCD.

El cifrador del César

Unos cincuenta años después del cifrador de Polybios, en el siglo I antes de J.C., aparece un cifrador básico conocido con el nombre genérico de cifrador del César en honor al emperador *Julio César* y en el que ya se aplica una transformación al texto en claro de tipo monoalfabética. Como se verá en un apartado posterior, el cifrador del César aplica un desplazamiento constante de tres caracteres al texto en claro, de forma que el alfabeto de cifrado es el mismo que el alfabeto del texto en claro pero desplazado 3 espacios hacia la derecha (módulo n, con n el número de letras del mismo). Se muestra el alfabeto y por tanto la transformación que utiliza este cifrador por sustitución de caracteres para el alfabeto castellano de 27 letras.

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

Alfabeto de cifrado del César para lenguaje castellano.

Ejemplo 2: Utilizando el cifrador del César según el alfabeto mostrado en la figura, cifre los siguientes mensajes:

m_1 = VINI, VIDI, VINCI. (Frase célebre de César: llegué, vi, vencí).
m_2 = AL CÉSAR LO QUE ES DEL CÉSAR.

Solución:Aplicando a cada carácter M_i su equivalente C_i de la tabla de la figura, se obtienen los siguientes criptogramas:

c_1 = YLPL, YLGL, YLPFL.
c_2 = DÑ FHVDU ÑR TXH HV GHÑ FHVDU.

A partir del ejemplo anterior, es fácil apreciar ciertas *debilidades* en este cifrador como, por ejemplo, la repetición de la cadena de caracteres *YL* en el criptograma primero y *FHVDU* en el segundo que entregan demasiadas pistas a un posible criptoanalista. Estos y otros puntos débiles del cifrador del César que por ahora no saltan a la vista serán analizados y comentados más adelante.

El cifrador de Alberti

En el siglo XVI *León Battista Alberti* presenta un manuscrito en el que describe un disco cifrador con el que es posible cifrar textos sin que exista una correspondencia única entre el alfabeto del mensaje y el alfabeto de cifrado como en los casos analizados anteriormente. Con este sistema, cada letra del texto en claro podía ser cifrada con un carácter distinto dependiendo esto de una clave secreta. Se dice entonces que tales cifradores usan más de un alfabeto por lo que se denominan *cifradores polialfabéticos,* a diferencia de los anteriores denominados *monoalfabéticos*.

Como se aprecia en la figura, el disco de Alberti presenta en su círculo exterior los 20 caracteres del latín, esto es, los mismos del alfabeto castellano excepto las letras H, J, Ñ, K, U, W e Y, y se incluyen los números 1, 2, 3 y 4 para códigos especiales. Por su parte, en el disco interior aparecen todos los caracteres del latín además del signo & y las letras H, K e Y. Al ser 24 los caracteres representados en cada disco, es posible definir hasta 24 sustituciones diferentes; es decir, dependiendo de la posición del disco interior la cantidad máxima de alfabetos de cifrado es igual a 24. Luego, para cifrar un mensaje, una vez establecida la correspondencia entre caracteres de ambos discos o, lo que es lo mismo, el alfabeto de cifrado, se repasa letra a letra el texto en claro del disco exterior y se sustituye cada una de ellas por la letra correspondiente del disco interior.

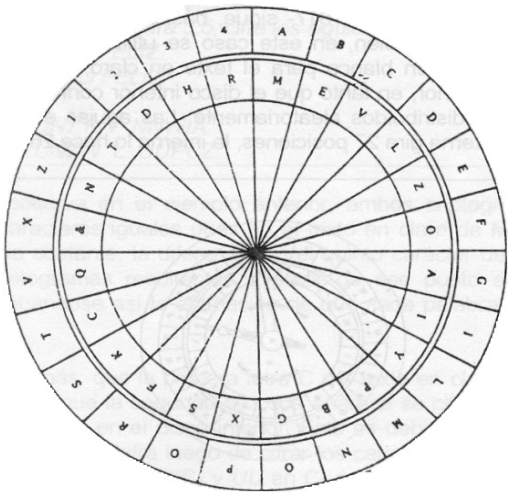


Figura 5.Disco cifrador de Alberti.

La innovación que supone este sistema consiste en que el alfabeto de sustitución puede ser cambiado durante el proceso de cifrado, por ejemplo cada *k* caracteres, simplemente girando el disco interior y por tanto utilizando otro alfabeto de sustitución.

Ejemplo 3: Cifre con el disco de Alberti de la figura, siendo su posición inicial la de coincidencia entre el número 1 del disco exterior y el signo & del disco interior, el siguiente mensaje:

```
m = EL DISCO DE ALBERTI ES EL PRIMER CIFRADOR
    POLIALFABÉTICO CONOCIDO.
```

Solución: Desplazamos el disco interior dos espacios en el sentido de las agujas del reloj y leemos el carácter cifrado en el disco interior bajo el carácter correspondiente del texto en claro del disco exterior, obteniéndose:

```
c = VA EOSMP EV HARVXFO VS VA BXOIVX MOLXHEPX
    BPAOHALHRVFOMP MPYPMOEP.
```

Estos métodos tradicionales se mantuvieron durante mucho tiempo. Si bien se mejoraron, especialmente los basados en el cifrado César, pues la permutación de letras en el mensaje permite un número casi infinito de combinaciones.

Durante siglos, la cifra de sustitución Monoalfabética había sido utilizado para asegurar el secreto. Como se vera mas adelante el desarrollo del análisis de frecuencia, primero en el mundo árabe y luego en Europa destruyo su seguridad. Para afrontar este reto surgieron las cifras polialfabéticas. Las primeras alternativas fueron el uso de dos alfabetos para cifrar (codificación bialfabética) donde a diferencia de la sustitución monoalfabética se cifra usando dos alfabetos de manera alterna.

	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
AC1	F	Z	B	V	K	I	X	A	Y	M	E	P	L	S	D	H	J	O	R	G	N	Q	C	U	T	W
AC2	G	O	X	B	F	W	T	H	Q	I	L	A	P	Z	J	D	E	S	V	Y	C	R	K	U	H	N

aquello ⇨ FENFAPD

Pero fue Blaise de Vigénere quien 1586 publica *Traicte des Chiffres*, definiendo lo que se ha venido denominando la cifra Polialfabética.

- Propone usar para la codificación hasta 26 alfabetos distintos.
- La palabra clave determina los alfabetos a utilizar.

El criptosistema de Vigènere

El sistema de cifrado de Vigenère (en honor al criptógrafo francés del mismo nombre) es un sistema polialfabético o de sustitución múltiple. Este tipo de criptosistemas aparecieron para sustituir a los monoalfabéticos o de sustitución simple, basados en el Cesar, que presentaban ciertas debilidades frente al ataque de los criptoanalistas relativas a la frecuencia de aparición de elementos del alfabeto. El principal elemento de este sistema es la llamada Tabla de Vigenère, una matriz de caracteres cuadrada, con dimensión, que se muestra en la tabla adjunta.

La clave del sistema de cifrado de Vigenère es una palabra escogida al azar que suele tener sentido para que sea fácil de recordar. Cada una de las letras de la tabla corresponde con el alfabeto elegido de la tabla adjunta para cifrar el mensaje.

Por ejemplo para la clave *k*=azar se utilizaran los alfabetos de la tabla correspondientes a los alfabetos A, Z, A y R de la tabla.

Una mejora sobre el cifrado de Vigenère fue introducida por el sistema de Vernam, utilizando una clave aleatoria de longitud igual a la del mensaje; la confianza en este nuevo criptosistema hizo que se utilizase en las comunicaciones confidenciales entre la Casa Blanca y el Kremlin, hasta, por lo menos, el año 1987.

	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

Figura 6. Tabla de Vigénere

3.2 Cifradores del siglo XIX

En el siglo XIX comienzan a desarrollarse diversos sistemas de cifra con las características polialfabéticas propuestas por Alberti, entre los que destacan el de discos concéntricos de Wheatstone en 1860 y el de cilindros de Bazeries en 1891.

El cifrador de Wheatstone

El criptógrafo de *Wheatstone* mostrado en la figura. -según un invento de *Decius Wadsworth* desarrollado en 1817- sigue, básicamente, el mismo algoritmo de cifra que el de Alberti. Ahora bien, en este caso se utiliza el alfabeto inglés de 26 caracteres más el espacio en blanco para el texto en claro, representado de forma ordenada en el disco exterior, en tanto que el disco interior contiene solamente los 26 caracteres del lenguaje distribuidos aleatoriamente. Las agujas están engranadas de forma que cuando la externa gira 27 posiciones, la interna lo hace 26.

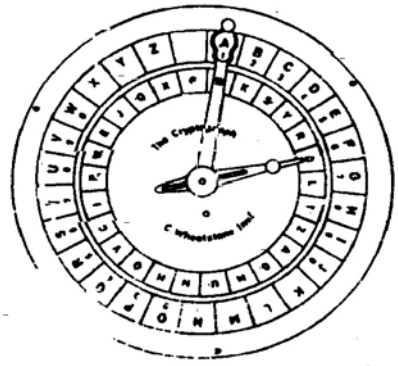


Figura 7. Máquina de cifrar de Wheatstone.

El método de cifra consiste en hacer girar la aguja externa en el sentido de las manecillas del reloj hasta hacer coincidir cada letra del texto en claro con la letra del disco externo y apuntar el carácter correspondiente que aparece en el círculo interior, incluso para el espacio en blanco. Observe que por la relación de giro de las agujas, éstas se van separando una posición o letra por cada vuelta, de forma que el alfabeto de cifrado será diferente cuando se cumpla cualquiera de estas tres condiciones:

a) Que se termine una palabra del texto en claro y por tanto demos un giro completo de la aguja mayor al buscar el espacio en blanco.

b) Que aparezcan letras repetidas y tengamos que dar toda una vuelta completa al buscar la segunda. No obstante, según los autores, en este caso es posible también omitir cifrar la letra repetida o bien cifrar ambas como una única letra poco usual, por ejemplo la letra Q.

c) Que las letras de una palabra no vengan en orden alfabético. Es decir, si ciframos la palabra *CELOS* no alcanzamos a dar la vuelta completa al disco exterior, en tanto que la palabra *MUJER* implica dos vueltas y *HOMBRE* significa tres.

La importancia de este cifrador está en que cada una de las palabras del mensaje influye en la forma en que se cifran las siguientes, una propiedad muy interesante y que precisamente utilizarán los cifradores modernos, sencillamente definiendo el concepto de palabra como bloque de bits para la cifra y aplicando lo que se denomina cifrado con encadenamiento.

Ejemplo 4: Con la máquina de cifrar de Wheatstone y suponiendo la posición inicial indicada en la figura, cifre los siguientes mensajes:

M_1 = CHICA FELIZ.
M_2 = CHICO FELIZ.

Solución:

C_1 = TUNZT T NNWIA.
C_2 = TUNZW L UUPCZ.

Como se observa en el ejemplo anterior, ambos criptogramas presentan los cuatro primeros caracteres iguales pues en el texto en claro de M_1 y M_2 también son iguales (CHIC). No obstante, la diferencia en el quinto carácter de los textos M_1 y M_2, hace que los criptogramas resultantes a partir de ese punto sean completamente diferentes, comprobándose así la afirmación de que cada palabra influye en el cifrado de la siguiente.

Observe, además, que la primera letra C del texto en claro en ambos casos se cifra como *T*, en tanto que la segunda vez que aparece se cifra como *Z*, precisamente un espacio hacia delante en el disco interior. Esto es debido al giro completo que se produce en la operación de cifra luego de cifrar los caracteres C, H e I. Por otra parte, los caracteres repetidos *TTNN* en C1 y *UU* en C2 se deben a una revolución completa del disco interior producida por dos caracteres contiguos en el texto en claro y que están separados 26 espacios como es el caso de los diagramas "A " y "FE". Por último, apréciese que una misma palabra repetida en el texto en claro se cifrará cada vez con un alfabeto distinto por la rotación completa producida por la búsqueda del espacio en blanco.

Por ejemplo el mensaje M = *TORA TORA*, palabra secreta usada como clave por los japoneses en el ataque a Pearl Harbor y cuyo significado es tigre, se cifrará como C = *XWQT Z KQBG.*

El cifrador de Bazeries

El cifrador de *Étienne Bazeries*, criptólogo francés nacido a finales del siglo XIX, está basado en el cifrador de ruedas de Jefferson, inventado unos 100 años antes por *Thomas Jefferson* reconocido como el padre de la criptografía americana. El criptógrafo mostrado en la figura consta de 20 discos, cada uno de ellos con 25 letras en su circunferencia, de forma que la clave se establece sobre la generatriz del cilindro, determinándose 25 alfabetos diferentes. Su funcionamiento es el siguiente: para cifrar el mensaje, primero se divide éste en bloques de 20 letras,

procediendo luego a su colocación en forma longitudinal en la línea del visor. El criptograma que se envía puede ser cualquiera de las 25 líneas, también llamadas generatrices del cilindro. Por ejemplo, si se elige la generatriz de distancia +2 en la figura, el mensaje M = JE SUIS INDECHIFFRABLE del visor se cifraría como C = LOVS PQUU TPUKEJHHCFDA.

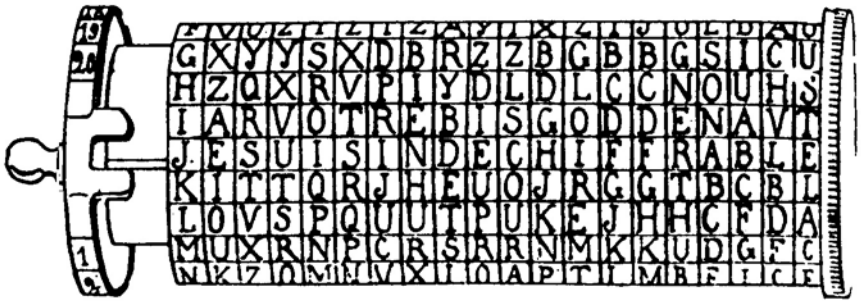


Figura 8. Máquina de cifrar de Bazeries.

Se puede elegir la misma distancia a la generatriz en la cual se lee el criptograma para todo el bloque o bien cambiar ésta en cada bloque o elemento del bloque, de forma que el número de combinaciones o alfabetos distintos en vez de ser solamente 25 podría crecer hasta el factorial de 25, un valor verdaderamente alto. Uno de estos posibles alfabetos podría ser elegir una secuencia de distancias, una vez introducido el mensaje en el visor, igual a -1,-2,-2,-1,1,2,2,1,-1,-2,-2,-1,1,2,2,1,-1,-2,-2,-1. Es decir, una vez se tenga el mensaje en claro en el visor, se envía como primer carácter del criptograma el que, en la misma columna, está desplazado una posición hacia atrás en el anillo; como segundo el que está desplazado dos posiciones atrás, el tercero también dos posiciones atrás, el cuarto una posición atrás, el quinto una posición hacia delante, el sexto dos adelante, etc., de manera que el criptograma forma una especie de *zig-zag* en torno al texto en claro, sin transmitir ningún carácter de éste puesto que la posición 0 no se encuentra en la secuencia indicada. Como es fácil observar, dicha secuencia sería la clave del sistema y, en este caso, su valor máximo sería igual todas las posible permutaciones es decir, 25! = $1{,}55 \times 10^{25}$, un valor muy grande aunque el sistema de cifra sería engorroso y poco práctico.

La operación de descifrado consiste en poner los caracteres del criptograma en el visor y buscar en alguna de las líneas el mensaje en claro o seguir el proceso inverso al comentado anteriormente. Como los bloques de criptograma tienen longitud de veinte caracteres, es prácticamente imposible que exista más de una solución con sentido.

Ejemplo 5: Considerando una representación del cifrador de Bazeries como la que se indica a continuación, cifre el mensaje mostrado en el visor de la generatriz 11 del disco:

M = INTENTA ROMPER LA CIFRA.

a) Con una distancia constante de +3 espacios.
b) Con la secuencia S de distancia de cifrado indicada:

S = 0,1,2,1,0,-1,-2,-1,0,1,2,1,0,-1,-2,-1,0,1,2,1.

	Fila																			
Disco	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
7	V	A	M	W	W	U	I	O	P	S	S	A	H	K	L	V	C	D	U	Q
8	M	O	J	F	J	K	L	M	A	C	H	Y	E	D	X	Z	G	Q	I	U
9	D	B	W	Q	D	S	B	D	Q	T	F	D	S	F	D	X	E	W	Q	G
10	A	W	K	Y	H	M	P	E	S	H	U	K	P	U	O	E	S	J	K	D
11	*I*	*N*	*T*	*E*	*N*	*T*	*A*	*R*	*O*	*M*	*P*	*E*	*R*	*L*	*A*	*C*	*I*	*F*	*R*	*A*
12	R	G	I	S	X	F	W	G	B	A	N	L	F	M	J	H	A	A	S	H
13	U	I	D	V	C	R	I	I	Z	D	D	C	Z	A	K	I	M	B	L	X
14	K	L	B	O	T	Z	H	Y	L	V	C	O	N	D	W	A	L	M	V	Z
15	W	H	S	K	L	P	O	U	I	E	E	P	D	N	C	G	Q	E	O	B

....

.....

Solución:

a) C = KLBOTZHYLVCONDWALMVZ.

b) Según la secuencia indicada, tomamos caracteres consecutivos de las línea 11,12,13,12,11,10,9,10,11,12,13,12,11,10,9,10,11,12,13,12 que se encuentran subrayados. C = IGDSNMBEOADLRUDEIALH.

Todos los sistemas comentados y muchísimos otros que se desarrollaron paralelamente en Europa, América y Asia, han sido criptoanalizados incluso sin contar con la ayuda de equipos informáticos. No obstante, la discusión de su criptoanálisis está fuera del objetivo de este libro, por lo que al lector interesado en tales temas históricos se le remite nuevamente al libro de Khan y a las publicaciones que se indican.

3.3 Máquinas de cifrar en el siglo XX

Ya entrado el siglo XX, aproximadamente unos 20 años antes de que estalle la Segunda Guerra Mundial, se desarrollan diversas máquinas de cifrar con rotores o ruedas que permiten un cifrado polialfabético, dando lugar a un importante número de claves secretas que, para aquel entonces, dificultaba *in extremis* el criptoanálisis. Este desarrollo a nivel industrial de los criptosistemas resulta lógico pues en aquellos años previos a dicha confrontación mundial, estaba todavía muy fresco en la memoria de todos, y en especial de gobernantes y militares, los efectos de la Primera Guerra Mundial, por lo que las medidas de seguridad ante el miedo al espionaje adquirían una importancia vital.

Ya en el siglo XX, durante la I Guerra Mundial, no cabía ingenuidad ni sistema de transposición de letras que no pudiese ser violado por los servicios de inteligencia de uno u otro bando. Por tanto, era necesario recurrir a sistemas mucho más eficaces para cifrar la información. Se consiguió, sin necesidad de ordenadores ni capacidad desbordante de cómputo, un tipo de cifrado que, en teoría, resultaba inviolable por definición. Se conocía por Cifrado Verman o "one time pad", haciendo referencia a que sus contraseñas eran de un sólo uso. Su fortaleza consistía en esto precisamente, que no existía una clave concreta que descubrir, sino que cada mensaje cifrado hacía uso de su propia contraseña. Como ya he dicho en alguna ocasión, para poder cifrar un mensaje es necesario: el mensaje, un algoritmo de cifrado y una contraseña. Si el algoritmo es público, el atacante debe centrarse en el descubrimiento de la contraseña, y si la contraseña

cambia en cada mensaje,¿para qué descubrirla? Si se intercepta otro mensaje, será inútil.

El cifrado Verman

Se basa en que ambos comunicadores poseen lo que se puede ver como una "gran clave" compuesta por un conjunto casi infinito y no repetitivo de letras realmente aleatorias. Ambos mantienen una copia idéntica de este largo listado de letras o números o caracteres, o todos combinados. Cada vez que alguno de ellos desea cifrar un mensaje, debe usar una parte, de longitud variable, de ese listado (normalmente el comienzo). Se podría ver como "arrancar" una parte de una "cinta" que contiene una ristra de números. Ahora, esa parte "arrancada" será la contraseña que se combinará con el texto (mediante un algoritmo) para obtener un mensaje cifrado. El algoritmo puede ser cualquier operación matemática permitida entre letras (o números, pues son fácilmente identificables unos con otros, trabajando con binarios). Por ejemplo, se pueden sumar las letras, o multiplicarse y el receptor, para descifrarlo, deberá realizar la operación inversa.

Cuando el mensaje llega a su destino, el receptor deberá usar una parte extraída de su "gran clave" de igual tamaño para poder realizar el proceso inverso y extraer del mensaje el texto plano. Una vez usadas ambas pequeñas partes de la "gran clave", son descartadas para siempre, utilizando cada vez una parte distinta aunque consecutiva. Así, se hace completamente imposible para un atacante conocer el mensaje cifrado, pues aunque conozca el algoritmo siempre le faltará un parámetro de la ecuación. Este es el único sistema absolutamente indescifrable siempre que la "cinta" se use una sola vez.

La máquina Enigma

Desde el siglo XIX y hasta la Segunda Guerra Mundial las figuras más destacadas fueron la del holandés Auguste Kerckhoffs y la del prusiano Friedrich Kasiski. Pero es en el siglo XX cuando la historia de la criptografía vuelve a presentar importantes avances. En especial durante las dos contiendas bélicas que marcaron al siglo: la Gran Guerra y la Segunda Guerra Mundial. A partir del siglo XX, la criptografía usa una nueva herramienta que permitirá conseguir mejores y más seguras cifras: las máquinas de cálculo. La más conocida de las máquinas de cifrado, posiblemente sea la máquina alemana Enigma: una máquina de rotores que automatizaba considerablemente los cálculos que eran necesarios realizar para las operaciones de cifrado y descifrado de mensajes. Para vencer al ingenio alemán, fue necesario el concurso de los mejores matemáticos de la época y un gran esfuerzo computacional. No en vano, los mayores avances tanto en el campo de la criptografía como en el de la informática no empezaron hasta entonces.

Tras la conclusión de la Segunda Guerra Mundial, la criptografía tiene un desarrollo teórico importante; siendo Claude Shannon y sus investigaciones sobre teoría de la información esenciales hitos en dicho desarrollo. Además, los avances en computación automática suponen tanto una amenaza para los sistemas existentes como una oportunidad para el desarrollo de nuevos sistemas. A mediados de los años 70 el Departamento de Normas y Estándares norteamericano publica el primer diseño lógico de un cifrador que estaría llamado a

ser el principal sistema criptográfico de finales de siglo: el Estándar de Cifrado de Datos o DES. En esas mismas fechas ya se empezaba a gestar lo que sería la, última revolución de la criptografía teórica y práctica: los sistemas asimétricos. Estos sistemas supusieron un salto cualitativo importante ya que permitieron introducir la criptografía en otros campos que hoy día son esenciales, como el de la firma digital.

