Introducción a la Criptología

José Galaviz Casas
Departamento de Matemáticas,
Facultad de Ciencias.

Arturo Magidin
Instituto de Matemáticas

Universidad Nacional Autónoma de México

Índice general

Introducción

Mientras exista el lenguaje, existirán las comunicaciones confidenciales: mensajes para una audiencia limitada. El principal problema es cómo llevar a cabio una comunicación confidencial. Si simplemente escribimos un mensaje, cualquiera lo puede leer y enterarse de su contenido.

Tradicionalmente, hay dos maneras en que se busca resolver este problema. El primero es esconder la existencia misma del mensaje: podríamos escribirlo con tinta invisible, por ejemplo. El estudio de comunicaciones donde se busca esconder la existencia misma del mensaje se llama *esteganografía.* Es el método favorito de los amantes clandestinos, y podemos ver en las tragedias teatrales las debilidades inherentes al método.

El segundo método consiste en cifrar o codificar el mensaje. En este caso, en vez de esconder la existencia del mensaje, se busca esconder su contenido, o hacerlo inintelegible para receptores no autorizados. El estudio de los sistemas de cifrado y su descifrado se llama *criptología.*

La criptología se divide a sí misma en dos partes: la *criptografía*, que es el estudio y diseño de sistemas y métodos para cifrar mensajes; y el *criptoanálisis*, que es el estudio del descifrado no autorizado de mensajes cifrados.

Estas notas están basadas en un curso que los autores impartieron en la Facultad de

Ciencias de la Universidad Nacional Autónoma de México entre enero y mayo de 2002. Se trata de una introducción a la criptología.

Empezamos estudiando los sistemas criptográficos clásicos: las substituciones monoalfabéticas (aquellas donde se usa un mismo alfabeto tanto para escribir como para cifrar), y vemos también su criptoanálisis. Procedemos después a las substituciones polialfabéticas, donde se usan varios alfabetos distintos para el cifrado, tanto desde el punto de vista criptográfico como criptoanalítico. Hacemos lo mismo para los sistemas poligráficos.

Estudiamos después otros sistemas históricos de criptografía, incluyendo sistemas mecánicos utilizados durante la Segunda Guerra Mundial, y el cifrado de Vernam. Brevemente mencionamos otros sistemas clásicos, como los códigos y los métodos de substitución.

En la segunda parte de las notas, estudiamos algunos de los sistemas modernos de criptografía, a saber el Estándar de Cifrado DES, y varios sistemas de criptografía de llave pública. También estudiamos el criptoanálisis de estos sistemas, poniendo especial énfasis enel criptoanálisis de DES y de RSA.

Durante las presentaciones, ponemos énfasis en los métodos matemáticos asociados tanto a la criptografía como al criptoanálisis. Si bien las matemáticas tienen un rol muy claro en sistemas modernos como RSA, resulta que también juegan un papel muy importante en la criptografía y criptoanálisis de los sistemas clásicos.

Terminamos hablando un poco sobre sistemas híbridos y protocolos de autentificación. Incluimos también apéndices que cubren datos estadísticos sobre el español, información sobre curvas elípticas, y otros temas de importancia para el texto pero que no forman parte directa de la presentación en el mismo.

Convenciones y nomenclatura

Vamos a utilizar las siguientes convenciones durante el texto:

Mensaje Colección de texto.

Texto claro El texto original del mensaje que se busca cifrar.

Criptotexto El texto resultante de cifrar el texto claro. También llamado **texto cifrado**.

Cifrar Transformar el texto claro en texto cifrado.

Sistema Criptográfico Un mecanismo o algoritmo para transformar el texto claro en criptotexto. También llamado **criptosistema**.

Alfabeto Colección de símbolos que se utilizan para escribir el texto, ya sea el texto claro o el criptotexto; los alfabetos usados para el texto claro y para el criptotexto no son necesariamente los mismos, aunque a menudo lo son.

Llave Información necesaria para descifrar un criptotexto.

Criptoanálisis La lectura no autorizada de mensajes obtenidos mediante un sistema criptográfico.

Enemigo Un recepto no autorizado que desea leer los mensajes cifrados, y quien no deseamos que los lea.

Es importante notar que las palabras **codificar** y **código** tienen un significado especial en Criptología, como veremos en el Capítulo 5; en particular, codificar **no** es lo mismo que cifrar, y es importante no confundir los términos.

Parte I

Sistemas Clásicos

Vamos a empezar con sistemas criptográficos que se utilizaron en la historia, empezando durante la República Romana, y hasta el Renacimiento. Todos los sistemas criptográficos de este capítulo comparten una característica: se tiene una correspondencia entre el alfabeto de escritura y el alfabeto de cifrado, y ésta correspondencia está fija durante todo el proceso de cifrado y de descifrado. Debido a ello, decimos que estamos usando únicamente un alfabeto de cifrado, lo que lleva al nombre de "substitución monoalfabético" o "cifrado monoalfabético."

1.1 Sistema criptográfico de César

Entre los sistemas criptográficos más antiguos de que tenemos noticia, destaca el sistema de cifrado cesáreo. Atribuido a Julio César, quien menciona que lo utilizó durante la Guerra de las Galias por el dominio de Europa central en contra de los pueblos celtas que las

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

Tabla 1.1: Alfabeto usado en la mayor parte del texto. Ocasionalmente añadiremos la letra Ñ. Las letras han sido indexadas (el índice es el número que aparece sobre la letra) a partir de cero.

habitaban. Suetonio en *La vida de los Césares* describe también el sistema utilizado por César.

El mecanismo de cifrado de César supone la existencia de un alfabeto, aquel en el que se escriben los mensajes que se desea cifrar. Este alfabeto es también utilizado para escribir los mensajes cifrados y se supone ordenado, en orden alfabético creciente. Según algunas fuentes César utilizó el alfabeto griego, según otras el latino de 24 letras. Nosotros usaremos aquí el alfabeto de 26 letras usual en la gran mayoría de los idiomas y que se muestra en la tabla 1.1.

Para cifrar un mensaje, además del alfabeto ordenado se requiere el mensaje mismo. El mensaje está escrito usando únicamente los símbolos del alfabeto. Un tercer elemento clave faltante para aplicar el mecanismo de cifrado de César: un número $k \in \{0, 1, \ldots, n-1\}$ donde n es el número de símbolos en el alfabeto (el tamaño del alfabeto). Este número es utilizado para desplazar el alfabeto mismo y obtener así una correspondencia entre los símbolos del mensaje original y los que se colocarán en el mensaje cifrado. Es decir, siempre que se encuentre el i-ésimo símbolo del alfabeto en el mensaje original, este será reemplazado por el símbolo $(i + k)$-ésimo del alfabeto en el mensaje cifrado. El desplazamiento del alfabeto es un mecanismo practico para obtener la correspondencia, en la tabla 1.2 se muestra la correspondencia del alfabeto para $k = 3$. Al parecer César siempre utilizó $k = 2$, pero nosotros podemos ser más generales.

Como se muestra en la Tabla 1.2, cada vez que en el texto original aparece la letra a este símbolo debe sustituirse en el mensaje cifrado por el símbolo `D`, la b por la `E` y así hasta llegar a la w que se substituye por la `Z`[1]. Al llegar a este punto se nos ha acabado el alfabeto leyéndolo linealmente, nos faltan por cifrar la x, la y y la z. Usamos para cifrar

[1]Hemos usado itálicas minúsculas para las letras del alfabeto en el mensaje original y mayúsculas tipográficas para las del cifrado, esto se hace por claridad y por razones históricas. Las letras mayúsculas han sido tradicionalmente usadas en los mensajes cifrados porque estos eran transmitidos generalmente por telégrafo y las máquinas de escribir de los telegrafistas, con las que eran transcritos los mensajes, sólo poseían mayúsculas.

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

Tabla 1.2: Correspondencia en cifrado de César con un desplazamiento de 3. Por claridad el alfabeto del texto original se ha puesto en minúsculas e itálicas, y el del texto cifrado en mayúsculas en tipo de letra de máquina de escribir. Esta convención la respetaremos a lo largo del texto.

éstas a la `A`, la `B`, y la `C`, respectivamente; i.e. le "damos vuelta al alfabeto", nos regresamos al principio. Esto lo hacemos intuitivamente porque nos faltan exactamente tres letras y las tres primeras son las únicas que no hemos usado. Formalmente hablando lo hacemos porque deseamos que la asociación de letras sea una función biyectiva.

Sabemos que el mensaje original debe estar escrito usando solamente símbolos del alfabeto. Como *cualquier* símbolo del alfabeto puede aparecer en un mensaje, esto significa que no podemos dejar símbolos del alfabeto original sin cifrar: para todo símbolo debemos estar seguros que otro lo va a reemplazar. Pero además si el destinatario recibe el mensaje cifrado debe poder descifrarlo correctamente, sin ambigüedad. Entonces, siempre que se encuentre algún símbolo debe estar seguro de que uno y sólo un símbolo es el que le corresponde en el mensaje original. No debe ocurrir que si ve una `B` no sepa cuál de un conjunto de letras poner para descifrar la `B`. El proceso debe ser invertible: la función de cifrado debe ser biyectiva.

Con la substitución de la tabla 1.2 podemos cifrar un mensaje sencillo como:

cuando despertó, el dinosaurio todavía estaba allí

Lo primero que debemos hacer es echar a perder la ortografía, vamos a quitar todos los acentos ortográficos. Luego reemplazamos cada letra por la que se le asocia según la tabla:

`FXDQGR GHVSHUWR HO GLQRVDXULR WRGDYLD HVWDED DOOL`

Si indexamos las letras del alfabeto, es decir les asignamos un número natural según su posición podemos trabajar más eficientemente. Esto ya lo hicimos en la Tabla 1.1.

a	b	c	d	e	f	g	h	i	j	k	l	m
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
A	C	E	G	I	K	M	O	Q	S	U	W	Y
n	o	p	q	r	s	t	u	v	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
A	C	E	G	I	K	M	O	Q	S	U	W	Y

Tabla 1.3: Correspondencia en cifrado de alfabeto decimado con un factor de 2.

Con el alfabeto indexado de esta manera podemos escribir explícitamente la formula que asocia un símbolo con otro para cifrar. La asociación mostrada en la tabla 1.2 es simplemente: $j \equiv (i + 3) \pmod{26}$ donde j es el índice del símbolo que cifra a aquel de índice i en el alfabeto. En general, si en un mensaje original se encuentra el símbolo de índice i de un alfabeto de n símbolos y al cifrarlo se le substituye por el símbolo de índice j del alfabeto desplazado k lugares a la derecha. Podemos simbolizar el cifrado mediante la fórmula $j \equiv (i + k) \pmod{n}$.

1.2 Cifrado con alfabeto decimado

En la sección anterior tratamos con el mecanismo de cifrado de César y concluimos que se podía expresar la función de asociación de símbolos del alfabeto como: $j = (i + k) \pmod{n}$. Podemos pensar ahora en un esquema similar, pero en el que el símbolo que reemplaza al i-ésimo es obtenido multiplicando i por algún número, en vez de sumar como en el esquema de César. Es decir, el i-ésimo símbolo es reemplazado en el texto cifrado por el j-ésimo, donde $j = (ki) \bmod n$. A este tipo de cifrado le denomina de *alfabeto decimado*[2].

Supongamos por ejemplo que $n = 26$ y $k = 2$. La asociación determinada por estos datos se muestra en la tabla 1.3.

Ciertamente esta asociación no sirve para cifrar, pues la función de cifrado no es biyectiva. Si el destinatario de una mensaje se encuentra con una *I* no sabe como descifrarla, pues tiene dos opciones: *r* y *e*. Tratemos con otro número. Por ejemplo, $k = 13$, que parece tan bueno como cualquier otro dado lo que sabemos hasta ahora.

La situación ahora es peor, pues sólo hay dos posibles resultados para la expresión

[2]Decimar es análogo a *diezmar*: eliminar uno de cada diez. En el esquema criptográfico de alfabeto decimado se elige para reemplazar uno de cada k símbolos en cada recorrido del alfabeto, y por ello el nombre.

$j = (ki) \pmod n$, a saber 0 y 13. Es decir sólo usaríamos A y N para cifrar todo el alfabeto.

Curiosamente los números que hemos probado son divisores de 26. Podríamos pensar que hay que elegir entonces a k de manera que no sea divisor de n.

Hagamos un experimento más pequeño. Supongamos que tenemos los números del 0 al 14, es decir trabajaremos módulo 15. Elegimos $k = 6$, que no divide a 15. Si nos ponemos a multiplicar y sacar módulos nos daremos cuenta de que la expresión $j = (6 \cdot i) \pmod{15}$, recorriendo todos los posibles valores de $i \in \{0, \ldots, 14\}$, sólo puede arrojar cinco resultados: 2, 8, 14, 5, y 11. Estos resultados se repiten periódicamente a lo largo de la secuencia $0, \ldots, 14$. El periodo de repetición, es decir, la distancia entre apariciones sucesivas del mismo número, es 5, que divide a 15; en efecto, $5|15$, pues $3 \times 5 = 15$. Curiosamente, 3 es el máximo común divisor de 6 y 15, en notación $\mathrm{mcd}(6, 15) = 3$. El periodo es $15/\mathrm{mcd}(6, 15)$. Eso nos podría hacer sospechar que el siguiente teorema es cierto:

Teorema 1.1 *La función $f\colon \mathbb{Z}_n \to \mathbb{Z}_n$, dada por*

$$j = f(i) = (ki) \pmod n,$$

donde $i, k \in \{0, \ldots, n-1\}$, es biyectiva si y sólo si $\mathrm{mcd}(k, n) = 1$. Es decir, si y sólo si k y n so primos relativos.

Necesitamos un pequeño lema para probarlo:

Lema 1.1 *Sean $a, b, c \in \mathbb{Z}$, y supongamos que $a|bc$. Si a y b son primos relativos, entonces $a|c$.*

Dem.: Como $\mathrm{mcd}(a, b) = 1$, existen enteros α y β tales que $\alpha a + \beta b = 1$. Multiplicando por c, tenemos que $\alpha ac + \beta bc = c$. Puesto que a divide a αac, y por hipótesis divide a βbc, se sigue que a divide a la suma, es decir, a c. □

Ahora podemos probar el teorema.

Dem.: Puesto que una función de un conjunto finito a sí mismo es biyectiva si y sólo si es inyectiva, si y sólo si es suprayectiva, vamos a probar que la función f es inyectiva si y sólo si k y n son primos relativos.

Supongamos que $i_1, i_2 \in \{0, \ldots, n-1\}$ son tales que $f(i_1) = ki_1 \equiv ki_2 = f(i_2) \pmod n$. Podemos suponer, sin pérdida de generalidad, que $0 \leq i_1 \leq i_2 < n$.

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

Tabla 1.4: Correspondencia en cifrado de alfabeto decimado con un factor de 5. La correspondencia es uno a uno.

Notamos que como $ki_1 \equiv ki_2 \pmod{n}$, tenemos que $n|ki_2 - ki_1 = k(i_2 - i_1)$.

Si k es primo relativo con n, entonces podemos usar el Lema 1.1 para concluir que $n|i_2 - i_1$. Pero $0 \leq i_2 - i_1 < n$, de manera que la única manera en que $n|i_2 - i_1$ es si $i_2 - i_1 = 0$; es decir, si k es primo relativo con n, entonces $f(i_1) = f(i_2)$ implica que $i_1 = i_2$, y f es inyectiva.

Conversamente, supongamos que k no es primo relativo con n, y sea $1 < d = \text{mcd}(k, n)$. Entonces $\frac{n}{d}$ y $\frac{k}{d}$ son enteros, y son menores que n y que d, respectivamente. Consideramos $f(0) = 0$ y $f(\frac{n}{d}) = \frac{nk}{d}$. Puesto que $\frac{nk}{d} = n\frac{k}{d}$, y $\frac{k}{d}$ es un entero, $\frac{nk}{d}$ es un múltiplo de n, y por lo tanto, $f(\frac{n}{d}) = 0$. Es decir, $f(0) = f(\frac{n}{d})$. Como $n > 0$, $\frac{n}{d} \neq 0$, de manera que f no es inyectiva.

□

Ahora sabemos qué se requiere para tener un esquema de cifrado con alfabeto decimado útil: hay que elegir el factor de decimación de tal forma que sea primo relativo al tamaño del módulo. Por ejemplo, si el factor es $k = 5$, con nuestros 26 símbolos tenemos la substitución mostrada en la Tabla 1.4

1.3 Cifrado afín

Podemos generalizar aún más nuestro método de cifrado. Podemos combinar los dos que tenemos hasta hora. En vez de sólo multiplicar o sólo sumar podemos hacer ambas cosas. Es decir, definimos la regla de substitución mediante la fórmula $j = (ri + k) \pmod{n}$, donde n es la cardinalidad de nuestro alfabeto, y $i, r, k \in \{0, \ldots, n-1\}$. A este tipo de substitución se le denomina cifrado o substitución afín.

Dado que en el esquema de cifrado afín también se utiliza un factor que multiplica al índice del símbolo a substituir, está sujeto a la misma restricción que el cifrado con alfabeto diezmado. Para que sea válido debe ocurrir que $\text{mcd}(r, n) = 1$. Dado que la suma solo desplaza los símbolos una cierta distancia, esta restricción basta para que la substitución sea biyectiva.

1.4 Alfabetos mezclados

Podemos generalizar aún más y definir el mapeo de substitución de una manera completamente aleatoria: desordenar completamente el alfabeto sin regla alguna y luego ponerlo en correspondencia con el alfabeto ordenado. Esta es, sin duda, la mayor generalización posible. A un esquema como éste se le llama cifrado por *alfabeto mezclado*.

Esto, sin embargo, trae consigo un problema práctico.

Si un par de personas pretenden usar un esquema criptográfico de alfabetos mezclados deben primero ponerse de acuerdo en el alfabeto desordenado que define la substitución. Este alfabeto no puede viajar, como un mensaje más entre los interlocutores: si el enemigo lo intercepta tiene toda la información para descifrar (y también cifrar) mensajes. Así que es necesario que los corresponsales se pongan de acuerdo en el alfabeto mezclado a utlizar antes de iniciar la correspondencia. Luego pueden irse a los lugares usuales desde donde envían y reciben mensajes. Pero de alguna manera deben poder recordar el alfabeto desordenado. Sería imprudente escribirlo en un papel que alguien pudiera robar, así que deben memorizarlo. Pero memorizar una secuencia completamente aleatoria de símbolos no es trivial.

Para resolver el problema de recordar la substitución sería mejor tener un método para generarlo a partir de alguna información más fácil de recordar. El método debe generar un alfabeto suficientemente desordenado. Hay varias maneras de hacerlo. La más común consiste en la elección de una palabra clave, que es la que intercambiamos con nuestro interlocutor en una reunión secreta, digamos `TRISCAYDECAFOBIA`. Si eliminamos las repeticiones de letras obtenemos `TRISCAYDEFOB`. Después escribimos el alfabeto que utilizamos en el orden usual y eliminamos aquellas letras que ya aparecen en `TRISCAYDEFOB`. Escribimos entonces las letras que quedan en el alfabeto en el orden en el que aparecen usualmente, y obtenemos:

`TRISCAYDEFOBGHJKLMNPQUVWXZ`,

lo que podemos usar como nuestro alfabeto desordenado definiendo así la correspondencia que aparece en la Tabla 1.5. A este tipo de alfabeto mezclado se le denomina un alfabeto *determinado por una palabra clave*. Por supuesto, no hay restricción en usar una palabra,

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

Tabla 1.5: Correspondencia en cifrado de alfabetos mezclados usando palabra la palabra clave `TRISCAYDECAFOBIA`.

y se puede usar una frase completa como base de la substitución.

Pero esto no es realmente muy aleatorio. Si el enemigo hace criptoanálisis de nuestros mensajes cifrados como veremos posteriormente, y logra adivinar suficientes letras de nuestra palabra clave probablemente con un golpe de vista logre adivinar las demás y se habrá ahorrado mucho trabajo. Para desordenar aún más el alfabeto podemos hacer lo siguiente: escribimos nuestra palabra clave sin repeticiones y luego procedemos a escribir bajo ella el alfabeto en el orden usual. Escribimos tantas letras del alfabeto como letras haya en la palabra clave sin repeticiones. Si faltan letras por escribir comenzamos de nuevo bajo la primera letra de la izquierda y hacemos esto hasta terminar con el alfabeto. Este procedimiento genera lo que mostramos a continuación:

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

Ahora procedemos a escribir el alfabeto leyendo cada *columna*, de arriba a abajo y de izquierda a derecha, del arreglo mostrado. Obtenemos:

`TGXRHZIJSKCLAMYNDPEQFUOVBW`,

bastante desordenado pero fácilmente reconstruible si se posee la palabra clave. Esta es la sucesión que usamos para definir la substitución: $a \mapsto$ `T`, $b \mapsto$ `G`, etc. Un alfabeto obtenido de esta manera se dice que es obtenido mediante *palabra clave mezclada*.

1.5 Complejidad y complicación

Supongamos que alguien pretende hacer criptoanálisis y trata de descifrar un mensaje cifrado con alguno de los métodos ya descritos sin tener la llave o información necesaria

para hacerlo: en el caso de cifrado de César, el desplazamiento; en el decimado, el factor; en el afín, el factor y el desplazamiento; en el alfabeto mezclado, la substitución o la manera de obtenerla. Si el criptoanalista tiene sólamente el texto cifrado, tendría que probar en principio varias posibles substituciones para cada símbolo del texto cifrado. Podemos preguntarnos ahora ¿cuántas posibles substituciones a lo más debe probar? O lo que es equivalente: si usáramos un programa que hiciera todas las posibles substituciones, dado el esquema criptográfico usado, ¿cuántas substituciones debe hacer?

En el caso de César es fácil hacer el cálculo. La substitución depende del desplazamiento, el valor de k. Como trabajamos módulo el tamaño del alfabeto, sumar usando $k = 33$ o $k = 59$ es lo mismo que usar $k = 7$, así que hay sólo 26 posibles valores para k. Uno de ellos, el cero, no hace substitución alguna; el criptotexto resultante es idéntico al texto claro. Así que tenemos 25 diferentes substituciones de César no triviales.

En el caso de alfabetos decimados podría parecer que la situación mejora, pues el proceso es más complicado. Pero de hecho no es así. Dada la restricción que el factor k debe ser primo relativo con el tamaño del alfabeto, las posibilidades se reducen drásticamente. Sólo hay doce números primos relativos con 26 menores que 26: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, y 25. El primero de ellos, correspondiente a $k = 1$, genera la substitución identidad, así que realmente sólo hay 11 posibilidades no triviales. ¡Menos que con cifrado de César!. Moraleja: *no es cierto que procesos más complicados generen mecanismos criptográficos más seguros.* La complicación y la complejidad no siempre van de la mano.

En el caso de substituciones afines, dado que estamos sujetos a la misma restricción que con los alfabetos decimados tenemos 12 posibilidades para elegir el factor. Luego aplicamos un desplazamiento, y éste puede tomar 26 diferentes valores. Tenemos entonces $12 \times 26 = 312$ posibilidades, de las cuales una es la identidad, así que en realidad tenemos 311 posibles substituciones afines no triviales.

El caso más general es el de alfabeto mezclado arbitrario. ¿Cuántas maneras de asociar las letras tenemos? Cada asociación está determinada por el orden que le asignemos a las letras en el alfabeto que ponemos bajo el alfabeto en orden convencional. Hay 26 maneras de elegir la primera letra de la izquierda, por cada una de estas 26 posibles elecciones hay 25 de la segunda letra, por cada una de estas 26×25 hay 24 de la tercera; en total hay 26! posibilidades:

$$26! = 403,291,461,126,605,635,584,000,000.$$

1.6 Criptografía y criptoanálisis

Hasta ahora hemos presentado diversos métodos para cifrar y descifrar mensajes. En todos ellos hay una característica común: dado un símbolo cualquiera del alfabeto, éste es reemplazado por un único símbolo del alfabeto. Cada vez que en el texto cifrado aparece una `M`, sabemos que habrá que reemplazarla por alguna letra, siempre la misma. De allí el calificativo *monoalfabéticos* para describir estos métodos de cifrado.

Procederemos ahora a hacer criptoanálisis de los sistemas monoalfabéticos. Es decir, presentaremos las técnicas usuales para descifrar mensajes cifrados cuando carecemos de la llave correspondiente. Pero antes conviene hacer algunas precisiones. Vamos a suponer que tenemos un criptotexto, que sabemos que fue cifrado mediante una substitución monoalfabética, y que no tenemos ninguna información adicional.

Para cifrar un mensaje usando el método de César se requiere, como hemos dicho, el alfabeto a utilizar en un orden predeterminado y el desplazamiento que define la función de reemplazo. Los mismos elementos son necesarios para que el destinatario de los mensajes pueda descifrarlos correctamente. Si damos por sentado que ambos interlocutores conocen el alfabeto ordenado, bastará que el receptor conozca, para cada mensaje recibido, el desplazamiento utilizado para cifrarlo; con eso basta para que pueda descifrar correcta y eficientemente el mensaje. Este elemento es al que llamaremos *la clave* o *la llave*.

Definición 1.1 La *clave* o *llave* de un sistema criptográfico es el conjunto de elementos necesarios para que el emisor de un mensaje pueda cifrarlo y el receptor puede descifrarlo eficientemente, usando los algoritmos correspondientes.

En los otros métodos de cifrado tenemos también los elementos similares: el alfabeto, el mensaje original, el mensaje cifrado, y la llave. En al caso de los alfabetos decimados la llave es el factor, en el cifrado afín es el factor y el desplazamiento, en los alfabetos mezclados es el alfabeto desordenado, o bien la palabra clave y el método para generarlo. En general tenemos el esquema mostrado en la Figura 1.1.

Hay que hacer énfasis en que cifrar poseyendo la clave y el texto original, o descifrar cuando estmos en posesión e la llave y el texto cifrado, es un procedimiento algorítmico. Sólo hay que seguir un conjunto ordenado de pasos que luego de un tiempo finito nos proporcionan lo que deseamos, a saber, el texto cifrado o el texto descifrado.

En contraste, si el enemigo logra capturar uno o varios mensajes cifrados pero no posee la clave para descifrarlos usando los algoritmos necesarios, entonces para tratar de descifrarlos tendrá que hacer un análisis del texto cifrado, y tendrá que hacer algunas hipótesis que

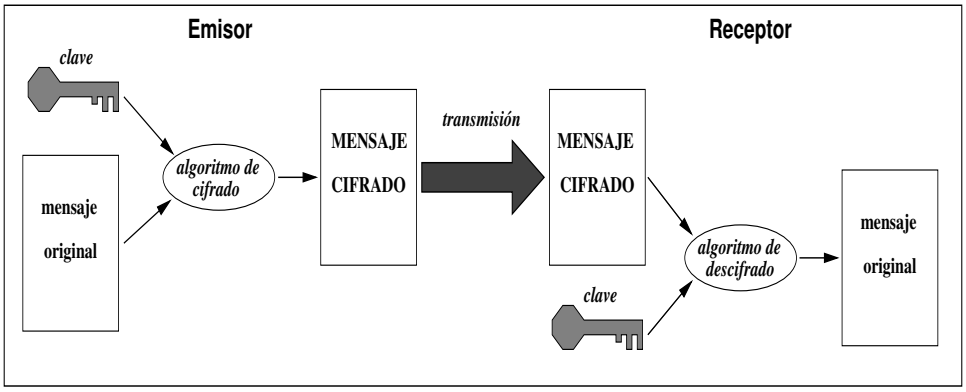

Figura 1.1: Esquema general de comunicación cifrada.

pueden o no ser ciertas acerca de las substituciones que se llevaron a cabo para obtener lo que el vé. Este, por supuesto, no es un procedimiento algorítmico, sino un proceso heurístico. Se plantean posibles soluciones parciales más o menos plausibles, se sigue un cierto camino que en ocasiones hay que desandar, nunca hay garantía de estar en la pista correcta; se da por terminado el proceso cuando se logra entender el mensaje recibido y se presume que es éste y no otro el que originalmente fue cifrado por el emisor. En este proceso la intervención humana es indispensable. Las computadoras nos pueden echar una mano haciendo algunas cosas por nosotros, como veremos, pero las decisiones inteligentes necesariamente las hacen las personas, al menos por ahora.

Cifrar y descifrar teniendo la clave es pues un procedimiento algorítmico, determinístico, criptográfico. Tratar de descifrar mensajes sin tener la clave es un procedimiento heurístico, azaroso, criptoanalítico. El proceso criptoanalítico necesariamente lo hace alguien, pero el proceso criptográfico puede hacerlo algo.

1.7 Criptoanálisis de sistemas monoalfabéticos

Procedemos ahora al estudio de criptoanálisis de los sistemas de substitución monoalfabética que hemos visto en éste capítulo. Puesto que los cifrados de César, diezmado, y afines son todos casos particulares del cifrado con alfabeto mezclado general, vamos a estudiar directamente el criptoanálisis de éste último.

1.7.1 Primer caso: cifrado monoalfabético con separación de palabras

Supongamos que recibimos el siguiente mensaje y que sospechamos que está cifrado usando un sistema monoalfabético:

```
GP AP OAUMW FG OM DMPIJM FG IATC
PCDMWG PC RAEGWC MICWFMWDG PC JMIG
DAIJC XEGDQC RAG YEYEM AP JEFMOUC
FG PCBOG IAPM RAG QCVGEM AP MPXEUAC
GUIAFC APM OMPSM GP MVXEOOGWC AP
WCIEP HOMIC T AP UMOUC ICWWGFCW.
```

El alfabeto es el usual de 26 letras que hemos estado usando y el mensaje cifrado ha conservado la separación de palabras del texto original. Suponemos también que el texto original estaba en español. Con todo esto en mente ¿Qué podemos hacer?

En cada idioma la frecuencia de uso de las letras es diferente. Es decir si nos fijamos en un texto largo escrito en inglés y contamos cuantas veces aparece en el texto cada letra, obtendremos ciertos números. Algunas letras serán más frecuentes que otras, y de hecho las podemos ordenar por frecuencia. Si en cambio hacemos lo mismo con un texto en español, las frecuencias de uso serán diferentes: habrá letras que son más frecuentes en español que en inglés y viceversa. Hablando en términos de teoría de la probabilidad, cada idioma tiene su propia distribución de probabilidades sobre el conjunto de letras del alfabeto.

Por otro lado, en los esquemas de substitución monoalfabética cada letra se mapea en otra y sólo una letra. Así que si el texto es suficientemente representativo de su idioma y se cifra con un sistema monoalfabético, entonces la letra más frecuente del idioma en que esté escrito el mensaje se mapeará a la letra más frecuente del texto cifrado, la segunda más frecuente del idioma a la segunda más frecuente del mensaje cifrado, etcétera. Claro está que eso de "suficientemente representativo" es un problema, ¿Qué significa tener un texto "típicamente inglés" o "típicamente español"? ¿De qué época? ¿De qué tema? ¿De qué estrato social? ¿De qué país? La respuesta es dificil, pero simplificamos, quizás demasiado, como se haría en estadística. Diremos que "suficientemente representativo" es equivalente a tener una "muestra suficientemente grande". Es decir, un texto largo. Entre más largo, lo supondremos más representativo.

Las frecuencias de aparición de las letras en el texto cifrado de arriba se muestran en la Tabla 1.6.

Veamos el texto cifrado. Hay que notar que hay una palabra de una sola letra `T`, observando las frecuencias de las palabras de una sola letra mostrada en el Apéndice A,

Letra	Frec.	%	Letra	Frec.	%
C	19	11.7	R	3	1.9
M	18	11.1	V	3	1.9
G	17	10.5	X	3	1.9
P	17	10.5	B	2	1.2
A	15	9.3	Q	2	1.2
I	10	6.2	T	2	1.2
E	9	5.6	Y	2	1.2
O	9	5.6	H	1	0.6
W	8	4.9	S	1	0.6
F	7	4.3	K	0	0.0
D	5	3.1	L	0	0.0
U	5	3.1	N	0	0.0
J	4	2.5	Z	0	0.0

Tabla 1.6: Tabla de frecuencias de los símbolos en el texto cifrado.

nos damos cuenta de que la más frecuente es la conjunción y así que podemos suponer que T $= y$. Palabras de dos letras hay varias: GP, AP, FG, OM, y PC. La P aparece la segunda y en la última de estas. Puesto que una palabra de dos letras en español tiene una vocal y una consonante, tenemos dos posibles suposiciones, ambas plausibles:

1. G y A son vocales y P es consonante; o
2. G y A son consonantes y P es vocal.

Pero la aparición de la palabra RAG (tercera palabra, tercer línea) nos hace pensar que la segunda opción no es muy buena, porque habría dos consonantes juntas en una palabra de tres letras. Así que mejor suponer que la primera opción es la buena. Si suponemos que P es consonante y G es vocal entonces también debemos suponer que F es consonante, porque de no serlo la palabra FG tendría dos vocales juntas. Luego notamos que la palabra OM y la aparición de APM (segunda palabra, quinto renglón) nos llevn a pensar que M es vocal, porque si no, en esa palabra de tres letras aparecerían dos consonantes juntas; suponer que M es vocal nos lleva a que O es consonante.