Capítulo 4:
Método de Sustitución Monoalfabética

Explicación de los métodos de sustitución monoalfabética, con énfasis en el cifrado Cesar, desarrollando un ejemplo para entender este método. También se implementa la forma de proceder ante su criptoanálisis.

En los métodos de sustitución, cada letra del mensaje se sustituye por otra distinta. De entre todos los métodos de sustitución los cifrado por sustitución monoalfabética son los más simples y han sido los más utilizados.

Algunos ejemplos en la historia que se dan en la historia son:

- Vatsyana cuenta en el Kamasutra que las mujeres deberían estudiar 64 artes, entre ellas se incluye el arte de la escritura secreta. Propone emparejar al azar las letras del alfabeto y luego sustituir cada una por su pareja.
- En *"La guerra de las Galias"* Julio Cesar cuenta como mando un mensaje sustituyendo las letras latinas por las griegas.
- Suetonio en "Vidas de Cesares", cuenta otro método de sustitución utilizado por Julio Cesar que consistía en sustituir las letras del mensaje original por la que esta tres lugares mas adelante en el alfabeto.

Es evidente que al utilizar cualquier cambio entre 1 y 25 lugares es posible generar 25 alfabetos cifrados distintos. De hecho, si permitimos cualquier combinación del alfabeto llano, el número de posibles alfabetos cifrados es aproximadamente de 400.000.000.000.000.000.000.000.000.

4.1 El sistema Cesar

El cifrado Cesar (o César) es uno de los más antiguos que se conocen. Debe su nombre al emperador Julio César, que presuntamente lo utilizó para establecer comunicaciones seguras con sus generales durante las Guerras Gálicas.

Matemáticamente, para trabajar con el cifrado Cesar, tomamos el alfabeto numérico. Así, trabajando con el alfabeto inglés (para nosotros resulta conveniente tomar este alfabeto, de uso más extendido en informática que el español; la única diferencia radica en el uso de la letra "ñ"), podemos tomar el alfabeto definido por, de la siguiente forma:

A=0	B=1	C=2	D=3	E=4	F=5
G=6	H=7	I=8	J=9	K=10	L=11
M=12	N=13	O=14	P=15	Q=16	R=17
S=18	T=19	U=20	V=21	W=22	X=23
Y=24	Z=25				

La regla de cifrado es c = (a * m + b) mod 26.

El cifrado Cesar utilizaba a = 1 y b = 3 y de esta forma, la regla anterior quedará c= (m+3) mod26, Lo cual se traduce sencillamente en que para encriptar una letra hemos de tomar su entero correspondiente y sumarle 3 para obtener la letra correspondiente. Para descifrar un texto tomamos la función inversa, definida por. Veamos un ejemplo sencillo, en el que se toma: queremos cifrar, con la clave, la palabra CESAR; en primer lugar, tomando el valor de cada letra, tenemos el equivalente numérico de la palabra:

C	E	S	A	R
2	4	18	0	17

A cada uno de los números anteriores le aplicamos la función, de forma que obtenemos el texto cifrado:

5	7	21	3	20
F	H	V	D	U

Este texto (FHVDU) es el resultado de cifrar la palabra CESAR con la clave elegida a=1 y m=3. Es lo que enviaríamos al receptor, que conociendo la clave acordada sería capaz de descifrarlo.

Veamos ahora el ejemplo contrario: somos los receptores de un mensaje del que sabemos que ha sido cifrado con la misma clave, y buscamos descifrar la cadena que nos ha sido enviada, FVYXYW. El valor de cada letra es

E	U	X	W	X	V
4	20	23	22	23	21

Tomando, tenemos el resultado

1	17	20	19	20	18
B	R	U	T	U	S

Como vemos, retornando cada número al alfabeto inglés obtenemos el texto en claro que nuestro emisor nos ha enviado: BRUTUS, equivalente al cifrado EUXWXV.

Sin embargo, el cifrado de cesar es muy vulnerable no ofreciendo una seguridad fiable en la transmisión de datos confidentes.

Cesar modificado

Al igual que el sistema César explicado anteriormente, éste método también realiza una sustitución monoalfabética del alfabeto usado por el mensaje por otro distinto, que viene dado por otro con una palabra clave como veremos en el ejemplo.

Ejemplo: Para la clave CESAR tendremos la siguiente sustitución:

```
k = CESAR
m = un texto mas cifrado

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

m = u n t e x t o m a s c i f r a d o
c = U L T R X T M K C Q S G B P C A M
```

4.2 Criptoanálisis de los métodos de sustitución Monoalfabética

Existen documentos que demuestran que ya en el siglo X, los árabes cifraban documentos de impuestos y asuntos del estado. Ellos son los inventores del criptoanálisis: Ciencia cuyo objetivo consiste en descifrar un mensaje sin conocer la clave.

La idea central, lo que se conoce como análisis de frecuencias, es que en las lenguas occidentales no todas las letras aparecen con la misma frecuencia, sino que unas lo hacen mucho más a menudo que otras. La frecuencia de las letras en castellano es aproximadamente la que sigue (en lo sucesivo seguiré la convención criptográfica por la cual las letras del texto plano, es decir, el texto original, se escriben en minúsculas, y las del texto cifrado, en mayúsculas):

e- 16,78%	r - 4,94%	y - 1,54%	j - 0,30%
a - 11,96%	u - 4,80%	q - 1,53%	ñ - 0,29%
o - 8,69%	i - 4,15%	b - 0,92%	z - 0,15%
l - 8,37%	t - 3,31%	h - 0,89%	x - 0,06%
s - 7,88%	c - 2,92%	g - 0,73%	k - 0,00%
n - 7,01%	p - 2,776%	f - 0,52%	w - 0,00%
d - 6,87%	m - 2,12%	v - 0,39%	

Lo primero que llama la atención es que entre las seis primeras letras suman más de dos tercios del total.

Por ejemplo, suponer el siguiente texto cifrado mediante una técnica de sustitución monoalfabética.

SBLY AFL FWY IL UYC VYDQGYC VEC ZYILBYCYC AFL SYWIFSLW EU ÑYVPBL EU EBDL K E UE SQLWSQE LC LU ILCLY IL LGEIQBCL IL UE GQIE SYDQIQEWE SYW CF ECZLBLME IYUYBYCE K CF GESQY ILCLCZLBEWDL IL LCSEZEB E UEC SEILWEC IL ILCLYC CQLVZBL SEVPQEWDLC EUPLBD LQWCDLQW

Contando los caracteres del texto cifrado, tenemos para cada letra:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
2	11	22	7	26	6	4	0	15	0	2	35	1	0	1	0	3	12	0	11	0	9	5	11	0	18	5

que ordenadas por frecuencias dan la siguiente tabla:

L	E	C	Y	I	Q	B	S	W	U	D	F	V	Z	G	P	A	K	M	Ñ	H	J	N	O	R	T	X
35	26	22	18	15	12	11	11	11	9	7	6	5	5	4	3	2	2	1	1	0	0	0	0	0	0	0

Probaremos a sustituir los dos caracteres más frecuentes en el texto cifrado por los dos caracteres más frecuentes en castellano. Entonces L = e y E = a y tenemos:

SBeY AFe FWY Ie UYC VYDQGYC VaC ZYIeBYCYC AFe SYWIFSeW aU ÑYVPBe aU aBDe K a Ua SQeWSQa eC eU IeCeY Ie eGaIQBCe Ie Ua GQIa SYDQIQaWa SYW CF aCZeBeMa IYUYBYCa K CF GaSQY IeCeCZeBaWDe Ie eCSaZaB a UaC SaIeWaC Ie IeCeYC CQeVZBe SaVPQaWDeC aUPeBD eQWCDeQW

Otro truco es analizar palabras cortas de dos y tres letras, en las que es frecuente encontrar la I (por los artículos determinados) y la s (formas plurales, partículas reflexivas...). Encontramos las siguientes:

UYC; eC; VaC; C4 y varios finales de palabra con C
aU; Ua; eU; UaC
que hacen pensar en que U = I y C = s:

SBeY AFe FWY Ie IYs VYDQGYs Vas ZYIeBYsYs AFe SYWIFSeW al ÑYVPBe al aBDe K a la SQeWSQa es el IeseY Ie eGaIQBse Ie la GQIa SYDQIQaWa SYW sF asZeBeMa IYIYBYsa K sF GaSQY IesesZeBaWDe Ie esSaZaB a las SaIeWas Ie IeseYs sQeVZBe SaVPQaWDes aIPeBD eQWsDeQW

Volviendo a la tabla de frecuencias, vemos que entre las más frecuentes tenemos la o del texto plano y la Y del cifrado. Al sustituir, varias palabras parecen confirmarlo:

SBeo AFe FWo Ie los VoDQGos Vas ZoIeBosos AFe SoWIFSeW al ÑoVPBe al aBDe K a la SQeWSQa es el Ieseo Ie eGaIQBse Ie la GQIa SoDQIQaWa SoW sF

asZeBeMa IoloBosa K sF GaSQo IesesZeBaWDe le esSaZaB a las SaIeWas le Ieseos sQeVZBe SaVPQaWDes alPeBD eQWsDeQW

Además, la I aparece como candidata para ser la d y la B como r:
SBeo AFe FWo le los VoDQGos Vas ZoIeBosos AFe SoWIFSeW al ÑoVPBe al aBDe K a la SQeWSQa es el Ieseo le eGaIQBse le la GQIa SoDQIQaWa SoW sF asZeBeMa IoloBosa K sF GaSQo IesesZeBaWDe le esSaZaB a las SaIeWas le Ieseos sQeVZBe SaVPQaWDes alPeBD eQWsDeQW

Quedaría:

Sreo AFe FWo de los VoDQGos Vas Zoderosos AFe SoWdFSeW al ÑoVPre al arDe K a la SQeWSQa es el deseo de eGadQrse de la GQda SoDQdQaWa SoW sF asZereMa dolorosa K sF GaSQo desesZeraWDe de esSaZar a las SadeWas de deseos sQeVZre SaVPQaWDes alPerD eQWsDeQW
Esto ya casi está: se ve que la Z es la p y que la S la c:

Sreo AFe FWo de los VoDQGos Vas Zoderosos AFe SoWdFSeW al ÑoVPre al arDe K a la SQeWSQa es el deseo de eGadQrse de la GQda SoDQdQaWa SoW sF asZereMa dolorosa K sF GaSQo desesZeraWDe de esSaZar a las SadeWas de deseos sQeVZre SaVPQaWDes alPerD eQWsDeQW

Quedaría:

creo AFe FWo de los VoDQGos Vas poderosos AFe coWdFceW al ÑoVPre al arDe K a la cQeWcQa es el deseo de eGadQrse de la GQda coDQdQaWa coW sF aspereMa dolorosa K sF GacQo desesperaWDe de escapar a las cadeWas de deseos sQeVpre caVPQaWDes alPerD eQWsDeQW

Lo que tenemos hasta ahora es los siguiente:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
E		S	I	L							U				Y	Z		B	C							

Si el criptoanalista me conociese sabría que puedo ser lo suficientemente torpe como para elegir la palabra EPSILON como clave. En tal caso completar la tabla sería trivial y el texto quedaría descifrado.

De todas maneras, tal intuición no nos es necesaria: basta mirar el texto para deducir las nuevas letras q y u de la reveladora secuencia AF, así como que W = n. Sustituyendo:

creo que uno de los VoDQGos Vas poderosos que conducen al ÑoVPre al arDe K a la cQencQa es el deseo de eGadQrse de la GQda coDQdQana con su aspereMa dolorosa K su GacQo desesperanDe de escapar a las cadenas de deseos sQeVpre caVPQanDes alPerD eQnsDeQn

Resumiendo lo obtenido hasta ahora, tenemos:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
E		S	I	L							U		W		Y	Z	A	B	C		F					

Observando la tabla de cifrado, la cosa parece bastante clara, y podemos deducir nuevas letras:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
E		S	I	L							U	V	W	X	Y	Z	A	B	C	D	F					

que dan:

creo que uno de los motQGos mas poderosos que conducen al ÑomPre al arte K a la cQencQa es el deseo de eGadQrse de la GQda cotQdQana con su aspereMa dolorosa K su GacQo desesperante de escapar a las cadenas de deseos sQempre camPQantes alPert eQnsteQn

Del texto anterior salta a la vista que Q = i. Además, si nos fijamos en las dos últimas palabras, parece claro que un viejo conocido nos dice que P = b:

creo que uno de los motiGos mas poderosos que conducen al Ñombre al arte K a la ciencia es el deseo de eGadirse de la Gida cotidiana con su aspereMa dolorosa K su Gacio desesperante de escapar a las cadenas de deseos siempre cambiantes Albert Einstein

El resto de letras es evidente. Terminamos:

creo que uno de los motivos mas poderosos que conducen al hombre al arte y a la ciencia es el deseo de evadirse de la vida cotidiana con su aspereza dolorosa y su vació desesperante de escapar a las cadenas de deseos siempre cambiantes Albert Einstein

La tabla de cifrado quedaría entonces así:

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

siendo la clave, efectivamente, EPSILON.

Capítulo 5:
Método de Sustitución Polialfabética

Explicación de los métodos de sustitución polialfabética, con énfasis en el cifrado de Vigénere, desarrollando un ejemplo para entender este método. También se implementa la forma de proceder para su criptoanálisis.

5.1 El criptosistema de Vigenère

El sistema de cifrado de Vigenère (en honor al criptógrafo francés del mismo nombre) es un sistema polialfabético o de sustitución múltiple. Este tipo de criptosistemas aparecieron para sustituir a los monoalfabéticos o de sustitución simple, basados en el Cesar, que presentaban ciertas debilidades frente al ataque de los criptoanalistas relativas a la frecuencia de aparición de elementos del alfabeto.

El principal elemento de este sistema es la llamada Tabla de Vigenère, una matriz de caracteres cuadrada que se muestra en la figura 8 (ver final del capítulo). La clave del sistema de cifrado de Vigenère es una palabra donde cada uno de las letras que la forman indica el alfabeto a utilizar correspondiente, donde cada uno de estos alfabetos se utilizaran de manera secuencial.

Veamos un ejemplo de aplicación del criptosistema de Vigenère: queremos codificar la frase *La abrumadora soledad del programador* utilizando la clave *prueba*. En primer lugar, nos fijamos en la longitud de la clave: es de seis caracteres, por lo que descomponemos la frase en bloques de longitud seis (aunque el último bloque es de longitud tres, esto no afecta para nada al proceso de cifrado):

laabru madora soleda ddelpr ograma dor

Ahora, aplicamos a cada bloque la clave *prueba* y buscamos los resultados como entradas de la tabla de Vigenère:

```
laabru    madora    soleda    ddelpr    ograma    dor
prueba    prueba    prueba    prueba    prueba    pru
arufsu    brxhsa    igfiea    suyoqr    exmena    sgm
```

Por ejemplo, la primera *“l”* del texto en claro corresponde a la letra cifrada “A” de alfabeto p, de la tabla de Vigenère (Figura 8). Finalmente, vemos que el texto cifrado ha quedado *arufsu brxhsa igfiea suyoqr exmena sgm*.

Este método de cifrado polialfabético se consideraba invulnerable hasta que en el S.XIX se consiguieron descifrar algunos mensajes codificados con este sistema, mediante el estudio de la repetición de bloques de letras (la distancia entre un bloque y su repetición suele ser múltiplo de la palabra tomada como clave).

Una mejora sobre el cifrado de Vigenère fue introducida por el sistema de Vernam, utilizando una clave aleatoria de longitud igual a la del mensaje; la confianza en este nuevo criptosistema hizo que se utilizase en las comunicaciones confidenciales entre la Casa Blanca y el Kremlin, hasta, por lo menos, el año 1987.

Si la clave de Vigenère tiene mas de 6 caracteres distintos, se logra una distribución de frecuencias en el criptograma uniforme, es decir más o menos plana, por lo que se difumina la redundancia del lenguaje.

Aunque pudiera parecer que usando una clave larga y de muchos caracteres distintos y por tanto varios alfabetos de cifrado, Vigenère es un sistema de cifra seguro, esto es falso. La redundancia del lenguaje unido a técnicas de criptoanálisis muy sencillas, como los métodos de Kasiski y del Índice de Coincidencia, permiten romper la cifra y la clave de una manera muy fácil y con mínimos recursos. Veamos un ataque por el método de Kasiski.

5.2 Ataque por el método de Kasiski

El método de Kasiski consiste en buscar repeticiones de cadenas de caracteres en el criptograma. Si estas cadenas son mayores o iguales a tres caracteres y se repiten más de una vez, lo más probable es que esto se deba a cadenas típicas del texto en claro (trigramas, tetragramas, etc., muy comunes en las lenguas occidentales) que se han cifrado con una misma porción de la clave.

Si se detectan estas cadenas, la distancia entre las mismas será múltiplo de la longitud de la clave. Luego, el máximo común divisor entre esas cadenas es un candidato a ser la longitud de la clave, digamos L.

Dividimos el criptograma en L subcriptogramas que han sido cifrados por una misma letra de la clave. Finalmente, en cada subcriptograma hacemos un ataque simple ahora de tipo estadístico monoalfabético.

La idea es buscar ahora a través de los tres caracteres más frecuentes en cada subcriptograma las posiciones relativas de las letras A, E y O que en castellano están separadas por 4 y 11 espacios. La letra de la posición que ocupe la letra A (A=0) será entonces la letra clave correspondiente.

Veamos todo lo dicho con un ejemplo. Sea el siguiente criptograma de 404 caracteres que vamos a criptoanalizar :

PBVRQ VICAD SKAÑS DETSJ PSIED BGGMP SLRPW RÑPWY EDSDE ÑDRDP CRCPQ MNPWK UBZVS FNVRD MTIPW UEQVV CBOVN UEDIF QLONM WNUVR SEIKA ZYEAC EYEDS ETFPH LBHGU ÑESOM EHLBX VAEEP UÑELI SEVEF WHUNM CLPQP MBRRN BPVIÑ MTIBV VEÑID ANSJA MTJOK MDODS ELPWI UFOZM QMVNF OHASE SRJWR SFQCO TWVMB JGRPW VSUEX INQRS JEUEM GGRBD GNNIL AGSJI DSVSU EEINT GRUEE TFGGM PORDF OGTSS TOSEQ OÑTGR RYVLP WJIFW XOTGG RPQRR JSKET XRNBL ZETGG NEMUO TXJAT ORVJH RSFHV NUEJI BCHAS EHEUE UOTIE FFGYA TGGMP IKTBW UEÑEN IEEU.

Entre otras, se observan las siguientes cadenas (subrayadas) en el criptograma:
3 cadenas GGMP, separadas por 256 y 104 posiciones.
2 cadenas YEDS, separadas por 72 espacios.
2 cadenas HASE, separadas por 156 espacios.
2 cadenas VSUE, separadas por 32 espacios.

Luego el período de la clave puede ser mcd {256, 104, 72, 156, 32} = 4. La clave tendrá cuatro caracteres, por lo tanto tomaremos del criptograma el carácter 1º, el 5º, el 9º, etc. para formar el primer subcriptograma C_A; luego el 2º, el 6º, el 10º, etc. para formar el subcriptograma C_B, y así hasta el subcriptograma C_D.

C_A = PQAAEPDMRÑEEDCNUSRIECNIONSAAETLUOLAUIEULMNIIEAAOOLU
MNARSOMRSISERNAISIRTMDTOORLIORRENENOAVSNIAEOFAMTEI

C_B = BVDÑTSBPPPDÑPPPBFDPQBUFNUEZCDFBÑMBEÑSFNPBBÑBÑNMKDPFQF
SJFTBPUNJMBNGDUNUFPFSSÑRPFTPJTBTETTJFUBSUTFTPBÑE

C_C = VISSSIGSWWSDCQWZNMWVOEQMVIYESPHEEXEEEWMQRPMVIST
MSWOMOEWQWJWEQEGDISSETEGOOSETYWWGQSXLGMXOHHECEEIGGIWEE

C_D = RCKDJEGLRYDRRMKVVTUVVDLWRKEYEHGSHVPLVHCPRVTVDJJDEIZ
VHSRCVGVXRUGGLJVEGEGRGTQGVJXGRKRZGUJRRVJHHUEYGKUNU

La frecuencia relativa observada en cada uno de los subcriptogramas es:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
C_A	11	0	2	3	12	1	0	0	11	0	0	5	6	9	1	10	2	1	9	7	4	5	1	0	0	0	0
C_B	0	14	1	6	4	12	1	0	0	4	1	0	3	6	8	6	14	2	1	6	9	7	1	0	0	0	1
C_C	0	0	1	2	18	0	7	3	7	1	0	1	7	1	0	0	2	6	1	12	3	0	3	12	3	2	1
C_D	0	0	3	5	7	0	12	6	1	7	5	4	1	1	0	6	2	1	13	2	3	7	14	0	2	3	2

La letra más frecuente del subcriptograma debería corresponder a la letra E del texto en claro, la segunda a la letra A y la tercera a la letra O.

Pero para mejorar la efectividad del criptoanalisis se utiliza frecuentemente la regla AEO.

- Si la posición relativa de la letra A es el valor 0, entonces la letra E está cuatro espacios a la derecha de la A (m+4 mod 27) y la letra O está 15 espacios a la derecha de la letra A (m+15 mod 27).
- Buscaremos en cada subcriptograma C_i las tres letras más frecuentes y que cumplan además con esta distribución.
- Es suficiente contar con estas tres letras para que el ataque prospere. No obstante, podemos afinar más el ataque si tomamos en cuenta la siguiente letra frecuente en castellano (S) en posición (m+19) mod 27.

En el ejemplo para C_A se observa que la única solución que cumple con esto es la que coincide la AEO (11, 12, 10) luego la letra clave sería la A. Para C_B elegimos BFP (14, 12, 14) por lo que la letra clave sería B. Para C_C elegimos EIS (18, 7, 12) por lo que la letra clave sería E. Para C_D elegimos RVG (13, 14, 12) por lo que la letra clave sería R.
La clave será k = ABER y m = “Para que la cosa no me sorprenda...”.

El índice de coincidencia IC

Cuando encontramos una longitud L de la clave por el método de Kasiski y rompemos el criptogama en L subcriptogamas, podemos comprobar que cada uno de ellos se trata efectivamente de un cifrado monoalfabético aplicando el concepto del índice de coincidencia IC.

$IC=\sum_{i=0}^{26} p_i^{\,2}$ para castellano mod 27: $IC = p_A^2 + p_B^2 + ... + p_Z^2 = 0{,}072$.

Aunque el estudio de este índice IC queda fuera del contexto de estos apuntes, como para el castellano mod 27 el IC = 0,072, en el ataque de Kasiski se comprueba que para cada subcriptograma su IC esté cercano a este valor. Si el IC es menor que 0,5 es muy probable que no estemos ante un cifrador monoalfabético sino uno polialfabético de periodo 2 o mayor.

En el ejemplo anterior, una vez roto el criptograma en cuatro tenemos: $IC_{C_A} = 0{,}070$; $IC_{C_B} = 0{,}073$; $IC_{C_C} = 0{,}075$; $IC_{C_D} = 0{,}065$.

	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

Figura 9. Tabla de sustitución de Vigénere.

Capítulo 6:
Método de Transposición

Explicación de los métodos de transposición, con énfasis en el cifrado de transposición a través de una matriz, desarrollando un ejemplo para entender este método.

Los Algoritmos de transposición consisten en la permutación de los símbolos del texto en claro de forma que desordena el orden lógico del texto. Si un mensaje consta de *n* letras se podrá transponer de *n!* formas. La reordenación se puede realizar desde un modo simple: escribiendo el mensaje letra a letra pero al revés, o utilizando complicados esquemas matriciales. Existen varios tipos de transposición:

6.1 Cifrado de transposición por grupos

Es periódico y consiste en aplicar una permutación ∏ a un grupo de caracteres del texto en claro.

∏ = 2, 4, 3, 1 lo que significa que la primera letra se permutará con la segunda, la segunda con la cuarta, la tercera no permuta y la cuarta con la primera.

Ejemplo:

Agrupamos en grupos de cuatro:

```
m = ATAC AMOS ALME DIOD IAZZ

c = TCAA MSOA LEMA IDOD AZZI
```

Para el descifrado se utiliza una permutación inversa (4, 1, 3, 2). Este método tiene como debilidad que es un sistema periódico.

6.2 Cifrado de transposición por series

Consiste en ordenar el mensaje como una cadena de submensajes, en donde cada una de las cadenas sigue una función serie.

Ejemplo:

```
Serie A: 1 3 5 7 9 …   (impares)
Serie B: 2 4 6 8 10…   (pares)

m = ATACAMOSALMEDIODIA

c = AAAOAMDOIA  TCMSMDOI
```

6.3 Cifrado de transposición por columnas

Este método consiste en colocar el texto original a cifrar en una matriz de N columnas, rellenamos la matriz por filas y a la hora de escribir el texto cifrado leemos la matriz por columnas.

```
Ejemplo:

m= NOS VAMOS PRONTO
n= 4    (nº de columnas)

N O S V
A M O S
P R O N
T O Z Z


C= NAPTOMROSOOZVSNZ
```

Para descifrar el método basta con contar el número de letras del mensaje y dividir por N. El resultado indicará la extensión de la columna y se lleva a cabo la misma operación que para cifrar.

Otra alternativa consiste en que cada columna se le asigna una de las letras que componen la palabra clave. Posteriormente las columnas se ordenan alfabéticamente.

Con el mismo ejemplo anterior tenemos que:

```
Ejemplo:

m = NOS VAMOS PRONTO
n = 4
Palabra clave: Hola
H O L A     A H L O
= = = =     = = = =
N O S V     V N S O
A M O S     S P O M
P R O N     N A O R
T O Z Z     Z T Z O


C= VSNZNPATSOOZOMRO
```

Capítulo 7: Máquina Enigma

Dentro del gran campo de la criptografía la Enigma marca el punto de inflexión entre la criptografía clásica y la moderna, entre la de antes y la de después de la existencia del ordenador. Este es el método de cifrado que se pudo hacerse con una maquina que utilizaba la corriente eléctrica pero con principios de funcionamiento mecánicos. El uso de esta maquina tuvo gran importancia en la Segunda guerra mundial, y después ha tenido una gran repercusión en la tecnología. Su existencia produjo avances decisivos en la tecnología que, evidentemente con muchos cambios, se han convertido en imprescindibles en la actualidad, como los ordenadores.

7.1 Orígenes

El principio en el que se basa la máquina Enigma es muy antiguo. Máquinas mecánicas para cifrar mensajes basadas en anillos y cilindros, el mismo principio que utiliza Enigma, ya fueron descritas por un romano llamado Aeneas Tacitus en el siglo IV a.C. Thomas Jefferson también inventó una máquina basada en unos anillos que rotaban alrededor de un eje común.

Pero la máquina Enigma fue patentada por un inventor alemán llamado Arthur Scherbius en 1918. Era una máquina electromecánica que tenía como finalidad facilitar la comunicación de documentos entre comerciantes y hombres de negocios de forma secreta. Su fácil utilización, pues era como la de una máquina de escribir, y la seguridad que suponía el cifrado con ella, la pusieron muy por delante de los métodos utilizados hasta entonces tanto civiles como militares.

Scherbius no tuvo un gran éxito comercial y decidió ofrecer su idea a los militares. El ejército al principio desestimó el invento de Scherbius, que les ofreció distintas versiones de la máquina, una con siete rotores (6 billones de combinaciones) o una con treinta rotores (100 trillones de combinaciones). Scherbius hizo cálculos aproximados de que incluso si el enemigo llegara a poseer máquinas de ocho rotores y también mensajes originales y sus equivalentes cifrados, requeriría del trabajo de 1000 operadores trabajando 24 horas al día durante 14 años y medio para encontrar la clave.

Scherbius y sus asociados continuaron mejorando su invento. Los rotores fueron modificados, podían extraerse de la máquina y podían cambiarse de orden. También añadieron un nuevo anillo a cada rotor que se podía ponerse en cualquiera de las 26 posiciones (una por cada letra) y que estaban marcados con números del 1 al 26 (en el modelo utilizado por la marina alemana) o las 26 letras del alfabeto.

Con estas modificaciones el ejército cambió su primera idea de rechazar la nueva máquina de Scherbius y Enigma se convirtió, a partir de 1923, en el canal de comunicación de las informaciones secretas y el espionaje del ejército alemán, con gran importancia sobretodo durante la Segunda Guerra Mundial. Por ejemplo, tuvo gran importancia en la Batalla del Atlántico, en la que los submarinos alemanes iban equipados con una Enigma para comunicarse información sobre la posición de los barcos con destino a Gran Bretaña y que hundían con facilidad.

Los alemanes vieron en la Enigma un código indescifrable aunque el enemigo fuera capaz de capturar una máquina y conocer su funcionamiento.

Hubo diversas variaciones en las máquinas Enigma y en los procedimientos de utilización a lo largo de la Segunda Guerra Mundial. Por ejemplo, se introdujo la utilización de 2 rotores más a los tres que ya se utilizaban y también, se introdujo una versión de la máquina con cuatro rotores en 1942.

7.2 Partes y Funcionamiento

La máquina Enigma constaba de:

1. Un teclado de 26 letras (como el de una máquina de escribir)
2. “Lamp board”o Tablero Luminoso. Un frontal con 26 bombillas, una para cada letra.
3. ”Steckerboard”, un frontal donde se podían hacer hasta 13 conexiones para emparejar letras mediante unas clavijas.
4. Reflector

La utilización era sencilla. Después de que el operador dispusiera la configuración inicial (posición inicial y orden de los rotores, conexiones del steckerboard, rotores y reflector utilizado), tecleaba el texto a cifrar y cada vez que una tecla era pulsada se iluminaba su equivalente letra en el texto cifrado. Entonces lo único que había que hacer era apuntar las letras que se iban iluminando y transmitir el mensaje.

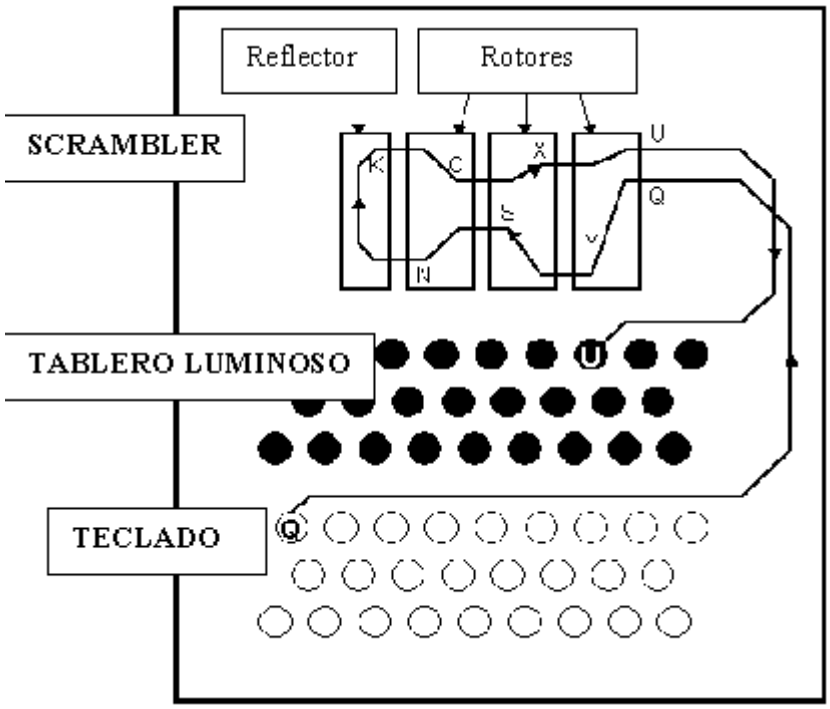


Figura 10. Grafico del cableado interno de la letra Q y para una combinación de rotores determinada.

Breve descripción de las partes de la maquina enigma

EL TECLADO

Solamente era un teclado en apariencia convencional, conectado a una serie de cables que hacían que una vez la máquina se conectaba al flujo eléctrico y se pulsaba una de las teclas se conectaba la corriente con uno de los cables de la entrada de la parte derecha del primer rotor.

LOS ROTORES

Los rotores eran unos discos circulares idénticos por fuera pero diferentes en su conexionado interior. Cada uno tenía 26 contactos (uno por cada letra) en cada lado y ambos lados estaban conectados mediante un cableado interno que era lo que los distinguía entre sí. En la máquina comercial solo había tres rotores que se podían intercambiar de posición.

Los rotores estaban numerados del I al III (más tarde los militares añadieron más, llegaron a existir 8, aunque solo se utilizaban 3 a la vez). Cada rotor tenía además unos discos o aros exteriores que servían para indicar la posición en la que se encontraba el rotor en ese momento mediante letras Variando la posición de estos se cambiaba la relación entre el cableado interior y la posición del rotor. La posición de los rotores no era siempre la misma, cambiaba cada vez que se pulsaba una tecla. El primer rotor giraba cada vez que se pulsaba una tecla, el segundo lo hacia cada vez que el primero completaba una vuelta, y el tercero cada vez que el segundo completaba la suya.

STECKERBOARD

Esta parte no existía en la máquina comercial diseñada por Scherbius, sino que fue añadida más tarde por los militares para hacer más complejo el cifrado y complicar aún más su lectura por parte del enemigo.

Se encontraba en la parte frontal de la máquina y consistía en una serie de 26 enchufes, uno para cada letra, que se podían conectar entre ellos (en parejas) hasta un máximo de 13 (habitualmente 6) conexiones posibles mediante unas clavijas. Se unían dos letras en cada conexión, entre el teclado y los rotores, lo que permitía intercambiar una letra por otra. Conectando esas letras antes de llegar a los rotores el número de posibles combinaciones de la máquina aumentó de forma considerable, dificultando el posible descifrado de los mensajes. El conexionado del steckerboard funcionaba en los dos sentidos, es decir, si por ejemplo se realizaba una conexión entre A y J la A pasaría a J, pero a su vez la J pasaría a A.

REFLECTOR

El reflector tampoco formaba parte de la máquina original. Se añadió para solucionar un problema como era la necesidad de tener dos claves diferentes para cifrar y descifrar el mismo mensaje, es decir diferentes configuraciones iniciales para cada función. Esto complicaba su uso y hacia que hubiera más posibles errores.

Lo que hacía era recibir la señal del último rotor después de haber pasado por todos y devolvérsela o reflejarla para que hiciera el camino contrario empezando por un contacto diferente de la cara izquierda del último rotor por el cual había llegado la corriente. Existían dos reflectores diferentes entre los que se podía elegir. La invención del reflectoe aportó que el cifrado y el descifrado fuesen simétricos, de forma que tanto una como otra función se realizaban con la misma clave.

Pero el reflector fue, en gran medida, un error, porque, si bien hizo la utilización de la máquina más sencilla, también supuso que una letra cifrada nunca coincidiese consigo misma, reduciendo el número de posibilidades y haciendo más fácil el trabajo del enemigo que quería descifrar el mensaje.

TABLERO LUMINOSO

El Tablero Luminoso era donde aparecían las letras una vez cifradas, estaba formado por unas bombillas que se encendían cada vez que se pulsaba una tecla. La corriente, después de pasar por las posibles conexiones del steckerboard, rotores en ambas direcciones y reflector)llegaba al Tablero Luminoso formado por 26 bombillas y en el que se encendía una, que era la correspondiente letra cifrada.

Funcionamiento

Cuando se pulsaba una letra la corriente pasaba primero por el steckerboard. Supongamos que se pulsa la B y esta está intercambiada en el steckerboard por la A. La corriente pasaba de la B a la A y llegaba al primer rotor por el contacto correspondiente a esta letra en el lado derecho y su cableado interno la llevaba hasta un contacto diferente de su lado izquierdo, por ejemplo G. La corriente entonces pasaba de G a un contacto del lado derecho del segundo rotor y éste la llevaba de nuevo mediante el cableado interno hasta el otro lado ,dando la señal W. Desde este contacto pasaba hasta la cara derecha del tercer rotor y se volvía a repetir el proceso. Supongamos que la salida del tercer y ultimo rotor sera la C. Cuando la corriente salía por el lado izquierdo del último rotor entraba en lo que llamamos reflector, que nos la retornaba también al lado izquierdo del tercer rotor pero en un contacto diferente, por ejemplo S.

Entonces se repetía el mismo proceso pero de forma contraria, de izquierda a derecha y siguiendo un camino a través del cableado de los rotores (diferente al anterior) obteniendo en el lado derecho del primer rotor una T.

Finalmente la corriente llegaba de nuevo al steckerboard, que podía tener o no conexionada esa letra con otra. En nuestro ejemplo pongamos que la T estaba conectada con la O. En este caso la corriente llegaría a la bombilla del Tablero Luminoso correspondiente a la O, que sería la letra cifrada.

El operador de nuestro ejemplo solo tendría que apuntar la letra de la bombilla que se había encendido y seguir tecleando el resto del mensaje e ir apuntando las letras equivalentes en el texto cifrado para después poder transmitir el mensaje. En la práctica, normalmente, había un operador que tecleaba y otro que iba tomando nota de las letras que se iluminaban en el Tablero Luminoso.

Además hay que mencionar que los rotores no estaban siempre en la misma posición, si no que iban girando cada vez que se pulsaba una letra, de manera que si pulsábamos una misma letra varias veces seguidas, las letras equivalentes del texto cifrado no tendrían porque ser las mismas.

El giro de los rotores estaba normalmente determinado de forma que cada vez que se pulsaba una tecla el rotor situado más a la derecha (llamado rotor rápido) avanzaba una posición. Cuando este rotor había dado una vuelta completa giraba entonces el segundo rotor (rotor medio) una posición. Habiendo realizado el rotor rápido una nueva vuelta completa el rotor medio volvía a avanzar una posición y así sucesivamente. De esta manera el rotor medio iba avanzando y cuando éste completaba una vuelta el rotor situado más a la izquierda (rotor lento) avanzaba una posición. Finalmente si se llegaba a escribir un mensaje tan largo que ya hubiera realizado el rotor lento un giro completo avanzaban los tres rotores una posición, de manera que se situaban en una nueva posición inicial y se seguía el mismo proceso.

No obstante este tipo de giro aunque era el que se utilizaba habitualmente podía ser modificado. El momento de giro de cada rotor estaba determinado por unas muescas que se encontraban en el disco exterior de cada uno y que indicaban el momento en le que tenían que girar. Modificando la situación de estas muescas se podía cambiar cada cuantas teclas pulsadas tenían que girar los rotores.

También cabe decir que nuestro operador debería haber apuntado la configuración inicial de la máquina antes de empezar (posición inicial y orden de los rotores, conexiones del steckerboard, rotores y reflector utilizado).

Ejemplo simplificado de la codificación de La Enigma.

Por simplicidad se supone un alfabeto formado por 6 letras y 2 rotores:

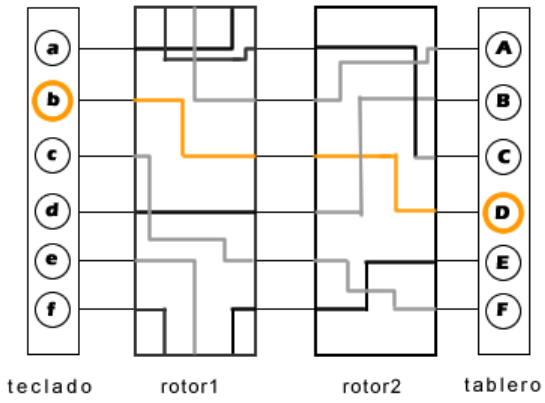


Figura11. Ejemplo simplificado de la codificación de la Enigma.

El cableado interno de los rotores es el que se muestra en la imagen. Por lo tanto cada rotor se encarga de realizar una sustitución monoalfabética.

Vamos a codificar la palabra **cafe**.

Para la primera vez que se pulse una tecla en el teclado se llevaran las siguientes sustituciones:

Alfabeto llano	A	B	C	D	E	F
Alfabeto cifrado 1	F	C	E	D	B	A
Alfabeto cifrado 2	C	A	D	B	F	E

Entonces la c será para la salida del primer rotor una E y la E para el segundo rotor será la F. Por lo tanto, c se convertirá en F.

Para la segunda letra el primer rotor girará 1/6. entonces el alfabeto cifrado 1 será distinto.

Alfabeto llano	A	B	C	D	E	F
Alfabeto cifrado 1	B	A	D	F	E	C
Alfabeto cifrado 2	C	A	D	B	F	E

Para este caso, la a será para la salida del primer rotor una B y la B para el segundo rotor será otra A. Por lo tanto, a se convertirá en A.

Y seguimos cifrando rotando el primer rotor por cada letra de manera que

cafe se cifrara como FAEE

7.3 Combinaciones

Las posibles combinaciones que ofrecía la máquina Enigma original eran bastante importantes. Como cada rotor tenía 26 posiciones y había tres rotores las combinaciones posibles serían 26 x 26 x 26 =17.576 posiciones iniciales.

Scherbius también sugirió unos posibles modelos con 8 rotores; lo que genera un total de $26^8 \approx 6$ billones de combinaciones. Y aún otro con 13 rotores y 100 trillones de combinaciones.

Cuando se hizo que los rotores se pudieran intercambiar de posición (en la Enigma militar de 3 rotores), estas 17.576 posiciones iniciales aumentaban, 17.576 x 6 (posibles combinaciones en la situación de los rotores) dando 105.456 posiciones posibles.

Además, se añadió un clavijero con 6 cables. El cable del clavijero permite intercambiar una letra por otra y hay 100.391.791.500 maneras de hacer 6 parejas de letras.

Por lo dicho el numero total de claves es **$26x26x26x6x100.391.791.500 = 10^6$**.

El 15 de Diciembre de 1938 los alemanes añadieron dos rotores más a los tres rotores disponibles, aunque solo se utilizaban tres a la vez. Las posibles combinaciones en la posición de los rotores aumentaban de 6 hasta:

$$\binom{5}{3}=\frac{5!}{(5-3)!}=\frac{5!}{2!}=60$$

Dando un resultado de 17.576 x 60 = 1.054.560. Y si a esto le añadimos la utilización del steckerboard, que podía soportar hasta un máximo de 13 conexiones entre el teclado y los rotores, la cifra de posibles combinaciones podía llegar desde 2 o 3 billones hasta unos asombrosos 10 cuatrillones según el número de conexiones utilizadas. Esto quiere decir que si 1000 operadores con máquinas que hubieran sido capturadas probaban 4 claves por minuto 24 horas al día, les hubiera llevado 900 millones de años probarlas todas.

Todo esto fue lo que convenció a los alemanes de que su código Enigma era totalmente indescifrable. Pero se equivocaron.

A finales de 1942 introdujeron una máquina que utilizaba cuatro rotores a la vez, que aumentaba, lógicamente, el número de combinaciones posibles, pero aún así no les fue posible mantener su código en secreto, ya era demasiado tarde, los aliados ya habían avanzado mucho sus conocimientos de cómo descifrar la máquina Enigma.

Ejemplo de cifrado

Ahora vamos a ver un ejemplo de un mensaje cifrado con Enigma. En este caso se trata de un mensaje real que data del 21 de Septiembre de 1938.

Es una comunicación codificada entre el alto mando militar alemán y sus tropas situadas en el frente. En la orden se recuerda la necesidad de defender y conservar las posiciones a toda costa, aunque el enemigo (los franceses) atacara con fuerzas mayores en número.

Este mensaje fue interceptado y descifrado por los servicios de inteligencia polacos. Se incluye en el ejemplo el texto original en alemán descifrado y su traducción al castellano.