La aparición de la palabra YEYEM (cuarta palabra, tercer línea), nos hace pensar que une de Y y M es vocal. La palabra HOMIC (segunda palabra, sexta línea) nos lleva a pensar que M es vocal porque de otro modo tendría muchas consonantes juntas, algo que no ocurre en español.

En síntesis, creemos que son vocales las siguientes letras: `G`, `A`, `C`, `M`, y `E`; y que `P`, `F`, y `O` son consonantes.

La letra `P` es entonces una consonante muy frecuente en español, así que debe ser *n*, *r*, *s*, ó *t*. Pero al ver las palabras `GP`, `AP`, y `APM` y consultar nuestra tabla de frecuencias de palabras de dos y tres letras (Apéndice A), no podemos suponer sino que es la *n* (las de dos letras nos pueden hacer pensar que es *l*, pero no hay una palabra de tres letras con *l* en el centro). Por lo tanto, la palabra `RAG` es una palabra muy frecuente de tres letras, y tiene dos vocales juntas. Viendo nuestra tabla de frecuencias de palabras de tres letras sospechamos que es *que*. Así que tenemos: `P` = *n*, `R` = *q*, `A` = *u*, y `G` = *e*. Eso está medio feo porque la *e* es la letra más frecuente del español y en nuestro texto la `G` es muy frecuente, pero no la más frecuente: le gana la `M`, a la que estamos suponendo vocal también. Puede que estemos mal. Es algo que siempre hay que tener en mente. Veremos el resto del desciframiento con algo de escepticismo, siempre listos para regresar y hacer nuevas hipótesis, deshaciendo lo que hemos hecho, en caso necesario.

Ahora bien, la palabra de dos letras de mayor frecuencia, que termina en una vocal distinta de *e* es *la*, por lo que pensamos que la palabra `OM` corresponde a *la*, ya que estamos suponiendo que `O` es consonante y la `M` es vocal.

Sólo nos faltan dos vocales, la *i* y la *o*, que deben corresponder al la `C` y la `E` o viceversa. La *o* es más frecuente en español que la *i*, como en nuestro texto la `C` es más frecuente que la `E` suponemos que `C` = *o* y `E` = *i*.

Si ponemos las substituciones que tenemos arriba del texto tenemos lo siguiente[3]:

```
en un  u a   e  a  an  a  e  u o
GP AP OAUMW FG OM DMPIJM FG IATC

no   e no quie o a o  a  e no  a e
PCDBWG PC RAEGWC MICWFMWDG PC JMIG

 u  o  ie  o que  i ia un  i a  o
DAIJC XEGDQC RAG YEYEM AP JEFMOUC

  e no  e  una que  o eia un an i uo
 FG PCBOG IAPM RAG QCVGEM AP MPXEUAC

e  u o una  an a en as i  e o un
```

[3]Violamos aquí nuestra convención de escribir el texto claro en letra itálica debido a cuestiones tipográficas

```
GVIAFC APM OMPSM GP MVXEOOGWC AP

 o in   a o y un  a  o  o  e o
WCIEP HOMIC T AP UMOUC ICWWGFCW
```

Se ve bastante bien por ahora: las vocales están bien distribuidas, y algunos conectivos están bien hechos. La distribución de las vocales es una muy buena indicación de que al menos hemos determinado correctamente cuáles letras del criptotexto corresponden a vocales.

Si nos fijamos ahora en la tercera palabra de la segunda línea, podemos pensar que `W` $= r$.

La `I` aparece mucho antecediendo a la letra que creemos es la u. La letra que más frecuentemente tiene a la u a su inmediata izquierda es la q, pero ésta ya está asignada. El segundo lugar lo tiene la s (véase la tabla de frecuencias de digramas en el Apéndice A). Pero la tercerr palabra de la cuarta línea sería *suna*, algo que no tiene mucho sentido. A la s como antecesor de u le sigue la p en frecuencia, pero *puna* tampoco es una buena opción. La siguiente letra en frecuencia es la c, y *cuna* parece razonable.

Si hacemos las substituciones `W` $= r$ y `I` $= c$, la última palabra del texto queda *corre?or*, por lo que suponemos que `F` $= d$. Si hacemos esta última substitución, la primera palabra de la quinta línea queda *e?cudo*, por lo que hacemos la substitución `V` $= s$. Y en la última palabra de la segunda línea tenemos *?ace*, por lo que suponemos `J` $= h$. La cuarta palabra de la segunda línea dice *acordar?e*, y la primera de la línea tres dice *?ucho*, por lo que suponemos que `D` $= m$. El estado actual de nuestro descifrado es el siguiente:

```
en un  u ar de  a mancha de cu o
GP AP OAUMW FG OM DMPIJM FG IATC

nom re no quiero acordarme no hace
PCDBWG PC RAEGWC MICWFMWDG PC JMIG

mucho  iem o que  i ia un hida  o
DAIJC XEGDQC RAG YEYEM AP JEFMOUC

de no  e cuna que  oseia un an i uo
FG PCBOG IAPM RAG QCVGEM AP MPXEUAC
```

```
escudo una  an a en as i  ero un
GVIAFC APM OMPSM GP MVXEOOGWC AP

rocin   aco y un  a  o corredor
WCIEP HOMIC T AP UMOUC ICWWGFCW
```

Quizás el lector ya habrá descifrado el texto completo, pero continuemos como si aún lo hemos hecho. Lo siguiente que podemos hacer es ver la primera palabra de la segunda línea. Parece evidente que `B` $= b$. La quinta palabra de la cuarta línea sugiere `Q` $= p$. Haciendo estas dos substituciones, notamos ahora que la segunda palabra palabra de la tercer línea sugiere que `X` $= t$; y la segunda palabra de la cuarta línea y, con mayor razón, la penúltima palabra de la quinta línea, sugieren `O` $= l$. Ahora tenemos:

```
en un lu ar de la mancha de cu o
GP AP OAUMW FG OM DMPIJM FG IATC

nombre no quiero acordarme no hace
PCDBWG PC RAEGWC MICWFMWDG PC JMIG

mucho tiempo que  i ia un hidal o
DAIJC XEGDQC RAG YEYEM AP JEFMOUC

de noble cuna que poseia un anti uo
FG PCBOG IAPM RAG QCVGEM AP MPXEUAC

escudo una lan a en astillero un
GVIAFC APM OMPSM GP MVXEOOGWC AP

rocin  laco y un  al o corredor
WCIEP HOMIC T AP UMOUC ICWWGFCW
```

Ya es fácil completar: `U` $= g$, `T` $= y$, `Y` $= v$, `S` $= z$, y `H` $= f$.

Nuestro texto descifrado queda:

```
en un lugar de la mancha de cuyo
GP AP OAUMW FG OM DMPIJM FG IATC

nombre no quiero acordarme no hace
```

```
PCDBWG PC RAEGWC MICWFMWDG PC JMIG

mucho tiempo que vivia un hidalgo
DAIJC XEGDQC RAG YEYEM AP JEFMOUC

de noble cuna que poseia un antiguo
FG PCBOG IAPM RAG QCVGEM AP MPXEUAC

escudo una lanza en astillero un
GVIAFC APM OMPSM GP MVXEOOGWC AP

rocin flaco y un galgo corredor
WCIEP HOMIC T AP UMOUC ICWWGFCW
```

Difícilmente podríamos encontrar algo más representativo del español (aunque sea del español del siglo de oro).

Ahora podríamos preguntarnos: ¿cómo fue construida la regla de substitución? ¿Habrá alguna manera de generar el alfabeto desordenado que se utilizó?

Escribamos nuestra asociación como se muestra a continuación, para ver si encontramos algún patrón:

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

Notemos la asociación de la `W`, la `X`, y la `Y`. Esas letras son contiguas en el alfabeto usual, pero en la asociación que descubrimos están separadas por un espacio. Algo así nos ocurrió cuando usamos `TRISCAYDECAFOBIA` para construir nuestro alfabeto desordenado. Si sólo nos fijamos en las posiciones impares de la asociación mostrada surge un patrón: $a =$ `M`, $c =$ `I`, $e =$ `G`, $g =$ `U`, $i =$ `E`. Podríamos apostar a que $k =$ `L`, una letra que nunca aparece en el texto cifrado. Lo que aparentemente se hizo para construir el alfabeto mezclado fue mediante palabra clave mezclada, con la llave `MIGUELDECERVANTES`. Una vez eliminadas las repeticiones, nuestro arreglo de columnas es:

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

La secuencia de letras que queda es entonces:

MBIFGHUJEKLODPCQRWVXAYNZTS.

1.7.2 Segundo caso: cifrado monoalfabético en bloques

En la sección anterior vimos que el criptoanálisis se facilitó mucho gracias a la separación de palabras, que se conservó luego de cifrar el mensaje. Por ejemplo, nos permitió identificar vocales, y deshechar posibles asignaciones que creaban palabras sin sentido.

Un criptógrafo sensato eliminará dicha división. Lo típico es encontrar un criptotexto donde las divisiones de palabras han desaparecido. La manera más común es reemplazarlas por divisiones artificiales fijas. El criptotexto entonces consiste en un número de bloques de tamaño fijo. Los bloques de cinco letras son comunes, nuevamente por razones históricas: los telegrafos cobraban los telegramas por palabra, y había reglas indicando que grupos de letras sin sentido no podían ser mayores de cinco letra.

Veamos un ejemplo. Supongamos que tenemos el siguiente criptograma que proviene de un texto en español:

```
DHPLH UJNIO PFTON OGFTL SJPVO
LOPJG QOPIO MTOKA ONIOG DJVAI
JDHPV JDAOG QOPGH OXKON AFOGQ
JGDJF TONQO FJPMT OTGJP HDJVO
ZRTDA HLOPJ NOPLO GJPOE TGIJJ
LQHPO ETGIH WADDA JFPSJ BOPKO
JNO
```

¿Por dónde empezamos ahora? En el caso anterior, las palabras cortas nos dieron una manera sencilla de empezar e identificar vocales. Pero ahora no tenemos división de palabras, así que no tenemos manera de identificar esas palabras cortas. Tendremos que recurrir a la estructura típica del idioma en que fue escrito el mensaje original. Nos referimos a cosas como que la b nunca va antecedida de n, o que la q siempre va seguida de u, por ejemplo. Esta última es una buena manera de empezar porque sabemos que la palabra *que* es muy frecuente en español (véase Apéndice A). Podemos tratar de identificar a la q y así tendremos a la u y probablemente alguna otra vocal (e ó i). A fin de cuentas las vocales son las letras más frecuentes y se reparten por todo el texto. Su espaciamiento también es fácil de identificar, pues cada sílaba debe tener al menos una vocal. Nuestro objetivo principal es entonces primero identificar a las vocales; la q es un objetivo secundario, pero veremos que

Letra	Frecuencia	Porcentaje	Letras	Frecuencia	Porcentaje
O	27	17.65	V	4	2.61
J	18	11.76	K	3	1.96
P	15	9.80	E	2	1.31
G	11	7.19	M	2	1.31
D	9	5.88	S	2	1.31
T	9	5.88	B	1	0.65
H	8	5.23	R	1	0.65
A	7	4.58	U	1	0.65
N	7	4.58	W	1	0.65
F	6	3.92	X	1	0.65
I	6	3.92	Z	1	0.65
L	6	3.92	C	0	0.00
Q	5	3.27	Y	0	0.00

Tabla 1.7: Tabla de frecuencias del segundo criptograma.

la información que tenemos que acumular para identificar las vocales nos permite también tratar de identificar a la *q*.

Lo primero que hacemos es observar la frecuencia de aparición de cada letra. Luego vamos a observar qué letras se juntan con cuáles y cuántas veces lo hacen. Cuando una letra va precedida o sucedida por otra se dice que hace contacto con ella. Así que necesitamos conocer los contactos de cada letra en el criptotexto y luego observar la frecuencia con la que cada letra hace contacto con cada otra. Esto se llama la frecuencia de *digramas* (bloques de dos letras). En la Tabla 1.7 se muestran las frecuencias de aparición de las letras en el texto cifrado, y en las Tablas 1.8–1.11, las frecuencias de los distintos digramas encontrados. En éstas últimas, en la columna izquierda debajo de cada letra aquellas que la preceden, junto a la frecuencia de tal digráfica; la columna de la derecha indica las letras que la suceden junto a su frecuencia. Por ejemplo, el criptotexto `A` está precedido por `K` una vez, y por `D` tres veces; mientras que el criptotexto `F` está sucedido por `J` una vez, y por `T` tres veces.

Para identificar a la *q*, buscamos letras que tengan sólo un único sucesor. Para identificar a las vocales, recordamos que las vocales tienden a ser sucesores y antecesores de todas las consonantes, pero que no se mezclan mucho entre sí; por otro lado, es más común que una consonante tenga pocos vecinos distintos, y que en su mayoría sean vocales. Buscamos entonces letras frecuentes, con muchos distintos vecinos, que no tengan mucho contacto entre ellos.

A		B		C		D		E		F		G	
K:1	D:1	J:1	O:1			G:2	A:3	O:2	T:2	P:1	J:1	O:5	D:2
V:1	F:1					J:2	D:1			G:1	O:1	J:2	F:1
D:3	H:1					H:1	H:2			A:1	P:1	P:1	H:1
N:1	I:1					T:1	J:3			J:2	T:3	T:3	I:2
W:1	J:1					A:1				O:1			J:2
	O:2					D:1							Q:3

Tabla 1.8: Contactos de las letras A–G en el criptograma.

H		I		J		K		L		M	
D:2	D:1	N:2	H:1	U:1	B:1	O:1	A:1	P:2	H:1	O:1	T:2
L:1	L:1	P:1	J:2	S:2	D:2	X:1	O:2	T:1	O:3	P:1	
G:1	O:1	A:1	O:2	P:2	F:2	P:1		O:1	Q:1		
P:1	P:3	G:2		D:3	G:2			H:1	S:1		
A:1	U:1			I:2	J:1			J:1			
Q:1	W:1			V:1	L:1						
I:1				Q:1	N:3						
				F:1	P:4						
				G:2	V:2						
				J:1							
				A:1							
				O:1							

Tabla 1.9: Contactos de las letras H–M en el criptograma.

N		O		P		Q		R		S	
J:3	A:1	I3	E:2	H:3	F:1	G:3	H:1	Z:1	T:1	L:1	J:2
O:4	I:2	T:4	F:1	O:7	G:1	N:1	J:1			P:1	
	O:3	N:3	G:5	J:4	H:1	L:1	O:3				
	Q:1	V:2	J:1	F:1	I:1						
		L:3	K:1		J:2						
		Q:3	L:1		K:1						
		A:2	M:1		L:2						
		H:1	N:4		M:1						
		K:2	P:7		O:2						
		F:1	T:1		S:1						
		P:2	X:1		V:2						
		B:1	Z:1								

Tabla 1.10: Contactos de las letras N–S en el criptograma.

T		U		V		W		X		Y		Z	
F:3	D:1	H:1	J:1	P:2	A:1	H:1	A:1	O:1	K:1				
M:2	G:3			J:2	J:1								
O:1	L:1				O:2								
R:1	O:4												
E:2													

Tabla 1.11: Contactos de las letras T–Z en el criptograma.

Las letras que sólo tienen un sucesor son: B, E, M, R, S, U, W, X, y Z. Pero la B antecede a O, la letra más frecuente en el criptograma, así que la O no es buen candidato para *u*. Lo mismo ocurre con la S y con la U, que anteceden a la J; y J es también demasiado frecuente para ser *u*. También ocurre algo similar con la W, que antecede a A. Lo contrario ocurre con la X y la Z, que aparecen a la derecha de la K y la R, y estas últimas son muy poco frecuentes para ser *u*. Así que nos quedan: E, M, y R. Si observamos la tabla de frecuencias de digramas en el Apéndice A, notamos que las letras que preceden a *q* son letras muy frecuentes: la *a*, la *e*, la *s*, por ejemplo. Pero eso no ocurre con los vecinos izquierdos de la R en el texto cifrado. La E, por su parte, sólo tiene un vecino izquierdo, muy frecuente ciertamente, pero sólo uno, mientras que esperamos que la *q* tenga más variedad como antecesores inmediatos. Así que nuestro mejor candidato a *q* es la M.

Si suponemos que M = *q* entonces T = *u*, lo que suena bien porque la T tiene de vecino derecho a la O, la más frecuente, buen candidato para *e*. Además, los contactos de tanto T como O sugieren una letra con gran variedad de contactos, probables vocales. Suponemos entonces que M = *q*, T = *u*, y O = *e*.

Ahora bien, la O se junta frecuentemente con la P (antecediéndola). La P tiene muchos posibles vecinos a la derecha y pocos (comparativamente) a la izquierda. Nosotros sabemos que las vocales tienden a tener muchos vecinos de ambos lados; estos dos indicios nos hacen pensar que P corresponde a alguna consonante muy frecuente, como *n*, *r*, ó *s*. Por otra parte la J es también muy frecuente, pero sí tiene equilibrados su lado derecho e izquierdo en la tabla de digramas, así que es buen candido para vocal. La G por su parte parece también consonante, y es muy frecuente, así que tiene las mismas posibilidades que la P. Por sí misma la P es más frecuente que la G, lo cual podría sugerir que P = *s*, y G = *n*, pero el digrama OP es más frecuente que OG. En español (véase el Apéndice A) es ligéramente más frecuente el digrama *en* que *es*, así que esto sugiere, por el contrario, que P = *n*, y G = *s*. Tenemos información contradictoria y tendremos que escoger una de ellas.

Cuando tenemos una situación así, con dos o más posibles caminos, y ambos se ven más o menos igual, no queda otra que probar uno de ellos, ver que ocurre luego de dos o tres pasos, y si las cosas no lucen bien (no se han completado palabras, se generan errores ortográficos, muchos diptongos, etc.), desandar el camino hecho desde la bifurcación. Cada decisión errónea puede acarrear futuras decisiones erróneas. En general el número de caminos puede crecer exponencialmente y lo único que podemos hacer es lo que en computación se conoce como *backtrack* o retroceso mínimo. No hay más que armarse de paciencia.

Supongamos que P = *s* y que la supuesta vocal *J* es de hecho la *a*. Substituyendo todo esto en nuestro criptograma tenemos:

```
  s    a  e s ue  en u   as e
DHPLH UJNIO PFTON OGFTL SJPVO

 esan  es e que   e  en  a
LOPJG QOPIO MTOKA ONIOG DJVAI

a  s  a  en  esn  e  e    en
JDHPV JDAOG QOPGH OXKON AFOGQ

an a  ue  e  asqu eunas   a e
JGDJF TONQO FJPMT OTGJP HDJVO

  u     esa  es e nase  un aa
ZRTDA HLOPJ NOPLO GJPOE TGIJJ

   se  un         a s a  es e
LQHPO ETGIH WADDA JFPSJ BOPKO

a e
JNO
```

Otra de las letras que aparenta ser una vocal es la `A`: tiene tantos vecinos a la izquierda como a la derecha, y tiene alta frecuencia; así que puede ser i o bien o. Por otra parte el digrama `AO` aparece dos veces en el texto, así que si suponemos que `A` $= o$ tenemos `AO` $= oe$, que es un digrama poco frecuente. En cambio el digrama ie sí es frecuente, así que sería mejor suponer que `A` $= i$. Las únicas letra más o menos equilibradas en vecinos derechos e izquierdos y con muchos vecinos que quedan son la `H` y la `D`, pero como la `D` aparece frecuentemente en nuestro texto precediendo o sucediendo a la que creemos que es la a, es decir a la `J`, nos hace pensar que no es la vocal que nos falta, porque aparecería mucho el digrama `ao` y el `oa`, algo poco frecuente en español. Así que suponemos que `H` $= o$, lo cual es plausible porque el digrama `HP` aparece frecuentemente en el texto y esto se traduciría en os, también frecuente en español.

Con esto en mente la `D` debe ser entonces una consonante muy frecuente; podriamos pensar en la r. Los digramas `DA` y `DJ` aparecen mucho en el texto, y corresponderían a ri y ra; pero entonces `DO`, que sería re, no aparece en el texto, y que debería ser más frecuente que los dos digramas ya mencionados. Tampoco aparece `DG`, que debería ser con mucho el más frecuente si `D` $= r$. La letra que si antecede mucho a la `O` es la `N`, así que `N` es mejor opción para r que `D`. Respecto a la `D`, dado que se junta mucho con la `J`, que creemos es la

a, pensamos que es la *l*, dado que el digrama *la* es muy frecuente en español. Sólo tenemos un dato contradictorio: la aparición del digrama `AD` seguido de `DA`. Si suponemos que `A` $= i$ y nos fijamos en la tabla de digramas con reversos frecuentes, esto sugeriría que `D` $= c$. Si ponemos en el texto ambas opciones nos daremos cuenta de que es mejor opción `D` $= l$. Poniendo nuestras nuevas substituciones tenemos:

```
los o  ar e s uer en u   as e
DHPLH UJNIO PFTON OGFTL SJPVO

 esan  es e que i er en la i
LOPJG QOPIO MTOKA ONIOG DJVAI

alos  alien  esno e  er i en
JDHPV JDAOG QOPGH OXKON AFOGQ

anla  uer e  asqu eunas ola e
JGDJF TONQO FJPMT OTGJP HDJVO

  uli o esa res e nase  un aa
ZRTDA HLOPJ NOPLO GJPOE TGIJJ

 ose  un o  illi a s a  es e
LQHPO ETGIH WADDA JFPSJ BOPKO

are
JNO
```

La `F` debe ser una consonante frecuente y tiende a aparecer mucho antes de vocales. También le anteceden la *s*, la *i*, la *a*, *n*, y la *e*. Sería raro que `F` fuera la *c* porque *cs* no es frecuente en español. Podría ser la *d*, pero debería aparecer más el digrama *de* de lo que aparece `FO`; y debería ser poco frecuente *du* que en nuestro caso sería `FT`. Podría ser que `F` $= t$, pero no parace muy probable porque aparece mucho `FT`, que correspondería a *tu*, y éste es poco frecuente en español. Además no aparece `FH` que debería ser muy frecuente porque *to* lo es en español. Podría también ser que `F` $= m$, lo que se ve bastente bien si lo sustituimos. Hacemos eso entonces:

```
los o  ar e smuer enmu   as e
DHPLH UJNIO PFTON OGFTL SJPVO
```

```
 esan  es e que i er en la i
LOPJG QOPIO MTOKA ONIOG DJVAI

alos  alien  esno e  er imen
JDHPV JDAOG QOPGH OXKON AFOGQ

anlam uer e masqu eunas ola e
JGDJF TONQO FJPMT OTGJP HDJVO

  uli o esa res e nase  un aa
ZRTDA HLOPJ NOPLO GJPOE TGIJJ

  ose  un o  illi ams a  es e
LQHPO ETGIH WADDA JFPSJ BOPKO

are
JNO
```

En el primer renglón leemos *mueren* y luego en el cuarto *muer?e* por lo que suponemos que suponemos que `Q` $= t$. También en el cuarto renglón tenemos *mas que una sola ?e?*, así que supondremos que `V` $= v$ y `Z` $= z$. En el primer y segundo renglones tenemos *mueren mu??as ve?es antes*. El primer y tercer signo de interrogación corresponden a la misma letra, la `L`, por lo que supondremos que `L` $= c$ y que `S` $= h$. A partir de ahora todo es más sencillo. Podemos descifrar el resto, y obtenemos:

```
losco barde smuer enmuc hasve
DHPLH UJNIO PFTON OGFTL SJPVO

cesan tesde quepi erden lavid
LOPJG QOPIO MTOKA ONIOG DJVAI

alosv alien tesno exper iment
JDHPV JDAOG QOPGH OXKON AFOGQ

anlam uerte masqu eunas olave
JGDJF TONQO FJPMT OTGJP HDJVO

zjuli ocesa resce naseg undaa
ZRTDA HLOPJ NOPLO GJPOE TGIJJ
```

```
ctose gundo willi amsha kespe
LQHPO ETGIH WADDA JFPSJ BOPKO

are
JNO
```

Con separación de palabras:

> *Los cobardes mueren muchas veces antes de que pierden la vida. Los valientes no experimentan la muerte mas que una sola vez. Julio César*, escena segunda, acto segundo. William Shakespeare.

Sería bueno averiguar cómo se construyó la asociación que produjo el criptograma. Tenemos:

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

esto es fácil de completar. Se trata de un alfabeto determinado por la palabra clave `JULIOCESAR`.

1.8 Las máximas de Kerckhoffs

En 1883 fue publicada una obra que sentó las bases de la criptografía moderna. Fue escrita por un holandés que, a pesar de tener siete nombres, es conocido sólo como Auguste Kerckhoffs. La obra es de hecho un libro publicado en varias números de el *Journal des Sciences Militaires* a partir del número de enero de 1883, bajo el título *La Cryptographie Militaire* [Ker83]. En ella, Kerckhoffs establece una serie de propiedades que deben poseer los sistemas criptográficos y que han servido desde entonces como preceptos para el diseño de dichos sistemas. A lo largo del tiempo estos preceptos han sido retomados en repetidas ocasiones por diferentes personas que han hecho aportaciones importantes a la criptografía, a la teoría de la información, y a la teoría de códigos. En una interpretación reciente podemos enunciarlos así:

1. No se debe subestimar al adversario. Siempre es bueno pensar que quien desea descifrar ilegalmente nuestros mensajes secretos es bastante más inteligente que nosotros y no bastante más tonto. Lo que puede parecernos muy seguro porque *nosotros* no podríamos descifrarlo puede no ser seguro.

2. Sólo un criptoanalista puede juzgar la seguridad de un sistema criptográfico. Ya vimos que un sistema criptográfico monoalfabético no ofrece mucha seguridad, algo que no sospecharíamos si sólo pensáramos que hay 26! posibles maneras de hacerlo. Esto se debe a que el criptoanalista no tiene que probarlas todas; las técnicas criptoanalíticas nos indican cuáles son las debilidades del sistema.

3. La seguridad de un sistema criptográfico debe residir exclusivamente en la llave. Kerckhoffs dijo literalmente que el sistema criptográfico "no debe exigir el secreto [del sistema mismo.] Este puede, sin inconveniente, caer en manos del enemigo". Es decir debemos suponer en todo momento que el enemigo conoce nuestro algoritmo de cifrado[4].

4. Las complicaciones superficiales pueden ser ilusorias. Un ejemplo de esto lo vimos al comparar el sistema de alfabeto decimado con el de César. A pesar de que el cifrado de César es más sencillo, es más débil el decimado desde el punto de vista combinatórico. En general el criptoanalista no necesariamente tiene que desandar paso a paso el camino seguido para cifrar para poder descifrar el mensaje.

5. Al juzgar la seguridad de una clase de métodos hay que considerar posibles errores de transmisión. Kerckhoffs originalmente escribió "es necesario que el sistema sea fácil de usar, que no demande tensión de la mente ni el conocimiento de una larga serie de reglas a observar". Cuando las cosas son demasiado complicadas para quien las usa, tiende a evitarse hacer todo lo que se debe hacer y la flojera puede resultar cara.

La primera máxima de Kerckhoffs originalmente decía que el sistema criptográfico "debe ser materialmente, si no matemáticamente, indescifrable". La tercera decía que "la clave debe poder comunicarse y retenerse sin tener que auxiliarse de notas escritas, y debe poder intercambiarse y modificarse por acuerdo de el emisor y el receptor de los mensajes". La cuarta (que originalmente era la quinta en el documento de Kerckhoffs) pensaba en términos de un aparato o máquina criptográfica, y era: "Debe ser portátil y el mantenimiento y funcionamiento no debe exigir la participación de muchas personas". En el libro de Kerckhoffs

[4]En 1995, por ejemplo, dos estudiantes de posgrado de la Universidad de California en Berkeley, David Wagner e Ian Goldberg, hicieron noticia cuando demostraron cómo romper fácilmente el sistema criptográfico que utilizaba el navegador *Netscape* para proporcionar seguridad. Parte de la seguridad del sistema residía en que Netscape no había publicado exactamente cómo funcionaba el algoritmo; a esto se le llama *seguridad por obscuridad.* En contraste, el estándar DES ha sido desde su origen un algoritmo públicamente conocido hasta en los más mínimos detalles y aún sobrevive

se incluye una cuarta regla que dice que el sistema "debe ser aplicable a la correspondencia telegráfica". Esa regla es poco mencionada porque es, de hecho, la única que ha caducado, o mejor dicho, actualmente resulta de ámbito muy limitado.

1.9 Tipos de ataque

Suponiendo que ya tenemos un sistema criptográfico, ya que podemos suponer que este sistema será atacado por el enemigo (un criptoanalista), es importante tener una idea sobre cuáles son los posibles ataques de los que dispone el criptoanalista.

Los tipos de ataque se suelen clasificar en cinco categorías dependiendo de lo que el criptoanalista posee para hacer su trabajo; cuanto más abajo de esta lista nos encontremos, más elementos tiene:

1. **Ataque de criptotexto conocido**. El criptoanalista posee un texto cifrado, su objetivo es descifrar el texto. Este es el ataque clásico, y es el ataque que utilizamos en éste capítulo.

2. **Ataque de texto conocido**. El criptoanalista posee un texto cifrado y sabe que cierto texto claro corresponde a un fragmento del criptotexto. El objetivo en este caso es obtener el descifrado completo del texto. Si se conoce el texto completo, el objetivo es encontrar la llave utilizada para cifrar.

3. **Ataque de texto elegido**. El criptoanalista puede escoger un texto claro para cifrarlo y así tener la pareja de texto claro y texto cifrado correspondiente. El objetivo es recuperar la clave.

4. **Ataque de texto elegido adaptable**. El criptoanalista escoge el texto a cifrar y, con base en la salida producida, puede elegir más texto a cifrar, y continúa así hasta recuperar la llave. El objetivo es recuperar la llave.

5. **Ataque de criptotexto elegido**. Este ataque surgió en el contexto específico de los sistemas de llave pública que veremos más adelante. El criptoanalista elige el texto a descifrar. El objetivo es recuperar la llave.

6. **Ataque de llave elegida**. El criptoanalista descubre relaciones entre llaves distintas, es decir, sabe que efectos producen en el texto cifrado ciertas características de las claves. El objetivo es recuperar el texto y la llave usada para ese texto. Este ataque es algo esotérico y se utiliza contra sistemas que tienen un manejo de llaves como DES.

7. **Ataque de garrote**. Es el menos elegante, pero ha probado ser el más efectivo; consiste en golpear, sobornar, chantajear, amenazar, engañar, o seducir a alguien para obtener la clave.

Nuestro criptoanálisis de las substituciones monoalfabéticas, aún en su forma más general, nos demuestran que éstas sólo ofrecen poca seguridad para la transmisión. Un mensaje relativamente largo puede ser descifrado aún con un ataque de criptotexto conocido (un criptoanalista experimentado puede descrifrar mensajes de treinta letras, cifrados mediante substituciones monoalfabéticas, en un par de horas).

Esto nos lleva necesariamente a buscar otros criptosistemas que proporcionen mayor seguridad para nuestras transmisiones.

2.1 Substitución con homófonos

La razón principal por la cual las substituciones monoalfabéticas son tan vulnerables al criptoanálisis es que preservan las frecuencias características del idioma. El conteo de

frecuencias y de contactos proporciona demasiada información al criptoanalista.

Con nuestra experiencia criptoanalítica, vemos que es necesario disfrazar las frecuencias si vamos a evitar la lectura no autorizada de nuestros mensajes.

Históricamente, la primer manera en que esto se intentó fue mediante el uso de *homófonos*. En vez de que cada letra del alfabeto original corresponda a una y sólo una letra del alfabeto de cifrado, tomamos una función con muchos valores, de manera que varios símbolos del texto cifrado puedan corresponder al mismo texto en el mensaje original. Es decir, en vez de una función del alfabeto original al alfabeto de cifrado, tenemos una relación en la cúal cada letra del alfabeto de cifrado está asociada a lo más a una letra del alfabeto original, pero cada letra del alfabeto original puede estar asociada a más de una letra del alfabeto de cifrado.

Varias letras que representan la misma letra del alfabeto original son llamadas *homófonos* de esa letra.

En otros casos, algunas letras del alfabeto de cifrado no corresponden a ninguna letra del alfabeto original; son agregadas aleatoriamente para producir confusión y disfrazar la información de frecuencias y contactos.

El primer sistema con homófonos se debe a Shihab al Qalqashandi, en el siglo XIV, y fue adoptado rápidamente por las cortes europeas. La idea es que las letras más frecuentes son representadas por varios símbolos, que son usados en proporciones iguales, de manera que las frecuencias queden completamente disfrazadas.

Vamos a ver un caso extremo que busca disfrazar las frecuencias completamente. Vamos a asignarle a cada letra del alfabeto un subconjunto de $S = \{00, 01, 02, \ldots, 99\}$, de tal manera que:

1. La colección de estos subconjuntos sea una partición de S; y
2. El número de elementos en el conjunto asociado a cada letra sea el entero más cercano a cien veces la frecuencia relativa de la letra.

Por ejemplo: para cifrar, reemplazamos cada letra por algún número, elegido aleatoriamente de el subconjunto de S asociado a la letra. Una posible asociación se muestra en la Tabla 2.1. Para descifrar, buscamos el número en la tabla y lo reemplazamos por la letra correspondiente.

```
 n  o  s     v  e  m  o  s     e  n    e  l
31 02 89    44 08 87 78 28    59 49   81 26
```

a	15, 16, 30, 33, 37, 38,53, 55, 57, 72, 91, 96
b	24
c	03, 39, 42, 67
d	04, 43, 61, 69, 88
e	08, 12, 20, 46, 47, 59, 64, 79, 81, 85, 90, 94, 97
f	60
g	29
h	05
i	14, 45, 50, 60, 73, 82
j	11
k	77
l	01, 26, 35, 71, 93, 98, 99
m	34, 87, 38
n	06, 17, 22, 31, 49, 58
o	02, 10, 41, 51, 66, 70, 75, 83
p	13,18
q	36
r	21, 25, 65, 68, 92, 95
s	00, 28, 52, 63, 74, 78, 89
t	07, 19, 23, 54, 84
u	09, 32, 62, 80
v	44
w	56
x	86
y	76
z	27

Tabla 2.1: Asociación de letras y números. Cada letra tiene una cantidad de códigos asociada correspondiente a su frecuencia relativa en español.

```
 c  a  f  e   a   l  a  s  t  r  e  s
03 37 40 81  57  71 72 52 54 25 85 63
```

Si elegimos los homófonos con cuidado, la distribución que se obtiene es completamente plana, de manera que el criptoanalista no tiene mucho por donde empezar (sobre todo si quitamos la separación de palabras).

Pero este sistema tiene dos problemas prácticos muy importantes: uno de seguridad, y uno de uso.

Primero, a menos que el remitente y destinatario tengan capacidades mentales extraordinarias y se puedan aprender la tabla, van a requerir de una copia física de la misma. Y cuando tenemos una copia escrita de la llave, se arriesga uno a que sea comprometida.

El otro problema es de uso práctico. Cuando se han utilizado estos métodos, la gente no es buena distribuyendo el uso de homófonos. En vez de usar todos los valores para, por ejemplo, la `e`, la gente tiende a aprenderse dos o tres de los homófonos, y usar esos siempre alternandolos de manera regular. Lo cual por supuesto derrota el propósito mismo del sistema.

Por lo anterior este sistema en poco práctico, a pesar de su fuerza teórica.

2.2 Cifrado de Vigenère

Los sistemas que vimos en el capítulo anterior son llamados *monoalfabéticos* pues utilizan un único alfabeto de cifrado. Para disfrazar las frecuencias, una idea es utilizar varios alfabetos de cifrado distintos, alternando entre ellos mediante algún esquema fijo.

El primer método polialfabético práctico fue diseñado por Blaise de Vigenère en 1586. El apellido simplemente indicaba que Blaise venía de un pueblo llamado Vigenère. Vigenère entró como secretario al servicio exterior francés, y tras algunos años de experiencia con cifrado y criptoanálisis, propuso un sistema relativamente avanzado, mediante una exposición muy clara.

Pero desgraciadamente, en esa época la moda criptográfica eran los nomenclatores, que se trataban de códigos donde palabras y frases completas eran reemplazadas por números[1]. Debido a ello, su sistema fue ignorado hasta finales del siglo XIX. Los criptólogos de ese

[1]La diferencia principal entre los códigos y los criptogramas es que en un código las unidades lingüísticas se respetan, mientras que en un criptograma son las unidades alfabéticas las que se respetan.

entonces añadieron insulto a la injuria, y degradaron el sistema de Vigenère a uno con mucha menor seguridad antes de asociarle su nombre.

El método que ahora se conoce como Vigenère utiliza la llamada *tabla de Vigenère* que aparece en la Tabla 2.2, y una llave, que es una palabra o frase clave. La idea es utilizar un monoalfabeto distinto para cada caracter del mensaje, en el orden especificado por la llave.

El primer renglón de la tabla tiene el alfabeto original; la primer columna consiste de indicadores como se verá mas abajo.

Los distintos renglones de la tabla representan los distintos alfabetos de substitución monoalfabética; cada uno es simplemente un cifrado de César.

Tanto el remitente como el destinatario conocen la palabra clave; digamos, por ejemplo, que se trata de la palabra `CLAVES`. Para cifrar un mensaje, escribimos la palabra varias veces, y escribimos el mensaje abajo:

C L A V E	S C L A V	E S C L A	V E S C L	A V E S C
e n u n l	*u g a r d*	*e l a m a*	*n c h a d*	*e c u y o*

Para cifrar, buscamos la letra de la palabra clave en la primer columna de la tabla, indicándonos que renglón usar. Luego buscamos la letra que queremos cifrar en el primer renglón, lo cual nos indica que columna de la tabla utilizar. El criptotexto correspondiente es entonces la entrada que está en esa columna y ese renglón.

Por ejemplo, la primer letra del mensaje es una *e*, indicando que debemos usar la columna que tiene una *e* hasta arriba; y la primer palabra de la clave es la C, indicando que usemos el renglón que empieza con C. La entrada en ese renglón y esa columna es una `g`, de manera que ese es nuestro criptotexto. Continuando de esa manera, tenemos:

C L A V E	S C L A V	E S C L A	V E S C L	A V E S C
e n u n l	*u g a r d*	*e l a m a*	*n c h a d*	*e c u y o*
`G Y U I P`	`M I L R Y`	`I D C X A`	`I G Z C O`	`E X Y Q Q`

Si pensamos en cada letra como un número $0 \leq x \leq 25$, módulo 26, entonces el proceso criptográfico es equivalente a sumar la letra de la clave a la letra del mensaje módulo 26 para obtener el valor de la letra del criptotexto.

El cifrado de Vigenère también es llamado “doble substitución”, y era considerado irrompible en el siglo XIX (excepto quizás si tenía uno la buena suerte de adivinar la llave).

	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	J	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	M	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 2.2: La tabla de Vigenère

Notemos que una misma letra del criptotexto puede representar a varias letras distintas del mensaje original: la segunda, décima, y vigésimo tercera letra del criptotexto son todas `Y`; sin embargo, la primer instancia representa a n, la segunda a d, y la tercera a u; simétricamente, aunque la segunda y cuarta letras del mensaje son n, están representadas en el criptotexto por letras distintas: `Y` e `I`, respectivamente. Por otro lado, la n en la posición décimo sexta vuelve a ser representada por `I`.

Si escogemos la llave razonablemente bien, las frecuencias de cada caracter en el criptotexto tienden a aplanarse, dando una distribución bastante uniforme.

2.3 El método original de Vigenère

Mencionamos en la sección anterior que el método que ahora se conoce por el nombre de Vigenère es una simplificación del método original.

El método original de Vigenère consistía en poner un alfabeto mezclado en el primer renglón, un alfabeto mezclado en la primer columna, y un alfabeto arbitrario en la tabla; dicho alfabeto era rotado (compuesto con cifrado de César) en cada renglón sucesivo. Tenemos un ejemplo en la Tabla 2.3.

Dicho cifrado es a veces llamada "cifrado de Alberti"; Alberti fue un criptoanalista y criptógrafo italiano, que sugirió un modelo de substitución polialfabética en el cual cada cierto número de letras (o cada palabra) se cambiaba el alfabeto de cifrado, utilizando algún esquema en el cual habían quedado de acuerdo el remitente y el destinatario.

Nostros vamos a usar el término *"cifrado de Alberti"* para referirnos a él. Una tabla con un alfabeto mezclado rotado va a ser llamada una *"tabla de Alberti"*, para diferenciar de la tabla de Vigenère usual. También le llamaremos así aunque sólo modifiquemos la primer columna o el primer renglón de la tabla de Vigenère (esto es equivalente a hacer una substitución monoalfabética antes, o después, de cifrar).

El método se puede hacer aún más complicado si en vez de utilizar un mismo alfabeto trasladado para los distintos renglones de la tabla de Vigènere (o de Alberti), usamos alfabetos arbitrarios no relacionados. Esto hace que la "llave" consista de los n alfabetos que usemos, más la palabra clave. Este fue el sistema utilizado por el ejercito americano durante la Segunda Guerra Mundial, diseñado por William Friedman, el padre del criptoanálisis moderno.

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

Tabla 2.3: Una tabla para el método original de Vigenère.

2.4 Criptoanálisis del sistema de Vigenère

Sabemos ya que en cada idioma las letras del alfabeto poseen una distribución de frecuencias característica; en el Capítulo 1 nuestro criptoanálisis de sistemas monoalfabéticos estaba fundamentado en este hecho. En las substituciones monoalfabéticas cada letra cambia su identidad, pero preserva su frecuencia. El panorama para hacer criptoanálisis de sistemas polialfabéticos, como los que hemos mencionado en este capítulo, luce escabroso, o mejor dicho, excesivamente plano. Los métodos de cifrado polialfabéticos tratan justamente de destruir la información de las frecuencias, pretenden "aplanar" la distribución de frecuencias de las letras y hacerlas indistinguibles (al menos desde el punto de vista del análisis de frecuencias). Así que ya no podemos basar el criptoanálisis de sistemas polialfabéticos en las frecuencias de las letras del criptograma, al menos no de inmediato. ¿Será posible recuperar la información de las frecuencias de alguna manera?

Hagamos una revisión de los elementos involucrados en el cifrado: se comienza con un texto claro y una palabra clave; la palabra clave se hace corresponder letra a letra con el texto claro. Como en general la palabra clave es bastante más corta que el texto a cifrar, la palabra clave se repite tantas veces como sea necesario para cubrir el texto completo. Cada pareja de letras (una letra de la palabra clave y la otra del texto original) se reemplaza por una letra en el texto cifrado; en general, en cada pareja de letras, la de la clave se usa para determinar uno de un conjunto de posibles alfabetos, que será usado para reemplazar la otra letra de la pareja, la del texto claro.

Denotemos con ℓ la longitud de la palabra clave (número de letras en la palabra clave) y con N la longitud del texto claro (número de letras sin considerar espacios ni puntuación). Como en general $\ell < N$, esto obliga a que la palabra clave sea repetida varias veces para cubrir todo el texto claro. Si denotamos por p_j la j-ésima letra del texto claro, donde $j \in \{0, \ldots, N-1\}$, y denotamos con c_i, $i \in \{0, \ldots, \ell - 1\}$ la i-ésima letra de la palabra clave, entonces la c_i corresponde a la p_j si y sólo si $i \equiv j \pmod{\ell}$.

La letra de la palabra clave c_i determina el alfabeto usado para cifrar la letra del texto claro p_j. Así que las letras del texto claro $p_0, p_\ell, p_{2\ell}, \ldots, p_{k\ell}$ son cifradas con el mismo alfabeto, a saber aquel determinado por c_0; $p_1, p_{\ell+1}, p_{2\ell+1}, \ldots, p_{k\ell+1}$ son cifradas con otro alfabeto, pero todas ellas con el determinado por c_1; $p_2, p_{\ell+2}, p_{2\ell+2}, \ldots, p_{k\ell+2}$ son cifradas con el alfabeto determinado por c_2; y así podemos continuar hasta llegar al conjunto de las letras $p_{\ell-1}, p_{2\ell-1}, p_{3\ell-1}, \ldots, p_{(k+1)\ell-1}$, las que son cifradas con el alfabeto determinado por $c_{\ell-1}$. Hemos dividido entonces todo el texto claro en ℓ subconjuntos distintos, cada subconjunto asociado a una letra diferente de la palabra clave y por tanto, cifrado usando un único alfabeto determinado por esa letra. Es decir, cada subconjunto es cifrado monoalfabéticamente. ¡Qué bien! Si logramos determinar el tamaño de la palabra clave entonces

podemos reducir nuestro criptoanálisis de un criptograma polialfabético a varios monoalfabéticos. Por cierto que a cada uno de estos monoalfabéticos no le podemos echar encima todos los recursos usados en el Capítulo 1, porque cada letra del criptograma monoalfabético está realmente separada varias posiciones de su sucesora en el texto claro. Es decir, aunque vamos a tener información de frecuencias, no vamos a tener información sobre contactos. Pero si el criptograma es suficientemente largo entonces esperamos que la información de las frecuencias baste.

2.4.1 La prueba de Kasiski

Ahora tenemos que encontrar una manera de estimar el tamaño de la palabra clave. Para evidenciar como podemos hacer esto consideremos un ejemplo. Supongamos que tenemos el siguiente texto claro[2]:

> *[...] la palabra es signo de la mente y la racionalidad humana, y con la palabra puede insultarse a Dios. No todo lo que es propio del hombre es necesariamente bueno. La risa es signo de estulticia. El que ríe no cree en aquello de lo que ríe, pero tampoco lo odia. Por tanto, reírse del mal significa no estar dispuesto a combatirlo, y en contraste, reírse indolentemente del bien significa desconocer la fuerza del bien, que se difunde por sí solo.*

El texto versa sobre la risa, por lo que son frecuentes palabras como *ríe* y *reírse*. Si no sólo consideramos palabras, sino secuencias de letras que se repiten, tenemos cosas como *la palabra*, *es signo de*, *significa*. Al cifrar se pone en correspondencia el texto claro con la palabra clave repetida. Si algunas de estas repeticiones empatan de la misma manera con la clave entonces son cifradas de la misma manera y esto resultará en repeticiones en el texto cifrado. Es decir, repeticiones de empate entre secuencias de letras del texto claro y la clave generan repeticiones en el criptograma; hay que enfatizar que no basta que una secuencia de letras se repita en el texto claro, es menester que estas repeticiones sean apareadas de la misma manera con la palabra clave. También es conveniente señalar que no siempre las repeticiones en el criptograma provienen de repeticiones de apareo entre el texto claro y la clave, algunas de las repeticiones en el criptograma son accidentales. Sin embargo, estas son raras por lo que tienden a ser cortas.

Para que ocurran las repeticiones de empate, es necesario y suficiente que la distancia entre algunas de las repeticiones de secuencia en el texto claro sea múltiplo de la longitud

[2]Tomado de: *El Nombre de la Rosa* de Humberto Eco; hora *Tercia* del *Segundo Día*, discusión acerca de la risa que sostienen fray Guillermo de Baskerville y fray Jorge de Burgos.

de la palabra clave. Por ejemplo, si ciframos nuestro texto usando el método de Vigenère con la imaginativa palabra clave: CLAVE, de longitud 5, obtenemos lo que se muestra en la Tabla 2.4.

En la tabla podemos observar que hay varias coincidencias de empate entre el texto claro y la clave, lo que ocasiona repeticiones de secuencia en el criptograma. Por ejemplo, la secuencia `NLPVPCMRV` se repite dos veces en el criptograma y proviene de dos empates idénticos de la secuencia *la palabra* con la palabra clave. De la misma forma podemos identificar otras repeticiones. Algunas de ellas se muestran en la Tabla 2.5 junto con la posición en la que comienza cada repetición (número total de letras que anteceden a la primera de la secuencia en cuestión) y la distancia entre apariciones (diferencia de posiciones).

En la Tabla 2.5 podemos ver que las distancias entre repeticiones de secuencias en el criptograma son divisibles entre 5, la longitud de la palabra clave, siempre que provienen de repeticiones de empate. En cambio las dos ultimas secuencias de la tabla se repiten a intervalos no divisibles por 5 y no corresponden a repeticiones de empate entre el texto claro y la clave. Éstas son las repeticiones accidentales que ya mencionamos. Hay que hacer notar que las repeticiones que nos interesan, las que provienen de empates iguales, son más largas que las accidentales. Esto por supuesto no siempre ocurre. En general es poco probable encontrar repeticiones accidentales largas pero puede haber muchas no accidentales cortas.

¿Qué hacemos si nos topamos con un criptograma que presumiblemente ha sido obtenido con un cifrado polialfabético?

1. De acuerdo a nuestro análisis procedemos a encontrar repeticiones de secuencias de letras en él, con la esperanza de que esas repeticiones no sea producto del azar, sino de repeticiones de empate entre la palabra clave y el texto claro original, mientras más largas sean las secuencias mayor certidumbre tenemos de que no son accidentales.

2. Registramos las posiciones en las que ocurren las repeticiones y obtenemos las distancias entre ellas.

3. De entre los divisores de las distancias, los más frecuentes son buenos candidatos para la longitud de la palabra clave.

4. Se agrupa cada letra del criptograma con todas aquellas que disten de ella en un múltiplo de la hipotética longitud de clave y se procede a hacer análisis de frecuencias sobre cada uno de los conjuntos obtenidos de esta manera.

Este método de criptoanálisis fue descubierto independientemente por dos personas a mediados del siglo XIX [Kah99, Sin99]: Charles Babbage, conocido por su diseño de dos máquinas computadoras mecánicas que nunca vieron la luz, y Friedrich Wilhelm Kasiski, un

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

Tabla 2.4: La clave con el texto claro y el criptotexto resultante.

Secuencia	Posiciones	Distancias	Factores
NLPVPCMRV	0, 50	50	2, 5, 5
IUDIBRQOE	9, 134	125	5, 5, 5
SFEMMG	155, 170	15	3, 5
CEDVUP	211, 271	60	2, 2, 3, 5
NMIYIAMEL	223, 298	75	3, 5, 5
GYO	125, 160	35	5, 7
GY	125,160,260,330	35, 100, 70	5, 7 / 2, 2, 5, 5 / 2, 5, 7
XG	264, 269, 289	5, 20	5 / 2, 2, 5
GP	165, 173	8	2, 2, 2
WY	123, 340	217	7, 31

Tabla 2.5: Algunas secuencias de letras que se repiten en el criptograma, sus posiciones, distancias y divisores primos de las mismas.

oficial del Trigésimo Tercer Regimiento de Infantería del Ejercito Prusiano. Posiblemente Babbage hizo su descubrimiento alrededor de 1854 [Sin99], pero nunca lo publicó. No se supo de este descubrimiento hasta que en el siglo XX, cuando algunos estudiosos de la obra de Babbage revisaron sus notas personales. Kasiski describió el método en su libro de 1863 *La Escritura Secreta y el Arte de Descifrar*[3]. Al parecer la obra de Kasiski no recibió mucha atención en su momento. El interés de Kasiski en la criptología decayó y prefirió dedicarse a la antropología, campo en el que fue más reconocido en su momento [Kah99]. El método resulta ser el fundamento en el que se basó el criptoanálisis moderno a principios del siglo XX. Hoy en día lo conocemos como *la prueba de Kasiski* [Lew00, Bau00].

2.4.2 Ejemplo de criptoanálisis usando la prueba de Kasiski

Ilustraremos el uso de la prueba de Kasiski con un ejemplo completo. Supongamos que hemos interceptado el siguiente mensaje cifrado con el método de Vigenère:

```
WWJEE SWPOI EMWVO NQOID YIATW AYSNS WWKUR TWXZE ADGLO AADGB
ILBIF CYNGO HXHEE BWUFE MAKNH MVEFC YNQIE RYNGO QWIMV EFCYN
QIERY NGOWW FAFBW UFEMA KHINB EKCCM BTWBI RBSFR LIDUW IUSLA
KRMIR MHAYM RCSDM AZAKL INGEF CIPBN WALID UWIUS RNERJ EASSV
CEATG ZOEAO ERJEA SSVCE ATGNH LNSJR KURZS BSNBE KCCMB HWAGO
FUJJK URVWW WIQAW BXEFP GSICV VAUXE YAKNX AQEKW CRVQM NTAZE
```

[3] *Die Geheimschriften und Dechiffrir-kunst.*

Repeticiones, distancias y divisores de las distancias			
Secuencia	**Frec.**	**Distancias**	**Factores primos**
WW	4	30, 78, 148	2, 3, 5 / 2, 3, 13 / 2, 2, 37
BWUFEMAK	2	48	2, 2, 2, 2, 3
NH	3	148	2, 2, 37
LIDUWIUS	2	42	2, 3, 7
IAFRXAQEK	2	36	2, 2, 3, 3
YNGO	3	30	2, 3, 5
JEASSVCEATG	2	18	2, 3, 3
NBEKCCM	2	114	2, 3, 19
AMU	2	54	2, 3, 3, 3
MVEFCYNQIERYNGO	2	18	2, 3, 3
CR	2	60	2, 2, 3, 5

Tabla 2.6: Resultados para el criptograma de la Sección 2.4.2.

```
SPLAQ AXNGE ATAMU TRNAN HDBPG AGEWO JNHMV SNNLD NDWBW OASMV
CRIAF RXAQE KMYLN VAMUQ HEUXH SHMAA FAIIV JYNIA FRXAQ EK
```

Supongamos también que el criptograma fue obtenido a partir de un texto claro en idioma español.

Procedemos a buscar secuencias de letras repetidas (ignorando, como hemos hecho hasta ahora, los espacios insertados artificialmente). Los resultados se encuentran en la Tabla 2.6.

Los divisores más frecuentes son 2, 3, y por supuesto 6, el producto de ambos. De éstos, el 2 y el 3 son demasiado pequeños como para ser la longitud de la palabra clave: usar claves tan cortas no es bueno, y por ende tampoco es frecuente. Así que lo más probable es que sea 6 el número que buscamos. Así que será entonces 6 nuestra estimación de la longitud de la palabra clave.

Si ahora dividimos el criptograma en seis subconjuntos y acomodamos cada uno de estos en una columna, obtenemos lo que se muestra a continuación.

El criptograma en bloques de seis

```
WWJEES WPOIEM WVONQO IDYIAT WAYSNS WWKURT WXZEAD GLOAAD GBILBI FCYNGO
HXHEEB WUFEMA KNHMVE FCYNQI ERYNGO QWIMVE FCYNQI ERYNGO WWFAFB WUFEMA
KHINBE KCCMBT WBIRBS FRLIDU WIUSLA KRMIRM HAYMRC SDMAZA KLINGE FCIPBN
```

```
WALIDU WIUSRN ERJEAS SVCEAT GZOEAO ERJEAS SVCEAT GNHLNS JRKURZ SBSNBE
KCCMBH WAGOFU JJKURV WWWIQA WBXEFP GSICVV AUXEYA KNXAQE KWCRVQ MNTAZE
SPLAQA XNGEAT AMUTRN ANHDBP GAGEWO JNHMVS NNLDND WBWOAS MVCRIA FRXAQE
KMYLNV AMUQHE UXHSHM AAFAII VJYNIA FRXAQE K
```

Hemos escrito nuestro criptograma en bloques de longitud seis. Estamos suponiendo que todas las letras que aparecen al principio de un bloque corresponden a la misma letra de la llave; todas las que aparecen como segunda letra de un bloque son de la misma llave; etc. Si tuvieramos espacios, hubiesemos escrito todo el criptograma con un solo bloque por renglón, para que cada columna correspondiera a la misma letra de la llave. Imaginemos entonces que los bloques se encuentran uno encima del otro y que tenemos una columna que está formada por las primeras letras de cada bloque, una por las segundas, etc.

Nuestra hipótesis es que cada columna está cifrada monoalfabéticamente. De hecho, dado que se trata de un criptograma de Vigenère, estamos suponiendo que cada columna es obtenida mediante un cifrado de César. Descifrar cada columna consiste entonces en encontrar el desplazamiento usado en el cifrado de César. Es decir, determinar una correspondencia texto-criptotexto. Para determinar ésto nos basaremos nuevamente en un análisis de frecuencias de las letras de cada columna. Lo más probable es que, dado que cada columna es un cifrado de César, alguna de las letras de mayor frecuencia de la columna sea aquella que substituye a la e, la letra más frecuente en español. Si logramos determinar esta letra, entonces habremos determinado el tamaño del desplazamiento usado para cifrar toda la columna. Esto nos dice algo más; si determinamos una pareja texto-criptotexto, también sabremos cuál renglón de la matriz de Vigenère se utilizó y por tanto cuál es la letra de la palabra clave que determina el cifrado de toda la columna.

Columna 1

```
WWHQKKWEKGSJKFWWWFKHWSWAXNAWGKEWSEGJKAWUIGFWFKSJWKAMAWFEWWFGSWMGFVK
```

La distribución de frecuencias se muestra en la Tabla 2.7.

La diferencia entre la letra más frecuente y la que le sigue es notable (poco más de 10 puntos porcentuales). Así que no es descabellado suponer que la letra `W` en esta columna del criptograma cifra a la e. Esto significa que el desplazamiento usado para el cifrado de César de esta columna es 18 (abusando de la notación $e{+}18 = \texttt{W}$). Si ésto es cierto entonces la letra a del texto original está siendo representada en esta columna del criptograma por la $a{+}18 = \texttt{S}$, por lo que la primera letra de la palabra clave sería `S`.

Letra	Frec.	%	Letra	Frec.	%
W	17	25.37	U	1	1.49
K	10	14.93	V	1	1.49
F	7	10.45	X	1	1.49
G	6	8.96	B	0	0.00
A	5	7.46	C	0	0.00
S	5	7.46	D	0	0.00
E	4	5.97	L	0	0.00
J	3	4.48	O	0	0.00
H	2	2.99	P	0	0.00
M	2	2.99	R	0	0.00
I	1	1.49	T	0	0.00
N	1	1.49	Y	0	0.00
Q	1	1.49	Z	0	0.00

Tabla 2.7: Distribución de frecuencias de la columna 1 del criptograma.

Columna 2

```
WWXWHRARCSPNMRPXUCCAIVAUNNMVLNRBDRNJNMBXDBCWRLVRWWNVAACRUICZBBNARJ
```

La distribución de frecuencias se muestra en la Tabla 2.8.

Aquí la elección no es evidente: las dos letras más frecuentes en la columan tienen frecuencias muy cercanas entre sí. No queda otra que probar ambas opciones y ver cuál de las dos se ajusta más a la distribución del español.

Supongamos que la letra e en texto claro está siendo cifrada en esta columna por la letra:

`R`. Esto significa un desplazamiento de 13 lugares, con lo que la a corresponde a la `N`. Con esta suposición la w corresponde a la `J` y ésta tiene una frecuencia del 3.03%, es decir dos ordenes de magnitud superior que la w. La j y la p corresponden a la `W` y a la `C`, respectivamente, cuyas frecuencias (9.09 en ambos casos) es muy superior a 0.54 y 2.32, las frecuencias relativas de la j y la p, respectivamente, en español.

`N`. Esto significa un desplazamiento de 9 lugares, con lo que la a corresponde a la `J`. Aparecen muy altas las frecuencias de quienes representan a la i, la t, y la m (la `Z`, la `K`,

Letra	Frec.	%	Letra	Frec.	%
R	9	13.64	L	2	3.03
N	8	12.12	P	2	3.03
A	6	9.09	H	1	1.52
C	6	9.09	S	1	1.52
W	6	9.09	Z	1	1.52
B	5	7.58	E	0	0.00
V	4	6.06	F	0	0.00
M	3	4.55	G	0	0.00
U	3	4.55	K	0	0.00
X	3	4.55	O	0	0.00
D	2	3.03	Q	0	0.00
I	2	3.03	T	0	0.00
J	2	3.03	Y	0	0.00

Tabla 2.8: Distribución de frecuencias de la columna dos del criptograma.

y la D respectivamente), pero en general esta suposición se ve bastante mejor que la anterior. Supondremos entonces que la segunda letra de la clave es J.

En la figura 2.1 aparece la gráfica comparativa de las frecuencias relativas de las letras en español y de las de la columna 2, desplazando el alfabeto 9 y 13 lugares el alfabeto. Con una gráfica como ésta podemos decidir qué desplazamiento hace corresponder mejor las distribuciones.

Columna 3

JKHIIMLJCILHYXOZFYCYUCGXGLUOOHYIMJHKXUWHYIYFLICKWCHCFYYYFUIOSXTGXY

La distribución de frecuencias se muestra en la Tabla 2.9

Otra vez tenemos una situación en la que el caso más probable se distingue mucho de los demás. La elección natural es suponer que la *e* está siendo cifrada por la Y. Esto significa que el desplazamiento de la columna es 20 y que la *a* está siendo cifrada entonces por la U. Así que suponemos que la tercera letra de la clave es U.

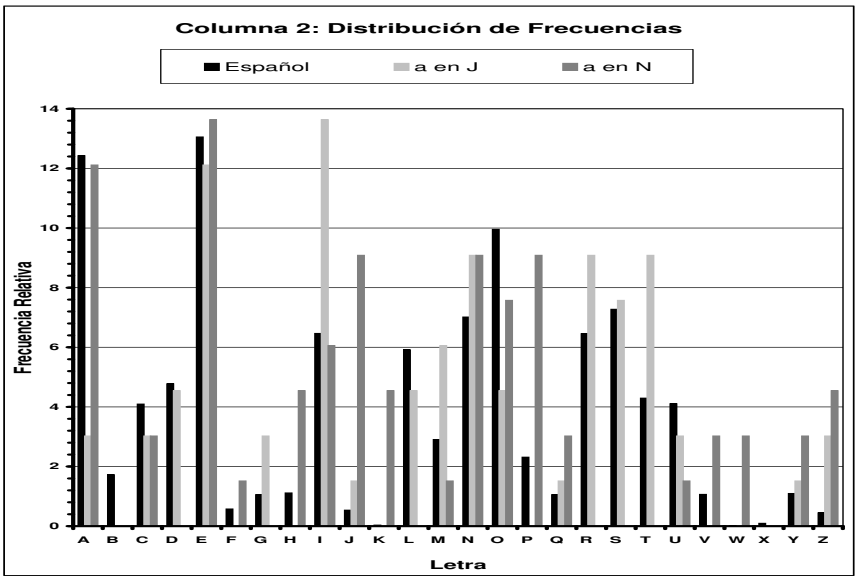

Figura 2.1: Gráfica de distribución de frecuencias para la columna 2.

Letra	Frec.	%	Letra	Frec.	%
Y	10	15.15	W	2	3.03
I	7	10.61	S	1	1.52
C	6	9.09	T	1	1.52
H	6	9.09	Z	1	1.52
X	5	7.58	A	0	0.00
F	4	6.06	B	0	0.00
L	4	6.06	D	0	0.00
O	4	6.06	E	0	0.00
U	4	6.06	N	0	0.00
G	3	4.55	P	0	0.00
J	3	4.55	Q	0	0.00
K	3	4.55	R	0	0.00
M	2	3.03	V	0	0.00

Tabla 2.9: Distribución de frecuencias de la tercer columna del criptograma.

Letra	Frec.	%	Letra	Frec.	%
E	14	21.21	Q	1	1.52
N	10	15.15	T	1	1.52
A	9	13.64	B	0	0.00
I	6	9.09	F	0	0.00
M	6	9.09	G	0	0.00
S	4	6.06	H	0	0.00
L	3	4.55	J	0	0.00
R	3	4.55	K	0	0.00
U	3	4.55	V	0	0.00
D	2	3.03	W	0	0.00
O	2	3.03	X	0	0.00
C	1	1.52	Y	0	0.00
P	1	1.52	Z	0	0.00

Tabla 2.10: Distribución de frecuencias de la cuarta columna del criptograma.

Columna 4

EUEMNIIEMCAMLAIEENMMSEOEEDQNAMNRAELUATOSILNAINEUIRDRASNNESPENEAEAN

La distribución de frecuencias se encuentra en la Tabla 2.10.

La elección clara es suponer que la *e* corresponde a la E, lo que significa un desplazamiento cero, es decir la *a* corresponde a la A. La cuarta letra de la clave sería entonces A.

Columna 5

EREVBRDABVQVNQEAMQBRRAFYANHQAVGBZANRQRAHABQFDGARQVBIINGGMLBABFZWQI

La distribución de frecuencias de la quinta columna se encuentra en la Tabla 2.11.

Igual que en el caso de la columna 2, aquí hay varias posibilidades a analizar. Tenemos que considerar cada una y comparar la distribución de frecuencias con la del español. Podemos suponer que la letra *e* del texto claro se ha desplazado hasta ocupar el lugar de la:

Letra	Frec.	%	Letra	Frec.	%
A	10	15.15	Z	2	3.03
B	8	12.12	L	1	1.52
Q	8	12.12	W	1	1.52
R	7	10.61	Y	1	1.52
V	5	7.58	C	0	0.00
G	4	6.06	J	0	0.00
N	4	6.06	K	0	0.00
E	3	4.55	O	0	0.00
F	3	4.55	P	0	0.00
I	3	4.55	S	0	0.00
D	2	3.03	T	0	0.00
H	2	3.03	U	0	0.00
M	2	3.03	X	0	0.00

Tabla 2.11: Distribución de frecuencias de la quinta columna del criptograma.

A. Esto significaría un desplazamiento de 22 lugares, con lo que la *a* es cifrada por la W. Esto significa que la *f*, la *u*, la *v*, y la *z* corresponden respectivamente en la B, la Q, la R, y la V. Todas ellas tienen frecuencias excesivamente altas respecto a las de las letras que representarían.

B. Esto significaría un desplazamiento de 23 lugares, con lo que la *a* correspondería a la X. La situación no mejora mucho respecto al caso anterior: ahora las letras Q, R, y V representan a la *t*, la *q*, y la *y* respectivamente, y siguen teniendo frecuencias demasiado altas respecto a lo que indica la distribución del español (véase la figura 2.2 donde se ilustra este caso y el anterior).

Q. Esto significaría un desplazamiento de 12 lugares, con lo que la *a* quedaría cifrada por la M. En este caso las letras M, D, y E representarían a la *a*, la *r*, y la *s* respectivamente, y tendrían una frecuencia demasiado baja para ello. Por otra parte las letras N, R, V, B, A, e I representarían a la *b*, la *f*, la *j*, la *p*, la *o*, y la *w* respectivamente, y tendrían entonces una frecuencia demasiado alta (véase la figura 2.3).