Este es la comunicación original tal como la interceptaron los polacos:

```
Fernschreiben H.F.M.No. 563

+ HRKM 13617 1807 -
AN HEERESGRUPPENKOMMANDO 2= 06 18 24
2109 -1750 - 3 TLE
```

```
1TL -172= 01 07 09 (en lápiz; AGI)
HCALN UQKRQ WUQTZ KFXZO MJFOY RHYZW VBXYS IWMMV WBLEB
DMWUW BTVHM RFLKS DCCEX IYPAH RMPZI OVBBR VLNHZ UPOSY EIPWJ
TUGYO SLAOX RHKVC HQOSV DTRBP DJEUK SBBXH TYGVH GFICA CVGUV
OQFAQ WBKXZ JSQJF ZPEVJ RO -

2TL - 166 - 25 02 05 (en lápiz; YBE)
ZZWTV SYBDO DMVWQ KWJPZ OCZJW XOFWP XWGAR KLRLX TOFCD SZHEV
INQWI NRMBS QPTCK LKCQR MTYVG UQODM EIEUT VSQFI MWORP RPLHG
XKMCM PASOM YRORP CVICA HUEAF BZNVR VZWXX MTWOE GIEBS ZZQIU
JAPGN FJXDK I -

3TL - 176 - 12 21 14 (en lápiz; LUN)

DHHAO FWQQM BMHTT YFBHK YYXJK IXKDF RTSHB HLUEJ MFLAC
ZRJDL CJZVK HFBYL GFSEW NRSGS KHLFW JKLLZ TFMWD QDQQV JUTJS
VPRDE MUVPM BPBXX USOPG IVHFC ISGPY IYKST VQUIO CAVCW AKEQQ
EFRVM XSLQC FPFTF SPIIU ENLUW O = 1 ABT GEN ST D H NR.
2050/38 G KDOS +
```

Nota:
Las posiciones de inicio de los rotores, 01 07 09 en el caso de la parte 1, no formaban parte del mensaje original, pero fueron añadidas en lápiz como una forma de ayudar en el proceso de desciframiento del mensaje. He añadido además las correspondencias en letras a estas posiciones de inicio, 01 07 09 corresponde a AGI.

DESCIFRADO:

```
Parte I:

AUF BEFEHL DES OBERSTEN BEFEHLSHABERS SIND IM
FALLE X Z X ZT X UNWAHRSCHEINLICHEN X FRANZOESISQEN
ANGRIFFS DIE WESTBEFESTIGUNGEN JEDER ZAHLENMAESSIGEN
UEBERLEGENHEIT ZUM TROTZ ZU HALTEN X

Parte II:

FUEHRUNG UND TRUPPE MUESSEN VON DIESER EHRENPFLIQT
DURQDRUNGEN SEIN X ABS X DEM GEMAESS BEHALTE IQ
MIR DIE ERMAEQTIGUNG ZUR PUFGABE DER BEFESTIGUNGEN
ODER AUQ VON TEILEN AUSDRUECKLIQ

Parte III:


PERSOENLIQ VOR X ABS X AENDERUNG DER ANWEISUNG
X OKH X GEN X ST X D X H X ERSTE ABT X NR X DREI DREI
ZWO EINS X DREI AQT G X KDOS X VOM JULI EINS NEUN DREI AQT
BLEIBT VORBEHALTEN X DER OBERBEFEHLSHABER DES HEERES
```

Y el mensaje original completo es:

```
AUF BEFEHL DES OBERSTEN BEFEHLSHABERS SIND IM
FALLE X Z X ZT X UNWAHRSCHEINLICHEN X FRANZOESISQEN
ANGRIFFS DIE WESTBEFESTIGUNGEN JEDER ZAHLENMAESSIGEN
UEBERLEGENHEIT ZUM TROTZ ZU HALTEN X
FUEHRUNG UND TRUPPE MUESSEN VON DIESER EHRENPFLIQT
DURQDRUNGEN SEIN X
ABS X DEM GEMAESS BEHALTE IQ MIR DIE ERMAEQTIGUNG
ZUR PUFGABE DER BEFESTIGUNGEN ODER AUQ VON TEILEN
AUSDRUECKLIQ PERSOENLIQ VOR X
ABS X AENDERUNG DER ANWEISUNG X OKH X GEN X ST
X D X H X ERSTE ABT X NR X DREI DREI ZWO EINS
X DREI AQT G X KDOS X VOM JULI EINS NEUN DREI AQT
BLEIBT VORBEHALTEN X
DER OBERBEFEHLSHABER DES HEERES
```

Y el mensaje en alemán reescrito sera:

```
Auf Befehl des Obersten Befehlshabers sind im Falle,
(z.Zt =) zur Zeit unwahrscheinlichen, Franzoesischen Angriffs
die Westbefestigungen jeder zahlenmaessigen Ueberlegenheit
zum trotz zu halten.
Fuehrung und Truppe muessen von dieser Ehrenpflicht
durchdrungen sein.
Dem gemaess behalte ich mir die Ermaechtigung zur Aufgabe
der Befestigungen oder auch von Teilen ausdruecklich
persoenlich vor.
Aenderungen der Anweisung OKH/Gen/St/D/H Erste Abt Nr.
3321/38
G/KDos vom Juli 1938 bleibt vorbehalten.
Der Oberbefehlshaber des Heeres
```

Aquí mostramos el texto descifrado original, traducido al castellano:

```
Por orden del Comandante en Jefe:
En el caso de ataques franceses, improbables en estos
momentos, a las fortificaciones del oeste, éstas deben
defenderse a toda costa, aún cuando las fuerzas del enemigo
sean numéricamente superiores.
Comandantes y tropas deben estar imbuidos del honor de esta
tarea.
De acuerdo con lo anterior, sólo yo tengo el derecho a
autorizar que las fortificaciones se abandonen parcial o
totalmente.
Me reservo el derecho de hacer cambios a la orden
OKH/Gen/St/D/h
1. Abt. Nr. 3321/38 GKDos de Julio de 1938.
El Comandante en Jefe del Ejército.
```

Comentarios:

Este mensaje tiene fecha del 21 de Septiembre de 1938, e iba dirigido al Comando 2º del Ejército. Cada unidad en concreto tenía una "Clave Diaria" (*Tagesschlüssel*) , aunque el intervalo con el que eran cambiadas estas claves cambió a lo largo de la guerra.

La Clave Diaria contenía información de cómo la máquina debía de ser configurada:

1.Orden de los rotores (*Walzenlage*): II, I, III
2.Posición de los rotores (*Ringstellung*): ZWD
3.Conexiones del steckerboard (*Steckerverbindungen*): EZ, BL, WR, IU, VM, JO

La primera parte del mensaje estaba en texto sin cifrar y contenía el remitente, el receptor, fecha y hora. El mensaje podía estar estructurado en diversas partes, para hacer que el posible descifrado por parte enemiga fuera más difícil. Se suponía que los mensajes estaban limitados a menos de 200 palabras, aunque esto no siempre se siguió.

Entonces elegía tres letras al azar, AGI para el primer segmento de este mensaje en particular. Tecleaba estas tres letras (la "Clave del Mensaje", (*Spruchschlüssel*) dos veces y se transmitían cifradas. Lo siguiente que hacía era teclear el resto del mensaje.

Cabe insistir en que este método aunque estaba en uso a finales de 1938, fue variando a medida que la guerra avanzaba.

El doble cifrado de la clave del mensaje fue un grave error, ayudando al descifrado por parte de los polacos. Aún peor era la actuación de algunos operadores que elegían diagonales del teclado (QAY, AAA, etc.). Cuando no hacían esto ponían las iniciales de las novias o obscenidades abreviadas.

En este caso el preámbulo del mensaje, sin encriptar nos dice:
AN HEERESGRUPPENKOMMANDO 2=
2109 -1750 - 3 TLE - FRX FRX - 1TL -172=

La primera línea nos indica que el mensaje va dirigido al Comando 2 del Ejército.
Y la segunda:
-La fecha, 2109, 21 de Septiembre
-La hora, 1750, 17:50 pm
-Las partes que tiene el mensaje, 3 TLE, este mensaje tiene tres partes.

Antes de cada parrafo se escribia
-Las letras del segmento del mensaje,numero de segmento 1TL-172, tiene 172 caracteres.

Las primeras seis letras, HCA LNU son la clave del mensaje cifradas dos veces. Entonces el operador llevaba la máquina hasta la posición de los rotores (correspondiente a la clave del dia) y tecleaba HCA LNU y veía como las bombillas AGI AGI se encendían una después de otra. En ese momento cambiaba los rotores hasta la posición AGI y descifraba el resto de la primera parte del mensaje.

La segunda parte empieza con 2TL-166 sin descifrar, que nos indica que la segunda parte del mensaje tiene 166 letras. El operador llevaba los rotores de nuevo hasta la posición FRX y descifraba ZZW TVS como YBE YBE, y llevando los rotores hasta esta nueva posición YBE descifraba la segunda parte del mensaje. Para la tercera parte el proceso era el mismo de forma que el mensaje separado en palabras quedaba así:

AUF BEFEHL DES OBERSTEN BEFEHLSHABERS SIND IM
FALLE X Z X ZT X UNWAHRSCHEINLICHEN X FRANZOESISQEN
ANGRIFFS DIE WESTBEFESTIGUNGEN JEDER ZAHLENMAESSIGEN
UEBERLEGENHEIT ZUM TROTZ ZU HALTEN X
FUEHRUNG UND TRUPPE MUESSEN VON DIESER EHRENPFLIQT
DURQDRUNGEN SEIN X
ABS X DEM GEMAESS BEHALTE IQ MIR DIE ERMAEQTIGUNG
ZUR PUFGABE DER BEFESTIGUNGEN ODER AUQ VON TEILEN
AUSDRUECKLIQ PERSOENLIQ VOR X
ABS X AENDERUNG DER ANWEISUNG X OKH X GEN X ST
X D X H X ERSTE ABT X NR X DREI DREI ZWO EINS
X DREI AQT G X KDOS X VOM JULI EINS NEUN DREI AQT
BLEIBT VORBEHALTEN X DER OBERBEFEHLSHABER DES HEERES

En este mensaje se observan ciertas convenciones:

X=Se utiliza para dividir periodos o palabras.
ABS=Indica un nuevo párrafo.
Q=Se utiliza en lugar de CH
ZXZT(z.Zt)=zur Zeit, es una abreviatura común en alemán y significa “en este momento”.

Aplicando estas correcciones se obtiene una versión final del mensaje en alemán, que se encuentra al principio del capítulo (MENSAJE ALEMÁN REESCRITO)

Cabe destacar que la máquina Enigma sólo tenía 26 letras de forma que los números debían de ser escritos, en este caso 3321=DREI DREI ZWO EINS. Pero como a mayor longitud del mensaje más facilidad para ser descifrado, en algunas máquinas Enigma de la marina las letras de la fila superior estaban etiquetadas con los números 1/Q, 2/W, ..., O/9, P/0. Para indicar el final y el principio de un número se utilizaba la letra Y. Con este método en uso, 3321 sería YEEWQY, de esta forma se ahorrarían 9 caracteres.

7.5 El desciframiento de La Enigma

Introducción

Los criptoanalistas polacos fueron capaces de descifrar La Enigma antes del inicio de la segunda guerra europea.

Pocos meses antes de empezar la contienda, los alemanes reforzaron la seguridad de La Enigma lo que produjo que Polonia no pudiera afrontar el nuevo reto. Además fue invadida por el ejército alemán el 1 de Septiembre del 39.

Durante el transcurso de la guerra, los criptoanalistas británicos ayudados por la información que les proporcionaron los polacos, consiguieron mejorar La Nueva Enigma mejorada.

Los factores que determinaron el desciframiento de la máquina fueron el miedo a ser invadidos, el espionaje, errores cometidos por los alemanes al codificar el mensaje y la incorporación de criptoanalistas con conocimientos en matemáticas.

El Biuro Szyfrów

En 1926, las potencias europeas empezaron a interceptar cifras completamente desconcertantes por la utilización de La Enigma. Cuando los aliados perdían las esperanzas por descifrar los mensajes los polacos crearon una nueva oficina de cifras llamada el Biuro Szyfrów.

Lo que forzó a Polonia al criptoanálisis de La Enigma es el factor miedo, ya que era presionada por la URRS, que quería extender el comunismo, y por los alemanes al sur que querían recuperar los terrenos que habían cedido a Polonia tras la I guerra europea.

El capitán Maksymilian Ciezki fue nombrado responsable del Biuro Szyfrów. El ya conocía el funcionamiento de La Enigma comercial, pero la versión militar tenía un cableado de los rotores distinto.

Los franceses habían firmado un tratado de cooperación militar con Polonia. Cumpliendo con este acuerdo, entregaron al Biuro Szyfrów unos documentos secretos, obtenidos mediante espionaje, con los planos completos de la Enigma. Los polacos lograron sacarle provecho a esta información.

Además de revelar los cableados internos de los modificadores, los documentos también explicaban el diseño de los libros de códigos utilizados por los alemanes. Cada mes, los operadores de La Enigma recibían un nuevo libro de códigos que especificaban las claves que debían usarse para cada día:

Posición del clavijero: (ejemplo A/L - P/R - T/D - B/W - K/F - O/Y)
Disposición de los modificadores : (2-3-1)
Orientación de los modificadores: Q-C-Q

Todos los mensajes de un mismo día se codificaban con una misma clave. Los alemanes decidieron mejorar la seguridad de las comunicaciones utilizando la clave del día para transmitir la clave del mensaje. La clave del mensaje era idéntica a la clave del día pero con una orientación diferente de los rotores.

La clave del mensaje era codificada dos veces al principio de cada mensaje. Los polacos se aprovecharon de la repetición de la clave del mensaje para descifrar la Enigma. Cada día se estudiaba detenidamente las 6 primeras letras de todos los mensajes interceptados. Por ejemplo podía recibir cuatro mensajes que comenzaban:

	1ª	2ª	3ª	4ª	5ª	6ª	
mensaje 1	L	O	K	R	G	M	……...
mensaje 2	M	V	T	X	Z	E	……...
mensaje 3	J	K	T	M	P	E	……...
mensaje 4	D	V	Y	P	Z	X	……...

Se sabía que la 1ª y la 4ª letra de los mensajes codificaban la misma letra del alfabeto llano. Lo mismo ocurría entre la 2ª y la 5ª y entre la 3ª y la 6ª.

Si en un mismo día se interceptaban muchos mensajes, se podían llegar a establecer las relaciones existentes entre la 1ª y la 4ª letra de todas las letras del alfabeto alemán. De la misma manera, se establecían las relaciones entre la 2ª y la 5ª y entre la 3ª y la 6ª.

Por ejemplo: Para la 1ª Y 4ª letras

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

Los polacos no tenían ni idea de la clave del día, así como tampoco de las claves del mensaje elegidas, pero sabia que daban como resultado las tablas de relaciones. Si la clave del día era diferente, sabían que las tablas de relaciones también serían diferentes.

Se define el concepto de cadena como *ciclos cerrados de relaciones entre los tres grupos de letras de las seis primeras letras del alfabeto cifrado*:

26 cadenas de la 1ª y la 4ª letras del texto cifrado:

A ► F ► w ► A	3 conexiones
B ► Q ► Z ► K ► V ► E ► L ► R ► I ► B	9 conexiones
C ► H ► G ► O ► Y ► D ► P ► C	7 conexiones
D ► P ► C ► H ► G ► O ► Y ► D	7 conexiones
…….	

Utilizando las otras dos tablas se construyen las otras dos tablas similares. Las cadenas cambiaban cada día, tanto las letras que le suponen como su longitud.

Nótese que el número de conexiones depende únicamente de la posición de los rotores pero no de la posición del clavijero .Por ejemplo si el clavijero intercambia la K con la H las cadenas resultarían:

A ► F ► W ► A	3 conexiones
B ► Q ► Z ► **H** ► V ► E ► L ► R ► I ► B	9 conexiones
C ► **K** ► G ► O ► Y ► D ► P ► C	7 conexiones
D ► P ► C ► **K** ► G ► O ► Y ► D	7 conexiones

…….

El numero total de posiciones de los rotores es igual al numero de disposiciones de los rotores (6) multiplicado por el numero total de orientaciones (17.576), es decir, 105.456. De este modo, en vez de tener que preocuparse sobre cual de las 10^{16} claves del día se asocian con el juego de cadenas particular, bastará con averiguar cuál de las 105.456 posiciones de los modificadores se asocia con el numero de conexiones de un día concreto.

Su equipo comenzó la laboriosa tarea de probar cada una de las 105.456 posiciones de los modificadores, catalogando la longitud de las cadenas generadas por cada una de ellas. Les costo un año tener completo el catalogo. Gracias a ello el Biuro Szyfrów pudo finalmente descifrar la enigma.

Aun quedaba establecer la posición del clavijero. A pesar que había alrededor de cien mil millones de posibilidades, ésta era una tarea relativamente fácil:

Alliveinbelrin ► Arrive in Berlin ---------- R ► L

Los polacos inventaron una versión mecanizada de su sistema de catalogación que podía buscar automáticamente las posiciones correctas de los modificadores. El invento es una adaptación de la máquina Enigma, capaz de probar rápidamente cada una de las 17.576 posiciones. Las unidades se llamaban bombas. De esta manera, si los alemanes modificaban ligeramente la manera de codificar los mensajes, se podía crear un nuevo catalogo en muy poco tiempo.

En 1938, los alemanes reforzaron la seguridad de la Enigma, añadiendo dos nuevos modificadores aunque simultáneamente sólo se utilizaban 3, lo que daba un total de 60 posiciones distintas) y utilizando un clavijero de 10 cables. Además la actividad de los espías franceses habían cesado.

El primer reto consistía en averiguar el cableado interno de los dos nuevos modificadores, pero lo que era todavía más preocupante era que también tenía que construir 60 bombas en lugar de 6, lo que superaba el presupuesto del Biuro Szyfrów.

La Enigma era el núcleo de la estrategia blitzkrieg de Hitler (guerra relámpago): la velocidad del ataque mediante la velocidad de las comunicaciones. El 27 de abril de 1939, Alemania se desdijo de su tratado de no agresión a Polonia. Langer decidió ofrecer todos los avances criptográficos a los aliados así como dos replicas de la enigma y los planos para construir las bombas. El 1 de septiembre, Alemania invade Polonia y empieza la II gran guerra Europea.

Bletchley Park

Al iniciarse la guerra, los responsables de la Sala 40 del Reino Unido decidieron contratar a matemáticos y científicos. Su lugar de trabajo fue Blechtley Park, en Buckinghamshire.

Hasta el 10 de mayo de 1940, los ingleses consiguieron muchos de los mensajes alemanes gracias a:

Hasta esta fecha, los alemanes siguieron utilizando la misma metodología de codificación: iniciar el mensaje cifrado con la clave del día repitiendo dos veces la clave del mensaje (aunque ahora la Enigma era algo mas compleja: 5 rotores y un clavijero con 10 cables). En consecuencia, los ingleses pudieron sacar provecho de la experiencia del Biuro Szyfrów.

Los alemanes cometieron graves errores de codificación: Uso indiscriminado del *cillis*: claves del mensaje obvias, como por ejemplo QWE.

El alto mando, tratando de asegurarse que las disposiciones de los modificadores fueran imprevisibles, no permitió que ningún modificador permaneciera en la misma posición dos días seguidos (si un día se utilizaba los modificadores 1-3-4, al día siguiente se podía utilizar 2-1-5 pero no el 2-1-4). Los criptoanalistas de Blechtley se dieron cuenta de ello, y consiguieron reducir a la mitad el número de pruebas a realizar en un mismo día.

De forma similar, tampoco permitieron que los cables del clavijero pudieran conectar una letra con sus vecinas en el alfabeto. De nuevo, el cumplimiento de esta regla redujo el numero de calves posibles.

Turing la figura mas destacada de Bletchley Park, revisando la biblioteca de mensajes cifrados de Bletchley, se dio cuenta de que, a veces, podía predecir parte del contenido de un mensaje sin descifrar basándose únicamente en el momento y el lugar en que fue enviado el mensaje.

Por ejemplo, la experiencia mostraba que los alemanes enviaban un parte meteorológico regular codificado todos los días poco después de las seis de la mañana. Por eso, un mensaje interceptado a las seis y cinco de la mañana contendría casi seguro la palabra watter (“tiempo en alemán”). Además, probablemente se trataban de las seis primeras letras del texto cifrado.

Turing define el concepto de **puntal:** combinación de letras del texto llano que puede asociarse con un fragmento del texto cifrado. Un mensaje interceptado a las 6 de la mañana empieza por ETJWPX y probablemente codifique la palabra watter. Una forma de encontrar la clave consiste en teclear la palabra watter utilizando cada una de las 150 x 10^{18} posibles claves, hasta que el resultado sea ETJWPX.

Para simplificar el problema, Turing decidió seguir la estrategia de Rejewski de separar los efectos del clavijero con el de los rotores. De esta forma, solo tendría que probar 1.054.560 posibilidades (60 x 17.576).

Turing se da cuenta de la presencia de rizos en algunos de los puentes.

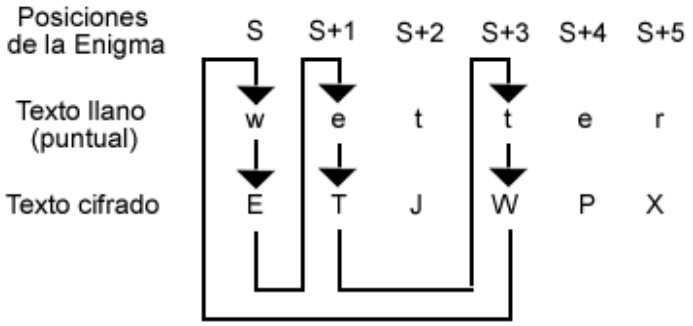


Figura 12. Ejemplo de puntal.

En la posición S, la Enigma codifica la “w” como E. En la posición S+1, la “e” como T y finalmente, en la posición S+3, la “t” como W.

Para automatizar el proceso Turing pensó en poner tres maquinas enigma trabajando en paralelo: La segunda maquina tendría sus modificadores un lugar por delante con respecto a la primera, la tercera maquina tendría sus modificadores tres lugares por delante y en la primera siempre se tecleará una w, en la segunda una e y en la tercera una t hasta que el resultado fuera ETW.

Todavía se tiene que probar 159 x 10 posiciones distintas de los modificadores.

A Turing también se le ocurrió una manera de anular el efecto del clavijero (ver figura adjunta). En el primer Enigma la corriente eléctrica entra en los modificadores y sale por alguna letra a la que llamaremos después de pasos por el clavijero L1 se transforma eb la letra **E**.

Esta letra **E** se conecta con un cable con la letra **e** de la segunda Enigma y cuando la corriente pasa por el segundo clavijero se vuelve a transformar en L1. En otras palabras, los dos clavijeros se han contrarrestado mutuamente.