R. Esto significaría un desplazamiento de 13 lugares, con lo que la *a* queda representada por la N. Esto implicaría que la *a*, representada por la N, tiene una frecuencia muy baja respecto a la que debería tener; y en menor medida lo mismo ocurre para la *r* y la *s*, representadas por la E y la F respectivamente. Por otra parte la *n*, representada por la A tiene una frecuencia muy alta y en menor medida lo mismo ocurre para la *z*,

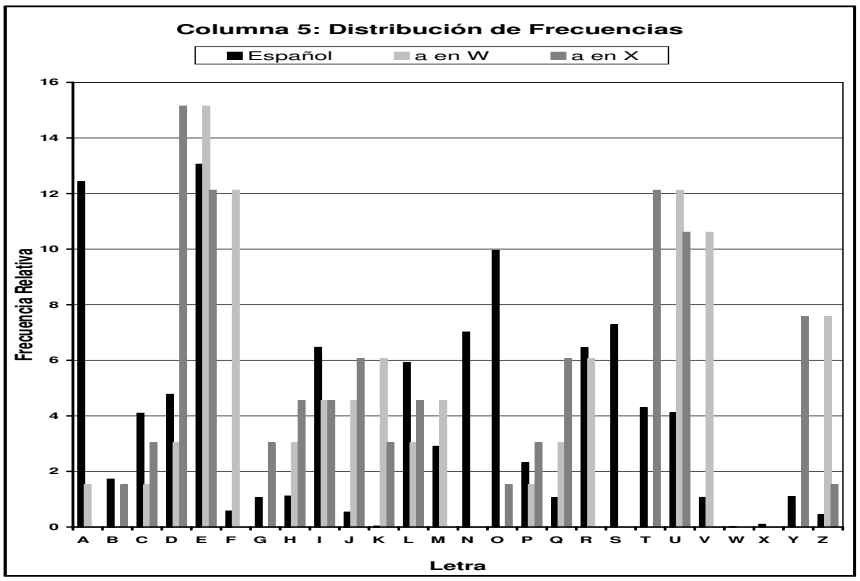

Figura 2.2: Gráfica de distribución de frecuencias para la columna 5, suponiendo desplazamientos de 22 y de 23 lugares.

representada por la `M`. Pero en general las cosas lucen bien, mucho mejor que en los otros tres casos (véase figura 2.3) De manera que vamos a suponer que la quinta letra de la clave es `N`.

Columna 6

```
STBEEMUSHVASVEMDAITCNTUATDEODEOSASSVENSMTIIBUETZAQPAISOOAANOEPEOEA
```

La distribución de frecuencias se encuentra en la Tabla 2.12.

Vemos una situación dificil si sólo vemos la tabla de frecuencias. Pero a primera posibilidad sería suponer que la `E` corresponde a la *e*, es decir desplazamiento cero, con lo que la sexta letra de la clave sería `A`. Esta hipótesis parece plausible porque hasta ahora la palabra clave es `SJUAN` y nuestra nueva suposición implicaría que la clave es `SJUANA`, lo que suena verosímil.

En síntesis, nuestra palabra clave propuesta es `SJUANA`. Si desciframos el criptograma usando el sistema de Vigenère con esta clave obtenemos lo siguiente:

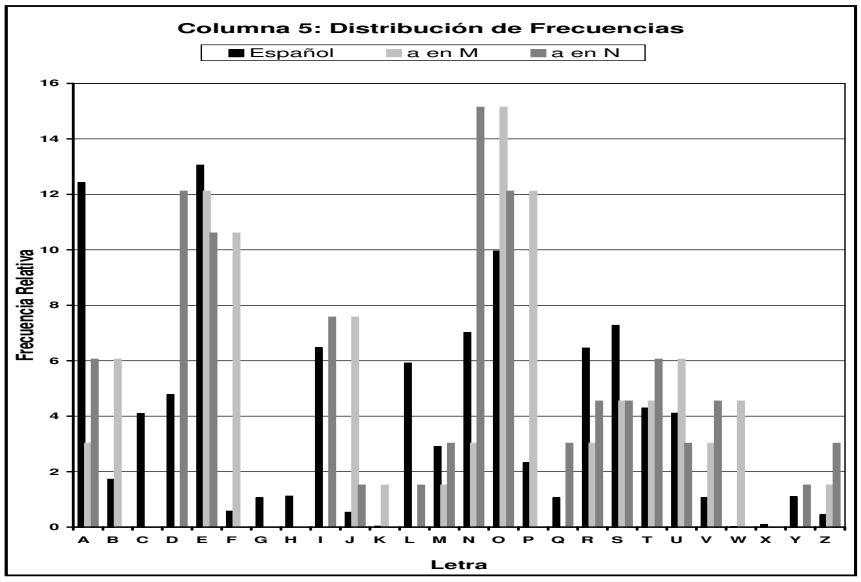

Figura 2.3: Gráfica de distribución de frecuencias para la columna 5 suponiendo desplazamientos de 12 y de 13 lugares.

Letra	**Frec.**	**%**	**Letra**	**Frec.**	**%**
E	10	15.15	C	1	1.52
A	9	13.64	H	1	1.52
S	8	12.12	Q	1	1.52
O	6	9.09	Z	1	1.52
T	6	9.09	F	0	0.00
I	4	6.06	G	0	0.00
D	3	4.55	J	0	0.00
M	3	4.55	K	0	0.00
N	3	4.55	L	0	0.00
U	3	4.55	R	0	0.00
V	3	4.55	W	0	0.00
B	2	3.03	X	0	0.00
P	2	3.03	Y	0	0.00

Tabla 2.12: Distribución de frecuencias de la sexta columna del criptograma.

Criptograma descifrado

enpers eguirm emundo queint eresas enquet eofend ocuand osoloi ntento
ponerb elleza senmie ntendi miento ynomie ntendi miento enlasb elleza
syonoe stimot esoros niriqu ezasya sisiem premec ausama sconte ntopon
erriqu ezasen mipens amient oqueno mipens amient oenlas riquez asynoe
stimoh ermosu raquev encida esdesp ojociv ildela sedade sniriq uezame
agrada fement idaten iendop ormejo renmis verdad escons umirva nidade
sdelav idaque consum irlavi daenva nidade s

Añadiendo puntuación, tenemos un soneto de Sor Juana Inés de la Cruz:

En perseguirme, Mundo ¿qué interesas?
¿En qué te ofendo, cuando sólo intento
poner bellezas en mi entendimiento
y no mi entendimiento en las bellezas?

Yo no estimo tesoros ni riquezas;
y así, siempre me causa más contento
poner riquezas en mi pensamiento
que no mi pensamiento en las riquezas.

Y no estimo hermosura que, vencida,
es despojo civil de las edades,
ni riqueza me agrada fementida,

teniendo por mejor, en mis verdades,
consumir vanidades de la vida
que consumir la vida en vanidades.

2.4.3 La prueba de Friedman

Ahora sabemos cómo podemos criptoanalizar un texto cifrado en un sistema polialfabético. Para ello necesitamos estimar la longitud de la palabra clave. La prueba de Kasiski nos proporciona buenas pistas al respecto, pero no es el único método para encontrar dicha longitud. Otra importante herramienta la proporciona la prueba de Friedman.

Alrededor de 1892 llegaron a los Estados Unidos, provenientes de Kishniev, Rusia, los miembros de la familia Friedman. Uno de ellos era el pequeño Wolfe, de un año de edad,

que luego cambió su nombre por el de William Frederick [Kah99]. William Friedman estudió genética en la Universidad de Cornell, y en 1914 entró a trabajar como genetista en los laboratorios privados Riverbank de George Fabyan. En los laboratorios Riverbank se trabajaba en varias áreas; genética era una de ellas, y criptoanálisis era otra. Friedman fue contratado con la idea de que trabajaría en el primero.

El principal interés del Departamento de Criptoanálisis de Riverbank era esclarecer la autoría de algunos sonetos atribuidos a Shakespeare, y que algunos piensan que en realidad fueron escritos por Francis Bacon. En el Departamento de Criptoanálisis trabajaba una joven experta en literatura inglesa, particularmente en la obra de Shakespeare, de nombre Elizebeth Smith (nótese que no es Elizabeth sino Elizebeth). No sabemos si a Friedman le interesó primero Elizebeth o el criptoanálisis, pero terminó trabajando en el Departamento de Criptoanálisis de Riverbank. Se casó con Elizebeth y en 1921 ambos se fueron al Departamento de Guerra norteamericano [Kah99].

Ya mencionamos que los sistemas criptográficos polialfabéticos pretenden "aplanar" la distribución de frecuencias (equivalentemente de probabilidades) característica del idioma, y hacerla lo más parecida posible a una distribución uniforme. Podríamos preguntarnos entonces: dado un criptograma polialfabético ¿qué lejos estará la distribución de las letras en él, respecto a la distribución uniforme?

Si denotamos por p_i la probabilidad de que el i-ésimo símbolo del alfabeto convencional de 26 letras aparezca, sabemos que $\sum_{i=1}^{26} p_i = 1$. Si el método de cifrado fuera ideal, entonces después de cifrar el texto todas las probabilidades serían iguales a $\frac{1}{26}$. Entonces una posible manera de medir la distancia a la distribución univorme sería calculos qué tanto varían las probabilidades p_i de $\frac{1}{26}$, es decir calcular los valores $p_i - \frac{1}{26}$. Cabe señalar que aquí estamos abusando de la notación: p_i está denotando la probabilidad de que el i-ésimo símbolo aparezca en el criptograma, pero dado que el tamaño del criptograma es finito, en realidad deberíamos de hablar de un estimador de la probabilidad, algo como la frecuencia relativa.

La expresión de arriba no es muy buena, pues si sumamos sobre todos los valores de i, como algunos sumandos pueden ser negativos y otros positivos, el total nos daría una idea equivocada de que tanto difieren las distribuciones. Usar valores absolutos complica las cosas, así que lo mejor es usar una técnica común en estadística, y tomar los cuadrados de los valores. Es decir,

$$\sum_{i=1}^{26} \left(p_i - \frac{1}{26} \right)^2 .$$

Podemos ahora manipular esta expresión como sigue:

$$\begin{aligned}\sum_{i=1}^{26}\left(p_i-\frac{1}{26}\right)^2 &= \sum_{i=1}^{26}\left[p_i^2-\frac{2}{26}p_i+\left(\frac{1}{26}\right)^2\right]\\ &= \sum_{i=1}^{26}p_i^2-\frac{2}{26}\sum_{i=1}^{26}p_i+\sum_{i=1}^{26}\left(\frac{1}{26}\right)^2\\ &= \sum_{i=1}^{26}p_i^2-\frac{2}{26}(1)+26\left(\frac{1}{26}\right)^2\\ &= \sum_{i=1}^{26}p_i^2-\frac{1}{26}\end{aligned} \tag{2.4.1}$$

Esta expresión nos dice qué tan diferente es la distribución de las letras en el texto respecto a la distribución uniforme. Por eso se le suele llamar el grado de aspereza o *roughness* [Sin66] del lenguaje. La suma de los cuadrados de las probabilidades la podemos interpretar como *la probabilidad de que al tomar aleatoriamente (sin reemplazo) dos letras del texto, ambas sean iguales.* Al valor de ésta probabilidad se le llama *el índice de coincidencias*, y fue una innovación de Friedman. Se trata de la primera vez que en criptoanálisis se proporciona una expresión analítica para obtener el valor de algo en vez de adivinarlo. Principalmente por esta aportación se le considera a Friedman el padre del criptoanálisis moderno. Con el índice de coincidencias, Friedman permitió el uso de todas las herramientas estadísticas para el criptoanálisis.

En la práctica sólo contamos con una muestra de texto, de manera que tenemos que estimar el índice de coincidencias. Supongamos que tenemos una muestra de texto T de N letras sobre el alfabeto convencional de 26 letras, y sea n_i el número de veces que se repite la i-ésima letra del alfabeto en el texto T. Entonces la probabilidad de que al tomar tomar dos letras resulta que ambas son la i-ésima es igual al número de éxitos entre el número de casos posibles:

$$\frac{\binom{n_i}{2}}{\binom{N}{2}}=\frac{n_i(n_i-1)}{N(N-1)} \tag{2.4.2}$$

Definición 2.1 El *índice de coincidencias* de una muestra de texto T sobre el alfabeto convencional de 26 letras es:

$$\mathrm{IC}(T)=\sum_{i=1}^{26}\frac{n_i(n_i-1)}{N(N-1)} \tag{2.4.3}$$

donde n_i es el número de apariciones de la i-ésima letra del alfabeto en T, y N es la longitud del texto T.

El valor del índice de coincidencias de textos escritos en un idioma particular tiende a parecerse, dado que está en función de las frecuencias de aparición de cada letra. Para textos en inglés suficientemente representativos el valor del índice de coincidencias es de aproximadamente 0.065 [Sin66, Lew00, Bau00]; para textos en español Pratt y Fletcher encontraron en 1939 que era de alrededor de 0.0755, Eyraud en 1953 obtuvieron el valor 0.0769, Solomon Kullback obtuvo 0.0775 en 1976, Randall Nichols [Nic] 0.0747 en 1995 sobre una muestra de 60,115 letras, y nosotros encontramos el valor de 0.0744 sobre una muestra de 319,858 letras, como puede verse en el Apéndice A. Este último es el valor que utilizaremos en adelante. El valor del índice de coincidencias para un texto aleatorio con distribución uniforme es de $1/26 \approx 0.038$.

¿Para qué nos sirve el índice de coincidencias en el criptoanálisis de sistemas polialfabéticos? Hemos dicho que el índice de coincidencias de textos escritos en un idioma particular debe parecerse a cierta constante, y que el índice de coincidencias de un texto aleatorio uniforme es también una constante, a saber 0.038. Si tenemos un criptograma monoalfabético, dado que cada letra es reemplazada por otra única letra, la probabilidad de que al elegir dos letras aleatorias estas sean la misma se conserva (i.e. es la misma que la del idioma original). En el cálculo del índice de coincidencias estamos ignorando la identidad de las letras y sólo nos interesa que sean la misma. Esto significa que el índice de coincidencias de un texto claro cualquiera escrito en algún idioma particular coincidirá con el que se calcule a cualquier criptograma obtenido de él mediante una substitución monoalfabética.

Pero si el criptograma se obitene mediante una substitución polialfabética, dado que se ha pretendido uniformar la distribución de letras, el valor del índice de coincidencias del criptograma deberá ser menor: cuanto más cercano a 0.038 mejor será el cifrado que se ha hecho. Esta "calidad" del cifrado, su cualidad para "desorganizar" el texto original, está en relación directa con la longitud de la palabra clave o, para ser más precisos, con el número de alfabetos diferentes usados en el cifrado. Así que, de alguna manera, el índice de coincidencias del criptograma nos proporciona información acerca del número del alfabetos usados para cifrarlo, y podemos pensar en usarlo para estimar la longitud de la clave.

Supongamos que tenemos un texto T de N letras cifrado polialfabéticamente. El índice de coincidencias del texto, es decir, la probabilidad de que al elegir aleatoriamente dos letras estas sean iguales es:

$$\mathrm{IC}(T) = \frac{M}{\binom{N}{2}} = \frac{2M}{N(N-1)} \tag{2.4.4}$$

donde M es el número de posibles selecciones de dos letras iguales.

Supongamos ahora que sabemos que el texto claro estaba originalmente en español, que al cifrar se usaron r alfabetos distintos, y que hemos acomodado el criptograma en r columnas. Cada columna tiene entonces aproximadamente N/r letras, o a lo más $(N/r)+1$ letras. Por simplicidad, suponemos que el número de letras de T es un múltiplo de r.

Calculemos M; para ello consideramos dos casos. Primero, el caso en que elegimos aleatoriamente dos letras de la misma columna sin reemplazo. Para elegir la primera tenemos N opciones (una letra cualquiera), y para la segunda tenemos que elegir alguna letra de la misma columna en la que está la primera elegida, pero excluyendo a ésta. Así que tenemos $(N/r)-1$ posibilidades. Como consideramos como casos iguales aquel en el que se elige la letra i-ésima y luego la j-ésima de la columna y aquel en el que se eligen en orden inverso, hay que dividir entre dos. En síntesis, el número de posibles elecciones de dos letras de la misma columna es:

$$\frac{N\left(\frac{N}{r}-1\right)}{2} \tag{2.4.5}$$

Tomamos ahora el caso en que elegimos dos letras de diferentes columnas. Tenemos N opciones para la primera letra y $N-(N/r)$ para la segunda, y luego dividimos entre dos. Entonces el número de posibles elecciones de dos letras de diferente columna es:

$$\frac{N\left(N-\frac{N}{r}\right)}{2} \tag{2.4.6}$$

Por definición el índice de coincidencias es la proporción de veces que se elige la misma letra en dos experimentos. Si las letras están en la misma columna de nuestro criptograma acomodado en r columnas, estamos suponiendo que pertenecen a la misma substitución monoalfabética y por tanto, la proporción de veces que elegiremos la misma letra en los dos experimentos será justamente el índice de coincidencias del idioma en el que estaba el texto claro (0.0744 en español). Por otra parte la proporción de veces que elegiremos dos letras iguales, dado que las tomamos de columnas diferentes, es menor al índice de coincidencias del idioma, de hecho esperamos que sea aproximadamente 0.038 (como si fuera texto aleatorio con distribución uniforme).

Así que el número de veces que esperamos obtener dos letras iguales si las elegimos aleatoriamente del criptograma es, usando las ecuaciones (2.4.5) y (2.4.6), y lo dicho en el párrafo anterior, está dado por:

$$\begin{aligned} M &\approx \left[\frac{N\left(\frac{N}{r}-1\right)}{2}\right](0.0744)+\left[\frac{N\left(N-\frac{N}{r}\right)}{2}\right](0.038) \\ &= (0.0744)\left(\frac{N(N-r)}{2r}\right)+(0.038)\left(\frac{N^2(r-1)}{2r}\right). \end{aligned}$$

Reemplazando M en la expresión (2.4.4) obtenemos:

$$\begin{aligned}
\mathrm{IC}(T) &\approx \frac{2}{N(N-1)}\left[(0.0744)\left(\frac{N(N-r)}{2r}\right)+(0.038)\left(\frac{N^2(r-1)}{2r}\right)\right] \\
&= \frac{1}{N-1}\left((0.0744)\frac{N-r}{r}+(0.038)\frac{N(r-1)}{r}\right) \\
&= \frac{1}{r(N-1)}\Big((0.0744)N-(0.0744)r+(0.038)Nr-(0.038)N\Big) \\
&= \frac{1}{r(N-1)}\Big((0.0744-0.038)N+r((0.038)N-0.0744)\Big) \\
&= \frac{1}{r(N-1)}\Big((0.036)N+r((0.038)N-0.0744)\Big). \qquad (2.4.7)
\end{aligned}$$

Ahora despejamos r de (2.4.7) y obtenemos:

$$\begin{aligned}
r &\approx \frac{1}{\mathrm{IC}(T)(N-1)}\Big((0.036)N+(0.038)Nr-(0.0744)r\Big) \\
&= \frac{0.036\,N}{\mathrm{IC}(T)(N-1)}+\left(\frac{(0.038)N-0.0744}{\mathrm{IC}(T)(N-1)}\right)r,
\end{aligned}$$

de donde:

$$r\left(1-\frac{(0.038)N-0.0744}{\mathrm{IC}(T)(N-1)}\right)=\frac{(0.036)N}{\mathrm{IC}(T)(N-1)}.$$

Finalmente:

$$\begin{aligned}
r &\approx \left(\frac{(0.036)N}{\mathrm{IC}(T)(N-1)}\right)\Big/\left(\frac{\mathrm{IC}(T)(N-1)-(0.038)N+0.0744}{\mathrm{IC}(T)(N-1)}\right) \\
&= \frac{(0.036)N}{\mathrm{IC}(T)(N-1)-(0.038)N+0.0744}. \qquad (2.4.8)
\end{aligned}$$

Ahora tenemos una expresión que nos estima el número de alfabetos usados para obtener el criptograma T a partir de un texto en español, usando un método polialfabético (véase [Lew00]).

2.4.4 Conceptos afines

Otra manera de interpretar el índice de coincidencias surge de pensar en poner en correspondencia dos textos cualesquiera. Si después nos fijamos en cuantas posiciones tienen ambos textos la misma letra obtendremos un cierto *número de coincidencias*. Si los textos

están en idiomas diferentes el número de coincidencias será bastante menor que el que resulta si los textos están en el mismo idioma. El número de coincidencias para dos textos en español es de aproximadamente el 7.44% del total de letras en los textos, por ejemplo. A este número se le suele denotar con la letra griega κ [Bau00].

Tomamos ahora dos textos, los ponemos en correspondencia, contamos sus coincidencias, y luego desplazamos uno de ellos un lugar a la izquierda haciendo que la letra que "sale" reaparezca por la derecha (lo que constituye una rotación del texto). Volvemos a contar las coincidencias y continuamos este proceso de rotación-conteo hasta regresar a la posición original del texto. Al final obtendremos una lista de números, tantos como la longitud de los textos (que suponemos son de la misma longitud por simplicidad). Si ahora obtenemos el promedio de estos números estamos en realidad obteniendo un cierto valor promedio de κ para ellos. A este valor se le denota por la letra griega χ [Bau00].

Si los dos textos son en realidad uno solo, es decir si ponemos un texto en correspondencia consigo mismo y obtenemos el promedio de las coincidencias sobre todas las posibles rotaciones del texto, entonces estaremos calculando el valor de χ para un texto consigo mismo, lo que suele denotarse por ψ.

Pero el promedio obtenido calculando ψ sobre un texto tiene un inconveniente. Cuando los dos textos, que son el mismo, están en correspondencia y ambos están en la posición inicial, antes de cualquier rotación, entonces, por supuesto, coinciden en *todas* sus posiciones y el valor de κ será exactamente 1 (100% de coincidencias). Este valor no es muy típico representativo del resto de los valores que se obtendrán con las rotaciones posteriores, así que convendría eliminarlo del promedio calculado por ψ. Al valor de ψ sin considerar este caso particular se le llama ϕ y es el que utilizamos como valor del índice de coincidencias en la pasada subsección.

En síntesis. Sean T_1 y T_2 dos textos cualesquiera de igual longitud (ℓ):

$\kappa(T_1, T_2)$ Es el número de posiciones en las que coinciden T_1 y T_2 puestos en correspondencia.

$\chi(T_1, T_2)$ Es el valor promedio de $\kappa(T_1, T_{2,(r)})$, donde $T_{2,(r)}$ el el resultado de rotar el texto T_2 r lugares, $r \in \{0, \ldots, \ell - 1\}$.

$\psi(T_1)$ Es el valor de $\chi(T_1, T_1)$.

$\phi(T_1)$ Es el valor promedio de $\kappa(T_1, T_{1,(r)})$ donde $T_{1,(r)}$ es el resultado de rotar el texto T_1 r posiciones, $r \in \{1, \ldots, \ell - 1\}$.

Normalmente el valor de κ para un idioma particular se denota con la kappa minúscula

y un subíndice: $\kappa_{es} = 0.0744$ es el valor para el español, $\kappa_{in} = 0.065$ para el inglés[4] y $\kappa_{un} = 0.038$ para el texto uniformemente distribuido.

2.4.5 Criptanálisis utilizando el índice de coincidencias

Supongamos que hemos recibido el criptograma que aparece a continuación y que sabemos que ha sido obtenido con un cifrado de Vigerère de un texto originalmente escrito en español:

```
WBTIA NZHGU ROILX BJYLW ENMHR BNRDK YMUSL XDOQU TUWIV RAJIX JFTSE
RTYKN FEIBD KYKXB VPXLU MMLCG PDZQS WAPKI YRTUL ZIXTX QYYAN DHVTH
UMQOI XWHBJ LIESS EFBCY BRRHC VRLCM BSTVP NTIKJ PTPOR HIVXD XRVNR
ULNXT VSRHM WWOOM YNUHM UQTWW VBFWN LVISG EUYDS EPDFG OMNZE IPNDX
IBSTU XVUHX XFLYU RIJMC CWIMN HCIJR OIUGY XWHAX HUBII LNSLX RRBWI
BHXPO NDIBA OCSOH UAWDB EMEEE XWWRX EVRDN IKOEE SNZUG LSUEG NGOMM
OUEDY EMIAF BIUBI SKXBN RFNRI LNJXP DDKYT NHKEL NKHUD QAEFU ESKXB
EEVQE MGJRB GKNIX WWQXP ONIMM EWGSD QEHLN RHRTH YDWCS XWWNR UIUQN
EOZQH LXVBR FNHXM ACWMO YQMGU SRIQQ EYVBI FEQHQ FKXAH UXRTY KROTP
JHDUL NJHXD BHUKR CGIQZ YNIMR XPDYU SMWRT EOMEF IVOGS BQYIT NGHFU
RUFKD SEPRH DVCNB ZSOCU SBAOL IOPEH ADABW PNUMX JRTYQ TUHBR ZXWSN
BXIAO SSVVU GXASF YUZKL IWRHI QGHYL RSGXH FSIUX ENIUR PUJJV XGKBU
MBXAT RGBQO VJRXE THUFT JGWEP NIKCN ZXGLA UMMUO XWSNT UTJQN EOYEB
QICVS QZKWP JRXWH ALITC IKEBQ YMKAS VMRAF IZZIX RRSKY UNBXW WRHJW
LOIEU NDIZN JXRWN HXMAW LEDPQ XIYIG XRQUF IBQXV HZEHQ JGIIU BBUAY
FHICN IKCNM TLDOY UVEWL XRQUF VXJXP FNRUT USKSO RINMW WTPDE YMIJF
TCDNB WMWWK PHYQY AYOWE GVZIT JPNIQ NIYVX FTHLB IBIPO TZXRI NZJAX
VFRTG CHJXR WHHIA XQTFD YBYZX MEIGR LYVCI KEHAB CLNGW SQDKC RXHXP
HCHYO DBMSF BCIAN ZEEPN RUXXF JYHRB MCYWX WHQUU TUWTH HYQHB NOJYL
RDKCN RTFDB RFQPO WSSBH FIVSK GHQHY KRPBH DCELY DSIIQ FQVIM OKPHN
BACWO IEUGU XMUOA SQEQK CNOEG DAPUA NDHVH YLUTX FWIVH RLIIC XPONH
YAYCG HLBSI VVIVL DUKGQ URTHT HUMMU ZTQDO QFICC ESVNO KCNSK EKVZU
LNIGV HZUHL XBGEW HHUTM SMSOR TIYDS OMYVQ UTJGM IQQYF TJGWI VNDWP
XPBIQ NOUGZ IXHRA TYYDW XVDDK YMUZT IVGKP QNGXP HFULD RFBEB YUNMW
RKMDC ELANB HVGBD KCRXH XHYUL MYZBG RDKYX XFLYD ZELTN VBGLR IYUNF
VIGDK YLNOE PLRDU LNZTR WRIYX DGBIV RTIVH GXPON CUANR HRDGE FWBOX
PONIY TXDKS PRJCW HZTSW EQFML OEDRY QYAYI XPDPE HTJQN EOYUJ IBCVE
VVUFU RGFSF BBIYD WHUXR SIVUO WIONU MXJRT TURWO VCCEI VHDIU KFXCG
```

[4]Para el inglés en general $\kappa_{in} \in [0.065, 0.069]$.

Letra	Frec.	%	Letra	Frec.	%
I	110	6.44	T	61	3.57
U	102	5.98	M	60	3.51
X	95	5.57	Q	60	3.51
H	89	5.21	S	56	3.28
R	86	5.04	G	55	3.22
N	82	4.80	L	52	3.05
Y	80	4.69	C	50	2.93
B	75	4.39	F	50	2.93
W	74	4.34	K	49	2.87
E	72	4.22	P	46	2.69
V	66	3.87	A	38	2.23
D	65	3.81	J	38	2.23
O	61	3.57	Z	35	2.05

Tabla 2.13: Frecuencias de letras en el criptograma polialfabético.

```
VZIYD SLIOY QGIKO EEPBB CVNFT CTHUY ZJVBN DQUOV QCGVD QEGWU WGIUB
TYIWH XUXRH UIUOV YDYJU UKWXR UBWIL XBJYL WENMZ IXWHC KMQNG XHRAO
MMUZT QDFUX WWOFS OVDYZ JCYVH PYYVM CEIQH UPWBG XVYVS CWBMF IUPUX
MB
```

Nuestro análisis de frecuencia del criptograma, que tiene 1707 letras, nos arroja los datos que aparecen en la Tabla 2.13. El índice de coincidencias del criptograma resulta ser 0.0413. Si revisamos en el Apéndice A la tabla del índice de coincidencias (IC) para textos cifrados polialfabéticamente en español, nos percataremos de que, según esa tabla se están usando alrededor de 10 alfabetos para cifrar el texto. Si usamos la expresión (2.4.8) obtenemos que $r \approx 10.85$. Podemos ahora hacer nuestro análisis de Kasiski para determinar el valor de r con mayor certidumbre.

En lo que sigue cambiaremos un poco la notación respecto al primer caso de estudio para abreviar la escritura de los casos diferentes para cada columna.

En la Tabla 2.14 se muestra el resultado de la prueba de Kasiski. Al ver la tabla del análisis de Kasiski podemos notar que:

- El 7 aparece en 8 renglones
- El 8 (2, 2, 2) aparece en 2 renglones

Secuencia	Distancia	Factores primos
RI	118, 113, 399, 63, 441, 27	2, 59 / 113 / 3, 7, 19 / 3, 3, 7 / 3, 3, 7, 7 / 3, 3, 3
VH	324, 10, 71, 194, 122, 143	2, 2, 3, 3, 3, 3 / 2, 5 / 71 / 2, 97 / 2, 61 / 11, 13
XWH	140, 472, 297, 612	2, 2, 5, 7 / 2, 2, 2, 59 / 3, 3, 3, 11 / 2, 2, 3, 3, 17
IV	117, 36, 108, 4, 99, 23	3, 3, 13 / 2, 2, 3, 3 / 2, 2, 3, 3, 3 / 2, 2 / 3, 3, 11 / 23
NIK	376, 144	2, 2, 2, 47 / 2, 2, 2, 2, 3, 3
NDI	495	3, 3, 5, 11
DJY	36, 297, 927, 63, 27	2, 2, 3, 3 / 3, 3, 3, 11 / 3, 3, 103 / 3, 3, 7 / 3, 3, 3
HRB	994	2, 7, 71
XPD	162, 945	2, 3, 3, 3, 3 / 3, 3, 3, 5, 7
WIV	1099, 117	7, 157 / 3, 3, 13
IANZ	999	3, 3, 3, 37
DCEL	243	3, 3, 3, 3, 3
RDKY	1323	3, 3, 3, 7, 7
XPON	126, 747, 261, 18	2, 3, 3, 7 / 3, 3, 83 / 3, 3, 29 / 2, 3, 3
THUM	1071	3, 3, 7, 17
ANDHV	1126	2, 563
XRQUF	45	3, 3, 5

Tabla 2.14: Análisis de repetición de secuencias. La columna del centro señala las distancia entre apariciones sucesivas de la secuencia en la columna de la izquierda. En la columna derecha está la descomposición en factores primos de cada distancia.

- El 9 (3, 3) aparece en 15 renglones
- El 10 (2, 5) aparece en 2 renglones
- El 11 aparece en 5 renglones

así que lo más probable es que se haya usado una clave de longitud 9 para cifrarlo.

El criptograma en 9 columnas se ve asi:

```
WBTIANZHG UROILXBJY LWENMHRBN RDKYMUSLX DOQUTUWIV RAJIXJFTS ERTYKNFEI
BDKYKXBVP XLUMMLCGP DZQSWAPKI YRTULZIXT XQYYANDHV THUMQOIXW HBJLIESSE
FBCYBRRHC VRLCMBSTV PNTIKJPTP ORHIVXDXR VNRULNXTV SRHMWWOOM YNUHMUQTW
WVBFWNLVI SGEUYDSEP DFGOMNZEI PNDXIBSTU XVUHXXFLY URIJMCCWI MNHCIJROI
UGYXWHAXH UBIILNSLX RRBWIBHXP ONDIBAOCS OHUAWDBEM EEEXWWRXE VRDNIKOEE
SNZUGLSUE GNGOMMOUE DYEMIAFBI UBISKXBNR FNRILNJXP DDKYTNHKE LNKHUDQAE
FUESKXBEE VQEMGJRBG KNIXWWQXP ONIMMEWGS DQEHLNRHR THYDWCSXW WNRUIUQNE
OZQHLXVBR FNHXMACWM OYQMGUSRI QQEYVBIFE QHQFKXAHU XRTYKROTP JHDULNJHX
DBHUKRCGI QZYNIMRXP DYUSMWRTE OMEFIVOGS BQYITNGHF URUFKDSEP RHDVCNBZS
OCUSBAOLI OPEHADABW PNUMXJRTY QTUHBRZXW SNBXIAOSS VVUGXASFY UZKLIWRHI
QGHYLRSGX HFSIUXENI URPUJJVXG KBUMBXATR GBQOVJRXE THUFTJGWE PNIKCNZXG
LAUMMUOXW SNTUTJQNE OYEBQICVS QZKWPJRXW HALITCIKE BQYMKASVM RAFIZZIXR
RSKYUNBXW WRHJWLOIE UNDIZNJXR WNHXMAWLE DPQXIYIGX RQUFIBQXV HZEHQJGII
UBBUAYFHI CNIKCNMTL DOYUVEWLX RQUFVXJXP FNRUTUSKS ORINMWWTP DEYMIJFTC
DNBWMWWKP HYQYAYOWE GVZITJPNI QNIYVXFTH LBIBIPOTZ XRINZJAXV FRTGCHJXR
WHHIAXQTF DYBYZXMEI GRLYVCIKE HABCLNGWS QDKCRXHXP HCHYODBMS FBCIANZEE
PNRUXXFJY HRBMCYWXW HQUUTUWTH HYQHBNOJY LRDKCNRTF DBRFQPOWS SBHFIVSKG
HQHYKRPBH DCELYDSII QFQVIMOKP HNBACWOIE UGUXMUOAS QEQKCNOEG DAPUANDHV
HYLUTXFWI VHRLIICXP ONHYAYCGH LBSIVVIVL DUKGQURTH THUMMUZTQ DOQFICCES
VNOKCNSKE KVZULNIGV HZUHLXBGE WHHUTMSMS ORTIYDSOM YVQUTJGMI QQYFTJGWI
VNDWPXPBI QNOUGZIXH RATYYDWXV DDKYMUZTI VGKPQNGXP HFULDRFBE BYUNMWRKM
DCELANBHV GBDKCRXHX HYULMYZBG RDKYXXFLY DZELTNVBG LRIYUNFVI GDKYLNOEP
LRDULNZTR WRIYXDGBI VRTIVHGXP ONCUANRHR DGEFWBOXP ONIYTXDKS PRJCWHZTS
WEQFMLOED RYQYAYIXP DPEHTJQNE OYUJIBCVE VVUFURGFS FBBIYDWHU XRSIVUOWI
ONUMXJRTT URWOVCCEI VHDIUKFXC GVZIYDSLI OYQGIKOEE PBBCVNFTC THUYZJVBN
DQUOVQCGV DQEGWUWGI UBTYIWHXU XRHUIUOVY DYJUUKWXR UBWILXBJY LWENMZIXW
HCKMQNGXH RAOMMUZTQ DFUXWWOFS OVDYZJCYV HPYYVMCEI QHUPWBGXV YVSCWBMFI
UPUXMB
```

Columna 1

```
WBFWUSFODOQLRUDWPHHVVDLWODHUXVSUGVFQOHSWCHDHDVKQGWRUDRLDPDRDKODP
UOUDGGHQOHRHVDVUDRYOPOUOQOQKQWRQHHHLWDROOGXODXVXOFDQBSGHDFLQLUDO
 VDDVODHRTSUEDTXUVTBROXHDQTYHLOFPUQEHYMVLWJRUPRHDFFSDDQBGPXTLYU
```

Letra	Frec.	%	Letra	Frec.	%
D	27	14.21	S	6	3.16
O	19	10.00	T	5	2.63
H	18	9.47	B	4	2.11
U	15	7.89	Y	4	2.11
Q	13	6.84	K	3	1.58
R	12	6.32	E	2	1.05
V	12	6.32	C	1	0.53
L	9	4.74	J	1	0.53
W	9	4.74	M	1	0.53
F	8	4.21	A	0	0.00
G	7	3.68	I	0	0.00
P	7	3.68	N	0	0.00
X	7	3.68	Z	0	0.00

Tabla 2.15: Distribución de frecuencias de la columna uno.

La distribución de frecuencias se encuentra en la Tabla 2.15. El indice de coincidencias de la primer columna es 0.0714.

Opciones:

1. $e \to$ D, $a \to$ Z. Desplazamiento de 25. Poco probable porque la frecuencia de la o (N) sería cero, yla de la i (H) demasiado alta, igual que la de w (V) que es poco usual en español.

2. $e \to$ O, $a \to$ K. Desplazamiento de 10. Poco probable porque la o (Y) tendría baja frecuencia, y la w (G) muy alta (mayor que la de o por ejemplo).

3. $e \to$ H, $a \to$ D. Desplazamiento de 3.

Siendo la tercera opción la mejor, suponemos que la primera letra de la clave es D.

Columna 2

```
BDBVGNUZBCGASBNHNQYNNCRENQCRLRGBNQNZPFNRNYYRCHVNBRYRQAWZNFRYNYYN
RYNOVRQFNZAYRPHBFDRRNNBNQMTBZNQNAYNBHDDNYVRVOQNVHNQHQNBAPNBDRGUR
 GZGVYYPAHRREDHRRVHQQRRCBEHVFRNBBBHRBNNRNNHHZNAZERBBAOQYDRRHWVP
```

Letra	Frec.	%	Letra	Frec.	%
N	33	17.37	P	5	2.63
R	28	14.74	E	4	2.11
B	19	10.00	O	3	1.58
H	14	7.37	U	2	1.05
Y	14	7.37	W	2	1.05
Q	13	6.84	L	1	0.53
V	10	5.26	M	1	0.53
A	8	4.21	S	1	0.53
Z	8	4.21	T	1	0.53
D	7	3.68	I	0	0.00
G	6	3.16	J	0	0.00
C	5	2.63	K	0	0.00
F	5	2.63	X	0	0.00

Tabla 2.16: Distribución de frecuencias de la segunda columna.

La distribución de frecuencias está en la Tabla 2.16. El índice de coincidencias de la columna es 0.0892.

Opciones:

1. $e \to$ N, $a \to$ J. Desplazamiento de 9. Poco probable, pues la frecuencia de a sería cero, la de o (X) también, y la de i (R) muy alta.
2. $e \to$ R, $a \to$ N. Desplazamiento de 13.
3. $e \to$ B, $a \to$ X. Desplazamiento de 23. Poco probable, pues la frecuencia de o (L) sería demasiado baja, al igual que la de n (K).

La mejor opción es la segunda, de manera que la segunda letra de la clave probablemente es N.

Columna 3

```
TKCBYZEQHUHUKBBHRHLODEDQUUKOULEIGEHYESTHIQBBERZODIQWEOEQTGBEIQUU
PEDYZLUQHUTUTEDTUKTHDDIIEEUUKHUIBQBSHKKCUZHDQYRUUREQYBQLQRIKDUKT
 KEEUQJYJUHIEKYTUUUYUIIHRQUQUIIBBWUTJUHDKRDDKIFEYTCHPQYUKJSUESU
```

Letra	Frec.	%	Letra	Frec.	%
U	31	16.32	O	4	2.11
E	19	10.00	S	4	2.11
Q	16	8.42	Z	4	2.11
H	15	7.89	C	3	1.58
I	14	7.37	G	2	1.05
K	14	7.37	P	2	1.05
D	12	6.32	W	2	1.05
B	11	5.79	F	1	0.53
T	11	5.79	A	0	0.00
Y	10	5.26	M	0	0.00
R	7	3.68	N	0	0.00
J	4	2.11	V	0	0.00
L	4	2.11	X	0	0.00

Tabla 2.17: Distribución de frecuencias de la tercer columna.

La distribución de frecuencias se encuentra en la Tabla 2.17. El índice de coincidencias es 0.0784.

Opciones:

1. $e \to$ U, $a \to$ Q. Desplazamiento de 16.

Con esto la tercera letra de la clave es Q

Columna 4

```
IYYFXUSHUSYMYUWIUYUKWLUFMOMIMCUIOMXNHIUJKYYMLLUUKYYOGMNSIOWMXMSM
UBIUIYUVYHYLIHIYXYUIXISMYFHMWXFYCHAIUYYUJIUYUYUHAIHFIXOIXUBCKXGI
 PLFFGUYIMMJXYDYFGFMFNNYFKMULYYICIPYLHCNHUUVLKIHMGIFUFFNYCIYNCX
```

La distribución de frecuencias se encuentra en la Tabla 2.18. El índice de coincidencias es 0.0854.

Opciones:

1. $e \to$ Y, $a \to$ U. Desplazamiento de 20.

Letra	Frec.	%	Letra	Frec.	%
Y	29	15.26	S	5	2.63
U	25	13.16	W	4	2.11
I	24	12.63	J	3	1.58
M	17	8.95	A	2	1.05
F	14	7.37	B	2	1.05
H	11	5.79	P	2	1.05
X	11	5.79	V	2	1.05
L	8	4.21	D	1	0.53
C	7	3.68	E	0	0.00
N	7	3.68	Q	0	0.00
K	6	3.16	R	0	0.00
G	5	2.63	T	0	0.00
O	5	2.63	Z	0	0.00

Tabla 2.18: Distribución de frecuencias de la cuarta columna.

2. $e \to$ U, $a \to$ Q. Desplazamiento de 16. Poco probable, pues la w (M) tendría frecuencia muy alta, y la o (E) frecuencia cero.

3. $e \to$ I, $a \to$ E. Desplazamiento de 4. Poco probable, pues r (V) y s (W) tendrían frecuencias muy bajas, y la b (F) muy alta.

Aparentemente entonces la cuarta letra de la clave es U.

Columna 5

```
AKBWWGKLKBLMUAMAXKTCPALMXVQLMMYLMGMIAUTWCAZCYILGCXAVWMMWKMIIWGMX
JQZVTVTIALYMVTUIWMLVIBKMVIBBPMVVLBCVTMXAIYIZTALXWLLKTIVTITIRCMQY
 QTQUIUVXQWMWTWKKXTKIMZOQCMTDUTYVLWKIMIIUILCICZQICAIAITMLWVZMWM
```

La distribución de frecuencias se encuentra en la Tabla 2.19. El índice de coincidencias es 0.0739.

Opciones:

1. $e \to$ M, $a \to$ I. Desplazamiento de 9.

Letra	Frec.	%	Letra	Frec.	%
M	24	12.63	Y	6	3.16
I	23	12.11	Z	6	3.16
T	16	8.42	G	4	2.11
L	15	7.89	P	2	1.05
W	15	7.89	D	1	0.53
V	14	7.37	J	1	0.53
A	12	6.32	O	1	0.53
K	11	5.79	R	1	0.53
C	10	5.26	E	0	0.00
X	8	4.21	F	0	0.00
Q	7	3.68	H	0	0.00
U	7	3.68	N	0	0.00
B	6	3.16	S	0	0.00

Tabla 2.19: Distribución de frecuencias de la columna cinco.

2. $e \rightarrow$ I, $a \rightarrow$ E. Desplazamiento de 4. La w (A) quedaría con frecuencia muy alta, y la o (S) muy baja.

Suponemos entonces que la quinta letra de la clave es I.

Columna 6

```
NXRNHLXXRARUNYWXXRXNXNNLJQNXLBDNMJAMDXJLNYXYDINZRDYCUUHAJNBAWUWJ
JINEJCUMYXDYHJKWWUZXBAXEBVRXJAXXNNWVMUXNBDUJUNNXDNNXNAJCYUPXNUUD
 NNBRKKMJOWCWNCRDAJABWJDPNUJRNXDNXBNEUJKDUNNWNZJJHNVNCJWNHUJZBB
```

La distribución de frecuencias se encuentra en la Tabla 2.20. El índice de coincidencias es 0.0840.

Opciones:

1. $e \rightarrow$ N, $a \rightarrow$ J. Desplazamiento de 9.

Supondremos que la sexta letra de la clave es J.

Letra	Frec.	%	Letra	Frec.	%
N	34	17.89	K	4	2.11
X	22	11.58	L	4	2.11
J	20	10.53	Z	4	2.11
U	16	8.42	E	3	1.58
D	12	6.32	V	3	1.58
W	11	5.79	I	2	1.05
B	10	5.26	P	2	1.05
A	9	4.74	O	1	0.53
R	9	4.74	Q	1	0.53
Y	7	3.68	F	0	0.00
C	6	3.16	G	0	0.00
H	5	2.63	S	0	0.00
M	5	2.63	T	0	0.00

Tabla 2.20: Distribución de frecuencias de la columna seis.

Columna 7

```
ZBRLASBVCOSOBFWQFPFSPBZORCGBCSSSORCRAEQOMOMWSCIIXGICWZRPPZHFQSRR
VCJWPIWOCBWZGQFHOSIDSOBWIOZARWJFGOOISZFRCSOCWDXFBJRAGORIISOHRORS
 GVOGOWCFIOCRHSOSSGSQWABOOZGFFDWFBGFSQROQQJBRZIGFJZSDCGROZOVIM
```

La distribución de frecuencias se encuentra en la Tabla 2.21. El índice de coincidencias es 0.0715.

Opciones:

1. $e \to$ O, $a \to$ K. Desplazamiento de 10. La w (G) quedaría con muy alta frecuencia, y la n (X) con muy poca, al igual que la o (Y).
2. $e \to$ S, $a \to$ O. Desplazamiento de 14.

Siendo la segunda opción la mejor, supondremos que la séptima letra de la clave es O.

Columna 8

```
HVHVXUEBGLGXXHKTJBWKBHTETGXJGTELUBWXBNNITWEXIXGXHBXEGTBKTEXBXRTT
```

Letra	Frec.	%	Letra	Frec.	%
O	25	13.23	P	5	2.65
S	20	10.58	D	4	2.12
R	17	8.99	H	4	2.12
F	14	7.41	V	4	2.12
C	13	6.88	M	3	1.59
G	12	6.35	X	2	1.06
I	12	6.35	E	1	0.53
W	12	6.35	L	1	0.53
B	11	5.82	K	0	0.00
Z	11	5.82	N	0	0.00
Q	8	4.23	T	0	0.00
A	5	2.65	U	0	0.00
J	5	2.65	Y	0	0.00

Tabla 2.21: Distribución de frecuencias de la séptima columna.

```
XVXLNKTKGGXBXNXXFLXXTCNGFGXTXLXTWJIVMTLHVLVYIHTLEXHHHSXKGKTXTATO
 XBXFEXETXOWXKXTEFWVXTXMWETMBVKHTJXESTOEANHZHXXITXEKHEWKETWBXF
```

La distribución de frecuencias se encuentra en la Tabla 2.22. El índice de coincidencias es 0.0925.

Opciones:

1. $e \to$ X, $a \to$ T. Desplazamiento de 19.

Concluimos que la octava letra de la clave es T.

Columna 9

```
GPCIHEERIIXWWIPFYHIEIVRDTVHYPVPXEGMPWIEELEIWIPVHXIPIIQNIPIPIPIEY
GSRXIEHPHEVGPECUSXTRUSRSESWRWEPHSYELSIYREIYVVVVYMPRUFSEEXSZPFSHM
              PGPSERISWMIEEWPPYEMVPVSSGQIEISUCYV
```

La distribución de frecuencias se encuentra en la tabla 2.23. El índice de coincidencias es 0.0845.

Letra	Frec.	%	Letra	Frec.	%
X	38	20.11	J	4	2.12
T	26	13.76	M	3	1.59
E	16	8.47	O	3	1.59
H	14	7.41	A	2	1.06
B	12	6.35	S	2	1.06
G	11	5.82	U	2	1.06
K	11	5.82	C	1	0.53
W	9	4.76	R	1	0.53
L	8	4.23	Y	1	0.53
V	8	4.23	Z	1	0.53
N	6	3.17	D	0	0.00
F	5	2.65	P	0	0.00
I	5	2.65	Q	0	0.00

Tabla 2.22: Distribución de frecuencias de la octava columna.

Letra	Frec.	%	Letra	Frec.	%
I	23	14.20	C	3	1.85
E	22	13.58	F	3	1.85
P	20	12.35	L	2	1.23
S	15	9.26	Q	2	1.23
V	12	7.41	T	2	1.23
R	9	5.56	D	1	0.62
Y	9	5.56	N	1	0.62
H	8	4.94	Z	1	0.62
W	8	4.94	A	0	0.00
G	6	3.70	B	0	0.00
X	6	3.70	J	0	0.00
M	5	3.09	K	0	0.00
U	4	2.47	O	0	0.00

Tabla 2.23: Distribución de frecuencias de la novena columna.

Opciones:

1. $e \rightarrow$ `I`, $a \rightarrow$ `E`. Desplazamiento de 4.

La novena y última letra de la clave sería entonces `E`. Aparentemente la palabra clave usada para cifrar el texto es: `DNQUIJOTE`.

Si desciframos usando esta palabra obtenemos:

todoseloc reyodonqu ijoteydij oqueelest abaallipr ontoparao bedecerle
yqueconcl uyeseconl amayorbre vedadquep udiesepor quesifues eotraveza
cometidoy seviesear madocabal leronopen sabadejar personavi vaenelcas
tilloexce ptoaquell asqueelle manoaseaq uienporsu respetode jariaadve
rtidoymed rosodeest oelcastel lanotrajo luegounli brodondea sentabala
pajayceba daquedaba alosarrie rosyconun cabodevel aqueletra iaunmucha
choyconla sdosyadic hasdoncel lassevino adondedon quijotees tabaalcua
lmandohin carderodi llasyleye ndoensuma nualcomoq uedeciaal gunadevot
aoracione nmitadoel aleyendaa lzolamand ydiolesob reelcuell ounbuengo
lpeytrase lconsumis maespadau ngentiles paldarazo siempremu rmurandoe
ntredient escomoque rezabahec hoestoman doaunadea quellasda masquelec
ineselaes padalacua llohizoco nmuchades envoltura ydiscreci onporquen
ofuemenes terpocapa ranoreven tarderisa acadapunt odelascer emoniaspe
rolasproe zasqueyah abianvist odelnovel caballero lestenial arisaaray
aalcenirl elaespada dijolabue nasenorad ioshagaav uestramer cedmuyven
turosocab alleroyle deventura enlidesdo nquijotel epregunto comosella
mabaporqu eelsupies edealliad elanteaqu ienquedab aobligado porlamerc
edrecibid aporquepe nsabadarl ealgunapa rtedelaho nraquelac anzasepor
elvalorde subrazoel larespond ioconmuch ahumildad quesellam abalatolo
sayqueera hijadeunr emendonna turaldeto ledoquevi viaalaste ndillasde
sanchobie nayayqued ondequier aqueellae stuviesel eserviria yletendri
aporsenor donquijot elereplic oqueporsu amorlehic iesemerce dquedeall
ineadelan tesepusie sedonysel lamasdeon atolosael laselopro metioylao
tralecalz olaespuel aconlacua llepasoca sielmismo coloquioq ueconlade
laespadap reguntole sunombreu dijoquese llamabala molineray queerahij
adeunhonr adomoline rodeanteq ueraalacu altambien rogodonqu ijoteques
epusiesed onysellam asedonamo lineraofr eciendole nuevosser viciosyme
rcedes

O bien, añadiendo puntuación:

> *Todo se lo creyó Don Quijote, y dijo que él estaba allí pronto para obedecerle, y que concluyese con la mayor brevedad que pudiese; porque si fuese otra vez acometido, y se viese armado caballero, no pensaba dejar persona viva en el castillo, excepto aquellas que él le mandase, a quien por su respeto dejaría.*

> *Advertido y medroso de esto el castellano, trajo luego un libro donde asentaba la paja y cebada que daba a los arrieros, y con un cabo de vela que le traía un muchacho, y con las dos ya dichas doncellas, se vino a donde Don Quijote estaba, al cual mandó hincar de rodillas, y leyendo en su manual como que decía alguna devota oración, en mitad de la leyenda alzó la mano, y dióle sobre el cuello un buen golpe, y tras él con su misma espada un gentil espaldarazo, siempre murmurando entre dientes como que rezaba. Hecho esto, mandó a una de aquellas damas que le ciñese la espada, la cual lo hizo con mucha desenvoltura y discreción, porque no fue menester poca para no reventar de risa a cada punto de las ceremonias; pero las proezas que ya habían visto del novel caballero les tenía la risa a raya. Al ceñirle la espada dijo la buena señora: Dios haga a vuestra merced muy venturoso caballero, y le dé ventura en lides. Don Quijote le preguntó como se llamaba, porque él supiese de allí adelante a quién quedaba obligado por la merced recibida, porque pensaba darle alguna parte de la honra que alcanzase por el valor de su brazo. Ella respondió con mucha humildad que se llamaba la Tolosa, y que era hija de un remendón, natural de Toledo, que vivía a las tendillas de Sancho Bienaya, y que donde quiera que ella estuviese le serviría y le tendría por señor. Don Quijote le replicó que por su amor le hiciese merced, que de allí en adelante se pusiese don, y se llamase doña Tolosa. Ella se lo prometió; y la otra le calzó la espuela, con la cual le pasó casi el mismo coloquio que con la de la espada. Preguntóle su nombre, y dijo que se llamaba la Molinera, y que era hija de un honrado molinero de Antequera; a la cual también rogó Don Quijote que se pusiese don, y se llamase doña Molinera, ofreciéndole nuevos servicios y mercedes.*

El fragmento de *El Ingenioso Hidalgo Don Quijote de la Mancha* de Miguel de Cervantes, en el que Don Quijote recibe el espaldarazo de quien él cree es un noble rey.

Hay que notar que, como ahora debe parecer claro, el índice de coincidencias de cada columna es similar al del texto en español, lo que evidencia que cada columna está cifrada monoalfabéticamente. Si al calcular el índice de coincidencias de cada columna, éste no es similar al del idioma supuesto para el texto sino que resulta menor, entonces lo más probable es que hayamos errado el tamaño estimado para la clave.

2.5 Criptoanálisis de Sistemas de Alberti

Recordemos que el llamado "sistema de Alberti" consiste en utilizar una tabla tipo Vigenère, con tres posibles generalizaciones:

1. El alfabeto del primer renglón puede ser un alfabeto mezclado, en vez del alfabeto en orden normal; es decir, el alfabeto que nos indica qué columna debemos utilizar puede ser un alfabeto mezclado. En este caso, decimos que se trata de un sistema con *secuencia de texto original mezclada* o con *columnas permutada*.

2. El alfabeto de la primer columna puede ser un alfabeto mezclado; es decir, el alfabeto que nos indica que renglón utilizar para cifrar una letra puede ser un alfabeto mezclado. En este caso, decimos que se trata de un sistema con *renglones permutados*.

3. El alfabeto que forma la parte central de la tabla, que es el que rotamos un lugar en cada renglón, puede ser un alfabeto mezclado en vez de un alfabeto normal. En este caso, decimos que se trata de un sistema con *secuencia de cifrado mezclada*.

Si conocemos la secuencia de texto original (es decir el orden en que las columnas corresponden a las letras del alfabeto original) y la secuencia de cifrado (es decir, el alfabeto que aparece en la tabla y que es "rotado" en cada renglón sucesivo), entonces podemos utilizar exactamente el mismo método que en el criptoanálisis de Vigenère para encontrar la solución. Simplemente buscamos cuál posición del alfabeto mezclado mejor corresponde con la distribución que tenemos, y proseguimos como antes.

Si conocemos la secuencia de texto original pero no la secuencia de cifrado, es decir, no sabemos cuál alfabeto fue usado en la tabla, entonces no podemos utilizar los métodos anteriores. En este caso, el método estándar de solución se conoce como *simetría de posiciones* o *simetría directa*. El método de simetría de posiciones depende de una palabra probable para empezar el análisis del criptograma. Se utiliza el hecho de que las columnas de la tabla se pueden reconstruir en su orden original a partir de la información que se vá obteniendo.

2.5.1 Simetría directa

Para explicar el método de simetría directa supongamos que tenemos un criptograma en el cual la secuencia de texto original es la estándar. Utilizando el índice de coincidencias, hemos determinado que la llave es de longitud seis. Nuestra información nos dice que el mensaje comienza con *"canta musa"*, y termina con la palabra *"bienaventurada."* Escribimos el mensaje en bloques de seis para facilitar la identificación de alfabetos iguales.

```
cantam usa
NPUANO LJYTQU RJWVNU HPJMSO ROYSQD JBZSSF OMIIFF HZODZW

RLZWIW LVOSQC EEZWNW CSSPDV IRAVFV NEBXGX WVODEW DPSADC
```

```
TSYWJX OYOWNO LAEAQY DSNILJ LAKADC WYZYGW AMIILZ JTSIKF

KYISNR RIZSQJ IPKANK LADLTV UBEHJC WNEIVR RTOYDF IPAVOT

                                                b ienave nturad a
DSBMQZ JTVCSD DSJIDC OSKXXW JJSIJC OJIXVN GAUIBF OUDMNR A.
```

Sabemos entonces, por ejemplo, que la N del primer alfabeto corresponde a la c, y podemos continuar esas substituciones a lo largo de todo el mensaje.

Armamos una tabla con esta información. Encabezando cada columna está la letra correspondiente; en cada renglón, el criptotexto que corresponde a ese alfabeto.

	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*
1	A		N						G					O							L					
2	P				A														J	V						
3	Y													U							D					
4	I																	M		A						
5	N																					B				
6		N		R	F							O														

Puesto que sabemos que cada renglón de la tabla de Alberti es una traslación de cualquier otro renglón, sabemos que la distancia entre letras en los renglones es contante. Por ejemplo, en el primer renglón, hay exactamente un hueco entre la A y la N. Eso quiere decir que en todos los renglones hay exactamente un espacio entre la A y la N. Por lo tanto, la N en el segundo renglón aparece en la columna de *g*, en el cuarto renglón en la columna de *v*, y en el último renglón en la columna de la *z*. Continuando de esta manera, podemos rellenar varios espacios de la tabla:

	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*
1	A		N		R	F		I	G					O	J	V					L		P	B	M	
2	P	B	M		A		N		R	F		I	G					O	J	V					L	
3	Y													U							D					
4	I	G						O	J	V				L		P	B	M		A		N		R	F	
5	N		R	F		I	G					O	J	V					L		P	B	M		A	
6		N		R	F		I	G					O	J	V					L		P	B	M		A

A su vez, esto permite llenar más nuestro mensaje:

```
cantam usa    es ia   a   m era    ob   e nc ade
NPUANO LJYTQU RJWVNU HPJMSO ROYSQD JBZSSF OMIIFF HZODZW

e   f  ut         a     p o hi ido c   g   t      a t
RLZWIW LVOSQC EEZWNW CSSPDV IRAVFV NEBXGX WVODEW DPSADC

  a m  n   am  e t      asn ue t       g  ac as  o  a e
TSYWJX OYOWNO LAEAQY DSNILJ LAKADC WYZYGW AMIILZ JTSIKF

    ad el   n ha ta  ue n o  b  m   g and e    e ha il
KYISNR RIZSQJ IPKANK LADLTV UBEHJC WNEIVR RTOYDF IPAVOT

       o         a   n      os am  ns  nb ienave nturad a
DSBMQZ JTVCSD DSJIDC OSKXXW JJSIJC OJIXVN GAUIBF OUDMNR A.
```

Ahora podemos usar esta información para obtener mas correspondencias del criptotexto. En los bloques 30 y 31, en el cuarto renglón, tenemos *g*ande*, lo cual sugiere que la `E` del tercer alfabeto de cifrado corresponde a la *r*; y en los bloques 12 y 13, tenemos *p*ohi*ido*, lo cual sugiere que la `D` del quinto alfabeto es una *r* también, y la `A` del cuarto una *b*. Con esta información, hacemos nuevamente uso de la simetría de posiciones para rellenar la tabla, y luego substituimos en el criptotexto.

	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*
1	A		N		R	F		I	G				U	O	J	V	E			D	L		P	B	M	Y
2	P	B	M	Y	A		N		R	F		I	G				U	O	J	V	E			D	L	
3	Y	A		N		R	F		I	G				U	O	J	V	E			D	L		P	B	M
4	I	G					U	O	J	V	E		D	L		P	B	M	Y	A		N		R	F	
5	N		R	F		I	G				U	O	J	V	E			D	L		P	B	M	Y	A	
6		N		R	F		I	G				U	O	J	V	E			D	L		P	B	M	Y	A

```
cantam usa  l es ial  apr m era  s ob   e nciade   om
NPUANO LJYTQU RJWVNU HPJMSO ROYSQD JBZSSF OMIIFF HZODZW

ey  f  uto    qu  a     pro hibido cuy g   tomo  ta tr
RLZWIW LVOSQC EEZWNW CSSPDV IRAVFV NEBXGX WVODEW DPSADC

  a m  ndo am uert y t dasn ue tr   d sg  acias  o  a e
TSYWJX OYOWNO LAEAQY DSNILJ LAKADC WYZYGW AMIILZ JTSIKF
```

```
 di ad el   n ha ta  ueun o mbr m   grand e osre habil
KYISNR RIZSQJ IPKANK LADLTV UBEHJC WNEIVR RTOYDF IPAVOT

t yr   o q  s t par  n      os am  nsi nb ienave nturad a
DSBMQZ JTVCSD DSJIDC OSKXXW JJSIJC OJIXVN GAUIBF OUDMNR A.
```

Continuando de esta manera, alternando entre análisis del texto y simetría de posiciones, obtenemos la tabla completa y el texto del mensaje.

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

```
cantam usacel estial aaprim erades obedie nciade lhombr
NPUANO LJYTQU RJWVNU HPJMSO ROYSQD JBZSSF OMIIFF HZODZW

eyelfr utodea quelar bolpro hibido cuyogu stomor taltra
RLZWIW LVOSQC EEZWNW CSSPDV IRAVFV NEBXGX WVODEW DPSADC

joalmu ndolam uertey todasn uestra sdesgr aciasc onlape
TSYWJX OYOWNO LAEAQY DSNILJ LAKADC WYZYGW AMIILZ JTSIKF

rdidad eleden hastaq ueunho mbrema sgrand enosre habili
KYISNR RIZSQJ IPKANK LADLTV UBEHJC WNEIVR RTOYDF IPAVOT

toyrec onquis topara nosotr oslama nsionb ienave nturad a
DSBMQZ JTVCSD DSJIDC OSKXXW JJSIJC OJIXVN GAUIBF OUDMNR A.
```

> *"Canta, Musa celestial, la primera desobediencia del hombre y el fruto de aquel árbol prohibido, cuyo gusto mortal trajo al mundo la muerte y todas nuestras desgracias con la pérdida del Edén, hasta que un Hombre más grande nos rehabilitó y reconquistó para nosotros la mansión bienaventurada."* *El Paraíso Perdido*, Libro Primero, de John Milton.

El método de simetría directa también se puede utilizar como método alternativo cuando conocemos la secuencia de cifrado (el alfabeto que va dentro de la tabla), pero no conocemos la secuencia de texto original. En ese caso, invertimos nuestra tabla, poniendo la secuencia de cifrado arriba, y en el interior de la tabla poniendo el equivalente de texto original, utilizando distintos renglones para distintos alfabetos. Cada renglón tiene entonces la secuencia de texto original en el orden en que aparece en la tabla.

2.5.2 Simetría indirecta y encadenamiento lineal

Cuando no conocemos la secuencia de texto original ni la secuencia de texto cifrado, no podemos recuperar la tabla de correspondencias con secuencias en el orden correcto. Sin embargo, ciertas relaciones en los intervalos entre letras son preservadas aún cuando las columnas de la tabla reconstruida no están en el orden correcto, y estas relaciones se pueden utilizar para recuperar la tabla completa.

Para ilustrar las relaciones entre los intervalos, supongamos que tenemos las siguientes dos tablas; la segunda se obtiene de la primera a base de reordenar las columnas para que la secuencia original sea la alfabética usual:

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

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

La observación importante es que al trabajar con la "tabla del criptoanalista" (la segunda tabla arriba), cualquier par de columnas y cualquier par de renglones, representan un intervalo de la matriz original. Por ejemplo, veamos las columnas de las letras *a* y *g* en la segunda tabla. Las letras `D` y `P` aparecen en el primer alfabeto (el primer renglón) en esas columnas, respectivamente. Si contamos la distancia entre `D` y `P` en la tabla original (la primera de arriba), notamos que el intervalo es de tamaño nueve. De manera similar, el intervalo entre las otras parejas en las dos columnas, `Q` y `A`, `U` y `F`, `M` y `Z`, también es nueve.

Para cualesquiera dos columnas que comparemos, el intervalo horizontal entre letras en cada renglón es constante. El intervalo no siempre es nueve, pues depende de las columnas que estemos comparando. Pero la distancia entre parejas en el mismo renglón en las mismas dos columnas siempre es constante.

Ahora comparemos las letras en el primer alfabeto de cifrado y el segundo en la segunda tabla. En la primer columna, las letras `D` y `Q` aparecen, y notamos que están a una distancia de diez en la secuencia original. Las letras `Q` y `X` aparecen en otra columna, y lo mismo con `C` y `P`, `E` y `R`. Todos estos pares están a una distancia de diez en la secuencia original. Lo mismo sucede con cualquier otro par elegido de la misma manera. La primer y segunda sucesión de cifrado en la segunda tabla están entonces separadas por un intervalo de diez espacios. Cada par de sucesiones (es decir, cada par de renglones) representa un intervalo distinto. Por ejemplo, el intervalo entre el primer y tercer renglón es trece, entre el primer y cuarto renglón es cinco, entre el segundo y tercero es cinco también, etc.

Hay varias maneras en que podemos usar esta relación entre intervalos para rellenar la tabla. Para ver cómo funciona la simetría indirecta, supongamos que tenemos un sistema polialfabético en el cuál hemos obtenido la siguiente información:

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	···
R				T				M		···
M				F						···
T						M				···

Si comparamos las columnas de *a* y de *e*, vemos que las letras `R` y `T`, y las letras `M` y `F` deben estar separadas por intervalos del mismo tamaño. No sabemos cuál es el tamaño (pues no conocemos la secuencia de cifrado original), pero sabemos que es el mismo en ambos casos.