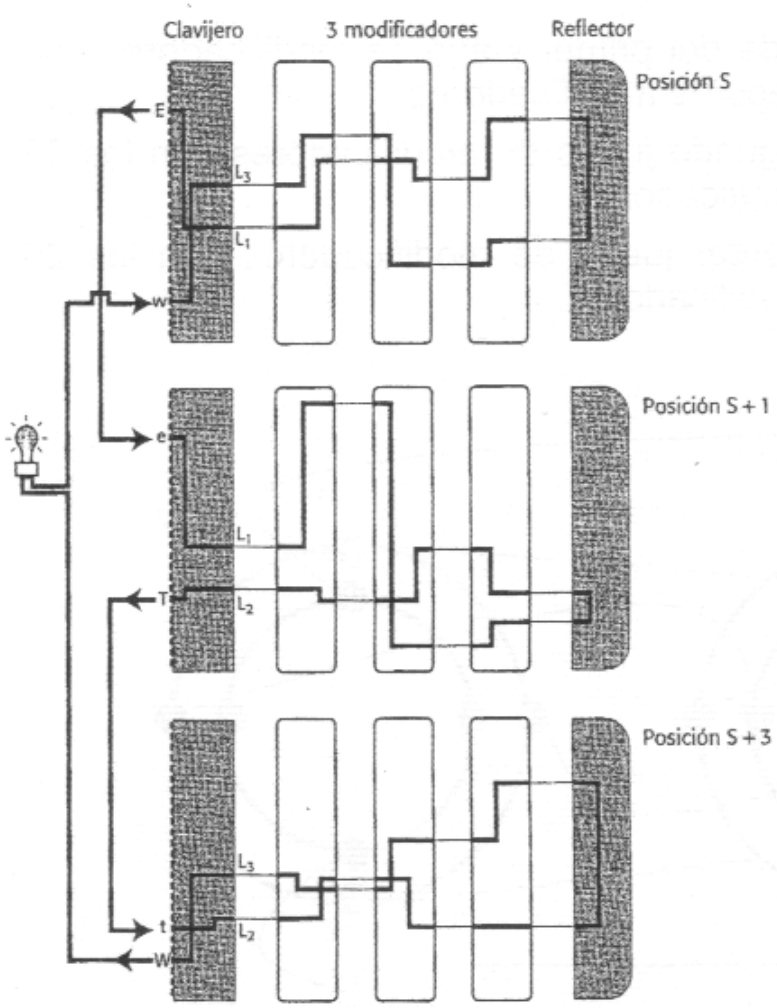


Figura 13. Ejemplo de conexión de tres máquinas Enigmas trabajando en paralelo.

De la misma forma, la corriente que sale de los modificadores de la segunda Enigma entra en el clavijero L2, antes de ser transformada en **T**. Esta letra **T** se conecta con la letra **t** de la tercera Enigma, y cuando la corriente pasa por el tercer clavijero, se vuelve a transformar en L2

En resumen: los clavijeros se compensaban mutuamente, de forma que Turing podía ignorarlos completamente.

Pero Turing desconocía el valor de L1, L2 Y L3, por ello tuvo que conectar los 26 dispositivos se salida del primer juego de modificadores con los 26 dispositivos de entrada del segundo juego de modificadores. También conectó los 26 dispositivos de salida del segundo juego de modificadores con los 26 dispositivos de entrada del tercer juego de modificadores y los 26 dispositivos de salida del tercer juego de modificadores con los 26 juegos de entrada del primer juego de modificadores. Cada uno de estos cables tenía una bombilla que se iluminaba con el paso de la corriente.

Se trataba de probar todas y cada una de las 17.576 posibles disposiciones de los modificadores. La complejidad del problema se ha reducido drásticamente (si

los modificadores cambiaban de orientación cada segundo solo se tardarían 5 horas en probar las combinaciones).

Aun faltaban por resolver un par de problemas. Como la Enigma opera con tres cualesquiera de los 5 modificadores disponibles, había que probar cada una de las 60 x 17.576 posibilidades (o construir 60 x 3 = 180 Enigmas trabajando en paralelo). Aun así, todavía faltaría por determinar la posición de los cables del clavijero, aunque ello no estrictamente necesario ya que este problema puede resolverse más fácilmente con una simple interpretación lingüística del texto descifrado. Además, los criptoanalistas de Bletchley se aprovecharón de una debilidad de la Enigma: una misma letra (por ejemplo, la a) nnunca se cifrará como ella misma (es decir, la a nunca generará una A en el texto cifrado).

Las bombas de Turing constaban de 12 juegos de modificadores conectados eléctricamente, lo que le permitía afrontar puntales de hasta 12 letras. La unidad completa tenia dos metros de altura, dos metros de longitud y un metro de anchura.

La primera bomba, bautizada con el nombre de “Victory”, llego a Bletchly Park el 14 de marzo de 1940, pero resulto ser mucho mas lenta de lo que se esperaba (tardaba una semana en encontrar la clave). Mejoraron el diseño y el 8 de agosto llego la nueva versión, a la que llamaron “Angus Dei”. En menos de 18 meses había ya quince bombas trabajando en paralelo. Si todo iba bien, la clave de la Enigma se podía encontrar en una hora .

Alan Turing y sus bombas fueron capaces de descifrar las claves del ejército alemán en Europa y África, junto con las de Lutwaffe. Sin embargo, la Kriegsmarine, marina alemana, utilizaba una Enigma mas sofisticada: poseía 8 modificadores (a elegir 3) y además, el reflector también giraba. Los operadores navales se cuidaban mucho de no mandar mensajes estereotipados, evitando la presencia de puntales.

El éxito de la estrategia alemana en el Atlántico se debía a la Enigma. En primer lugar, los submarinos rastreaban el océanos en busca de convoyes aliados. En cuanto uno de ellos divisaba un objetivo, llamaba al resto de submarinos, transmitiendo las coordenadas del lugar en un mensaje cifrado.

Los ingleses reaccionaron de dos maneras, una forzando a los alemanes a transmitir puntales. La RAF se encargaban de tirar minas sobre barcos alemanes, provocando que éstos transmitiera advertencias a otros barcos. Lógicamente, en estos casos se transmitirían las coordenadas del lugar en un mensaje cifrado. La segunda robando los libros de códigos de los barcos y submarinos capturados (pellizcos).

Después de la guerra, las proezas de Bletchley Park se mantuvieron en escrito silencio. De echo, el Reino Unido había capturado miles de maquinas Enigma y las distribuyo entre sus antiguas colonias. De esta manera, pudieron descifrar sus comunicaciones durante los años siguientes.

Todos los empleados de Bletchley Park tuvieron que jurar guardar escrito secreto sobre su actividad durante la guerra. Muchos de ellos tuvieron que sufrir adversidades de sus conciudadanos, pues les trataban de cobardes y traidores.

El secreto de Bletchly Park terminó en 1974, cuando el servicio secreto británico permitió que el capitán F. W. Winterbothnam publicara un libro titulado THE ULTRA SECRET.

Alan Turning no vivió los suficiente para recibir ningún reconocimiento público. En 1952, mientras denunciaba un robo a la policía, reveló ingenuamente que mantenía una relación homosexual. La policía lo detuvo y lo acusó de flagrante indecencia contraria a la sección 11 del Alta de Enmienda de la Ley Penal de 1885. Los periódicos informaron del juicio y Turning fue humillado públicamente. El gobierno británico le retiró su acreditación como miembro de seguridad. Se le prohibió trabajar en proyectos de investigación relacionados con el desarrollo del ordenador. Fue obligado a consultar un psiquiatra y tuvo que someterse a un tratamiento de hormonas, que le dejo impotente y obeso, Durante los dos años siguientes sufrió una gran depresión y el 7 de junio de 1954 se suicidó.

Capítulo 9: Guión de Practicas

Explicación de el software desarrollado a modo de guión de practicas con la documentación y la descripción de cada uno de los programas ejemplos, resueltos y propuestos para ayudar a la comprensión de los métodos que simulan.

ANEXO: HERRAMIENTAS PARA EL ANÁLISIS

Este anexo trata de explicar y documentar las herramientas que usaran durante las prácticas para poder descifrar los textos cifrados por los diferentes métodos.

ANALIZADOR DE FRECUENCIAS (frecuencias.c)

DESCRIPCIÓN
Programa que analiza las frecuencias en que aparecen los caracteres en los documentos de texto. El programa calcula número de caracteres válidos, número de cada uno de los 26 caracteres y la frecuencia, en %, en la que se presentan en el texto.

SINOPSIS
frecuencias [-f file name] [-v version] [-h help]

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar

DETECTOR DE SECUENCIAS REPETIDAS (cadenas_repetidas.c)

DESCRIPCIÓN
Programa que detecta secuencias de caracteres idénticas en el texto, indicando el nombre de la cadena que se repite la posición en la que se encuentra ésta y la distancia hasta la siguiente cadena.

SINOPSIS
cadenas_repetidas [-f file name] [-m number Char] [-v version] [-h help]

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar
`-m number`	numero mínimo de caracteres de las cadenas repetidas

UNIR Y SEPARAR TEXTOS (separar_textos)

DESCRIPCIÓN
Programa que divide un fichero de texto en varios o junta varios ficheros de texto en uno de la siguiente forma: extrae del fichero original los caracteres que están distanciados entre sí una cierta cantidad m. Útil para descifrar cifras de Vigénere.

.

SINOPSIS
separar_textos [-f file name] [-c | -p] [-v version][-h help]

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar
`-c`	indica que lo que se quiere es convertir un fichero en muchos
`-p`	indica que lo que se quiere es unir muchos ficheros en uno
`-n number`	numero para indicar la cantidad de ficheros que se quieren dividir el original o el numero de ficheros que van a formar un nuevo fichero.

CAMBIAR CARACTERES (cambiar_letras.c)

DESCRIPCIÓN
Cambia todos los caracteres del texto iguales al introducido por el indicado.

SINOPSIS

cambiar_letras [-f file_name] [-c char] [-d char] [-v version] [-h help]

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar
`-c char`	indicar el carácter del texto para cambiar
`-d char`	indicar el carácter por el que se quiere cambiar el anterior

1º MÉTODO DE SUSTITUCIÓN MONOALFABETICA

Esta práctica trata de ayudar a comprender el método clásico de cifrado por sustitución monoalfabética y conocer los ataques que se pueden hacer al texto cifrado.

DESCRIPCIÓN DEL MÉTODO

El método de sustitución monoalfabético consiste en sustituir cada letra del mensaje por otra que corresponde a otro alfabeto distinto.

EJEMPLOS:

-Relacionar pares de letras entre si y luego sustituir cada letra del mensaje original por su pareja.

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

Intercambiando la fila las letras de la fila de arriba por las letras de la fila de abajo tendremos:

```
mensaje original (M): ENCONTRÉMONOS A MEDIANOCHE
mensaje cifrado  (C): USMQSZLUCQSQN V CUXGVSQMBU
```

-Permutación del alfabeto por desplazamiento de n posiciones del original desplazamiento N: **3**

```
Alfabeto llano   M: 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
Alfabeto cifrado 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 C
```

-Permutación del alfabeto a través de una palabra clave.

```
Palabra clave: Criptografia
Alfabeto llano   M: 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
Alfabeto cifrado C: C R I P T O G A F H J K L M N Q S U V W X Y Z B D E
```

(nota: Las letras repetidas de la clave no se repiten en el alfabeto).

-Permutando el alfabeto original por una palabra clave situada en el medio del alfabeto.

La palabra clave situada en el medio del alfabeto se puede realizar con una permutación de desplazamiento del mensaje cifrado con la palabra clave.

```
Palabra clave: Criptografia
Desplazamiento N: 14
Alfabeto llano    M: 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
Alfabeto cifrado  C: C R I P T O G A F H J K L M N Q S U V W X Y Z B D E
Alfabeto cifrado C1: N Q S U V W X Y Z B D E C R I P T O G A F H J K L M
```

DESCRIPCIÓN DE LA APLICACIÓN

DESCRIPCIÓN
La aplicación proporciona la posibilidad de cifrar un texto de caracteres (mod26, de A a Z sin Ñ) con un método de cifrado por sustitución monoalfabética. Se introduce un clave para cifrar el texto y/o un desplazamiento, es decir se puede cifrar por permutación de las letras originales con una clave y/o un desplazamiento.

SINOPSIS
sust_monoalfabetica [-f file_name] [-m position] [-k key] [-v version] [-h help]

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar
`-c`	indicar que se quiere cifrar
`-d`	indicar que se quiere descifrar
`-k clave`	es la clave con la que se va a cifrar el texto.
`-m numero`	numero de desplazamientos que se van a cifrar

EJEMPLOS

Para el texto del fichero **"texto.txt"**:

"**Este texto esta cifrado por un método de sustitución monoalfabética"**

La salida a:
./sust_monoalfavetica -c -f texto.txt -k criptografía -m 14

será un fichero **"texto.cod"** con el siguiente contenido:

VGAVAVKAIVGANSZWONUIPIOFRCVAIUIUVGFGAZAFSZIRCIRINEWNQVAZSN

Para recuperar el texto original a partir de l texto cifrado habrá que decodificar el fichero **"texto.cod"** con:
./sust_monoalfabetica -d -f texto.cod -k criptografía -m 14

CRIPTOANÁLISIS DEL MÉTODO

Para descifrar un texto cifrado por un método de sustitución monoalfabética se utiliza el método de análisis frecuencia. En este método de sustitución cada letra siempre se cifra por una misma letra por lo que el método se hace vulnerable y fácil de atacar estadísticamente.

EJERCICIOS

EJERCICIO 1
Escribe un texto de unas pocas letras y codifícalo por el método de sustitución con distintas claves. Desplazárlo y visualizar las salidas. Comprobar sus frecuencias.

EJERCICIO 2
Dado **texto1.cod** un texto cifrado, descífralo dando el alfabeto en claro, la clave (si la tiene) y el texto completo descifrado.
¿Que tipo de cifrado es?

EJERCICIO 3
Dado **texto2.cod** un texto cifrado, descífralo dando el alfabeto en claro la clave (si la tiene) y el texto completo descifrado.
¿Que tipo de cifrado es?

EJERCICIO 3
Dado **texto3.cod** un texto cifrado, descífralo dando el alfabeto en claro la clave (si la tiene) y el texto completo descifrado.
¿Que tipo de cifrado es?

2º MÉTODO DE SUSTITUCIÓN POLIALFABÉTICA (Vigenère)

La intención de la práctica es intentar comprender el método de Vigenère así como la forma de proceder ante su criptoanálisis.

DESCRIPCIÓN DEL MÉTODO

Método surgido con el fin de evitar el descifrado de textos cifrados con un método monoalfabético. Vigénere propone utilizar 26 alfabetos para cifrar y los alfabetos se seleccionan por la palabra clave utilizada.

Ejemplo

	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

Tabla con los 26 alfabetos

Dada la palabra clave HIELO se utilizaran para cifrar los alfabetos correspondientes a la tabla con H, I, E, L, O

```
Clave: H I E L O H I E L O H I E L O H I E L O H I E
    m: t e x t o a c i f r a r p o r v i g i n e r e
    c: A M B E C H K M Q F H Z T Z F C Q K T B L Z I
```

La ventaja es que resulta inexpugnable para el análisis de frecuencia y tiene un gran número de claves.

DESCRIPCIÓN DE LA APLICACIÓN

DESCRIPCIÓN
La aplicación proporciona la posibilidad de cifrar un texto de por el método de Vigénere. Se introduce un texto a cifrar y un clave para cifrar el texto que indican los alfabetos a utilizar según la tabla de los 26 alfabetos de Vigénere.

SINOPSIS
vigenere [-c|-d] [-f file_name] [-k key] [-v version][-h help]

OPCIONES

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar
`-c`	indicar que se quiere cifrar
`-d`	indicar que se quiere descifrar
`-k clave`	es la clave con la que se va a cifrar el texto, Es decir el numero (numero de caracteres de la clave) y los alfabetos con los que se van a cifrar el texto por sustitución polialfabética, de manera que si la clave es HOLA se usaran los alfabetos con desplazamiento de tantas posiciones como indica el carácter según la tabla de cifrado.

EJEMPLOS

Para el texto del fichero **“texto.txt”**:

“**Este texto esta cifrado por el método de sustitución polialfabética de Vigénere.”**

La salida a
./vigenere -c -f texto.txt -k hielo

será un fichero **“texto.cod”** con el siguiente contenido:

“**OCDODOKDYOCDHESPBHLYZYBOVWODYLYLOCFCDSDFESYXZYVSHVPHIODSEHLOGSQSXOBO”**

Para recuperar el texto original a partir de l texto cifrado habrá que decodificar el fichero **“texto.cod”** con:

./viginere -d -f texto.cod -k hielo

CRIPTOANÁLISIS DEL MÉTODO

Para descifrar el método basta con saber el número de alfabetos N utilizados ya que se puede separar el texto original en N subtextos monoalfabéticos y realizarle un análisis de frecuencia en cada uno de ellos.

EJERCICIOS

EJERCICIO 1

Crear un texto de unas pocas líneas y cifrar con una clave por el método de Vigénere después descifrar el fichero codificado y compruebe que la salida corresponde al fichero original.

EJERCICIO 2

Dado el fichero de texto **texto4.cod** codificado obtenga la clave con la que se cifra el/los alfabeto/s con los que se cifra y el texto en claro.
¿Es un cifrado polialfabético o monoalfabético?

3º MÉTODO DE TRANSPOSICIÓN DE MATRIZ

El objetivo de la práctica es el conocer el método de transposición por matrices y la forma de proceder ante su criptoanálisis.

DESCRIPCIÓN DEL MÉTODO

Cifrar por un método de transposición consiste en alternar el orden de los símbolos sin sustituir éstos por otros.

Método de Transposición por MATRIZ
Se coloca el texto en claro en una matriz de N Columnas (o M filas) y se rellena la matriz con los símbolos por filas. Después se escribe el texto cifrado leyendo la matriz por columnas.

```
M = TEXTO PARA CIFRAR CON MATRIZ
N=4

T  E  X  T
O  P  A  R
A  C  I  F      El texto cifrado será
R  A  R  C      C = TOAROTEPCANRXAIRMITRFCAZ
O  N  M  A
T  R  I  Z
```

DESCRIPCIÓN DE LA APLICACIÓN

DESCRIPCIÓN
Esta aplicación realiza un cifrado por transposición de un texto con, utilizando una matriz de tantas filas y columnas como se le indiquen. El texto a cifrar y descifrar se coloca en orden en la matriz por filas y se lee por columnas. Si el número de elementos de la matriz es mayor que el número de caracteres del texto esto se rellenara con el carácter 'Z'.

```
Por ejemplo “Este texto” en una matriz de 4 filas seria:

    ESTE
    TEXT y el texto cifrado es ETOSEZTXZETZ
    OZZZ
```

Para descifrar el texto se escribe por columnas y se lee por filas.

SINOPSIS
trasp_matriz [-c|-d] [-f file_name] [-m row] [-n column] [-v version][-h help]

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar
`-c`	indicar que se quiere cifrar
`-d`	indicar que se quiere descifrar

`-m numero`	número de filas de la matriz para cifrar el texto
`-n numero`	número de columnas de la matriz para cifrar el texto

EJEMPLO

Para u fichero **texto.txt** cuyo contenido es :

"**Este es un texto para cifrar por el método de transposición por matriz"**

ejecutando
./trasp_matriz -c -f texto.txt -m 5

se obtiene un fichero texto.cod con el el siguiente contenido

"EOOTNSPRRPTAEAOERLNREAMSMSCEPAUITOTNFOSRTRDIIEAOCZXRDIZTPEOZ"

para descifrar texto.cod escribir
./trasp_matriz -d -f texto.txt -m 5

y se obtiene texto.dec con el contenido igual que al comienzo.

CRIPTOANÁLISIS DEL MÉTODO
Para descifrar un texto cifrado por el método de sustitución basta con conocer o probar distintos valores del tamaño de la matriz.

EJERCICIOS

EJERCICIO 1
Crear un texto de unas pocas líneas y cifrar con por el método de transposición de matrices después descifrar el fichero codificado y compruebe que la salida corresponde al fichero original.

EJERCICIO 2
Dado el fichero de texto **texto5.cod** codificado diga el método con el que es cifrado.
Obtenga el texto original.

EJERCICIO 3
Dado el fichero de texto **texto6.cod** codificado obtenga el texto original.

4º LA ENIGMA

El objetivo de la practica es conocer el funcionamiento de la enigma la forma de cifrar de la maquina y como los alemanes cifraban los mensajes durante la Segunda Guerra Mundial. También como supieron descifrar los mensajes codificados de la enigma.

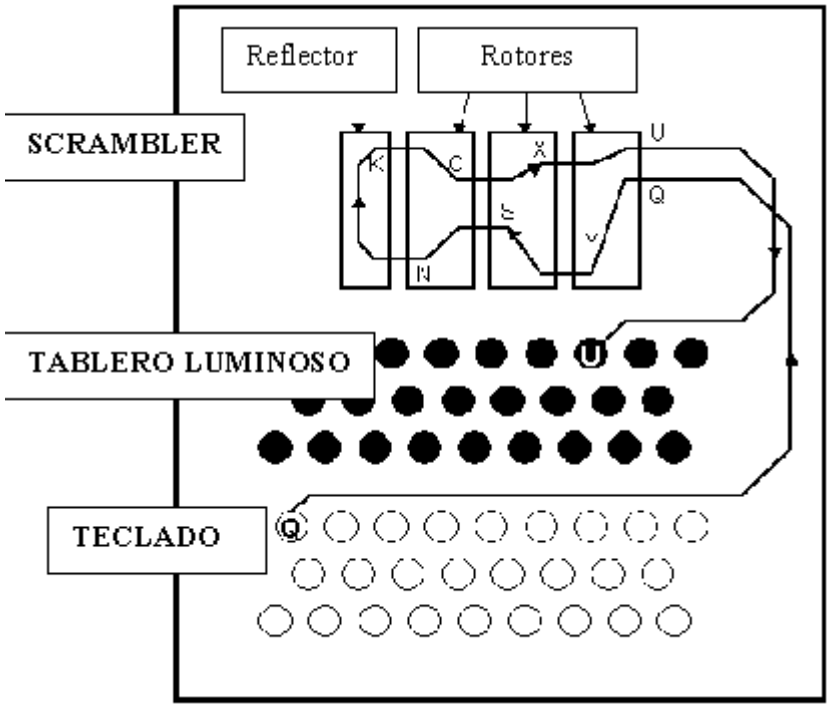
DESCRIPCIÓN DE LA MÁQUINA

La máquina Enigma constaba de:

Un teclado de 26 letras (como el de una máquina de escribir)

1. “Lamp board” o Tablero Luminoso. Un frontal con 26 bombillas, una para cada letra.
2. ”Scrambler” donde se encuentran los rotores en un eje común.
3. ”Steckerboard”, un frontal donde se podían hacer hasta 13 conexiones para emparejar letras mediante unas clavijas.
4. Reflector

La utilización era sencilla, después de que el operador dispusiera la configuración inicial (posición inicial y orden de los rotores, conexiones del steckerboard, rotores y reflector utilizado), tecleaba el texto a cifrar y cada vez que una tecla era pulsada se iluminaba su equivalente letra en el texto cifrado. Entonces lo único que había que hacer era apuntar las letras que se iban iluminando y transmitir el mensaje.