El mismo intervalo aparece cuando comparamos el primer y el tercer renglón, donde `R` y `T` aparecen en la primer columna. Puesto que el intervalo es el mismo entre cualquier par de letras en la misma columna de los renglones uno y tres, y sabemos que `M` y `F` tienen el mismo intervalo que `R` y `T`, podemos agregar la letra `F` en la columna de *i*, tercer renglón, debajo de la `M` en el primer renglón.

Cada vez que establecemos una relación del intervalo entre un par de letras en un patrón rectangular como el de arriba, y tenemos tres de las cuatro letras, también en un patrón rectangular, en otro lugar de la tabla, podemos agregar la cuarta letra para completar el patrón. Las parejas deben tomarse en la misma dirección en ambos patrones, pues la distancia entre la primera y tercera sucesión de cifrado (es decir, el primer y tercer renglón) no es el mismo que entre la tercera y primera sucesión.

Con información en la tabla, podemos también usar otra técnica, llamada encadenamiento lineal, para continuar llenando el resto de la tabla.

Supongamos que hemos obtenido la siguiente información de un criptotexto que usó cinco alfabetos:

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

La idea es obtener listas que marcan todas las relaciones que resultan de comparar cada sucesión de cifrado con cada otra sucesión de cifrado. Por ejemplo, al comparar la primera con la segunda sucesión, notamos que la distancia de `E` a `C` es la misma que de `C` a `H`, de `H` a `K`, de `K` a `O`, etc. Es decir, la sucesión:

```
E  C  H  K  O
```

es una decimación (con intervalo desconocido) de la sucesión original. Si hacemos esto con la tabla de arriba, obtenemos las siguientes sucesiones (cada bloque representa un segmento de la decimación del alfabeto original; la distancia entre letras sucesivas en cada renglón es constante, pero puede variar de renglón a renglón):

```
1-2: ECHKOR  TWZM  FJN  IL  AS

1-3: EHOV  TZB  SIP  WM  KR  FN

1-4: FZP  WL  KE  SV  JM  OC  TI  AR

1-5: OSTFX  IZN  ER  WJ  KA  CV

2-3: CHKORV  WZMB  FJN  LP  AS

2-4: AOE  NMP  SRC  FWI  JZL

2-5: LZJX  CRS  MN  OA  WF

3-4: XFTSO  NZIVC  BP  ML  RE  JW
```

3-5: HRA BNX PZF KVS MJ IT

4-5: PN LJ EA VT CS IF ZX

Notemos que las cadenas producidas al comparar los renglones 1 y 2, y las obtenidas al comparar los renglones 2 y 3, tienen obviamente el mismo intervalo (en ambas tenemos el fragmento CHKOR, por ejemplo). Puesto que al comparar los renglones 1 y 2 obtuvimos el fragmento IL, y al comparar los renglones 2 y 3 el fragmento LP, los podemos "encadenar" para obtener un fragmento más largo, ILP; de manera similar obtenemos el fragmento TWZMB. Tenemos pues ahora los siguientes fragmentos de una decimación del alfabeto de cifrado usado:

```
E C H K O R V
  T W Z M B
    F J N
    I L P
     A S
```

El siguiente paso es relacionar estas cadenas con las otras. Si examinamos los intervalos y patrones de las letras en los fragmentos de otros renglones, podemos agregarlos usando las mismas reglas. Por ejemplo, si nos fijamos en los fragmentos obtenidos en 1-3, notamos que el fragmento EHOV representa una decimación, con intervalo dos, del primer fragmento listado arriba. Concluimos entonces que las distancias entre letras vecinas en el renglón 1-3 es el doble de la distancia que en el renglón 1-2 y en el renglón 2-3. Las podemos agregar, indicando un lugar vacío (cuyo contenido aún no conocemos) por un punto, y obtenemos

```
E C H K O R V
  T W Z M B
    F J N
  A S . I L P
```

Las cadenas del renglón 3-4 se obtienen de las de arriba contando cinco letras hacia atrás, como vemos con las letras V y C del fragmento NZIVC. Esto nos permite atar todos los fragmentos en una misma cadena larga. Al agregar todas las combinaciones, cada una con su propia regla, obtenemos el alfabeto casi completo:

```
E C H K O R V . A S . I L P T W Z M B F J N . . X .
```

Esta sucesión representa una decimación de la sucesión original. Como `V`, `W`, y `X` están a una distancia constante de nueve, podemos tomar una decimación de ésta de distancia 9 para producir la sucesión:

`V W X . Z . A M E S B C . F H I J . L N O P . R T .`

lo cual sugiere el alfabeto basado en la palabra `GAMES`:

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

Una vez conocido el alfabeto de cifrado, podemos regresar a nuestra tabla de substitución polialfabética, y reordenar las columnas usando este alfabeto en cada renglón, para obtener:

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

Las letras que no se usaron se pueden determinar regresando al mensaje original y descifrando. En este caso, el alfabeto original (que va arriba de la tabla) resulta ser obtenido mediante palabra clave mezclada, usando la llave `OLYMPIC`.

Nuevamente, el método de simetría indirecta requiere de conocer algunas palabras del mensaje, o de proceder a partir de una palabra probable.

2.5.3 Superposición de Kerckhoffs

Dos mensajes idénticos cifrados por dos claves distintas son llamados *isólogos*.

Cuando tenemos isólogos tales que los tamaños de las llaves son distintos, es posible recuperar la sucesión de cifrado original sin necesidad de utilizar palabra probable o de recuperar ninguna parte del texto original. Después podemos reducir los criptogramas a una substitución monoalfabétca que podemos resolver rápidamente.

Hay dos técnicas, dependiendo de si ambos fueron cifrados con la misma tabla de Vigenère o con distintas tablas, pero las ideas son esencialmente las mismas.

Por ejemplo, supongamos que recibimos los siguientes dos mensajes, de la misma fuente:

```
RTDQU XACPP QZCCK PPQPU VZNDA HLTYW YIRDD BLMIS RXRLS GPCIF NEWAL
CZNDZ UXBVL IQQMT KPVQV UQBLW BUXFZ ODPUA HCQEX MJWGL EMMAN KPDQG
LSMLI GVLTK SPBJA FNQSG PCXCN DZMAB VJTBM USVUC BMXBF NHCPL XNPVE
JUCGW WWYLC MQQHM XJZRV RRXIK NEIYT CKNTZ UXACP PPUAH PQRWP XHGPP
QQXHN MMAMK HGWDZ GLSRT VBKLT HBVZJ DIFEV BUZCK LQAMM HRQWI SQHRW
QAMMH CWEEJ XAZDM GSUSC WCLRA CKXDM MAJFT WIAFX XWHWA DHVNP MOAHV.

VXIOW SSVTT NQLXY QQTOC LANXE TUODH BWIQJ AMVUO MXVCO XTHEW ROLSN
XIOLS LHUOL SGYRO WILLZ AEAME VXBFM AINWC LVUOA BCBVA NPNXH XPXGQ
UCXXT SZTNH CWVPT FTWOX THMVR ABBOW RKUCN CFOUN OWFTQ TYYVT LSESH
PLCOS ZASBO EUUXB TEIDW DMHTX NTNRA UWTRH YCMIE WOXTH IWAAT SLXTT
NFTSF PPIND YUWXT QUCFC LXTTN FTSFP PINDY UWNOS IUPBG SHTXB GIYGW
CEWRX PHGKQ CMXKV YOLSE SHPLC OSZAS BOEUU XBTEI DWDMH TXNTN RAUWT.
```

Como tienen exactamente la misma longitud, y se recibieron en un periodo poco de tiempo, sospechamos que se tratan de isólogos.

Usando análisis de coincidencias hemos descubierto que el primer mensaje fue cifrado usando siete alfabetos distintos, y el segundo con seis alfabetos distintos (y suponemos que esas son las longitudes de las llaves respectivas).

Comparando los dos mensajes posición a posición, notamos que los dos mensajes tienen coincidencia (es decir, la misma letra en la misma posición) en las posiciones 23, 42, 65, 84, 107, 126, 149, 168, 191, 210, 233, 252, 275, 294, y 317. Estas se pueden dividir en dos grupos, uno empezando en 23 y el otro en 42, donde entre cada término sucesivo del grupo hay una distancia de 42; 42 es el mínimo común múltiplo de las longitudes de las llaves. El primer grupo de coincidencias ocurre cuando el primer mensaje está utilizando el alfabeto número dos de la llave; y el alfabeto número cinco de la segunda llave es el que está en uso en esas posiciones en el segundo mensaje. Esto nos indica que el segundo alfabeto de la primer llave, y el quinto alfabeto de la segunda llave, son el mismo alfabeto. De manera similar, el segundo grupo de coincidencias nos indica que el séptimo alfabeto de la primer llave y el sexto de la segunda son el mismo alfabeto.

También sabemos, por ejemplo, que la `R` del primer alfabeto de la primer clave corresponde a la misma letra en el mensaje original que la `V` del primer alfabeto de la segunda clave, puesto que estas letras son la primera que aparece en cada criptotexto. Ahora, armamos una tabla para organizar toda esta información. La tabla tiene once renglones, una para cada alfabeto distinto que se utilizó en el mensaje (siete en la primer clave, seis en la segunda, pero dos de ellos iguales a dos de la primer clave). El primer renglón corresponde al primer alfabeto de la primer llave; el segundo al segundo alfabeto de la primer llave, que

es igual al quinto de la segunda; el séptimo renglón corresponde al séptimo alfabeto de la primer llave, que es igual al sexto de la segunda, etc. Primero, en cada columna vamos poniendo la letra que aparece en esa posición del mensaje, en el renglón correspondiente. La tabla comienza de la siguiente manera (las letras itálicas provienen del primer criptotexto, las letras negritas del segundo; hacemos la distinción aquí por claridad, pero no hay necesidad de hacerla normalmente):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(1,1)	*R*							*C*							*K*
(2,1)(5,2)		*T*			W				*P*		N				
(3,1)			*D*							*P*					
(4,1)				*Q*							*Q*				
(5,1)					*U*							*Z*			
(6,1)						*X*							*C*		
(7,1)(6,2)						S	*A*					Q		*C*	
(1,2)	V						S						L		
(2,2)		X						V						X	
(3,2)			I						T						Y
(4,2)				O						T					

Notamos que en cada columna aparecen letras que corresponden a la misma letra en el mensaje original. Comparamos la columna 2 con la columna 14. Ambas tienen una X en el segundo alfabeto de la segunda llave. La columna dos nos dice que esa letra corresponde a lo mismo que T en el segundo alfabeto de la primer llave, y la columna catorce que es igual a C en el séptimo alfabeto de la primer llave. Podemos combinar las dos columnas para hacer una sola, que sería:

	···		···
(1,1)	···		···
(2,1)(5,2)	···	T	···
(3,1)	···		···
(4,1)	···		···
(5,1)	···		···
(6,1)	···		···
(7,1)(6,2)	···	C	···
(1,2)	···		···
(2,2)	···	X	···
(3,2)	···		···
(4,2)	···		···

Eso nos dice que `X` del segundo alfabeto de la segunda llave, `C` del séptimo alfabeto de la primer llave (que es igual al sexto de la segunda), y `T` del segundo alfabeto de la primer llave (que es igual al quinto alfabeto de la segunda llave) corresponden a la misma letra del mensaje original.

Podemos ahora continuar, consolidando columnas. Al final, obtenemos la siguiente tabla:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
$(1,1)$	G	Z	V	K	N	C	F	R	B	M						Y	H		X									P	
$(2,1)(5,2)$	T	L	N	W	B	P	Q	F	M	X	S	E	J			D		U	R					O	A				
$(3,1)$	Q	D	E	V	I	M	P	C		W			H	S	R			T					Y						
$(4,1)$	B	W	Q	G	Z	A	I	E	L	H	M		R	P											X		C		
$(5,1)$	P	R	S	U	A	K	M		J	V	Y	Z			O		Q					H				G			
$(6,1)$	A	U	M	C	X	D	L	Z	R	F	J	K			T														Q
$(7,1)(6,2)$	C	X	T	H	S	I	B	N	A	J	P	Q	D				F			W	V								
$(1,2)$	O	M	G	L	T	E	N	V	S		C		Y			K		R					B				I	
$(2,2)$	X	C	L	S	H	V	W	K		E	R	D								B		T		A		Q	Y		I
$(3,2)$	V	I	R	Y	F	T	U	H	Q	Z				O	A						L								
$(4,2)$	U	A	O	X	C	Q	T	G	P				K	N					D										

Cada columna de esta tabla representa la misma letra en el mensaje original, aunque varias columnas son homófonos de la misma letra (que no pudimos consolidar por falta de información).

Podemos ahora empezar a usar simetría indirecta y encadenamiento lineal en esta misma tabla. Dependiendo de cuales relaciones lineales utilicemos, obtenemos una decimación de la sucesión original del mensaje. Por ejemplo, un posible resultado es la sucesión:

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

Podemos simplemente utilizar esta decimación para obtener la tabla final, o notar que las posiciones de la `W`, `X`, `Y`, y `Z` sugieren que la sucesión original era:

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

Utilizando esta información, terminamos la tabla; una vez identificadas las columnas correspondientes, la nueva tabla tiene únicamente 21 columnas (el mensaje original sólo usa 21 letras distintas). El resultado es:

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

Cada columna representa un fragmento de la columna de Alberti original, en algún orden.

Podemos ahora asignar una letra aleatoria distinta a cada columna, por ejemplo, `A` a la primera, `B` a la segunda, etc, y los usamos para substituir el mensaje original por una substitución monoalfabética.

Por ejemplo, el principio del primer mensaje es:

```
R  T  D  Q  U    X  A  C  P  P    Q  Z  C  C  K
1  2  3  4  5    6  7  1  2  3    4  5  6  7  1
```

Los números indican cuál alfabeto debemos usar en cada letra.

Buscamos la `R` en el primer alfabeto del primer mensaje. Está en la columna 8, correspondiente a `H`, así que substituimos la `R` por `H`. La `T` en el segundo renglón corresponde a la columna `A`; la `D` en el tercero a `B`, y continuamos. La `A` en el séptimo corresponde a `I`, y luego tomamos la `C` y volvemos a empezar en el primer renglón. Haciendo las substituciones, obtenemos:

```
HABCD EIFFG CLDAD FGCAB OBCBF QGCND TGBFM IBFGC IBHBL DADFG CCBEG
ABCBE DEGCB ECPCC DFDCJ BLIBJ ADEPB UBLDA DFGCR GKNAB CKGAH DFBCM
GEJBE DJGCD KGAIA PEGLD ADBFC BEGAG CMPAG BECPH AGEGG CMPAG BEFDH
IBAAD JBKGA JGAJG EJBOD MBEFD CCGKN ADCPE DEIFF GLDAD SGNBA EDAFG
CPEDE IFFGL DADBE MGEHA DAFGC PEDEI FFGLD ADHAD BAFGC DHGJG CODHD
AFGCD FDCHI EIBNF DCBEF DHIBA ADJBK GAJGA JGEJB ODMBE FDCCG KNADC.
```

Esto representa una substitución monoalfabética del mensaje original, el cual puede ser

resuelto usando las técnicas desarrolladas para ello. El resultado final es el poema principal de *El Señor de los Anillos* de J.R.R. Tolkien:

> Tres anillos para los reyes elfos bajo el cielo
> Siete para los señores enanos en sus salas de piedra.
> Nueve para los hombres mortales, condenados a morir.
> Uno para el Señor Oscuro, en su trono oscuro.
>
> En la Tierra de Mordor, donde yacen las sombras.
> Un Anillo para gobernarlos, un Anillo para encontrarlos,
> Un Anillo para traerlos a todos y atarlos a las tinieblas.
> En la Tierra de Mordor, donde yacen las sombras.

El método de Superposición de Kerckhoffs se puede usar siempre que tengamos dos mensajes idénticos cifrados con llaves de distinta longitud, aun si las llaves no tienen ninguna letra en común. En ese caso, tendremos más renglones.

El único problema surge si las dos llaves tienen la misma longitud, pues en ese caso no habrá manera de identificar columnas, a menos que tengamos suficiente información para decir si algún alfabeto de una llave es el mismo a un alfabeto de la otra llave, en otra posición.

El método de Superposición de Kerckhoffs se puede usar también *aún si se usaron tablas de Alberti distintas*. En ese caso, al momento de hacer simetría de posiciones y encadenado lineal, hay que mantener la información obtenida de las distintas copias separadas, pues provienen de tablas distintas.

3.1 Sistema poligráfico de Playfair

En los capítulos anteriores vimos como el análisis estadístico de la frecuencia de aparición de cada letra contribuye al criptoanálisis de textos cifrados monoalfabéticamente y polialfabéticamente. Una posibilidad para dificultar el criptoanálisis, echando a perder por completo esta información estadística, es cambiar la unidad fundamental de substitución criptográfica, que hasta ahora ha sido la letra. Podríamos usar, por ejemplo, conjuntos de letras. En ésto se basan los sistemas poligráficos. Ya no reemplazaremos letras individuales por otras, sino que ahora tomaremos pequeñas secuencias de letras y las reemplazaremos.

El nombre proviene de *grafo*, que significa letra; tanto las substituciones monoalfabéticas como las polialfabéticas son sistemas monográficos, pues actúan sobre letras individuales. Un sistema *bigráfico* o *digráfico* actúa sobre pares de letras; *trigráfico* sobre tripletes de letras; *n-gráfico* sobre bloques de n letras. En general, un sistema es poligráfico si actúa

sobre bloques de n letras, con $n > 1$. Más formalmente, siguiendo a [Sin66] decimos que un sistema poligráfico es aquél que reemplaza cada conjunto de n letras del texto original por un conjunto de n letras en el criptotexto, con $n > 1$.

Desde el punto de vista matemático, se trata de una substitución monoalfabética en la cuál el "alfabeto" original y el "alfabeto" de cifrado son la colección de todos los posibles bloques de n-letras.

El primer sistema tipo poligráfico del que tenemos noticia es el de Giovanni Baptista Porta (1563) [Kah99]. El sistema de Porta estaba basado en una tabla que, como la de Vigenère, tenía indexados los renglones y las columnas por las letras del alfabeto. Las entradas de la tabla eran un conjunto de símbolos extraños. Dado un par de letras consecutivas del texto claro, la primera se utiliza como el índice del renglón r y la segunda como el índice de la columna c, luego se busca en la tabla el símbolo en la entrada (r, c) y este reemplaza al par de letras que le dieron lugar. Esto constituye el germen de un sistema poligráfico, pero no exactamente uno, pues cada pareja de letras es sustituído por un símbolo, y no por otro bloque de letras.

El primer sistema poligráfico verdadero que conocemos es el de *Playfair*. Recibe su nombre de el noble inglés Lyon Playfair, quien *no* lo inventó alrededor de 1854 [Kah99]. Quien realmente inventó el sistema fue Charles Wheatstone, que también inventó su propio telégrafo poco antes del de Morse y estudiaba óptica, electromagnetismo, y acústica entre otras cosas. Wheatstone le presentó su sistema a su buen amigo Playfair, con quien se entretenía descifrando los mensajes de enamorados secretos que aparecían en el London Times. Playfair presentó el sistema de Wheatstone en una cena de gala a la que asistió toda la alta sociedad inglesa, incluyendo a la reina Victoria y a muy diversos funcionarios. Desde entonces el sistema se conoce por el nombre de su padrino Playfair y no por el de su verdadero inventor. Por cierto Playfair nunca tuvo intención de plagiar el sistema de Wheatstone, que se convirtió después en el cifrado de trinchera del ejército inglés y fue utilizado en la Guerra de los Boers.

El sistema de Playfair reemplaza digramas (pares de letras consecutivas). Intuitivamente esto dificulta el criptoanálisis porque:

- Se pierden las frecuencias de las letras individuales.
- Se reduce el número de elementos disponibles para su análisis a la mitad. En un texto de 100 letras sólo hay 50 digramas. Esto dificulta el uso de la estadística.
- Hay sólo 26 letras, pero hay $26 \times 26 = 676$ digramas. La letra *e* constituye aproximadamente, el 13% del total de letras en un texto en español, pero el digrama más frecuente, *en*, constituye sólo el 2.5% del total de digramas. La complejidad absoluta

del sistema es mucho mayor. Para poder distinguir la frecuencia verdadera del ruido, se requiere mucho más texto.

En el sistema de Playfair el alfabeto se acomoda en un arreglo de 5×5 letras, llamado un *cuadrado de Playfair*. Como el alfabeto tiene 26 y no 25 letras, entonces debemos eliminar una de ellas. En inglés se elige generalmente eliminar la j [Sin66, Lew00], una letra poco usual, fusionándola con la i; es decir, siempre que en un digrama aparezca la letra j será considerada como i. En nuestro caso resulta más conveniente eliminar la w, que después de todo no es una letra propiamente española. Por ejemplo, un cuadrado de Playfair sería el siguiente:

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

Supongamos que en el texto claro nos encontramos con el digrama ordenado p_1p_2. Éste será substituido en el criptograma por el digrama ordenado c_1c_2, que se obtiene mediante el siguiente algoritmo [Sin66, Lew00]:

1. Se localizan p_1 y p_2 en el cuadrado.
2. *Regla del cuadrado:* Si p_1 y p_2 no coinciden ni en renglón ni en columna entonces definen un rectángulo dentro de la matriz, del cual tanto p_1 como p_2 son dos de las esquinas; las otras dos esquinas son c_1, aquella en el mismo renglón que p_1, y c_2, aquella en el mismo renglón que p_2. Este caso está ilustrado en la Figura 3.1.A.
3. *Regla del renglón:* Si p_1 y p_2 están en el mismo renglón, entonces c_1 es la letra que está a la derecha de p_1 y c_2 la que está a la derecha de p_2. Si p_1 ó p_2 están en el borde derecho de la matriz entonces se considera que la letra de la derecha es aquella en el extremo izquierdo y sobre el mismo renglón de la matriz. Este caso está ilustrado en la Figura 3.1.B.
4. *Regla de la columna:* Si p_1 y p_2 están en la misma columna, entonces c_1 es la letra que está abajo de p_1 y c_2 la que está abajo de p_2. Si p_1 ó p_2 están en el borde inferior de la matriz entonces se considera que la letra de abajo es aquella en el borde superior y sobre la misma columna de la matriz. Este caso está ilustrado en la figura 3.1.C.
5. Si p_1 y p_2 coinciden en renglón y columna (es decir, son la misma letra), se inserta una letra arbitraria entre ellas, usualmente x, para romper el digrama. Luego se procede

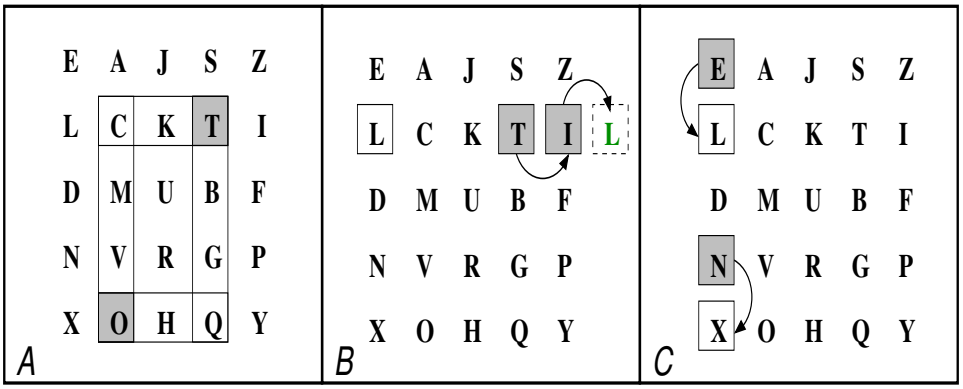

Figura 3.1: Substitución de digramas en el método de Playfair. En A se ilustra el caso en el que las letras no coinciden en renglón ni columna; en este caso el digrama que reemplaza a *to* es `CQ`. En B se ilustra el caso en el que las letras coinciden en renglón; en este caso el digrama *it* se reemplaza por `LI`. En C se ilustra el caso en el que las letras coinciden en columna; en este caso el digrama *en* se reemplaza por `LX`.

a cifrar por separado los digramas p_1, x y p_2 con la letra que siga. Otra alternativa es substituir p_2 por una letra arbitraria.

El último caso es similar a lo que se hace si el número de letras en el texto claro no es par. Se añade una letra comodín para formar el último digrama.

Podemos formular con mayor precisión el esquema de cifrado si utilizamos notación de matrices. Sean $M_{i,j}$ y $M_{k,m}$ las posiciones en la matriz M de la primera y segunda letra del digrama de texto claro respectivamente $i, j, k, m \in \{0, \ldots, 4\}$. Las reglas 1, 2, y 3 formuladas arriba corresponden a:

1. Si $i \neq k$ y $j \neq m$ entonces el digrama que reemplaza es el constituido por las letras en $M_{i,m}$ y $M_{k,j}$, en ese orden.
2. Si $i = k$ y $j \neq m$ entonces el digrama que reemplaza es el constituido por las letras en $M_{i,(m+1)}$ y $M_{i,(j+1)}$, en ese orden, donde los valores $m + 1$ y $j + 1$ son tomados módulo 5.
3. Si $i \neq k$ y $j = m$ entonces el digrama que reemplaza es el constituido por las letras en $M_{(i+1),j}$ y $M_{(k+1),j}$, en ese orden, donde los valores $i + 1$ y $k + 1$ son nuevamente tomados módulo 5.

En el esquema original de Wheatstone si las dos letras del digrama eran iguales y su

posición en la matriz era $M_{i,j}$ entonces se reemplazaban por el digrama en el que ambas letras son la de la posición $M_{R-i-1,C-j-1}$ donde R es el número de renglones en la matriz y C es el número de columnas. De hecho las matrices de Wheatstone podían tener el último renglón más corto que los anteriores, y no ser cuadradas [Kah99].

Para descifrar hay que aplicar el proceso a la inversa. Sean $M_{i,j}$ y $M_{k,m}$ las posiciones en la matriz M de la primera y segunda letra del digrama del criptotexto, respectivamente, con $i, j, k, m \in \{0, \ldots, 4\}$. Para descifrarlo:

1. Si $i \neq k$ y $j \neq m$ entonces el digrama que reemplaza es el constituido por las letras en $M_{i,m}$ y $M_{k,j}$, en ese orden.
2. Si $i = k$ y $j \neq m$ entonces el digrama que reemplaza es el constituido por las letras en $M_{i,(m-1)}$ y $M_{i,(j-1)}$, en ese orden, con los valores $m-1$ y $j-1$ tomados módulo 5.
3. Si $i \neq k$ y $j = m$ entonces el digrama que reemplaza es el constituido por las letras en $M_{(i-1),j}$ y $M_{(k-1),j}$, en ese orden, con los valores $i-1$ y $k-1$ tomados nuevamente módulo 5.
4. No puede ocurrir que ambas letras del digrama sean iguales. Pero habrá que leer el resultado final para eliminar las letras espurias que se añadieron para romper digramas de la misma letra.

Hay una formula mnemotécnica usada por el ejército [otA90], para utilizar el algoritmo de Playfair en los casos en que las letras coinciden en renglón o columnas: CDDI y CDDA; esto es: *Cifrar Derecha, Descifrar Izquierda* y *Cifrar Debajo, Descifrar Arriba*[1].

Si nuestro texto a cifrar fuera:

[...] *mitos. Los mitos se generan* [...]

procedemos a dividirlo en digramas:

mi to sl os mi to ss eg en er an

Ahora nos percatamos que tenemos el digrama *ss* y tenemos que romperlo añadiendo una *x* entre las *s*. Esto va a dejar un número impar de letras, así que al final debemos también añadir una *x*. La secuencia resultante de digramas es:

[1]El mnemotécnico original en inglés es: ERDL y EBDA, es decir: *Encipher Right, Decipher Left* y *Encipher Below, Decipher Above.*

mi to sl os mi to sx se ge ne ra nx

Una de las ventajas del segundo método de eliminación de digrámas repetidos, sustituír la segunda letra por una x, es que se puede hacer "en línea" al encontrar un digrama que consiste de una letra repetida, y no hay necesidad de modificar los digramas siguientes.

El cifrado, usando nuestra matriz de arriba, queda como sigue:

claro	*mi*	*to*	*sl*	*os*	*mi*	*to*	*sx*	*se*	*ge*	*ne*	*ra*	*nx*
`cifrado`	`FC`	`CQ`	`ET`	`QA`	`FC`	`CQ`	`EQ`	`ZA`	`NS`	`XL`	`VJ`	`XE`

Hay que notar que la secuencia de criptotexto `FCCQE` corresponde a la secuencia de texto claro *mitos*. Pero ahora este tipo de repeticiones no ocurrirán a menos que ambas apariciones ocurran en posiciones pares, o ambas en impares[2]. Por ejemplo la secuencia *os* de *los* y de *mitos* no genera repeticiones en el criptograma porque una aparición ocurre en posición par y la otra en una impar.

Lo que sí ocurre siempre en el cifrado de Playfair es que si el digrama p_1p_2 se cifra como c_1c_2, entonces el digrama (p_2, p_1) se cifra como (c_2, c_1). Ésto puede servir de elemento para el criptoanálisis. Esta situación desaparece en el *método de los cuatro cuadrados*, que veremos en la siguiente sección.

3.2 Sistema de cuatro cuadrados

En el método de los cuatro cuadrados se coloca el alfabeto completo (salvo nuestra w), en el orden usual, en un cuadrado de 5×5. Este cuadrado se coloca dos veces en el primer y cuarto cuadrante (es decir, esquinas opuestas) de un cuadrado de 10×10, capaz de contener cuatro cuadrados menores de 5×5. En los otros dos cuadrantes se ponen sendos alfabetos mezclados diferentes. Por ejemplo:

[2] De hecho, la repetición de la letra `E` al final en este caso fue accidental; se debe a que la `X` y la `E` están en la misma columna.

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

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

Para cifrar el digrama p_1p_2 en el texto claro, se busca p_1 en el primer cuadrante (superior izquierdo, alfabeto en orden usual) y se busca p_2 en el cuarto cuadrante (inferior derecho, alfabeto en orden usual). Supongamos que hemos encontrado a p_1 en la entrada $M_{i,j}$ de la matriz y a p_2 en la $M_{k,m}$, con $i, j \in \{0, \ldots, 4\}$ y $k, m \in \{5, \ldots 9\}$. Ahora se aplican las mismas reglas que conocemos, y el cifrado es el digrama ordenado c_1c_2, donde $c_1 = M_{i,m}$, y $c_2 = M_{k,j}$. Nótese que las letras del texto claro se buscan el los cuadrantes 1 y 4 y las del criptograma en los cuadrantes 2 y 3 (superior derecho e inferior izquierdo). En este esquema al cifrar los reversos de digramas claros no necesariamente se obtiene lo mismo que al revertir el cifrado del digrama claro.

Hagamos un ejemplo. Supongamos que deseamos cifrar la frase:

[...] *se está cuerdo, me aventuro a decir,*

Usando los cuatro cuadrados mostrados previamente obtenemos:

claro	*se*	*es*	*ta*	*cu*	*er*	*do*	*me*	*av*	*en*	*tu*	*ro*	*ad*	*ec*	*ir*
cifrado	SV	UP	OA	MX	GP	EI	NR	IQ	UJ	OZ	SH	UC	GA	BO

Como podemos notar, el criptotexto que corresponde a *se* es SV, pero el que corresponde a *es* no es VS, sino UP.

Una variación sencilla se obtiene si en vez de usar el alfabeto en su orden usual en los cuadrantes 1 y 4, se utiliza un alfabeto mezclado arbitario en cada cuadrante.

3.3 Sistema de dos cuadrados

Los sistemas de dos cuadrados son una simplifación del sistema de cuatro cuadrados. Hay dos versiones: vertical y horizontal.

Sistema vertical de dos cuadrados

Este sistema usa dos matrices de 5×5, uno encima del otro. Normalmente ambos cuadrados tienen un alfabeto mezclado. Por ejemplo, véase la Figura 3.2

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

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

Figura 3.2: Un ejemplo de dos cuadrados verticales. El cuadrado de arriba se obtuvo con la palabra clave `CALABOZOS`, y el de abajo con `DRAGONES`.

Para cifrar, usamos la regla del cuadrado, igual que en el caso de los cuatro cuadrados, cuando se pueda. La primer letra del digrama se busca en el cuadrado superior, y la segunda en el inferior. Si las dos letras que se busca cifrar están en la misma columna, el digrama es "cifrado" consigo mismo. Nótese que la primer letra de un digrama de criptotexto siempre proviene del cuadrado de arriba, y la segunda del cuadrado de abajo.

Por ejemplo, para cifrar *la luna brilla* con los dos uadrados de la Figura 3.2, tenemos:

Texto claro	*la*	*lu*	*na*	*br*	*il*	*la*
Criptotexto	LA	CX	PR	AG	GP	LA

y el digrama *la* se transforma en sí mismo.