DESCRIPCIÓN DE LA APLICACIÓN

DESCRIPCIÓN

La aplicación consiste en un simulador software de la máquina Enigma. El programa recibe el nombre del fichero de texto a cifrar y del fichero de configuración con los rotores, la posición inicial de estos y el cableado del steckerboard. La aplicación toma cada carácter del texto como si se pulsara en la máquina uno a uno y guarda en un fichero con extensión ".cod" los valores obtenidos de cifrarlos, Después rota el primer rotor, cuando este ha dado una vuelta completa rotara el siguiente y así sucesivamente. Como se dispone de un reflector el codificar será de la misma manera que decodificar el texto.

El fichero de configuración debe tener la siguiente sintaxis:

```
########################################
#                                      #
# Fichero de configuración de la enigma #
#                                      #
########################################

[rotors]           #rotores
QWERTYUIOPLKJHGFDSAZXCVBNM
ASDFGHJKLMNBVCXZQWERTYUIOP
QWASZXERDFCVTYGHBNUIJKMOPL


#reflector
[reflector]
AABBCCDDEEFFGGHHIIJJKKLLMM

#iniciacion de rotores
[init_rotors]
ABB

#steckerboard (opcional)
[steckerboard]
AH
VM

#debe tener fin de linea
```

La opción [rotors] corresponde al cableado interno de los rotores, representados con caracteres, que significan la sustitución que realiza cada rotor. Para variar el orden de los rotores hay q variar el orden en el fichero, no pueden haber caracteres repetidos.

La opción [reflector] corresponde al reflector en la propia máquina y su forma de actuar es que las conexiones se realizan según los emparejamientos dos a dos de los diferentes caracteres, (por ejemplo, la posición 0 y 1 tienen los caracteres A A pues significan que en la maquina hay un cable que el giro lo devuelve así).

La opción [init_rotors] corresponde a la posición inicial de los rotores, es decir el numero de giros que realiza cada rotor para configurar la enigma antes de comenzar (clave del día o del mensaje).

Por último la opción [steckerboard] que corresponde a las parejas que corresponderían en la máquina original al clavijero donde la letra a la que corresponda se sustituye por la que esta unida. No deben de haber más de 12 parejas dispuestas en una columna.

SINOPSIS

enigma -c|-d -f file name -e configfile name -n num_rotors [-s steckerboard]
[-v version][-h help].

OPCIONES

`-h`	muestra la ayuda del programa
`-v`	muestra la versión del programa
`-f file_name`	se utiliza para indicar al programa el nombre del fichero a analizar
`-c`	indicar que se quiere cifrar el texto
`-d`	indicar que se quiere descifrar el texto
`-e`	nombre del fichero de configuración de la Máquina.
`-n numero`	número de rotores.
`-s`	Indica que se va a usar el steckerboard por lo tanto comprobara fichero de configuración

EJEMPLOS

Dado un fichero de texto texto.txt que se compone de:

"**Aqui se va a cifrar con la enigma**"

Y una clave de mensaje **CFG** para un fichero de configuración inicial con la clave del día ABB:

Entonces la manera de proceder al cifrado será:

Cifrar la clave del mensaje dos veces con esta configuración: **CFGCFG**

./enigma –c –f init.txt –e enigma.conf –n 3 -s

Que se obtendrá una salida cifrada: **KJHGHI** y se guardara en el fichero resultado.txt.

cat init.cod > resultado.txt

y luego cambiar en el fichero de configuración de la enigma la opción:
[init_rotors]
CFG

y luego volver a cifrar el texto original texto.txt con la nueva clave:

./enigma –c –f texto.txt –e enigma.conf –n 3 -s

y se obtendrá un fichero texto .cod con el siguiente valor:

TPKXJIUKAFLSVWWJWFUNWRESAR

cat texto.cod >> resultado.txt

y se obtendrá un fichero resultado.txt con:

KJHGHITPKXJIUKAFLSVWWJWFUNWRESAR

CRIPTOANÁLISIS DE LA ENIGMA

Para el criptoanálisis de la enigma se debe leer del capitulo 7 el apartado referente a este tema, como se generan las cadenas con el conjunto de mensajes del día y el desarrollo del diccionario. Se han creado una serie de programas que calculan el diccionario con todas las posibles combinaciones de los rotores. De manera que comparando el número de conexiones de los mensajes de un día con los del diccionario se puede obtener la clave del día ahorrando probar todas las claves posibles.

Generador de textos cifrados de la enigma

DESCRIPCIÓN
Este programa simula la captura de mensajes de un día explicada en el capítulo dedicado a la Enigma, generando un fichero con cadenas de caracteres (longitud de dos veces la clave del día) se que corresponderían con la primera letra de un texto codificado por la enigma (clave del mensaje codificada repetida dos veces).

SINOPSIS
./gen_fichero_cifrado –n number_rotors

RESULTADO

```
.....
.....
GUXOFQ
EISHTP
QJCJHC
GVBOMB
PQHERI
SXIQLE
MADYGH
OENWBD
GQGORS
ICBFUB
FIXRTQ
HXFNLF
......
```

Calculador de las cadenas de conexión en un conjunto de mensajes de un día.

DESCRIPCIÓN
Este programa crea las cadenas correspondientes en lo explicado en el capitulo 7 sobre el desciframiento de la enigma según el fichero del programa antes comentado.
Analiza el 1º y 4ª, 2º y 5º y 4º y 6 carácter de el fichero antes mostrado (codifican la misma letra), crean las tablas y comprueba las conexiones que tiene cada carácter.

SINOPSIS
./calc_cadenas –n number_rotors

RESULTADO

```
tabla 1
ABCDEFGHIJKLMNOPQRSTUVWXYZ
PUAIHRONFCGTYVWEJKQDZSLMXB

tabla 2
ABCDEFGHIJKLMNOPQRSTUVWXYZ
GEUYBSNATHOQWVKCRDXZFMILPJ

tabla 3
ABCDEFGHIJKLMNOPQRSTUVWXYZ
YBCHUFSIEZKOTDJRAMPLXGNQWV

tabla 1
A 10 conexiones
B 3  conexiones
C 10 conexiones
D 10 conexiones
E 10 conexiones
F 10 conexiones
G 10 conexiones
H 10 conexiones
I 10 conexiones
J 10 conexiones
K 10 conexiones
L 10 conexiones
M 3  conexiones
N 10 conexiones
O 10 conexiones
P 10 conexiones
Q 10 conexiones
R 10 conexiones
S 10 conexiones
T 10 conexiones
U 3  conexiones
V 10 conexiones
W 10 conexiones
X 3  conexiones
Y 3  conexiones
Z 3  conexiones
.............................
```

Software que realiza el diccionario con todas las cadenas

DESCRIPCIÓN

Este programa, dado una configuración inicial de rotores, obtiene las cadenas mediante el programa anteriormente descrito utilizando todas las posibles configuraciones iniciales de los rotores, así como para cualquier permutación de estos. El resultado se guardara en un fichero llamado diccionario.txt.

El fichero de configuración solo debe llevar información referida a la posición inicial de los rotores.

```
[rotors]          #rotores
QWERTYUIOPLKJHGFDSAZXCVBNM
ASDFGHJKLMNBVCXZQWERTYUIOP
QWASZXERDFCVTYGHBNUIJKMOPL
```

Con los rotores crea el propio fichero de configuración del enigma con todas las configuraciones y llama a los programas, enigma, gen_ficheros_cifrados y calc_cadenas, para que le calcule las cadenas. El fichero utiliza los otros dos ficheros anteriores para calcular las cadenas de una determinada posición. Por lo que diccionario.txt será igual a la salida de calc_cadenas pero de forma repetida con la configuración inicial de los rotores a los que corresponde.

```
……….
rotors_init ABB

tabla 1
ABCDEFGHIJKLMNOPQRSTUVWXYZ
PUAIHRONFCGTYVWEJKQDZSLMXB

tabla 2
ABCDEFGHIJKLMNOPQRSTUVWXYZ
GEUYBSNATHOQWVKCRDXZFMILPJ

tabla 3
ABCDEFGHIJKLMNOPQRSTUVWXYZ
YBCHUFSIEZKOTDJRAMPLXGNQWV

tabla 1
A 10conexiones
B 3conexiones
C 10conexiones
D 10conexiones
E 10conexiones
…….
```

EJERCICIOS

EJERCICIOS

EJERCICIO 1
Crear un texto de unas pocas líneas y cifrar con la máquina Enigma de la forma antes explicada con una clave de mensaje después descifrar el fichero codificado y compruebe que la salida corresponde al fichero original.

EJERCICIO 2
Dado un mensaje codificado **texto7.cod** sabiendo que la clave del día es:
132
DCF

a-h
d-c
e-r
t-j

Obtén el mensaje decodificado.

(utiliza enigma7.conf)

EJERCICIO 3
Dado un mensaje codificado **texto8.cod** del que no se conoce la clave del día pero si las cadenas de los mensajes del día, Dado el diccionario.
Calcular:
El mensaje original.
Las parejas de los clavijeros.

(utiliza enigma8.conf).

EJERCICIO 4
Dado una posición de rotores para una enigma de 2 rotores. Obtener el diccionario y descifrar el mensaje texto9.cod.

(diccionario.conf)

Anexo: Código C implementado

0. cripto.h

```
#ifndef _CRIPTO_H_
#define _CRIPTO_H_

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define OPT_F 1
#define OPT_C 2
#define OPT_D 4
#define OPT_K 8
#define OPT_M 16
#define OPT_N 32
#define OPT_E 64
#define OPT_P 128
#define OPT_S 256
#define ALFABET_26 26
#define ALFABET_36 36

#endif
```

1. sust_monoalfabetica.c

```
//--- sust_monoalfabetica.c ------


#include "cripto.h"

static char *prgname;
static char versión[]="1.0";
char usage[] = "Usage: %s -f file_name -m position -k key [-v versión]
                [-h help]\n";

int main(int argc, char **argv){
    FILE *fd1, *fd2;
    int alphabet[ALFABET_26];
    char newalphabet[ALFABET_26];
    char *name_in , *name_out, *aux;
    int ctrlopt = 0, c, i, j, w;
    char *key;
    int move = 0;

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;

    for (i = 0; i < ALFABET_26; i++){alphabet[i]=0;newalphabet[i]=0;}

    while(1)
    {
        c = getopt(argc, argv, "vhcdf:k:m:");
        if (c == EOF) break;
```

```
        switch (c)
        {
                case 'h': printf (usage, prgname);
                          fputs ("-h\t\tprint a options summary\n",
                                 stdout);
                          fputs ("-v\t\tprogram version\n", stdout);
                          fputs ("-c\t\tto encrypt the file\n", stdout);
                          fputs ("-d\t\tto desencrypt the file\n",
                                 stdout);
                          fputs ("-f\t\tname of the input data file\n",
                                 stdout);
                          fputs ("-m\t\tmove right positions in the
                                 alphabet\n", stdout);
                          fputs ("-k\t\tkey to replace the common
                                 alphabet\n", stdout);
                          exit (0);
                case 'v': fprintf (stderr, "%s\n", version);
                          exit (0);
                case 'c': ctrlopt |= OPT_C;
                          break;
                case 'd': ctrlopt |= OPT_D;
                          break;
                case 'f': name_in = optarg;
                          ctrlopt |= OPT_F;
                          break;
                case 'k': key=optarg;
                          ctrlopt |= OPT_K;
                          break;
                case 'm': move=atoi(optarg);
                          ctrlopt |= OPT_M;
                          break;
                default : printf(usage, prgname);
                          exit (0);
        }
}

if (!(ctrlopt & OPT_F))if (!(ctrlopt & OPT_F)) {
       perror(" -f option required");
       exit(1);
}
if((ctrlopt & OPT_C )){
    if((ctrlopt & OPT_D )){
        perror("incompatible -c and -d optoins");
        exit(1);
    }
}else if(!(ctrlopt & OPT_D)){
    perror("-c or -d option required");
    exit(1);
}
if(w=strlen(name_in)>0){
   name_out = (char *)malloc(w+4);
   strcpy(name_out,name_in);}

else{
   perror("file name error");
   exit(1);
}
if (move < 0 ){ perror("move < 0");exit(1);}
if (key==NULL){ perror("key = null");exit(1);}
```

```
 if((ctrlopt & OPT_K )){
     w=0;
     for ( i = 0; i < strlen ( key ) ; i++ ) {
           if (isalpha(*(key+i)))
           {
                 if (alphabet[toupper(*(key+i))-'A']==1)continue;
                       newalphabet[w] = toupper(*(key+i));
                       c=(toupper(*(key+i))-'A');
                       w++;
                       alphabet[toupper(*(key+i))-'A']=1;
                 }
           }
      for(j=0 ; j < ALFABET_26 ; j++ ){
           c=(c+1)%ALFABET_26;
           if(alphabet[c]==1)continue;
           else{
                 newalphabet[w]=c+'A';w++;
           }
      }

 }else{
    for(j = 0; j < ALFABET_26 ; j++ )
       newalphabet[j]=j+'A';

 }

 if ( ctrlopt & OPT_M ){
     if(move<0){perror("move < 0");exit(1);}
     char aux [ALFABET_26];

     for ( i = 0; i < ALFABET_26  ; i++){
             aux[(26+i-move)%ALFABET_26]=newalphabet[i];
     }
     strcpy(newalphabet,aux);
 }

 if( ctrlopt & OPT_C ){
     aux = memchr(name_out,'.',strlen(name_out));
     if ( aux!=NULL ) *aux='\0';

     strcat(name_out,".cod");

}else {
     aux = memchr(name_out,'.',strlen(name_out));
     if ( aux!=NULL ) *aux='\0';
     strcat(name_out,".dec");
}
if ((fd2 = fopen(name_out, "w")) == NULL) {
     perror("open file error"); exit(1);}

if ((fd1 = fopen(name_in, "r")) == NULL) {
     perror("open file error"); exit(1);}
free(name_out);

w=0;
```

```
    if( ctrlopt & OPT_C ){
        while (!feof(fd1))
        {
            c = getc(fd1);
            if (!isalpha(c)) continue;
            else
            {
                putc(newalphabet[toupper(c)-'A'],fd2);
            }
             w++;
             if((w%80==0)){w=0;putc('\n',fd2);}
        }
    }
    else{
        while (!feof(fd1))
        {
            c = getc(fd1);
            if (!isalpha(c)) continue;
            else
            {
                for(i=0;i<ALFABET_26;i++)
                        if (toupper(c)== newalphabet[i]){
                                putc(i+'A',fd2);
                                break;
                        }
            }
            w++;
            if((w%80==0)){w=0;putc('\n',fd2);}
        }
    }
    fclose(fd1);
    fclose(fd2);
    exit(0);
}
```

2. vigenere.c

```
//--- vigenere.c -----


#include "cripto.h"
#include <math.h>
static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s -c|-d -f file name -k key [-v version][-h
                    help]\n";

int main(int argc, char **argv)
{

    FILE *fd1, *fd2;
    int alphabet[ALFABET_26];
    char *key, *validkey;
    char *name_in, *name_out, *aux;
    int ctrlopt = 0, c, i, w, j, temp;
    int finLinea=0;
```

```
prgname = strrchr(argv[0], '/');
if (prgname == (char *)NULL) prgname = argv[0];
else prgname++;

for (i = 0; i < ALFABET_26; i++) alphabet[i] = 0.0;

while(1)
{
    c = getopt(argc, argv, "vhcdf:k:");

    if (c == EOF) break;

    switch (c)
    {
            case 'h': printf (usage, prgname);
                      fputs ("-h\t\tprint a options summary\n",
                             stdout);
                      fputs ("-v\t\tprogram version\n", stdout);
                      fputs ("-c\t\to crypt the file\n", stdout);
                      fputs ("-d\t\to decrypt the file\n", stdout);
                      fputs ("-f\t\tname of the input data file\n",
                             stdout);
                      fputs ("-k\t\tkey to crypt the file\n",
                             stdout);
                      exit (0);
            case 'v': fprintf (stderr, "%s\n", version);
                      exit (0);
            case 'c': ctrlopt |= OPT_C;
                      break;
            case 'd': ctrlopt |= OPT_D;
                      break;
            case 'f': name_in = optarg;
                      ctrlopt |= OPT_F;
                      break;
            case 'k': ctrlopt |= OPT_K;
                      key=optarg;
                      break;
            default : printf(usage, prgname);
                      exit (0);
    }
}

if (!(ctrlopt & OPT_F))if (!(ctrlopt & OPT_F)) {
       perror(" -f option required");
       exit(1);
}

if((ctrlopt & OPT_C )){
    if((ctrlopt & OPT_D )){
        perror("incompatible -c and -d optoins");
        exit(1);
    }
}else if(!(ctrlopt & OPT_D)){
     perror("-c or -d option required");
     exit(1);
}

if(!(ctrlopt & OPT_K)){
     perror(" -k option required");
     exit(1);
}
```

```
    else{
        if(key==(char *)NULL)exit(1);
        validkey=(char *)malloc(strlen(key));
        for(i=0;i<strlen(validkey);i++)*(validkey+i)=0;
             j=0;
             for(i=0;i<strlen(key);i++){
                 if(!isalpha(*(key+i)))continue;
                      *(validkey+j)=*(key+i);
                      alphabet[toupper(*(key+i))-'A']=1;
                      j++;
             }
    }
    if(i=strlen(name_in)>0){
          name_out = (char *)malloc(i+4);strcpy(name_out,name_in);}
    else{
          perror("file name error");
          exit(1);
    }

    if((ctrlopt & OPT_C ))
    {
        aux = memchr(name_out,'.',strlen(name_out));
        if ( aux!=NULL ) *aux='\0';
        strcat(name_out,".cod");

   }else {
        aux = memchr(name_out,'.',strlen(name_out));
        if ( aux!=NULL ) *aux='\0';
        strcat(name_out,".dec");
   }

   if ((fd2 = fopen(name_out, "w")) == NULL) {
              perror("file open error");
              exit(1);
   }
   if ((fd1 = fopen(name_in, "r")) == NULL) {
        perror("file open error");
        exit(1);
   }
   free(name_out);

   while (!feof(fd1)){
        for (i=0;i<strlen(validkey);i++)
            temp=((toupper(*(validkey+i)))-'A');
            if((ctrlopt & OPT_D ))temp=26-temp;
            if (!feof(fd1)){
                      c=getc(fd1);
                  if(isalpha(c)){
                       c = (((toupper(c)-'A')+temp)%26)+'A';
                       putc(c,fd2);
                       finLinea++;
                       if((finLinea%80==0)){finLinea=0;putc('\n',fd2);}
                  }else i--;
             }else break;
                                                                        }
   }
   fclose(fd1);
   fclose(fd2);
}
```

3. trasp_matriz.c

```
//---trasp_matriz.c-----

#include "cripto.h"

static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s -c|-d -f file_name -m row -n column [-v
version][-h help]\n";

int main(int argc, char **argv)
{
    FILE *fd1, *fd2;
    char c;
    char *name_in, *name_out,*aux ;
    int numChar;
    int column=0,row=0;
    int i,j,w;
    int temp;
    int ctrlopt = 0;
    int numZ;
    int finLinea=0;

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;
    while(1)
    {
        c = getopt(argc, argv, "vhcdf:m:n:");

        if (c == EOF) break;

        switch (c)
        {
                case 'h': printf (usage, prgname);
                          fputs ("-h\t\tprint a options summary\n",
                                 stdout);
                          fputs ("-v\t\tprogram version\n", stdout);
                          fputs ("-c\t\tencrypt the file\n", stdout);
                          fputs ("-d\t\tdesencrypt the file\n", stdout);
                          fputs ("-f\t\tname of the input data file\n",
                                 stdout);
                          fputs ("-m\t\tnumber of rows\n", stdout);
                          fputs ("-n\t\tnumber of columns\n", stdout);
                          exit (0);
                case 'v': fprintf (stderr, "%s\n", version);
                          exit (0);
                case 'c': ctrlopt |= OPT_C;
                          break;
                case 'd': ctrlopt |= OPT_D;
                          break;
                case 'f': name_in = optarg;
                          ctrlopt |= OPT_F;
                          break;
                case 'm': row=atoi(optarg);
                          ctrlopt |= OPT_M;
                          break;
```

```
                case 'n': column=atoi(optarg);
                          ctrlopt |= OPT_N;
                          break;
                default : printf(usage, prgname);
                          exit (0);
        }
}

if (!(ctrlopt & OPT_F))if (!(ctrlopt & OPT_F)) {
            perror(" -f option required");
            exit(1);
 }

 if((ctrlopt & OPT_C )){
     if((ctrlopt & OPT_D )){
          perror("incompatible -c and -d optoins");
          exit(1);
 }
 }else if(!(ctrlopt & OPT_D)){
         perror("-c or -d option required");
         exit(1);
 }

if(!(ctrlopt & OPT_M )){
     if(!(ctrlopt & OPT_N )){
         perror("-n or -m option required");
         exit(1);
 }
}
numChar=countChar(name_in);
if(temp=strlen(name_in)>0){
    name_out = (char *)malloc(temp+4);
    strcpy(name_out,name_in);
}
else {
    perror("name of file error");
    exit(1);
}

if(column<=0){
      column=numChar/row;
      if ( numChar > row*column) column++;
      numZ=column*row-numChar;
}else if(row<=0){
     row=numChar/column;
     if ( numChar > row*column) row++;
     numZ=column*row-numChar;
}else if ( numChar > column*row ){
     perror("number of char > columns * rows\n");
     exit(1);
}else numZ=column*row-numChar;


if (ctrlopt & OPT_D){
    temp=row;row=column;column=temp;
}

if(ctrlopt & OPT_C){
     aux = memchr(name_out,'.',strlen(name_out));
     if ( aux!=NULL ) *aux='\0';
     strcat(name_out,".cod");
```

```
   }else {
        aux = memchr(name_out,'.',strlen(name_out));
        if ( aux!=NULL ) *aux='\0';
        strcat(name_out,".dec");
   }
   if ((fd2 = fopen(name_out, "w")) == NULL) {
        perror("open file error");
        exit(1);
   }

   free(name_out);

   for(j=0;j<column;j++){
        if ((fd1 = fopen(name_in, "r")) == NULL) {
             perror("open file error");
             exit(1);}
        for(w=0;w<j;w++){
            if(!feof(fd1)){
                c=getc(fd1);
                if ( !isalnum(c) )w--;
            }
         }
         while(!feof(fd1))
         {
             c=getc(fd1);
             if ( !isalnum(c))continue;
             else
             {
                 if(islower(c)){
                      putc(toupper(c),fd2);
                      finLinea++;
                      if((finLinea%80==0)){finLinea=0;putc('\n',fd2);}
                 }else {
                      putc(c,fd2);
                      finLinea++;
                      if((finLinea%80==0)){finLinea=0;putc('\n',fd2);}
                 }
             }
             for(i=0;i<column-1;i++){
                  if(feof(fd1))break;
                  c=getc(fd1);
                  if ( !isalnum(c) )i--;
             }
        }
        if ((j>=column-numZ)){
             putc('Z',fd2);
             finLinea++;
             if((finLinea%80==0)){finLinea=0;putc('\n',fd2);}
        }
        fclose (fd1);
   }
   fclose (fd2);
}

int countChar( char *name) {
        FILE * fd;
        char c;
        fd=fopen(name,"r");
        int numChar=0;
```

```
        while ( !feof(fd) ){
                c=getc(fd);
                if (!isalnum(c)) continue;
                else
                {
                        if (isdigit(c))numChar++;
                        else if (isupper(c))numChar++;
                        else if (islower(c))numChar++;
                }
        }
        fclose(fd);
        return numChar;
}
```