El caso en que las letras del texto claro no son modificadas en el criptotexto se llama una *transparencia*. Una debilidad del sistema de dos cuadrados es que a la larga, alrededor de

el 20% de las digráficas del criptograma serán transparencias. Esto facilita la identificación y el desciframiento.

Para descifrar, buscamos la primer letra en el cuadrado de arriba y la segunda en el de abajo. Si están en la misma columna, las dejamos como estaban. Si están en columnas distintas, usamos la regla del cuadrado para descifrar.

Sistema horizontal de dos cuadrados

En este caso colocamos los dos cuadrados de 5×5 uno al lado del otro en posición horizontal. Para cifrar, buscamos la primer letra en el cuadrado de la izquierda, y la segunda en el de la derecha. Si las letras están en renglones distintos, aplicamos la regla del cuadrado. Si están en el mismo renglón, invertimos su orden antes de ponerla. Nótese que la primer letra de un digrama cifrado es obtenida del cuadrado de la derecha, y la segunda del cuadrado de la izquierda; es por esto que dos letras en la misma columna deben ser invertidas al cifrar.

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

Figura 3.3: Un ejemplo de dos cuadrados horizontales, obtenidos con los mismos cuadrados base que el caso de dos cuadrados verticales.

Por ejemplo, nuevamente ciframos *la luna brilla*, usando los dos cuadrados horizontales en la Figura 3.3:

Texto claro	*la*	*lu*	*na*	*br*	*il*	*la*
Criptotexto	AL	DV	PA	RB	FP	AL

Para descifrar, se busca la primera letra en el cuadrado de la derecha, y la segunda en el de la izquierda. Si están en renglones distintos, se aplica la regla del cuadrado. Si están en el mismo renglón, se invierte la digráfica.

3.4 Un poco de teoría de números

Antes de pasar al siguiente método de cifrado necesitamos algunos conocimientos de teoría de números.

Hemos estado trabajando todo el tiempo con las letras del alfabeto, en nuestro alfabeto convencional tenemos 26 letras que podemos numerar, como hemos hecho ya en repetidas ocasiones, del 0 al 25 en orden lexicográfico creciente. A partir de esta numeración podemos trabajar ya no con las letras mismas, sino con los números que las representan y hasta podemos pensar en que nuestras transformaciones operan aritméticamente sobre los números en vez de sobre las letras. Cambiamos nuestros problemas de transformar letras a el problema de transformar números.

Pero ahora tenemos que pensar en que a partir de un mensaje cualquiera, hecho de letras (números del 0 al 25), debemos obtener otro mensaje hecho de letras que cifre al primero. Esto restringe nuestras operaciones: debemos obtener números del 0 al 25 a partir de números del 0 al 25. Nuestras operaciones tendrán lugar en $\mathbb{Z}_{26}$, el conjunto de enteros módulo 26. También es usual denotar a los enteros módulo k como $\mathbb{Z}/k\mathbb{Z}$. Las sumas, las restas, y los productos de dos elementos cualesquiera de $\mathbb{Z}_{26}$ darán como resultado un elemento de $\mathbb{Z}_{26}$.

Hay que notar que toda resta en $\mathbb{Z}_{26}$ puede escribirse como una suma. Por ejemplo $(8-12) \pmod{26} = (8+14) \pmod{26} = 22$. En general $(a-k) \pmod{m} = (a+(m-k)) \pmod{m}$.

Con la multiplicación y la división la situación es diferente. Hay elementos de $\mathbb{Z}_{26}$ que no tienen inverso multiplicativo. Por ejemplo no existe ningún elemento de $x \in \mathbb{Z}_{26}$ tal que $2\,x = 1 \pmod{26}$, es decir: $2x \equiv 1 \pmod{26}$. Si hubiera, por definición x es tal que:

$$26|(2\,x - 1),$$

es decir:

$$(2 \times 13)|(2\,x - 1).$$

Pero un número par no puede dividir a un impar.

Para establecer exactamente cuáles residuos tienen inverso multiplicativo, necesitaremos algunos teoremas elementales de Teoría de Números.

Ya sabemos que a $\equiv b \pmod{m}$ significa que $m|(a - b)$. Recordemos también el siguiente hecho evidente:

Teorema 3.1 *Sean a, b y c en $\mathbb{Z}$. Si $a|b$ y $b|c$ entonces $a|c$.*

```
MCD(a, b)
1  if b = 0 then
2      return a
3  else
4      return MCD(b, a (mod b))
5  endif
```

Figura 3.4: Versión recursiva del algoritmo de Euclides para obtener el máximo común divisor de dos números a y b.

llamada	**a**	**b**	**cociente**
0	97	77	1
1	77	20	3
2	20	17	1
3	17	3	5
4	3	2	1
5	2	1	2
6	1	0	

Tabla 3.1: Valores de a y b, argumentos del algoritmo de Euclides en las llamadas recursivas sucesivas para encontrar mcd$(97, 77)$.

Dem.: Como $a|b$ entonces es posible escribir $b = ax$ para algún $x \in \mathbb{Z}$. Además $b|c$ entonces $c = by$ para algún $y \in \mathbb{Z}$. Sea $z = xy \in \mathbb{Z}$. Entonces $c = axy = az$ por lo que $a|c$. □

Recordemos ahora el algoritmo de Euclides para encontrar el máximo común divisor de dos números. El algoritmo se estudia en los cursos elementales de álgebra, y es mostrado en la Figura 3.4 en su versión recursiva.

Observemos un ejemplo de ejecución de este algoritmo. En la tabla 3.1 se muestran los valores de a y b en las llamadas recursivas que se efectúan para encontrar el mcd de 97 y 77, que por cierto son primos relativos y por tanto mcd$(97, 77) = 1$.

Por la manera en que se efectúa la llamada en la línea 4 del algoritmo (fig. 3.4) vemos que en la tabla 3.1 tenemos en la llamada i como valor de a (llamémosle a_i en adelante) el valor que tenía b en la llamada anterior (llamémosle b_{i-1}). Por otra parte el argumento que jugará el papel de b en la llamada que ocurre en la línea 4 es a mod b, el residuo que resulta de dividir a por b. Así que podemos escribir $a_i = q_i\, b_i + b_{i+1}$ con $0 \leq b_{i+1} < b_i$; es decir el siguiente valor de b será el residuo de dividir la a actual entre la b actual. Pero el valor

actual de a es en realidad el que tenía b en la llamada anterior como habíamos observado, así que:

$$b_{i-1} = q_i\, b_i + b_{i+1} \tag{3.4.1}$$

En esencia la ecuación (3.4.1) dice que cualquier residuo (los valores de b) resultado del proceso puede escribirse como combinación lineal de sus dos predecesores. Sólo tenemos que reescribir la ecuación como $b_{i+1} = -q_i\, b_i + b_{i-1}$. Pero si esto es posible entonces podemos irnos hacia atrás escribiendo cada residuo de la expresión como combinación lineal de sus predecesores hasta llegar a a y b. Como el último residuo es $\mathrm{mcd}(a, b)$ esto quiere decir que el máximo común divisor de dos números puede escribirse como combinación lineal de ellos. En particular si los dos números a y b son primos relativos entonces existen $\alpha, \beta \in \mathbb{Z}$ tales que:

$$1 = \alpha\, a + \beta\, b \tag{3.4.2}$$

Esto nos da el siguiente Teorema:

Teorema 3.2 *Si $d = \mathrm{mcd}(a, b)$ entonces el algoritmo de Euclides (extendido) garantiza que existen $\alpha, \beta \in \mathbb{Z}$ tales que:*

$$d = \alpha\, a + \beta\, b$$

Ahora bien, si estamos en $\mathbb{Z}_b$ ¿quién es α? la expresión (3.4.2) dice que $\alpha\, a \equiv 1 \pmod{b}$ así que α resulta ser el inverso multiplicativo de a en $\mathbb{Z}_b$.

Al algoritmo de Euclides modificado para obtener la combinación lineal que expresa el mcd se le suele llamar el *algoritmo de Euclides extendido,* y en particular es útil para calcular inversos multiplicativos como hemos visto. Un tratamiento bonito de esto puede hallarse en [BW00].

En nuestro ejemplo de $\mathrm{mcd}(97, 77)$ tenemos lo siguiente:

$$\begin{aligned}
1 &= 3 - 2 \\
&= 3 - (17 - 5(3)) = 6(3) - 17 \\
&= 6(20 - 17) - 17 = 6(20) - 7(17) \\
&= 6(20) - 7(77 - 3(20)) \\
&= 27(20) - 7(77) \\
&= 27(97 - 77) - 7(77) \\
&= 27(97) - 34(77).
\end{aligned}$$

Como $-34 \equiv 63 \pmod{97}$ $(97 - 34 = 63)$, concluimos que el inverso de 77módulo 97 es 63.

Tomando esto en consideración pensemos en el inverso multiplicativo de un elemento en un conjunto $\mathbb{Z}_m$. Encontrar el inverso de un número $a \in \mathbb{Z}_m$ consiste en hallar $x \in \mathbb{Z}_m$ tal que $ax = 1$ tomando el producto módulo m. Es decir:

$$\mathrm{a}x \equiv 1 \pmod{m} \tag{3.4.3}$$

Esta ecuación tiene solución, y es única, si y sólo si a y m son primos relativos, es decir $\mathrm{mcd}(a, m) = 1$.

Lema 3.1 *Si $r|a$ y $r|b$, entonces $r|\alpha\, a + \beta\, b$ para todo $\alpha, \beta \in \mathbb{Z}$.*

Dem.: Existe x tal que $rx = a$, y existe y tal que $ry = b$. Por lo tanto,

$$\alpha\, a + \beta, b = \alpha\, rx + \beta\, ry = (\alpha\, x + \beta\, y)r,$$

de manera que $r|\alpha\, a + \beta\, b$, como aseguramos. □

Teorema 3.3 *Un elemento a de $\mathbb{Z}_m$ posee inverso multiplicativo módulo m si y sólo si* $\mathrm{mcd}(a, m) = 1$.

Dem.: Supongamos primero que $b \in \mathbb{Z}$ es tal que $ab \equiv 1 \pmod{m}$. Entonces $m|(ab - 1)$, es decir, existe una $\beta \in \mathbb{Z}$ tal que $\beta\, m = ab - 1$. Por lo tanto, $ab - \beta\, m = 1$, de manera que el máximo común divisor de a y m debe dividir al 1; la única manera es si $\mathrm{mcd}(a, m) = 1$.

Conversamente, supongamos que $a \in \mathbb{Z}_m$, y que $\mathrm{mcd}(a, m) = 1$. Entonces existen $\alpha, \beta \in \mathbb{Z}$ tales que

$$\alpha\, a + \beta\, m = 1.$$

Entonces $-\beta\, m = \alpha\, a - 1$, es decir, $m|\alpha\, a - 1$. Por lo tanto, $\alpha\, a \equiv 1 \pmod{m}$, de manera que α es un inverso multiplicativo módulo m de a. □

Los números de $\mathbb{Z}_{26}$ que tienen inverso son entonces los siguientes:

número	1	3	5	7	9	11	15	17	19	21	23	25
inverso	1	9	21	15	3	19	7	23	11	5	17	25

que son justamente aquellos que son primos relativos con 26.

3.5 Sistema de Hill

En 1929 en *The American Mathematical Monthly* apareció un artículo del profesor Lester S. Hill, de Hunter College de Nueva York [Hil29]. En el artículo Hill describe un sistema criptográfico basado en manipulaciones algebraicas del alfabeto, aplicándole transformaciones lineales. En 1931 amplió el trabajo anterior [Hil31].

Si las letras corresponden a enteros positivos del 0 al 25 y todas nuestras operaciones son en $\mathbb{Z}_{26}$, entonces podemos ver un digrama como una pareja ordenada de números (p_1, p_2), es decir, un vector en $\mathbb{Z}_{26}^2$. Por ejemplo, el digrama *af* corresponde al vector $(0, 5)$.

Si ahora aplicamos una transformación lineal a este vector, multiplicándolo por una matriz módulo 26, obtenemos un nuevo vector (c_1, c_2) en $\mathbb{Z}_{26}^2$ que corresponde a un digrama. Es decir, podemos cifrar un digrama p_1p_2 multiplicando por una matriz M de 2×2:

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \pmod{26} \tag{3.5.4}$$

Por ejemplo, para cifrar el texto: *con diez cañones* usando la matriz de cifrado:

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$$

procedemos a dividir el texto en digramas y escribir los vectores correspondientes a cada digrama.

Aplicamos la transformación a cada vector; por ejemplo, para el primer digrama (*co*) tenemos:

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 2 \\ 14 \end{pmatrix} = \begin{pmatrix} 22 \\ 4 \end{pmatrix} \pmod{26}.$$

La letra que corresponde al 22 es la `W`, y la que corresponde al 4 es la `E`, así que el digrama *co* se transforma en `WE`. El resultado de cifrar todo el texto se muestra en la Tabla 3.2.

claro	co	nd	ie	zc	an	on	es
vec. claro	$(2, 14)$	$(13, 3)$	$(8, 4)$	$(25, 2)$	$(0, 13)$	$(14, 13)$	$(4, 18)$
vec cripto	$(22, 4)$	$(25, 8)$	$(10, 16)$	$(25, 9)$	$(0, 13)$	$(22, 5)$	$(4, 16)$
criptograma	`WE`	`ZI`	`KQ`	`ZJ`	`AN`	`WF`	`EQ`

Tabla 3.2: Cifrado de Hill del ejemplo.

¿Cómo desciframos? Dada la matriz del cifrado:

$$M = \begin{pmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{pmatrix},$$

necesitamos otra matriz:

$$M' = \begin{pmatrix} m'_{1,1} & m'_{1,2} \\ m'_{2,1} & m'_{2,2} \end{pmatrix}$$

tal que si (p_1, p_2) es el vector asociado a un digrama de texto claro y

$$M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \pmod{26},$$

entonces:

$$M' \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \pmod{26}.$$

Es decir:

$$M'M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \pmod{26}.$$

Esto debe ser válido para *todo* vector (p_1, p_2). Así que, si denotamos con I_2 a la matriz identidad de 2×2, necesitamos que $M'\,M = I_2 \pmod{26}$. Es decir, M' es la inversa de M módulo 26. Bueno, esto no parece muy complicado pero sí lo es. La inversa puede ni siquiera existir. De existir, como podemos ver en cualquier texto de álgebra lineal:

$$M^{-1} = \frac{1}{|M|} \begin{pmatrix} m_{2,2} & -m_{1,2} \\ -m_{2,1} & m_{1,1} \end{pmatrix} \pmod{26}.$$

He ahí el problema, $1/|M| \pmod{26}$ es el inverso multiplicativo del determinante de M módulo 26. Pero el inverso de un número módulo 26 es algo que, según vimos en la sección anterior, no siempre existe.

Esto significa que no cualquier matriz de 2×2, con entradas en $\mathbb{Z}_{26}$, puede ser usada para cifrar porque si la matriz elegida no tiene inversa módulo 26, entonces no es posible descifrar (se puede probar que la matriz tiene inverso si y sólo si multiplicación por ella corresponde a una función inyectiva; si la función no es inyectiva no puede ser usada para cifrar, como ya vimos).

En la sección anterior vimos cuales eran los requisitos para que exista el inverso de un número módulo m. En particular listamos todos los elementos de $\mathbb{Z}_{26}$ que tienen inverso. La matriz de nuestro ejemplo tiene determinante:

$$\begin{vmatrix} 9 & 4 \\ 5 & 7 \end{vmatrix} = 63 - 20 = 43 \equiv 17 \pmod{26}.$$

Ya que $\text{mcd}(17, 26) = 1$ sabemos que sí existe el inverso multiplicativo de 17 modulo 26. A saber, $1/17 = 23 \pmod{26}$ así que la inversa de M es:

$$M^{-1} = 23 \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \pmod{26} = \begin{pmatrix} 161 & -92 \\ -115 & 207 \end{pmatrix} \pmod{26} = \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix}.$$

El lector puede comprobar que multiplicando módulo 26 esta matriz por los vectores que aparecen en el tercer renglón de la tabla 3.2 se obtienen los del segundo renglón.

Por otro lado, la matriz:

$$\begin{pmatrix} 2 & 12 \\ 1 & 14 \end{pmatrix}$$

tiene determinante 2, que no es primo relativo con 26, así que no tiene inverso y por tanto la matriz tampoco tiene inversa. No se puede usar para cifrar.

Cuando trabajamos módulo 26 hay dos matrices de 2×2 que son su propia inversa. Por ejemplo,

$$\begin{pmatrix} 2 & 3 \\ 25 & 24 \end{pmatrix}$$

es una de ellas. A estas matrices se les denomina *auto-involutivas*. Tienen la propiedad que si se utiliza una matriz auto-involutiva para cifrar, esa misma matriz sirve para descifrar.

Nada impide que en vez de usar matrices de 2×2 se utilicen matrices más grandes. En general una matriz de $n \times n$ cifra n-gramas. En [Hil31] Hill propone algunas matrices de 3×3.

El método de Hill es el primero con el que nos topamos que posee una característica relevante: La clave para cifrar (la matriz) no es necesariamente igual a la clave para descifrar (la matriz inversa). A un sistema con esta cualidad se le denomina *asimétrico*. Más adelante, cuando abordemos el tema de criptografía de llave pública veremos la utilidad de algunos sistemas criptográficos asimétricos más poderosos. En el sistema de Hill las claves de cifrado y de descifrado no son la misma, pero si se conoce una de ellas, la otra es fácilmente calculable. Si no fuera así podríamos darle a todo mundo una matriz de cifrado de la que conocemos su inversa, con la seguridad de que cualquiera puede cifrar mensajes con ella, pero nadie puede calcular fácilmente la inversa que descifra.

Otra característica importante es que el método de Hill permite cifrar n-gramas para n grande de manera bastante sencilla. No se requieren grandes tablas de n-gramas ni complicados procesos para realizar el cifrado.

3.6 Criptoanálisis de sistemas poligráficos

El primer paso para hacer criptoanálisis de un sistema poligráfico es reconocer que lo que tenemos es un sistema poligráfico. Es decir, poder diferenciar un criptotexto obtenido usando un sistema monoalfabético o polialfabético de uno que obtenemos mediante un sistema poligráfico.

Uno comienza, al igual que con los criptotextos de capítulos anteriores, calculando la tabla de frecuencia monoalfabética, y los índices de coincidencia. Normalmente, las fórmulas que tenemos nos indicarían que se trata de un sistema de n alfabetos, cuando estamos viendo el resultado de un sistema n-gráfico. Pero al calcular los índices de coincidencia de cada alfabeto, nuevamente obtenemos resultados demasiado pequeños para que se trate de un sistema polialfabético.

Si hemos desechado la posibilidad de que se trate de un criptotexto producido mediante un sistema polialfabético, nuestra siguiente posibilidad sería un sistema poligráfico; el número de alfabetos y las distancias entre repeticiones nos indican el valor probable de n. Un análisis de frecuencia de n-grafos puede apoyar esa hipótesis.

Una vez identificado el texto como el resultado de un sistema poligráfico, y una vez determinado el tamaño de las gráficas, podemos proceder a identificar el sistema particular que se utilizó. En el sistema de Playfair, por ejemplo, nunca hay digráficas que consistan de una misma letra repetida. En los sistemas de dos cuadrados, muchas digráficas naturales ocurren en el criptotexto, o bien sus reversos (naturales cuando se trata del sistema vertical, sus reversos en el horizontal). El sistema de cuatro cuadrados normalmente se identifica por eliminación: a diferencia de Playfair, incluye digráficas que consisten de una letra repetida; no tiene una proporción alta de digráficas que son de alta frecuencia en el lenguaje original, ni tales que la digráfica invertida lo tenga.

Las técnicas para iniciar la solución de un sistema de digráficas, una vez determinado el sistema, son similares a las necesarias para iniciar la solución de un sistema monoalfabético. Esencialmente, usamos datos de frecuencia, repetición de patrones, y contactos de *digráficas* para desarrollar posibles valores del texto original. Una vez obtenido algo de texto original, podemos usar esa información para identificar nuevas porciones del texto cifrado, y continuamos de esa manera hasta obtener la solución total. El sistema particular usado se utiliza para identificar mas correspondencias, como veremos abajo.

3.7 Criptoanálisis de Playfair

El criptoanálisis del sistema de Playfair requiere de identificar alguna correspondencia entre texto original y criptotexto. Esto se puede hacer analizando frecuencias de digráficas (que requiere de mucho más datos que la frecuencia monoalfabética), o analizando patrones de palabras comunes.

Una propiedad del sistema de Playfair que ayuda tanto para reconocer Playfair mismo como para su criptoanálisis es exclusiva al sistema de Playfair, de entre todos los sistemas poligráficos que hemos estudiado: cada vez que una digráfica del texto original se repite en el orden inverso, el criptotexto correspondiente también se repite en el orden inverso. Por ejemplo, si el texto *ea* corresponde al criptotexto `RB`, entonces el criptotexto `BR` corresponde al texto *ae*. Otra propiedad muy útil es la simetría de toro[3] que tiene la tabla de Playfair, lo cual nos permite reconstruir la tabla sin tener que preocuparnos de quien estaba "realmente" en la esquina superior izquierda, por ejemplo.

El criptoanálisis de Playfair comienza con una identificación de cierto criptotexto, y luego utiliza esa información para ir reconstruyendo la tabla de Playfair utilizada para el cifrado. Para ilustrar el sistema, supongamos que hemos recibido el siguiente criptotexto:

```
TCVKS TEVAN MLAEM FKCTS HNLAF NMYEM FUDZK
ETSGM JKVAP TMYBL FUNFD KLIOP IMKZQ SISAF
IMPCN BVRMK NJKEP OILKO TJBDH KMEAT GMDJG
MMAPA HKDMM ZAMAE MFKCD JGMAT GMIDK XAVTS
TMNKD MKJRO CPKND ZTLCQ JKBPV MSINJ IMKSF
MABYH MDKJE XPTNB CZKJE XENAV CPHNH KKHMA
CPJKA OEBDJ FNLNT ALMAB MEKXJ KFNKE GMMAP
AHKDM IFMGI NKSFM ABKHE KTLAE MFKCL SMKKJ
PYTPK EGMAT GMLZ.
```

Sabemos que la palabra *infierno* y la palabra *cielo* aparecen varias veces cada una.

Un análisis de índice de coincidencias nos dice que el Índice de Coincidencias del texto es $IC = 0.0629$, y que el número de alfabetos es aproximadamente $r = 1.29$. Puesto que la distribución no es monoalfabética, reordenamos el criptotexto en dos columnas para facilitar su análisis:

[3] El toro es el nombre matemático de una superficie en forma de dona; se obtiene tomando un cuadrado, identificando el lado superior con el inferior para formar un cilindro, y luego identificando el lado derecho e izquierdo, para formar un tubo cerrado.

```
TC VK ST EV AN ML AE MF KC TS HN LA FN MY
EM FU DZ KE TS GM JK VA PT MY BL FU NF DK
LI OP IM KZ QS IS AF IM PC NB VR MK NJ KE
PO IL KO TJ BD HK ME AT GM DJ GM MA PA HK
DM MZ AM AE MF KC DJ GM AT GM ID KX AV TS
TM NK DM KJ RO CP KN DZ TL CQ JK BP VM SI
NJ IM KS FM AB YH MD KJ EX PT NB CZ KJ EX
EN AV CP HN HK KH MA CP JK AO EB DJ FN LN
TA LM AB ME KX JK FN KE GM MA PA HK DM IF
MG IN KS FM AB KH EK TL AE MF KC LS MK KJ
PY TP KE GM AT GM LZ
```

El Índice de Coincidencias de la primer columna es 0.0639, y el de la segunda es 0.0563. En ninguno de los dos casos se tiene una distribución monoalfabética, lo cual sugiere que se trata de un sistema poligráfico, que utiliza digráficas. La falta de digráficas repetidas sugiere que se trata de un sistema de Playfair.

Buscamos repeticiones, y obtenemos las siguientes:

Criptotexto	Número de veces	Posición
`AE MF KC`	3 veces	13, 119, 269
`AT GM`	3 veces	99, 129, 289
`MA PA HK DM`	2 veces	107, 243
`DJ GM`	2 veces	103, 125
`KJ EX`	2 veces	183,193

Esto sugiere las siguientes correspondencias:

```
in fi er no          nf ie rn
MA PA HK DM          AE MF KC
```

Notamos que ésto implicaría que *cielo* aparece como *ci el o** en al menos dos repeticiones, y no como **c ie lo*, pues en el segundo caso tendríamos la digráfica `MF` en otro bloque de repeticiones correspondientes a *cielo*.

Para empezar la reconstrucción de la tabla de Playfair, de ser posible buscamos comenzar con una digráfica en la cual hay una letra en común entre la digráfica cifrada y la digráfica original.

Pero de no ser posible (como en el caso de arriba), empezamos con dos pares de digráficas que tengan al menos dos letras en común. Por ejemplo, el par (*in*, MA), y el par (*fi*, PA).

Ahora consideramos todas las posibles maneras en las que estos pares pueden corresponder en una tabla de Playfair. Por ejemplo, hay tres maneras en las que la digráfica *in* pudo corresponder a la digráfica MA: ya sea con la regla del cuadrado, del renglón, o de la columna. En cada caso respectivo, la tabla de Playfair se vería:

```
I   M                                   I
                                        M

                      IM   NA

                                        N
A   N                                   A
```

donde los espacios representan un número desconocido de renglones o columnas (incluyendo cero) entre ellos.

Pero ahora notamos que también sabemos que *fi* corresponde a PA, y buscamos ver cómo podría esa correspondencia acoplarse con las tablas que ya teníamos. Supongamos, por ejemplo, que $in \mapsto$ MA resultó usando la regla del cuadrado. En ese caso, no es posible que $fi \mapsto$ PA sea el resultado de aplicar la regla del cuadrado, pues en ese caso A tendría que estar en el mismo renglón que I. De igual manera, no puede ser el resultado de la regla del renglón; de manera que la única manera en la que estas dos correspondencias pueden ser ciertas si la primera se debió a la regla del cuadrado, es si la segunda se debió a la regla de la columna. En cuyo caso, nuestro cuadrado de Playfair sería:

```
F
P

I    M
A    N
```

Supongamos ahora que la asignación $in \mapsto$ MA fue el resultado de la regla del renglón. Entonces $fi \mapsto$ PA no puede ser el resultado de la regla del renglón ni de la columna, y debe ser el resultado de la regla del cuadrado. En cuyo caso, nuestra aproximación al cuadrado de Playfair sería:

```
P       F

IM      NA
```

Finalmente, si la asignación $in \mapsto$ `MA` se debió a la regla de la columna, entonces la asignación $fi \mapsto$ `PA` es imposible. De manera que la primer asignación no se pudo deber a la regla de la columna. Esto nos deja dos posibilidades para nuestra aproximación a la tabla de Playfair, que es consistente con estas dos asignaciones:

```
F
P                                       P       F


I       M                               IM      NA
A       N
```

Continuamos ahora tratando de acoplar las otras parejas que conocemos a estas tablas, viendo cuáles son las posibilidades válidas.

Por ejemplo, consideremos la asignación $nf \mapsto$ `AE`. Si la tabla de Playfair corresponde a la primera opción de arriba, entonces esta asignación no puede ser por regla de la columna, pues `N` y `A` están en columnas distintas; ni por regla del renglón, que requeriría que `F` y `A` estén en el mismo renglón. Por lo tanto, tendría que ser por la regla del cuadrado. Por otro lado, si la tabla corresponde a la segunda opción arriba listada, entonces esta asignación no puede deberse ni a la regla del renglón ni de la columna, pues `F`, `N`, y `A` no están ni en el mismo renglón ni en la misma columna; de manera que también se tendría que deber a la regla del cuadrado. Nuevamente tenemos, pues, dos posibilidades para la tabla:

```
F       E                               P       EF
P

I       M                               IM      NA
A       N
```

También tenemos la asignación $ie \mapsto$ `MF`; pero ésta es incompatible con la segunda opción de arriba, que daría $ie \mapsto$ `NP`. De manera que la única posibilidad para la tabla de Playfair que produjo el mensaje, con esas asignaciones, es la primera.

Veamos ahora la asignación $no \mapsto$ `DM`. No puede ser debida a la regla del renglón, pero tanto la del cuadrado como la de la columna son compatibles. En el caso de la primera, obtenemos la tabla:

```
F    E
P

I    M    O
A    N    D
```

En el caso de la segunda, obtenemos la tabla:

```
  F       E
  P

          O
  I       M
  A       N
          D
```

Pero, claro, se trata de una tabla de cinco por cinco, de manera que podemos "acoplar" los dos pedazos de la tabla para obtener que en el segundo caso nuestra aproximación a la tabla de Playfair es entonces:

```
F      E
P      O
I      M
A      N
       D
```

Aquí podemos ver el uso de la simetría de toro del sistema de Playfair. No importa cuál renglón pongamos primero, siempre y cuando mantengamos el orden de los renglones correctamente.

Nuestras dos posibilidades son entonces:

```
F  E           F  E
P              P  O
I  M  O        I  M
A  N  D        A  N
               D
```

Para continuar, tomamos dos parejas de correspondencias, $er \mapsto$ `HK` y $rn \mapsto$ `KC`. Si la primera se debe a la regla del renglón, puesto que `E` y `N` están en la misma columna (en ambas posibilidades arriba), tendríamos la situación siguiente:

```
EH  RK

N
```

y no habría manera de hacer la correspondencia $rn \mapsto$ `KC`. Si se debiera a la regla de la columna, tendríamos a `E`, `H`, `R`, y `K` en la misma columna; pero también sabemos que al menos `M` y `N` van en la columna de `E` (nuevamente, de acuerdo a ambas posibilidades arriba), lo cual nos da al menos seis letras en la misma columna de Playfair, lo cual por supuesto es imposible. De manera que la asignación $er \mapsto$ `HK` se debe, necesariamente, a la regla del renglón. Como `R` y `N` no pueden estar en el mismo renglón (el lugar correspondiente está ocupado por la `K`), y como `R` y `K` están en el mismo renglón, la única posibilidad para la asignación $rn \mapsto$ `KC` sería un diagrama como el siguiente:

```
E    H

K    R

N    C
```

Ahora buscamos acoplar este diagrama parcial con nuestras dos reconstrucciones parciales de la tabla de Playfair de arriba. No cabe con la segunda, pues en la segunda ya tenemos cinco letras en la columna de `E`, y tendríamos que agregar una sexta, a saber `K`. Eso quiere decir que la única posibilidad es la primera, y al acoplar este diagrama de seis letras, obtenemos la siguiente reconstrucción parcial:

```
F   E       H
P

    K       R

I   M   O
A   N   D   C
```

Es importante notar que el orden de las columnas no está fijo, ni tampoco el de los bloques de renglones. El renglón de la `K` podría ir arriba del renglón de la `F`. Sólo los renglones listados contiguamente deben permanecer en esa relación unos con otros. También es posible que el renglón de la `K` fuera el mismo que el de la `P`. Simplemente no sabemos todavía.

Usamos esta información parcial para regresar a nuestro criptograma, y obtener algunas correspondencias más; y también utilizamos las correspondencias que ya teníamos. Por ejemplo, tenemos en el quinto renglón:

```
ni nf ie rn o
AM AE MF KC DJ
```

Eso quiere decir que $o* \mapsto$ `DJ`. Como la `O` está arriba de la `D`, `J` tiene que estar en la misma columna en la tabla de Playfair.

Sabemos que *cielo* aparece como *ci el o**; como ya tenemos que $ci \mapsto$ `A*`, podemos adivinar de nuestra tabla de repeticiones que *ci el* $\mapsto$ `AT GM`. Eso nos dice que los digramas `DJ`, `ID`, y `LZ` corresponden todos a *o**.