4. enigma.c

```
//---enigma.c---


#include "cripto.h"

#include <time.h>
#define MAX 1024
#define STECK_MAX 6
#define REP_KEY 2
#define MAXROTORS 12
#define INI_OPT 1
#define ROT_OPT 2
#define STK_OPT 4
#define REF_OPT 8

static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s -c|-d -f file name -e configfile name -n
num_rotors [-s steckerboard] [-v version][-h help]\n";


static char  steckerboard [ALFABET_26];
static char ** rotors;
static char ** config_rotors;
static char reflector [ALFABET_26];
static int num_rotors=3;
static char * key;

int main(int argc, char ** argv)
{
    FILE *fd1, *fd2, *fd_conf;
    int * alphabet[ALFABET_26];
    char * name_conf;
    char * name_in, *name_out, *aux;
    int ctrlopt=0, c, i, w, j, temp;
    int rotor_ctrl=0;

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;
    for (i=0;i<ALFABET_26;i++) steckerboard[i]=0;
    while(1)
```

```
{
    c = getopt(argc, argv, "vhcdsf:e:n:");
    if (c == EOF) break;
    switch (c)
    {
            case 'h': printf (usage, prgname);
                      fputs ("-h\t\tprint a options summary\n",
                        stdout);
                      fputs ("-v\t\tprogram version\n", stdout);
                      fputs ("-c\t\tto crypt the file\n", stdout);
                      fputs ("-d\t\tto decrypt the file\n", stdout);
                      fputs ("-f\t\tname of the input data file\n",
                        stdout);
                      fputs ("-e\t\tname of the config file\n",
                        stdout);
                      fputs ("-s\t\tto use steckerboard\n", stdout);
                      fputs ("-n\t\tnumber of rotors\n", stdout);
                      exit (0);
            case 'v': fprintf (stderr, "%s\n", version);
                      exit (0);
            case 'c': ctrlopt |= OPT_C;
                      break;
            case 'd': ctrlopt |= OPT_D;
                      break;
            case 'e': ctrlopt |= OPT_E;
                      name_conf=optarg;
                      break;
            case 'f': name_in = optarg;
                      ctrlopt |= OPT_F;
                      break;
            case 'n': num_rotors = atoi(optarg);
                      ctrlopt |= OPT_N;
                      break;
            case 's': ctrlopt |= OPT_S;
                      break;
            default : printf(usage, prgname);
                      exit (0);
    }
}
if (!(ctrlopt & OPT_F))if (!(ctrlopt & OPT_F)) {
      perror(" -f option required");
      exit(1);
}
if((ctrlopt & OPT_C )){
    if((ctrlopt & OPT_D )){
        perror("incompatible -c and -d optoins");
        exit(1);
    }
}else if(!(ctrlopt & OPT_D)){
    perror("-c or -d option required");
    exit(1);
}

if (!(ctrlopt & OPT_E)) {
    perror(" -e option required");
    exit(1);
}
if (!(ctrlopt & OPT_N)) {
    perror(" -n option required");
    exit(1);
}
```

```
 if( num_rotors <= 0 ) {
     perror("error num_rotors < 0");
     exit(1);
 }



 else{
     if( num_rotors > MAXROTORS ){
         perror("error num_rotors > MAXROTORS");
         exit(1);
     }
     rotors=(char **)malloc(num_rotors);
     config_rotors=(char **)malloc(num_rotors);
     for(i=0;i<num_rotors;i++){
         rotors[i]=(char *)malloc(ALFABET_26);
         config_rotors[i]=(char *)malloc(ALFABET_26);
     }
     key=(char *)malloc(num_rotors);
 }
 if( i=strlen(name_in)>0){
     name_out = (char *)malloc(i+4);
     strcpy(name_out,name_in);
 }
 else{
     perror("file name error");
     exit(1);
 }

 fill_rotors(name_conf);
 fill_reflector(name_conf);
 if (ctrlopt & OPT_S)
        fill_steckerboard(name_conf);
 init_rotors(name_conf);

 if((ctrlopt & OPT_C ))
 {
     aux = memchr(name_out,'.',strlen(name_out));
     if ( aux!=NULL ) *aux='\0';
     strcat(name_out,".cod");

}else {

     aux = memchr(name_out,'.',strlen(name_out));
     if ( aux!=NULL ) *aux='\0';
     strcat(name_out,".dec");
}

if ((fd2 = fopen(name_out, "w")) == NULL) {
     perror("open file error");
     exit(1);
}
if ((fd1 = fopen(name_in, "r")) == NULL) {
     perror("open file error");
     exit(1);
}
free(name_out);
temp=0,rotor_ctrl=0;
```

```
   while(!feof(fd1)){
       if(feof(fd1))break;
       c=getc(fd1);
       if(!isalpha(c)){continue;}
       c=keyboard_preshed(c,ctrlopt);
       putc(c,fd2);
       rotor_ctrl++;
       for( i=0 ; i < num_rotors ; i++ ){
          if((i!=0)&&(rotor_ctrl%(ALFABET_26*i)==0)){
             rotate_rotors(rotors[i]);
             if (i==(num_rotors-1))rotor_ctrl=0;
           }
           if(i==0)rotate_rotors(rotors[i]);
       }
   }
   putc('\n',fd2);
   fclose(fd1),fclose(fd2);
}

rotate_rotors(char * rotor ){
   int i , temp;
   char aux [ALFABET_26];
   for( i = 0 ; i < ALFABET_26 ; i++ ){
        temp=((i+'A')-rotor[i]);
        temp=((i+1)%ALFABET_26)-temp;
        if(temp<0)temp=ALFABET_26+temp;
        temp=temp%ALFABET_26;
        aux[(i+1)%ALFABET_26]=temp+'A';
   }
   for( i = 0 ; i < ALFABET_26 ; i++ ) rotor[i]=aux[i];
}

fill_rotors(char *name_conf){

   int c,i,j;
   FILE * fd_conf;
   if ((fd_conf = fopen(name_conf, "r")) == NULL) {
           perror("error open config file");
           exit(1);
   }
   char  buff [1024];
   char * t;
   while(!feof(fd_conf)){
       fgets ( buff,1024,fd_conf);
       if((t=strstr(buff,"[rotors]"))!=NULL){
            for(i=0;i<num_rotors;i++)
                 for(j=0;j<ALFABET_26;j++){
                     if(!feof(fd_conf)){
                         if( !isalpha( c=getc(fd_conf) ) ){
                                j-- ; continue; }
                         rotors[i][j]=toupper(c):
                         config_rotors[i][j]=toupper(c);
                     }
                 }
       }
   }
   fclose(fd_conf);
}
```

```
fill_reflector(char * name_conf){

   int c,j;
   char * t;
   FILE * fd_conf;
   if ( (fd_conf = fopen(name_conf, "r")) == NULL) {
        perror("error open config file");
        exit(1);
   }
   char  buff [1024];
   while(!feof(fd_conf)){
       fgets ( buff,1024,fd_conf);
       if((t=strstr(buff,"[reflector]"))!=NULL){
            for(j=0;j<ALFABET_26;j++){
                if(!feof(fd_conf)){
                    if( !isalpha( c=getc(fd_conf) ) ){ j-- ; continue; }
                    reflector[j]=toupper(c);
                }
            }
       }
   }
   fclose(fd_conf);
}

fill_steckerboard(char * name_conf){

   int c,i,j;
   FILE * fd_conf;
   if ( (fd_conf = fopen(name_conf, "r")) == NULL) {
        perror("error open config file");
        exit(1);
   }
   char  buff [1024];
   char * t;
   char  temp  [STECK_MAX][2];
   for (i=0 ; i < STECK_MAX ; i ++ )
            for(j=0 ; j<2 ; j++){
                 temp[i][j]='0';
            }

   while(!feof(fd_conf)){
       fgets ( buff,1024,fd_conf);
       if((t=strstr(buff,"[steckerboard]"))!=NULL){
          for (i=0 ; i < STECK_MAX ; i ++ )
            for(j=0 ; j<2 ; j++){
                if(!feof(fd_conf)){
                    c=getc(fd_conf);
                    if( c == '#' | c=='[') break;
                    if( !isalpha( c ) ){ j-- ; continue; }
                    temp[i][j]=toupper(c);
                }else  break;
           }
       }
   }
   for (i=0 ; i < STECK_MAX ; i ++ ){
        if(temp[i][0]-'A'<0)break;
        steckerboard[temp[i][0]-'A']=temp[i][1];
        steckerboard[temp[i][1]-'A']=temp[i][0];
   }
   fclose(fd_conf);
}
```

```
init_rotors(char * name_conf){

   FILE * fd_conf;
   char buff[1024];
   char * t=NULL;
   int i=0,j=0;
   if ((fd_conf = fopen(name_conf, "r")) == NULL) {
  perror("error open config file");
  exit(1);
   }
   while(!feof(fd_conf)){
        fgets ( buff,1024,fd_conf);
        if((t=strstr(buff,"[init_rotors]"))!=NULL){
              if(!feof(fd_conf))
              fgets ( buff,1024,fd_conf);
              for(j=0;j<num_rotors;j++){
                    for(i=0;i<toupper(buff[j])-'A';i++)
                            rotate_rotors(rotors[j]);
              }
        }
   }

int keyboard_preshed( int c , int ctrlopt){

       int temp=0,i,j;
       if((ctrlopt & OPT_C ))
            if( steckerboard[toupper(c)-'A']!=0 )
                c=steckerboard[toupper(c)-'A'];
       for( i = 0 ; i < num_rotors ; i++ ){
           c=(int)rotors[i][toupper(c)-'A'];
       }
       temp=reflector[c-'A'];
       for(i = 0; i < ALFABET_26; i++){
           if((i+'A')==c)continue;
           if(temp==reflector[i]){c=i+'A';break;}
       }
       for( i = num_rotors ; i>0 ; i-- ){
           for(j = 0; j < ALFABET_26; j++){
               if(c==(int)rotors[i-1][j]){c=j+'A';break;}
           }
       }
       if((ctrlopt & OPT_D ))
          if( steckerboard[c-'A']!=0 )
                c=steckerboard[c-'A'];
       return c;
}

file_Correct(char * name_conf){
    int config_ctrl=0;
    char * p;
    int i,j,c;
    FILE * fd;
    char buff [1024];
    int alphabet [26];
    char * aux;
    int count=0;
    if ((fd = fopen(name_conf, "r")) == NULL) {
           perror("error open config file");
           exit(1);
    }
```

```
while (!feof(fd)){
    for(j=0;j<26;j++)alphabet[j]=0;
    p=fgets(buff,1024,fd);
    if(buff[0]=='#')continue;
    else if(buff[0]==' '){
          while(1){
               c=*p;
               p++;
               if(c==' ')continue;
               else if(c=='\n'|c=='#')break;
               else {
                     perror("sintaxy config file error");
                     exit(1);
                }
          }
     }
     else if(buff[0]=='\n')continue;
     else if((aux=strstr (buff,"[rotors]"))!=NULL){
          if((config_ctrl&ROT_OPT)){
               perror("sintaxy config file error");
               exit(1);
           }
           config_ctrl |= ROT_OPT;
           p=strchr(aux,']');
           while(1){
           p++;
           if ((p[0]=='#')|(p[0]=='\n'))break;
           if(p[0]==32)continue;
           else{
              perror("sintaxy config file error");
              exit(1);
           }
    }
    for(i=0;i<num_rotors;i++){
       if(!feof(fd)){
           p=fgets(buff,1024,fd);
       }else{
           perror("sintaxy config file error");
           exit(1);
       }
       if(!isalpha(buff[0])){
            perror("sintaxy config file error");
            exit(1);
       }
       while(1){
           c=*p;
           p++;
           if(isupper(c)){
                if(alphabet[c-'A']>i){
                    perror("sintaxy config file error");
                    exit(1);
                }
                alphabet[c-'A']++;
           }
           else if(c==' ')continue;
           else if(c=='\n'|c=='#')break;
           else{
                perror("sintaxy config file error");
                exit(1);
           }
       }
```

```
        for(j=0;j<26;j++)
              if(alphabet[j]<i+1){
                      perror("sintaxy config file error");
                      exit(1);
              }
         p=NULL;}
    }
    else if((aux=strstr (buff,"[reflector]"))!=NULL){
         if((config_ctrl&REF_OPT)){
            perror("sintaxy config file error");
            exit(1);
         }
    config_ctrl|=REF_OPT;
    p=strchr(aux,']');
    while(1){
        p++;
        if ((p[0]=='#')|(p[0]=='\n'))break;
        if(p[0]==32)continue;
        else{perror("sintaxy config file error");exit(1);}
    }
    if(!feof(fd)){
        p=fgets(buff,1024,fd);
    }else{perror("sintaxy config file error");exit(1);}
    if(!isalpha(buff[0])){
        perror("sintaxy config file error");
        exit(1);
    }
    while(1){
          c=*p;
          p++;
          if(isupper(c)){
              alphabet[c-'A']++;
          }
          else if(c==' ')continue;
          else if(c=='\n'|c=='#')break;
          else{
             perror("sintaxy config file error");
             exit(1);
          }
    }
    for(j=0;j<26;j++)
         if((alphabet[j]!=0)&&(alphabet[j]!=2)){
               perror("sintaxy config file error");
               exit(1);
         }
}

else if((aux=strstr (buff,"[steckerboard]"))!=NULL){
     if((config_ctrl & STK_OPT)){
           perror("sintaxy config file error");exit(1);}
     config_ctrl|=STK_OPT;
     p=strchr(aux,']');
     while(1){
          p++;
          if ((p[0]=='#')|(p[0]=='\n'))break;
          if(p[0]==32)continue;
          else{perror("sintaxy config file error");exit(1);}
     }
```

```
         for(i=0;i<6;i++){
            if(!feof(fd)){
                p=fgets(buff,1024,fd);
            }else{
                perror("sintaxy config file error");exit(1);}
            if(i==0&!isalpha(buff[0])){
                 perror("sintaxy config file error");exit(1);}
            if(!isalpha(buff[0]))break;
            while(1){
                 c=*p;
                 p++;
                 if(isupper(c)){
                    alphabet[c-'A']++;
                 }
                 else if(c==' ')continue;
                 else if(c=='\n'|c=='#')break;
                 else{
                      perror("sintaxy config file error");
                      exit(1);}
                 }
            }
for(j=0;j<26;j++){
if(alphabet[j]!=1&&alphabet[j]!=0){
     perror("sintaxy config file error");
     exit(1);}
     if(alphabet[j]==1)count++;
}
if((count>12)|(count%2!=0)){
     perror("sintaxy config file error");exit(1);};
}
else if((aux=strstr (buff,"[init_rotors]"))!=NULL){
     if((config_ctrl & INI_OPT)){
           perror("sintaxy config file error");exit(1);}
     config_ctrl|=INI_OPT;
     p=strchr(aux,']');
     while(1){
           p++;
           if ((p[0]=='#')|(p[0]=='\n'))break;
           if(p[0]==32)continue;
           else{perror("sintaxy config file error");exit(1);}
     }
     if(!feof(fd)){
        p=fgets(buff,1024,fd);
     }else{perror("sintaxy config file error");exit(1);}
     i=0;
     while(1){
          c=*p;
          p++;
          if(isupper(c)&&i<num_rotors){i++;continue;}
          else if(c==' ')continue;
          else if(c=='\n'|c=='#')break;
          else{
                perror("sintaxy config file error");
                exit(1);}
     }
}
else{perror("sintaxy config file error");exit(1);}
bzero(buff,1024);
}
fclose(fd);
}
```

5. frecuencias.c

```
//--- frecuencias.c -----

#include "cripto.h"

static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s -f file name [-v version][-h help]\n";

int main(int argc, char **argv)
{

    FILE *fd;
    int alphabet[ALFABET_26];
    char *name;
    int ctrlopt = 0, c, i;
    int numChar=0;

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;

    for (i = 0; i < ALFABET_26; i++) alphabet[i] = 0.0;

    while(1)
    {
        c = getopt(argc, argv, "vhf:");

        if (c == EOF) break;

        switch (c)
        {
                case 'h': printf (usage, prgname);
                          fputs ("-h\t\tprint a options summary\n",
                                 stdout);
                          fputs ("-v\t\tprogram version\n", stdout);
                          fputs ("-f\t\tname of the input data file\n",
                                 stdout);
                          exit (0);
                case 'v': fprintf (stderr, "%s\n", version);
                          exit (0);
                case 'f': name = optarg;
                          ctrlopt |= OPT_F;
                          break;
                default : printf(usage, prgname);
                          exit (0);
        }
    }
    if (!(ctrlopt & OPT_F)) {
       perror("-f option required");
       exit(1);
    }
    if ((fd = fopen(name, "r")) == NULL) {perror("not open"); exit(1);}
```

```
    while (!feof(fd))
    {
        c = getc(fd);
        if (!isalpha(c)) continue;
        else
        {
             alphabet[toupper(c)-'A']++;
             numChar++;
        }
    }

    printf("\n Number Chars = %d \n",numChar);
    printf("CHAR\tNUM\tFREQ\n");
    for (i = 0; i < ALFABET_26; i++)
    {
        printf(" %c\t%d\t%f\n",'A'+ i ,
           alphabet[i],((float)alphabet[i]/numChar*100));
    }
    exit(0);
}
```

6. cadenas_repetidas.c

```
//--- cadenas_repetidas----

#include "cripto.h"

#define MAXLINE 1024

static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s -f file name -m number char [-v version]
[-h help]\n";

int countChar( char *name);

int main (int argc,char ** argv){

    FILE *fd1, *fd2;
    int group=-1 ,numChar ,count = 0;
    int temp = 0, i = 0, w = 0, ctrlopt = 0;
    char c, c1, c2;
    char *name;
    int distance, fd1_cont, fd2_cont;
    int * position;
    char search [MAXLINE];

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;


    while (1)
    {
        c = getopt(argc, argv, "vhf:m:");

        if (c == EOF) break;

        switch (c)
        {
```

```
            case 'h': printf (usage, prgname);
                      fputs ("-h\t\tprint a options summary\n", stdout);
                      fputs ("-v\t\tprogram version\n", stdout);
                      fputs ("-f\t\tname of the input data file\n",
                             stdout);
                      fputs ("-m\t\tnumber of char in the group\n",
                             stdout);
                      exit (0);
            case 'v': fprintf (stderr, "%s\n", version);
                      exit (0);
            case 'f': name = optarg;
                      ctrlopt |= OPT_F;
                      break;
            case 'm': group=atoi(optarg);
                      ctrlopt |= OPT_M;
                      break;
            default : printf(usage, prgname);
                     exit (0);
        }
    }

    if (!(ctrlopt & OPT_F)) {
perror(" -f option required");
exit(1);
    }
    if (!(ctrlopt & OPT_M)) {
perror(" -m option required");
exit(1);
    }
    if (group<=0){perror(" m<0 "); exit(1);}

    if ( (numChar = countChar(name)) <= 0 ){
          perror(" numChar <=0 ");
          exit(1);
    }else{
         position = malloc(numChar);
         for( i = 0; i<numChar ; i++ ){
              *(position+i)=0;

         }
    }

    while ( count < (numChar-group) )
    {
         distance=0;w=0;
         if ((fd1 = fopen(name, "r")) == NULL) {perror("open"); exit(1);}
         if ((fd2 = fopen(name, "r")) == NULL) {perror("open"); exit(1);}
         for( i=0 ; i<count ; i++ ){
              if(!feof(fd1)&!feof(fd2)){
                 c=getc(fd1);
                 getc(fd2);
                 if(!isalpha(c)){i--;continue;}
                 w++;
              }
         }
         do{
              fd1_cont=ftell(fd1);
              c1=getc(fd1);
         }while(!isalpha(c1));

         for( i=0 ; i<group ; i++ ){
```

```
             if(!feof(fd2)){
                c=getc(fd2);
                if(!isalpha(c)){i--;continue;}
                w++;
             }
        }
        temp=0;
        while(!feof(fd2)){

            fd2_cont=ftell(fd2);
            c2=getc(fd2);

            if(!isalpha(c2))continue;

            if(((*(position + w))>0)&temp==0){w++;continue;};

            if (c2==c1){
                search[temp]=c1;
                do{
                  c1=getc(fd1);
                }while(!isalpha(c1));
                temp++;
            }
            else{
                fseek(fd1,fd1_cont,0);
                c1=getc(fd1);

                if (temp>=group){

                      fseek(fd2,fd2_cont,0);
                      distance=w-temp-count;
                      for( i=0 ; i<temp ; i++ )
                         (*(position+(count+distance+i)))++;
                      printf("%s en posicion %d a distancia %d\n ",
                              search,ftell(fd1)-1,distance);
                      memset(search,'\0',MAXLINE);w--;
                      while( getchar()!='\n' );
                }
                temp=0;
            }

            w++;
        }
        if (temp>=group){
                      fseek(fd1,fd1_cont,0);
                      distance=w-temp-count;
                      for( i=0 ; i<temp ; i++ )
                         (*(position+(count+distance+i)))++;
                      printf("%s en posicion %d a distancia %d\n ",
                              search,ftell(fd1),distance);
                      memset(search,'\0',MAXLINE);
                      while( getchar()!='\n' );
        }

      fclose(fd1);
      fclose(fd2);
      count++;
   }
   exit(0);

}
```

```
int countChar( char *name) {
FILE * fd;
char c;
fd=fopen(name,"rw");
int numChar=0;
while ( !feof(fd) ){
c=getc(fd);
if (!isalpha(c)) continue;
else numChar++;
        }
fclose(fd);
return numChar;
}
```

7. cambiar_letras.c

```
//--- cambiar_letras.c -----

#include "cripto.h"

static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s -f file_name -c char1 -d char2 [-v version]
[-h help]\n";

int main(int argc, char **argv)
{

    FILE * fd_io;
    char * name_in;
    int num , i ,ctrlopt;
    char c1,c2,c;

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;

    while(1)
    {
        c = getopt(argc, argv, "vhf:c:d:");
        if (c == EOF) break;
        switch(c)
        {
                case 'h': printf (usage, prgname);
                          fputs ("-h\t\tprint a options summary\n",
                                 stdout);
                          fputs ("-v\t\tprogram version\n", stdout);
                          fputs ("-f\t\tname of the input data file\n",
                                 stdout);
                          fputs ("-c\t\tchar1\n", stdout);
                          fputs ("-d\t\tchar2\n", stdout);
                          exit (0);
                case 'v': fprintf (stderr, "%s\n", version);
                          exit (0);
                case 'f': name_in = optarg;
                          ctrlopt |= OPT_F;
                          break;
                case 'c': ctrlopt |= OPT_C;
```

```
                        if(strlen(optarg)>1)exit(1);
                        c1 = optarg[0];
                        break;
                case 'd': ctrlopt |= OPT_D;
                        if(strlen(optarg)>1)exit(1);
                        c2 = optarg[0];
                        break;
                default : printf(usage, prgname);
                        exit (0);
        }
    }


    if (!(ctrlopt & OPT_F)){
        perror("-f option required");
        exit(1);
    }
    if (!(ctrlopt & OPT_C)){
        perror("-c option required");
        exit(1);
    }
    if (!(ctrlopt & OPT_D)){
        perror("-d option required");
        exit(1);
    }

    if((!isalpha(c1))|(!isalpha(c2))){
        perror("c1 and c2 not char");
        exit(1);
    }


    if ((fd_io = fopen(name_in, "r+")) == NULL) {
        perror("not open");
        exit(1);
    }

while(!feof(fd_io)){
   i=ftell(fd_io);
   c=getc(fd_io);
        if(!isalpha(c))continue;
        if (toupper(c)==toupper(c1)){
            fseek(fd_io,i,0);
            putc(tolower(c2),fd_io);
        }
    }
    fclose(fd_io);
    exit(0);
}
```

8. separar_texto.c

```
//---separar_texto.c-----

#include "cripto.h"

#define MAXLINE 256

static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s -f file name -c|-p [-v version][-h help]\n";

int main(int argc, char ** argv)
{
    FILE * fd_in;
    FILE ** fd_out;
    char * name_in;
    char ** name_out;
    int num , i ,ctrlopt;
    char c;
    char temp[MAXLINE];

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;

    while(1)
    {
        c = getopt(argc, argv, "n:vhf:cp");

        if (c == EOF) break;
        switch (c)
        {
                case 'h': printf (usage, prgname);
                          fputs ("-h\t\tprint a options summary\n",
                                 stdout);
                          fputs ("-v\t\tprogram version\n", stdout);
                          fputs ("-f\t\tname of the input data file\n",
                                 stdout);
                          fputs ("-n\t\tnumber of inputs or outputs data
                                 file\n", stdout);
                          fputs ("-c\t\t1 file to n files\n", stdout);
                          fputs ("-p\t\tn files to 1 file\n", stdout);
         exit (0);
                case 'v': fprintf (stderr, "%s\n", version);
                          exit (0);
                case 'f': name_in = optarg;
                          ctrlopt |= OPT_F;
                          break;
                case 'n': num = atoi( optarg );
                          ctrlopt |= OPT_N;
                          break;
          case 'c': ctrlopt |= OPT_C;
                          break;

                case 'p': ctrlopt |= OPT_P;
                          break;
                default : printf(usage, prgname);
                          exit (0);
        }
    }
```

```
if (!(ctrlopt & OPT_F)){
    perror("-f option required");
    exit(1);
}
if (!(ctrlopt & OPT_N)){
    perror("-n option required");
    exit(1);
}

if((ctrlopt & OPT_C )){
    if((ctrlopt & OPT_P )){
       perror("incompatible -c and -p optoins");
       exit(1);
}
}else if(!(ctrlopt & OPT_P)){
       perror("-c or -p option required");
       exit(1);
}
if(strlen(name_in)>0){ name_out = (char **)malloc ( num );}
else{
    perror("file error");
    exit(1);
}

fd_out = (FILE **)malloc (num);

for(i=0;i<num;i++){
    name_out[i] = (char *)malloc(strlen(name_in+1));
    fd_out[i]=(FILE *)malloc(sizeof(FILE *));
    bzero(name_out[i],sizeof(name_out[i]));
    bzero(temp,sizeof(temp));
    strcpy(name_out[i] , name_in);
    sprintf(temp,"%d",i);
    strncat(name_out[i] ,temp,1);
}

if (ctrlopt & OPT_C){
   if ((fd_in = fopen(name_in, "r")) == NULL){
     perror("file error, open");
     exit(1);
   }
   for(i=0 ; i<num ; i++ )
        if ((fd_out[i] = fopen(name_out[i], "w")) == NULL) {
             perror("file error, open"); exit(1);}

    i=0;
    while(1){
      if(feof(fd_in))break;
      c=getc(fd_in);
      if (!isalpha(c)) continue;
      putc(toupper(c),fd_out[i]);
      i++;
      if(i==num)i=0;
    }
}else
if (ctrlopt & OPT_P){
    if ((fd_in = fopen(name_in, "w")) == NULL) {
         perror("file error, open"); exit(1);}
```

```
         for(i=0 ; i<num ; i++ )
             if ((fd_out[i] = fopen(name_out[i], "r")) == NULL) {
                       perror("file error, open");
                       exit(1);}
        i=0;
        while(1){

            if(feof(fd_out[i]))break;
            c=getc(fd_out[i]);
            if (!isalpha(c)) continue;
            putc(toupper(c),fd_in);
            i++;
            if(i==num)i=0;
        }
    }else
      exit(1);


    fclose(fd_in);
    for(i=0 ; i<num ; i++ ){
        putc('\n',fd_out[i]);
        fclose(fd_out[i]);
    }
    free(fd_out);
    free(name_out);
    exit(0);

}
```

9. gen_ficheros_cifrados.c

```
//--- gen_ficheros_cifrados.c

#include "cripto.h"
#include <time.h>

#define MAX_ROTORS 12

int main(int argc, char ** argv){

int c,i,j,cuenta=0;
char fin=0;
FILE * fd1,*fd2;
char * clave;
char sentencia [1024];
char temp [10];
char ** alphabet;
int num_rotors=3;
int ctrlopt=0;
```

```
while(1)
{
   c = getopt(argc, argv, "n:");
   if (c == EOF) break;
        switch (c)
        {
            case 'n': num_rotors = atoi(optarg);
                      ctrlopt |= OPT_N;
                     break;
          default : perror("options error");
                               exit (1);
              }
        }

if (!(ctrlopt & OPT_N)) {
perror(" -n option required");
exit(1);
    }
    if( num_rotors <= 0 ) {
        perror("error num_rotors < 0");
exit(1);
    }
if( num_rotors > MAX_ROTORS ) {
        perror("error number of rotors");
exit(1);
    }
clave=(char *)malloc(num_rotors);
alphabet=(char **)malloc(num_rotors);
for(i=0;i<num_rotors;i++)
            alphabet[i]=(char *)malloc(ALFABET_26);
for(i=0;i<num_rotors;i++){
for(j=0;j<ALFABET_26;j++)
                   alphabet[i][j]=0;
}
gen_aleatorio();

while(!fin){
cuenta=0;
if ((fd1 = fopen("mess.txt", "w")) == NULL) {
                  perror("open"); exit(1);}
for(j=0;j<num_rotors;j++){
c=(get_char()+'A');
clave[j]=c;
alphabet[j][c-'A']=1;
}
for(i=0;i<2;i++){
for(j=0;j<num_rotors;j++){
putc(clave[j],fd1);
}
}
putc('\n',fd1);
fclose(fd1);
sprintf(sentencia,"./enigma -c -f mess.txt
                        -e enigma.conf -n %d",num_rotors);
system(sentencia);
system("cat mess.cod >>resultados.txt");
cuenta=0;
for(i=0;i<num_rotors;i++){
for(j=0;j<ALFABET_26;j++)cuenta+=alphabet[i][j];
}
if(cuenta==(ALFABET_26*num_rotors))
```

```
fin=1;
}
system("rm mess.*");

}

gen_aleatorio(){
   int stime;
   long ltime;
   ltime=time(NULL);
   stime=(unsigned)ltime/2;
   srand(stime);
   return;
}

int get_char(){
     return  rand()%26;
}
```

10. calc_cadenas.c

```
//--- calc_cadenas.c -----

#include "cripto.h"
#include <time.h>

#define MAX_ROTORS 12

int main(int argc, char ** argv){

int i=0,j=0,temp=0;
int conexiones=0;
int num_rotors=3;
char ** alphabet;
char buff [256];
FILE * fd1, * fd2;
int ctrlopt=0;
int c=0;

while(1)
{
    c = getopt(argc, argv, "n:");
    if (c == EOF) break;
    switch (c)
    {
      case 'n': num_rotors = atoi(optarg);
                ctrlopt |= OPT_N;
                break;
      default : perror("options error");
                exit (1);
    }
}
if (!(ctrlopt & OPT_N)) {
    perror(" -n option required");
    exit(1);
}
if( num_rotors <= 0 ) {
   perror("error num_rotors < 0");
   exit(1);
```

```
}

if( num_rotors > MAX_ROTORS ) {
    perror("error number of rotors");
    exit(1);
}
alphabet=(char **)malloc(num_rotors);
for(i=0;i<num_rotors;i++){
     alphabet[i]=(char *)malloc(ALFABET_26);
}

for(i=0;i<num_rotors;i++){
   for(j=0;j<ALFABET_26;j++){
        alphabet[i][j]=0;
   }
}

if ((fd1 = fopen("resultados.txt", "r")) == NULL) {
    perror("file resultados.txt is not created"); exit(1);}
if ((fd2 = fopen("cadenas.txt", "w")) == NULL) {
    perror("error to create cadenas.txt"); exit(1);}

while(!feof(fd1)){
     bzero(buff,sizeof(buff));
     fgets(buff,256,fd1);
     for(i=0;i<num_rotors;i++){
         if(alphabet[i][buff[i]-'A']==0){
             alphabet[i][buff[i]-'A']=buff[i+num_rotors];
         }
     }
}
for(i=0;i<num_rotors;i++){
    fprintf(fd2,"rotor %d\n",(i+1));
    for(j=0;j<ALFABET_26;j++)
       fprintf(fd2,"%c",(j+'A'));
    fprintf(fd2,"\n");
    for(j=0;j<ALFABET_26;j++){
       fprintf(fd2,"%c",(alphabet[i][j]));
    }
    fprintf(fd2,"\n");
}
for(j=0;j<num_rotors;j++){
    fprintf(fd2,"\ntabla %d\n",j+1);
    for(i=0;i<ALFABET_26;i++){
        temp=i;
        do{
             conexiones++;
             temp=(alphabet[j][temp])-'A';
        }while(i!=temp);
        fprintf(fd2,"%c\t%d\tconexiones\n",i+'A',conexiones);
        conexiones=0;
    }
}
fputs("\n**********************************\n",fd2);
fclose(fd1);
fclose(fd2);
}
```

11. diccionario.c

```
//--- diccionario.c -----

#include "cripto.h"
#include <time.h>
#define ROT_OPT 2

#define MAX_ROTORS 12


char reflector[] = "AABBCCDDEEFFGGHHIIJJKKLLMM \n";
static char * init;
char steckerboard[] = "  \n";
static char ** rotors;
static int num_rotors = 3;
static char ** config_rotors;
static int * comb_base;
static int ** combinaciones;

static char *prgname;
static char version[]="1.0";
char usage[] = "Usage: %s  -e configfile name -n num_rotors [-h help]\n";

int main(int argc, char ** argv){

char condicion=1;
int c=0,i=0,j=0,z=0;
int cuenta=0;
int * p;
int temp;
int ctrlopt=0;
char * name_conf;
char sentencia [256];
char sentencia2 [256];
char sentencia3 [256];
char string [256];
char ** rotors_aux;
while(1)
{
    c = getopt(argc, argv, "n:e:h");
    if (c == EOF) break;
    switch (c)
    {
        case 'h': printf (usage, prgname);
                  fputs ("-h\t\tprint a options summary\n",
                         stdout);
                  fputs ("-e\t\tname of the config file\n",
                         stdout);
                  fputs ("-n\t\trotors number\n", stdout);
                         exit (0);
        case 'n': num_rotors = atoi(optarg);
                  ctrlopt |= OPT_N;
                  break;
        case 'e': name_conf = optarg;
                  ctrlopt |= OPT_E;
                  break;
        default : printf(usage, prgname);
                  exit (0);
    }
}
```

```
if (!(ctrlopt & OPT_N)) {
    perror(" -n option required");
    exit(1);
}

if (!(ctrlopt & OPT_E)) {
    perror(" -e option required");
    exit(1);
}

if( num_rotors <= 0 ) {
    perror("error num_rotors < 0");
    exit(1);
}

if( num_rotors > MAX_ROTORS ) {
    perror("error number of rotors");
    exit(1);
}

comb_base=(int *)malloc(num_rotors);
for(i=0;i<num_rotors;i++){
    comb_base[i]=1;
}

rotors=(char **)malloc(num_rotors);
rotors_aux=(char **)malloc(num_rotors);
combinaciones=(int **)malloc(factorial(num_rotors));
config_rotors=(char **)malloc(num_rotors);
for(i=0;i<num_rotors;i++){
    rotors[i]=(char *)malloc(ALFABET_26);
    config_rotors[i]=(char *)malloc(ALFABET_26);
    rotors_aux[i]=(char *)malloc(ALFABET_26);
}
for(i=0;i<factorial(num_rotors);i++){
    combinaciones[i]=(int *)malloc(num_rotors);
}
init=(char *)malloc(num_rotors);
file_Correct(name_conf);
fill_rotors(name_conf);
rellenar_combinaciones();

for(i=0;i<num_rotors;i++){
     for(j=0;j<ALFABET_26;j++){
        rotors_aux[i][j]=rotors[i][j];
     }
}


for(z=0;z<factorial(num_rotors);z++){
    string[0]=0;
    for(j=0;j<num_rotors;j++){
        sprintf(sentencia3,"%d ",combinaciones[z][j]);
        strcat(string,sentencia3);
    }
    sprintf(sentencia3,"echo \"rotor position %s\n
                            \">>diccionario.txt",string);
    system(sentencia3);
    for(i=0;i<num_rotors;i++){
         init[i]=0;
    }
```

```
    for(i=0;i<num_rotors;i++){
          for(j=0;j<ALFABET_26;j++){
                rotors[i][j]=rotors_aux[(combinaciones[z][i])-1][j];
          }
    }

    while(condicion){
        string[0]=0;
        for(j=0;j<num_rotors;j++){
             sprintf(sentencia3,"%c ",init[j]+'A');
             strcat(string,sentencia3);
        }
        sprintf(sentencia3,"echo \"\ninit position
                       %s\n \">>diccionario.txt",string);
        system(sentencia3);
        create_enigma_conf();
        sprintf(sentencia,"./gen_ficheros_cifrados
                                    -n %d",num_rotors);
        system(sentencia);
        sprintf(sentencia2,"./calc_cadenas -n %d",num_rotors);
        system(sentencia2);
        system("cat cadenas.txt >> diccionario.txt");
        system("rm cadenas.txt");
        system("rm resultados.txt");
        if(final())condicion=0;
        else rotate();
   }
   condicion=1;
}
puts("\nFIN\n");
}

int final(){
  int i=0;
  int temp='Z'-'A';
  for(i=0;i<num_rotors;i++){
    if(init[i]!=temp)return 0;
  }
  return 1;
}



rellenar_combinaciones(){

  int i=0,j=0,w=0,k=0,valid=1;
  char condicion=1;
  char aux [num_rotors];
  for(j=0;j<num_rotors;j++){
     aux[j]=0;
  }
  for(i=0;i<factorial(num_rotors);i++)
      for(j=0;j<num_rotors;j++){
          combinaciones[i][j]=0;
      }
  i=0;
```

```
  while(condicion){
     valid=1;
     for(j=0;j<num_rotors;j++){
          aux[comb_base[j]-1]++;
}
for(j=0;j<num_rotors;j++){
    if(aux[comb_base[j]-1]!=1)valid=0;
}

if(valid){
   for(j=0;j<num_rotors;j++){
   combinaciones[w][j]=comb_base[j];
}
w++;
}
if(w==factorial(num_rotors))condicion=0;
else if(comb_base[i]==num_rotors){
      comb_base[i]=1;
      k=1;
      do{
          comb_base[i+k]++;
          if(comb_base[i+k]>num_rotors){
            comb_base[i+k]=1;
            k++;
          }else break;
      }while((i+k)!=num_rotors);
}
        else comb_base[i]++;


for(j=0;j<num_rotors;j++){
    aux[j]=0;
}
}

}

create_enigma_conf(){
     FILE *fd;
     int i=0,j=0;
     fd=fopen("enigma.conf","w");
     fputs("[rotors] \n",fd);
     for(i=0;i<num_rotors;i++){
               for(j=0;j<ALFABET_26;j++){
               fprintf(fd,"%c",rotors[i][j]);
      }
      fputs(" \n",fd);
       }
      fputs("[reflector] \n",fd);
      fputs(reflector,fd);
      fputs(" \n",fd);
      fputs("[init_rotors] \n",fd);
      for(j=0;j<num_rotors;j++){
               fprintf(fd,"%c",init[j]+'A');
      }
      fputs(" \n",fd);
      fputs("[steckerboard]\n",fd);
      fputs(steckerboard,fd);
      fputs(" \n",fd);
      fclose(fd);
}
```

```
fill_rotors(char *name_conf){

   int c,i,j;
   FILE * fd_conf;
   if ((fd_conf = fopen(name_conf, "r")) == NULL) {
         perror("error open config file");
         exit(1);
   }
   char  buff [1024];
   char * t;
   while(!feof(fd_conf)){
       fgets ( buff,1024,fd_conf);
       if((t=strstr(buff,"[rotors]"))!=NULL){
            for(i=0;i<num_rotors;i++)
   for(j=0;j<ALFABET_26;j++){
     if(!feof(fd_conf)){
     if( !isalpha( c=getc(fd_conf) ) ){ j-- ; continue; }
     rotors[i][j]=toupper(c);
     config_rotors[i][j]=toupper(c);
   }
}
       }
   }
   fclose(fd_conf);
}




rotate(){
   int i=0;
   char condicion=1;
   while(condicion){
       if(i==num_rotors)condicion=0;
       else{
          init[i]++;
          if(init[i]==ALFABET_26){
               init[i]=0;
               i++;
               }else condicion=0;
          }
       }
}

file_Correct(char * name_conf){
   int config_ctrl=0;
   char * p;
   int i,j,c;
   FILE * fd;
   char buff [1024];
   int alphabet [26];
   char * aux;
   int count=0;

   if ((fd = fopen(name_conf, "r")) == NULL) {
   perror("error open config file");
   exit(1);}
```

```
while (!feof(fd)){
   for(j=0;j<26;j++)alphabet[j]=0;
      p=fgets(buff,1024,fd);
      if(buff[0]=='#')continue;
      else if(buff[0]==' '){
         while(1){
             c=*p;
             p++;
             if(c==' ')continue;
             else if(c=='\n'|c=='#')break;
             else {
                 perror("sintaxy config file error");
                 exit(1);}
             }
      }
      else if(buff[0]=='\n')continue;
      else if((aux=strstr (buff,"[rotors]"))!=NULL){
          if((config_ctrl&ROT_OPT)){
               perror("sintaxy config file error");
               exit(1);}
          config_ctrl |= ROT_OPT;
          p=strchr(aux,']');
          while(1){
               p++;
               if ((p[0]=='#')|(p[0]=='\n'))break;
               if(p[0]==32)continue;
               else{perror("sintaxy config file error");exit(1);}
          }
          for(i=0;i<num_rotors;i++){
          if(!feof(fd)){
              p=fgets(buff,1024,fd);
          }else{
              perror("sintaxy config file error");
              exit(1);}
          if(!isalpha(buff[0])){
              perror("sintaxy config file error");
              exit(1);}
          while(1){
              c=*p;
              p++;
              if(isupper(c)){
                 if(alphabet[c-'A']>i){
                      perror("sintaxy config file error");
                      exit(1);}
                 alphabet[c-'A']++;
              }
              else if(c==' ')continue;
              else if(c=='\n'|c=='#')break;
              else{
                 perror("sintaxy config file error");
                 exit(1);}
              }
for(j=0;j<26;j++)if(alphabet[j]<i+1){
   perror("sintaxy config file error");
   exit(1);}
   p=NULL;
}
}
else{
perror("sintaxy config file error");
```

```
    exit(1);}
   bzero(buff,1024);
 }
 fclose(fd);
}

int factorial(int numero){
int i=0;
int calculo=1;
for(i=1;i<=numero;i++){
calculo=calculo*i;
}
return calculo;
}
```

Conclusión

Para finalizar este trabajo sobre criptografía clásica hay que destacar que se han cumplido todos los objetivos propuestos anteriormente, dando una visión detallada de cada uno de los métodos analizados así como de La Enigma, objetivo final de este proyecto.

También se han analizado las formas de criptoanálisis de estos métodos desarrollando software que ayuda al criptoanálisis de los sistemas estudiados con numerosos ejemplos para facilitar la comprensión de estos.

Se ha desarrollado la Guía de Practicas que ayuda al uso de los programas, con ejemplos de uso de estos con textos para poder cifrar y descifrar usando las diferentes herramientas.

Por ultimo, destacar los conocimientos aprendidos tanto en programación como en tecnicas de cifrado y criptoanálisis

Bibliografía

- “Cryptography Demystified” de Jonh E. Hershey
- Libro Electrónico de Criptografía Clásica Escuela Universitaria de Informática Universidad Politécnica de Madrid
  www.criptored.upm.es/software/sw_m001a.htm
- http://es.tldp.org/Manuales-LuCAS/doc-unixsec/unixsec-html/node304.html
- http://www.criptored.upm.es/guiateoria/gt_m001a.htm
- Curso de Seguridad Informática y Criptografía por la Universidad Politécnica de Madrid