Para poder acomodar la correspondencia $ci \mapsto$ `AT`, la única posibilidad en la tabla que ya tenemos es:

```
          J

F   E       H
P

    K       R

I   M   O   T
A   N   D   C
```

Esto nos permite llenar más digramas en nuestro criptotexto; por ejemplo, `ID` corresponde a *oa*, etc. Rellenando, y sabiendo que $el \mapsto$ `GM`, tenemos:

```
 t                nf ie rn    ec    ea
TC VK ST EV AN ML AE MF KC TS HN LA FN MY

                           i          ae n
EM FU DZ KE TS GM JK VA PT MY BL FU NF DK

   i              i      a
LI OP IM KZ QS IS AF IM PC NB VR MK NJ KE

 i     m       er    ci el o  el in fi er
PO IL KO TJ BD HK ME AT GM DJ GM MA PA HK

no       ni nf ie rn o  el ci el oa
DM MZ AM AE MF KC DJ GM AT GM ID KX AV TS
```

```
   m  no     t a   m
TM NK DM KJ RO CP KN DZ TL CQ JK BP VM SI

d        ei       on        i
NJ IM KS FM AB YH MD KJ EX PT NB CZ KJ EX

 m    a  ec er re in a     di    o  ea
EN AV CP HN HK KH MA CP JK AO EB DJ FN LN

ic                ea    el in fi er no
TA LM AB ME KX JK FN KE GM MA PA HK DM IF

   ma    ei    re       nf ie rn
MG IN KS FM AB KH EK TL AE MF KC LS MK KJ

   i     el ci el
PY TP KE GM AT GM LZ
```

Tenemos pues que ML debe corresponder a **i*. Como la I está en el mismo renglón que M en nuestra tabla, la L debe estar a la derecha de I. Es decir,

```
          J

F   E       H
P

    K       R

IL  M   O   T
A   N   D   C
```

También tenemos que TS corresponde a *o**, y que TL corresponde a **i*, con lo cual obtenemos más correspondencias. Puesto que LS corresponde a *o**, y ya tenemos todo el renglón (las cinco entradas; sólo falta determinar el orden), eso quiere decir que S está en la columna de O; como LZ también corresponde a *o**, Z también está en esa columna. De aquí obtenemos

```
F   E   H
P

    K   R

IL  M   T   O
A   N   C   D

            J

            S

            Z
```

Como `DJ` corresponde a *od*, la `J` debe estar inmediatamente abajo de la `D`. También tenemos que $ou \mapsto$ `MZ`, de manera que nuestra tabla ahora se convierte en:

```
F   E   H
P

    K   R

IL  M   T   O
A   N   C   D
            J

            S

    U       Z
```

Como $ar \mapsto$ `CP`, `K` y `R` van en el mismo renglón que `P`. Eso obliga a la `S` a estar en el renglón de `F` o de `P`. Pero si `S` está en el renglón de `F`, entonces no habría donde acomodar el renglón de `U`. De manera que debemos de tener:

```
F   E   H
P   K   R   S

IL  M   T   O
A   N   C   D
            J

    U       Z
```

Para poder empalmar los dos bloques de dos renglones, la única posibilidad es:

```
IL  M   T   O
A   N   C   D
F   E   H   J
P   K   R   S
    U       Z
```

Nuevamente usamos la simetría del toro, pues da lo mismo si ponemos el renglón de `U` arriba o abajo.

Con esta tabla podemos ahora descifrar mucho más. Por ejemplo, en el último renglón tenemos:

```
re ne *i nf ie rn o
RE EK TL AE MF KC LS
```

y el * debe ser `M`, `L`, u `O`. La `L` es la opción clara, lo cual nos permite poner tres columnas en orden:

```
ILT   M   O
A C   N   D
F H   E   J
P R   K   S
      U   Z
```

Tenemos que `EN PO IL KO`, al final del tercer y principio del cuarto renglón, corresponde a *en si *i sm*, y la `I` debe corresponder a `O` o a `M`. Eso nos dice que la `M` debe estar a la izquierda inmediata de la `I`, y eso nos deja a la `O` en cualquier extremo. Nuestra tabla de Playfair as ahora

```
M I L T O
N A   C D
E F   H J
K P   R S
U       Z
```

donde las columnas y renglones ya se encuentran en su orden correcto.

Esto nos permite suponer que la tabla es:

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

Con ello desciframos el resto del mensaje. Obtenemos:

```
yt up ro fu nd oi nf ie rn or ec ib ea tu
TC VK ST EV AN ML AE MF KC TS HN LA FN MY

nu ev os en or el es pi ri tu lx ev ae ns
EM FU DZ KE TS GM JK VA PT MY BL FU NF DK

im is mo su pr op ia mo ra da yp ue de en
LI OP IM KZ QS IS AF IM PC NB VR MK NJ KE

si mi sm oh ac er un ci el od el in fi er
PO IL KO TJ BD HK ME AT GM DJ GM MA PA HK

no ou ni nf ie rn od el ci el oa qu ip or
DM MZ AM AE MF KC DJ GM AT GM ID KX AV TS
```

```
lo me no se st ar em os li br es aq ui po
TM NK DM KJ RO CP KN DZ TL CQ JK BP VM SI

de mo sr ei na rc on se gu ri da dy se gu
NJ IM KS FM AB YH MD KJ EX PT NB CZ KJ EX

nm ip ar ec er re in ar es di gn od ea mb
EN AV CP HN HK KH MA CP JK AO EB DJ FN LN

ic io na un qu es ea en el in fi er no va
TA LM AB ME KX JK FN KE GM MA PA HK DM IF

le ma sr ei na re ne li nf ie rn oq ue se
MG IN KS FM AB KH EK TL AE MF KC LS MK KJ

rv ir en el ci el ox
PY TP KE GM AT GM LZ
```

> *"Y tú, profundo infierno, recibe a tu nuevo señor. El espíritu lleva en sí mismo su propia morada, y puede en sí mismo hacer un cielo del infierno o un infierno del cielo. Aquí por lo menos estaremos libres. Aquí podemos reinar con seguridad. Y reinar, según mi parecer, es digno de ambición, aunque sea en el infierno. Vale más reinar en el infierno que servir en el cielo."*

Es parte del monólogo de Lucifer en el Infierno, en el Canto I de *El Paraíso Perdido*, de John Milton.

Para resolver sistemas de Playfair como éste, es importante recordar que hay que probar todas las posibilidades en cada paso, y tratar de hacer las cosas de manera sencilla, para que tengamos pocas posibilidades en cada paso. Es muy fácil ignorar una posibilidad cuando tratamos de armar la tabla parcial, así que es muy importante ser ordenado y trabajar sistemáticamente. Para evitar que el trabajo se vuelva demasiado complejo, hay que buscar digráficas que tengan pocas posibles posiciones en la construcción, y que permitan ordenar los renglones y columnas cuando sea posible.

3.8 Criptoanálisis de cuatro cuadrados

Al igual que con el sistema de Playfair, el sistema de cuatro cuadrados requiere, para poder iniciar el criptoanálisis, que podamos identificar o asumir parte del texto. Esto se puede hacer con tablas de frecuencias de digráficas, con palabras probables, estudio de patrones, o con inicios y terminaciones típicas.

Si ya sabemos que el mensaje fue cifrado con un sistema digráfico, normalmente lo identificamos como el resultado de un sistema de cuatro cuadrados a base de eliminación; por ejemplo, tiene digráficas que consisten de la repetición de una misma letra, lo cual es imposible en Playfair.

Consideremos el siguiente mensaje, que fue cifrado con el método de cuatro cuadrados. Nuestra información indica que el mensaje comienza con *amigos*, termina en *dura*, y contiene varias veces la palabra *ambición* o *ambicioso*.

```
SF DD LS TH IL MK PI KM NJ PS HK NO NR TL TR AJ ML RN AM AC LM ZU JD
IO TI NT MP SN SJ AS SN MK CJ IU CU OH MO KG EF NV EM LS HH JL TL PD
SJ AM MU QR JO TL YB ZU EM AB AM AS SF AM YJ MO MR SS NL RK RH LR YN
BX AS LS TI RN SF AB AM AN FI LM RL NI TL LG JO MU SP ZN YN AD RL NV
SO AS SN SS CJ JL FI HK TK SF DD KM HK SS EF CO LV BZ TR IS SN SF BR
SS JO NX NZ QI FI ES ZJ SS CJ JL FI HK TK VI NX NZ TI PD KM SP SE LM
KS EJ MU IX SC LS CL ZN DY LS PS RA IO TH IL XI LM NY PR AS CL TO OK
MU MK IU NI PL AN NT EG FI AN SS RJ TK AM RL NI NL JL FI HK TK IU OK
EA QC IL QI EM LS TF SP AS BL AC AF JG KS SN SJ AS SN VK CJ JL FI HK
LI EO SS BI TI QL AE LI JM SM AN CY NL
```

Las repeticiones relevantes son el criptotexto `CJ JL FI HK TK`, que aparece a la mitad del sexto renglón, y al final del séptimo; `JL FI HK TK` al final del décimo, y `JL FI KH` en el duodécimo. Eso sugiere, en vista de la palabra probable, que `JL FI KH TK` representa el final de *ambicioso*, y `JL FI KH` un fragmento de *ambición*. Entonces, `JL FI KH TK` es *mb ic io so*, y `CJ JL FI HK LI` es **a mb ic io n**.

También sabemos que `SF DD LS` es *am ig os*, y que `CY NL` es *du ra*. Con esta información, vamos llenando los cuadrados, y obtenemos

```
      a b c d e     C   S
      f g h i j       D F   H
      k l m n o       J   L
      p q r s t     N       T
      u v x y z

J         L I       a b c d e
            D       f g h i j
      F     K       k l m n o
              S     p q r s t
            Y       u v x y z
```

Sabemos que la `J` está en el tercer cuadrante, en el renglón de la `a` del cuarto cuadrante, pues `CJ` corresponde a **a*. Como no sabemos en qué columna está, la ponemos fuera del cuadrado pero indicando el renglón en el que está. Usamos esta tabla para hacer un descifrado parcial.

En el sexto renglón, `SF DD KM KH SS EF` queda como *am ig ** io er ***, y una buena opción es *am ig om io er a**. De manera que `KM` corresponde a *om*, y `EF` a *a**. Eso nos da:

```
      a b c d e     C   S         E
      f g h i j       D F   H
      k l m n o       J K L
      p q r s t     N       T
      u v x y z

J         L I       a b c d e
            D       f g h i j
      F     K M     k l m n o
            S       p q r s t
            Y       u v x y z
```

Utilizamos esta tabla parcial para descifrar mas texto. Como `CJ` es **a*, y la `J` está en el primer renglón de abajo, en el sexto renglón tenemos que `SS CJ JL FI KH TK` debe ser uno de *er aa mb ic io so*, *er ba mb ic io so*, y *er ea mb ic io so*. La opción obvia es la primera, lo cual coloca a `J` al principio del cuadrado de abajo a la izquierda, y permite más descifrado.

Esto sugiere probar `JULIO` como base del cuadrado inferior izquierdo, pues además da buenas digráficas en el resto del texto. Y en el renglón doce tendríamos `CJ JL FI HK LI EO SS`, que tiene traducción parcial *aa mb ic io nd e* er*, con la letra faltante necesariamente *b*, *d*, o *e*. La opción clara es *b*, lo cual también coloca la `E` del cuadro superior derecho. Tenemos pues

```
a b c d e     C E S
f g h i j       D F   H
k l m n o       J K L
p q r s t     N       T
u v x y z

J U L I O     a b c d e
      D       f g h i j
F     K M     k l m n o
        S     p q r s t
      Y       u v x y z
```

y rápidamente sugerimos que el primer renglón del segundo cuadrante sea `CESAR`.

Con esto desciframos aun más texto. Puesto que `ES` corresponde a *eq* (renglón siete), tenemos que `ZJ` debe ser *ue* o *ui*, y por estar en el renglón de la `J` del tercer cuadrante, *ue* es lo obvio.

```
a b c d e     C E S A R
f g h i j       D F   H
k l m n o       J K L
p q r s t     N       T
u v x y z             Z

J U L I O     a b c d e
      D       f g h i j
F     K M     k l m n o
        S     p q r s t
      Y       u v x y z
```

Parece claro que el lugar después de la `L` del segundo cuadrante debe corresponder a `M`, y la última posición del tercer cuadrante a `Z`. Entre la `F` y la `K` del tercer cuadrante la izquierda van `G` y `H` (`I` y `J` ya están antes), y continuando de esta manera, el resto del cuadrado se llena. Obtenemos:

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

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

Utilizando esta tabla para descifrar el mensaje, obtenemos:

> *"Amigos, romanos, compatriotas. Prestadme atención. Vengo a sepultar a César, no a alabarlo. El mal que los hombres hacen les sobrevive; el bien es a menudo enterrado con sus huesos. Sea también así con César. El noble Brutus dice que César era ambicioso. Amigo mío era, leal y justo para mí, pero Brutus dice que era ambicioso, y Brutus es hombre honorable. Muchos cautivos trajo a Roma, y con sus rescates llenó las arcas públicas. ¿Era eso en César ambicioso? Las lágrimas de los pobres hacían llorar a César, y la ambición debería ser de índole más dura."*

Es parte del monólogo de Marco Antonio, Escena II, Acto III, de *Julio César*, de William Shakespeare.

3.9 Criptoanálisis de cuatro cuadrados con alfabetos mezclados y de dos cuadrados

La solución de un criptograma en el cual se utilizó el sistema de cuatro cuadrados, pero donde cada cuadrante contiene un alfabeto mezclado, es un poco distinta. No la vamos a demostrar en detalle, pues consiste de una mezcla entre los métodos que acabamos de ver, y el método usado para el criptoanálisis de Playfair. En general, se requiere más texto conocido o asumido para resolver éste sistema que el inmediato anterior.

Para resolver el sistema, tomamos una hoja y la dividimos en cuatro regiones, que corresponden a los cuatro cuadrantes del sistema. Procedemos a poner en cada cuadrante

las parejas de texto-criptotexto que hemos identificado, respetando su relación rectangular. Cada nueva pareja se debe poner en renglones y columnas nuevas, a menos tengan uno o más valores en común con entradas anteriores, lo cual permite identificar renglones o columnas. Luego buscamos acoplar esta información de manera similar a la que usamos en el criptoanálisis de Playfair.

La solución de un sistema de dos cuadrados, ya sea horizontal o vertical, es parecida. Primero dividimos nuestra hoja de trabajo en dos regiones. Reconstruimos la matriz igual que en el caso de cuatro cuadrados mezclados, poniendo parejas de digráficas en arreglos rectangulares, con nuevos renglones y columnas en cada paso. Cuando tenemos una transparencia, las dos letras ocurren en el mismo renglón o misma columna.

3.10 Criptoanálisis de sistema de Hill

Para facilitar el estudio, haremos criptoanálisis del sistema de Hill aplicado a una sustitución digráfica; es decir, donde las matrices utilizadas para la transformación son de dos por dos.

En este caso, recordemos que en realidad el cifrado de Hill consiste en una transformación lineal. Si pudiéramos obtener dos o más correspondencias entre texto y criptotexto, las podríamos usar para obtener un sistema de congruencias, donde las entradas de la matriz son las incógnitas. Dos correspondencias darían cuatro congruencias, que puede ser suficiente para resolver el sistema de manera única. Si no lo son, entonces las distintas posibilidades se pueden probar contra el criptotexto, o nuevas congruencias pueden ser obtenidas, para obtener una solución.

Todo esto, por supuesto, asumiendo que el criptoanalista conoce la correspondencia entre letras y números que está siendo utilizada.

Consideremos, por ejemplo, el siguiente mensaje; suponemos que ya hemos identificado el sistema como un sistema de Hill, usando la correspondencia usual de letras a números ($a \leftrightarrow 0, \ldots, z \leftrightarrow 25$). Nuestro servicio de inteligencia nos ha indicado que el mensaje comienza con *si nosotros*.

```
MS LF AW XE GS NX IM AW HL HC SL MD QQ DZ SU QV UW DP HX IO GN AA RS
AC HB HX IO AP YH CV HX BT SL QG IO AE QQ BE ZL GH QM VX MS QT CK WX
SH II YW NN MS FV CK PP ML QM LF SV CK OR KV GS BZ PV CK YC WX HB YC
GS MS AW IM CE UI QC GU XD NH DZ GN OI SI DY SH GL
```

Tenemos, pues, las siguientes correspondencias:

$$\begin{pmatrix} \texttt{M} \\ \texttt{S} \end{pmatrix} = \begin{pmatrix} 12 \\ 18 \end{pmatrix} \longmapsto \begin{pmatrix} 18 \\ 8 \end{pmatrix} = \begin{pmatrix} s \\ i \end{pmatrix}$$
$$\begin{pmatrix} \texttt{L} \\ \texttt{F} \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix} \longmapsto \begin{pmatrix} 13 \\ 14 \end{pmatrix} = \begin{pmatrix} n \\ o \end{pmatrix}$$
$$\begin{pmatrix} \texttt{A} \\ \texttt{W} \end{pmatrix} = \begin{pmatrix} 0 \\ 22 \end{pmatrix} \longmapsto \begin{pmatrix} 18 \\ 14 \end{pmatrix} = \begin{pmatrix} s \\ \texttt{o} \end{pmatrix}$$
$$\begin{pmatrix} \texttt{X} \\ \texttt{E} \end{pmatrix} = \begin{pmatrix} 23 \\ 4 \end{pmatrix} \longmapsto \begin{pmatrix} 19 \\ 17 \end{pmatrix} = \begin{pmatrix} t \\ r \end{pmatrix}$$
$$\begin{pmatrix} \texttt{G} \\ \texttt{S} \end{pmatrix} = \begin{pmatrix} 6 \\ 18 \end{pmatrix} \longmapsto \begin{pmatrix} 14 \\ 18 \end{pmatrix} = \begin{pmatrix} o \\ s \end{pmatrix}$$

Con estas identificaciones, tenemos dos opciones, que normalmente son equivalentes: podemos buscar la matriz de cifrado, y luego calcular su inverso para encontrar la de descifrado. O bien podemos tratar de calcular la matriz de descifrado directamente, y obtener la de cifrado calculando su inversa (módulo 26, por supuesto). La elección de cuál de los dos caminos elegir depende de cuál sistema de congruencias sea más sencillo. En general, conviene más un sistema que tiene ceros en los coeficientes y no en los resultados. En este caso, vamos a calcular la de descifrado, aprovechando el 0 en la tercer correspondencia.

Recordemos que buscamos la matriz

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \pmod{26}$$

que hace las transformaciones indicadas arriba. Usado la tercer transformación, tenemos pues que

$$\begin{aligned} 0a + 22b \equiv 22b &\equiv 18 \pmod{26} \\ 0c + 22d \equiv 22d &\equiv 14 \pmod{26} \end{aligned}$$

Eso nos dice que $11b \equiv 9 \pmod{13}$; esta congruencia es equivalente a

$$\begin{aligned} -2b &\equiv -4 \pmod{13}, \\ b &\equiv 2 \pmod{13}. \end{aligned}$$

De manera análoga, $11d \equiv 7 \pmod{13}$ nos da como resultado que $d \equiv 3 \pmod{13}$.

Esto nos dice que el valor de b, módulo 26, es 2 ó 15; y el de d es 3 ó 16.

Para encontrar valores para a y c, usaremos la primer y última ecuación. La razón es que la coordenada dos de ambos vectores es igual, y entonces va a ser fácil cancelar la b y la d para obtener una congruencia en a y una en c. La primer transformación nos dice que

$$\begin{aligned} 12a + 18b &\equiv 18 \pmod{26} \\ 12c + 18d &\equiv 8 \pmod{26} \end{aligned}$$

y la última que

$$\begin{aligned} 6a + 18b &\equiv 14 \pmod{26} \\ 6c + 18d &\equiv 18 \pmod{26} \end{aligned}$$

Restando la tercera de la primer congruencia obtenemos que $6a \equiv 4 \pmod{26}$, o que $3a \equiv 2 \pmod{13}$, de donde concluimos que $a \equiv 5 \pmod{13}$. De la segunda y cuarta tenemos que $6c \equiv -10 \equiv 16 \pmod{26}$, y de ahí que $c \equiv 7 \pmod{13}$.

Por lo tanto,

$$\begin{aligned} a &= 5 \text{ ó } 18; \\ b &= 2 \text{ ó } 15; \\ c &= 7 \text{ ó } 20; \\ d &= 3 \text{ ó } 16. \end{aligned}$$

Ahora tenemos que encontrar los valores exactos. Podríamos intentar las 16 posibilidades, o podemos proceder un poco más cuidadosamente. De la segunda transformación, $\texttt{LF} \mapsto no$, tenemos que $11a + 5b \equiv 13 \pmod{26}$, de manera que a y b deben tener paridad opuesta. También obtenemos que $11c + 5d \equiv 14 \pmod{26}$, de manera que c y d tienen la misma paridad. Por lo tanto,

$$\begin{aligned} a = 5,\ b = 2 \quad &\text{ó} \quad a = 18,\ b = 15; \\ c = 7,\ d = 3 \quad &\text{ó} \quad c = 20,\ d = 16. \end{aligned}$$

Como los múltiplos pares de 13 son 0 módulo 26, para determinar cual valor usar necesitamos congruencias dónde los coeficientes de a y b sean de paridad distinta (si son de la misma paridad, no hay manera de distinguir las dos posibilidades, pues al final habremos sumado un múltiplo par de 13; lo mismo para c y d). Esto nos lo da la cuarta correspondencia, $\texttt{XE} \mapsto tr$, que es equivalente a las congruencias

$$\begin{aligned} 23a + 4b &\equiv 19 \pmod{26} \\ 23c + 4d &\equiv 17 \pmod{26} \end{aligned}$$

Sustituyendo $a = 18$, $b = 15$ obtenemos $23a + 4b = 474 \equiv 6 \pmod{26}$, de manera que $a = 3$, $b = 2$. Y sustituyendo $c = 20$, $d = 16$ tenemos que $23c + 4d = 524 \equiv 4 \pmod{26}$, así que $c = 7$ y $d = 3$.

Entonces la matriz de descifrado es

$$\begin{pmatrix} 5 & 2 \\ 7 & 3 \end{pmatrix}$$

De una vez calculamos su inverso. El determinante es $15 - 14 = 1$, así que el inverso es simplemente

$$\begin{pmatrix} 3 & -2 \\ -7 & 5 \end{pmatrix} = \begin{pmatrix} 3 & 24 \\ 19 & 5 \end{pmatrix},$$

que es la matriz de cifrado.

Utilizando la matriz de descifrado, obtenemos que el texto es:

> *Si nosotros hemos ofendido, piensa ésto y todo queda arreglado: Que te haz dormido aquí mientras las visiones aparecían. Así pues, buenas noches a todos. Denme sus aplausos si somos amigos, y Robin se disculpara[x].*

que es el Epílogo de *Sueño de una Noche de Verano*, de William Shakespeare.

El sistema de Hill se presta fácilmente a aumentar el valor de n, es decir, pasar de un sistema digráfico a un sistema n-gráfico con n grande. En ese caso, cada correspondencia de una n-gráfica de texto original a una n-gráfica de criptotexto resulta en n congruencias, cada una con n incógnitas (las n^2 entradas de la matriz). Procederíamos como antes, aunque la resolución de estos sistemas es, por supuesto, más complicada y lleva a mayor posibilidad de ambigüedades.

4.1 Cifrado de Vernam y seguridad perfecta

4.1.1 Comunicaciones telegráficas a principios del siglo XX

En 1837 Samuel Morse en Estados Unidos y Sir Charles Wheatstone (el mismo que inventó el cifrado de Playfair) inventaron, independientemente, un telégrafo. El diseño de Morse era más barato, y no requería de los cinco hilos de transmisión que tenía el de Wheatstone, así que fue el que prevaleció, junto con el código asociado a la transmisión: el código Morse. Más tarde, en 1857, Wheatstone inventó un mecanismo que disminuía considerablemente la posibilidad de errores al transmitir mensajes. El mecanismo de Wheatstone consistía en un teclado, donde el telegrafista tecleaba el mensaje e enviar; ese teclado producía como salida una cinta de papel perforada con el código Morse como se muestra en la figura 4.1; después sólo había que colocar la cinta en el dispositivo transmisor, que la leía y enviaba las señales que le correspondían. Esta misma idea fue reutilizada en 1880, cuando

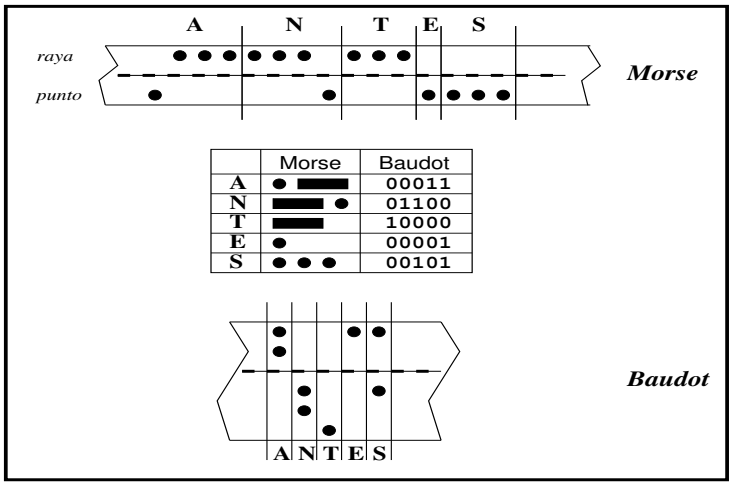

Figura 4.1: Codificación telegráfica en una cinta usando (arriba) el código Morse y (abajo) el código Baudot. En el caso de Morse cada raya está representada por tres puntos (hoyos) en la parte superior de la cinta, esto porque la señal para representar una raya debía ser tres veces más larga que la señal del punto. Hemos puesto un punto para representar un agujero. En Baudot los agujeros equivalen a un `1`, la ausencia de agujero a un `0`.

Emile Baudot inventó un nuevo código telegráfico que, igual que el de Morse, posee sólo dos símbolos fundamentales. Pero a diferencia de el código Morse, en el código de Baudot todos los símbolos tienen la misma longitud. El código Morse fue pensado para minimizar la cantidad promedio de símbolos usados para transmitir un mensaje. El código de Baudot, en cambio, para maximizar la expresividad del código. Por ejemplo, en el código de Baudot hay símbolos especiales que cambian el significado de todos los que le siguen en la transmisión: `10111` (23 en decimal[1]) en código de Baudot representa a la letra `Q`, pero si va precedido de una aparición de el código `11011` (27) entonces su significado cambia para representar el número `1` (véase Tabla 4.1). El código `11111` (31) regresa la transmisión a su estado original, actuando como el inverso del 27, al que se le asigno el nombre inglés de *shift*. El código de Baudot ocupó el lugar del de Morse en las transmisiones telegráficas. El sistema ideado por Wheatstone fue mejorado y adaptado para funcionar en código Baudot: el telegrafista tecleaba el mensaje que era perforado en una cinta. Luego ésta era leída por un aparato transmisor y el mensaje se enviaba. El aparato receptor recibía el mensaje que era nuevamente perforado en una cinta que más tarde era leída por un mecanismo impresor que entregaba el mensaje legible en una hoja de papel. Este era el panorama general de las comunicaciones telegráficas a principios del siglo XX [Kah99].

[1]Estamos considerando el código como la representación de un número entero en base 2, es decir, en binario.

Dec.	**Binario**	**Símbolo**	**Símbolo (con shift)**	**Dec.**	**Binario**	**Símbolo**	**Símbolo (con shift)**
0	00000			16	10000	T	5
1	00001	E	3	17	10001	Z	"
2	00010	*lf*	*lf*	18	10010	L	)
3	00011	A	-	19	10011	W	2
4	00100	*espacio*	*espacio*	20	10100	H	#
5	00101	S	*bell*	21	10101	Y	6
6	00110	I	8	22	10110	P	0
7	00111	U	7	23	10111	Q	1
8	01000	*cr*	*cr*	24	11000	O	9
9	01001	D	$	25	11001	B	?
10	01010	R	4	26	11010	G	&
11	01011	J	’	27	11011	*shift*	*shift*
12	01100	N	,	28	11100	M	.
13	01101	F	!	29	11101	X	/
14	01110	C	:	30	11110	V	;
15	01111	K	(	31	11111	$shift^{-1}$	$shift^{-1}$

Tabla 4.1: El código de Baudot. El código *lf* significa cambio de línea (*line feed*), mientras que *bell* es un sonido, y *cr* es retorno de carro (*carriage return*).

4.1.2 Cifrado *en línea* de Vernam

En 1917, en plena Primera Guerra Mundial, Razelmnod Parker de la compañía AT&T fue sido encargado de un proyecto que tenía como uno de sus objetivos proponer un esquema criptográfico susceptible de usarse en comunicaciones telegráficas [Kah99]. Parker asignó esta tarea a uno de los ingenieros en su equipo de trabajo, que se llamaba Gilbert S. Vernam.

La idea de Vernam fue introducir un nuevo artefacto entre el teclado del telegrafista y el dispositivo transmisor. Ese mismo aparato era también puesto entre el dispositivo receptor y la impresora. El aparato en cuestión recibía dos cintas perforadas como entrada, una con el mensaje (claro o cifrado) y otra con una clave, y era el encargado de cifrar y descifrar el mensaje. Ambas operaciones eran hechas de la misma forma, es decir, aplicando la misma función. Hay que hacer notar que, dado que se utiliza el código de Baudot en el que todo se representa con ceros y unos (falso y verdadero, agujero y espacio) lo que se busca es, de hecho, una función booleana. El valor de cada posición puede verse como un bit.

Si la función aplicada para cifrar y para descifrar es la misma, es decir, es reversible, entonces dicha función debe garantizar que no ocurran cosas como esta:

P	**C**	**M**
0	0	0
0	1	0
1	0	0
1	1	1

En la tabla aparecen todas las posibles combinaciones de bits en el mensaje claro (columna **P**) y palabra clave (columna **C**), arbitrariamente podemos pensar en que un `1` representa un agujero de la cinta y un `0` la ausencia de agujero. Si la columna **M** representa el bit mensaje cifrado estamos en problemas. El primer y el tercer renglón de la tabla tienen el mismo valor de mensaje pero no provienen del mismo valor de texto claro. Aplicar la función a las columnas **C** y **M** no da como resultado la columna **P**. La solución es una función muy conocida por los computólogos, se le suele llamar disyunción exclusiva, XOR, suma booleana, o suma módulo 2:

P	**C**	**M**
0	0	0
0	1	1
1	0	1
1	1	0

Ahora sí, si se aplica la misma función a las columnas **M** y **C** se obtiene de regreso la columna **P**. Podemos pensar la función XOR como una comparación: si ambos bits de argumento coinciden la función vale cero, si son diferentes vale uno.

Vernam le llamó a su método *cifrado en línea* (*on-line encipherment*) [Kah99], dado que no se requiere de una persona que cifre y otra que descifre. Con lo que sabemos hasta ahora resulta fácil ver que el cifrado de Vernam es, esencialmente, un cifrado polialfabético con una tabla de 32×32 entradas; cada renglón de la tabla es un alfabeto mezclado diferente. Si solamente usáramos dos bits la tabla sería:

	00	**01**	**10**	**11**
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

La entrada (i, j) de la matriz es simplemente el XOR de i y j.

Si por ejemplo ciframos el texto *antes* con la clave `CLAVE`, usando el código de Baudot tendríamos lo mostrado en la Tabla 4.1.2.

C	L	A	V	E
01110	10010	00011	11110	00001
00011	01100	10000	00001	00101
a	n	t	e	s
01101	11110	10011	11111	00100
F	V	W	$shift^{-1}$	*espacio*

Tabla 4.2: Ejemplo de cifrado de Vernam. Los renglones 1, 4 y 6 constituyen la palabra clave, el texto claro original y el texto cifrado respectivamente; los renglones 2, 3 y 5 contienen el código Baudot de la clave, del texto claro y del texto cifrado (el XOR de los dos anteriores).

Hablemos un poco más sobre el funcionamiento de la máquina de Vernam. La cinta del mensaje entra, junto con la cinta de la clave, al dispositivo de cifrado/descifrado, pero... ¿Qué sucede si las cintas son de diferentes longitudes? Podría ocurrir que la cinta de clave se acabara antes que la de mensaje. Para que eso no ocurra a Vernam se le ocurrió pegar los extremos de la cinta de clave uno con otro, formando un círculo. Así cuando "se acabara" volvería a comenzar. Pero ahora hay otro problema: ¿De qué longitud hacer la cinta de

clave? ya sabemos que los sistemas poligráficos son susceptibles a la prueba de Kasiski (conocida para la época de Vernam), así que no es buena idea usar claves cortas, y de hecho no es conveniente usar una misma clave por mucho tiempo, pues el criptoanalista puede recabar suficiente texto cifrado como para proceder al criptoanálisis. Poner cintas circulares largas no es buena idea porque resultan inmanejables y hay que tener en cuenta que la misma cinta debe estar tanto del lado del emisor como del receptor, y transportar un rollo grande de papel hace al sistema vulnerable.

A uno de los miembros del equipo de Vernam, Morehouse, se le ocurrió poner varias cintas. Combinar, de hecho, cintas largas y cortas, de tal forma que una clave corta (primaria) cifre una clave más larga (secundaria). Si una de ellas mide 100 caracteres y la otra 850, la combinación nos da un total de $17,000$ caracteres antes de comenzar a repetir el ciclo de clave usada para cifrar mensajes (la longitud es el mínimo común múltiplo de las longitudes de las dos cintas). Sin embargo, Mauborgne, un criptoanalista [Kah99] notó que con suficiente tráfico era posible utilizar superposición de Kerckhoffs y recuperar la clave secundaria.

Usar más de una vez una clave hace que el criptoanalista posea dos textos diferentes cifrados con la misma clave. Dadas las cualidades de la función XOR esto hace posible que el criptoanalista sepa donde coinciden dos letras del texto claro y donde no. Si, por ejemplo, nuestros textos claros *hola* y *dato* fueran cifrados con la misma clave, a saber: `JQRT`, obtendríamos lo siguiente en código Baudot:

clave	`J`	`Q`	`R`	`T`
Baudot clave	`01011`	`10111`	`01010`	`10000`
Claro 1	*h*	*o*	*l*	*a*
Baudot 1	`10100`	`11000`	`10010`	`00011`
Cifrado 1	*shift*	`K`	`O`	`W`
Bc 1	`11111`	`01111`	`11000`	`10011`
Claro 2	*d*	*a*	*t*	*o*
Baudot 2	`01001`	`00011`	`10000`	`11000`
Cifrado 2	*lf*	`H`	`G`	*cr*
Bc 2	`00010`	`10100`	`11010`	`01000`
(**Baudot 1**) XOR (**Baudot 2**)	`11101`	`11011`	`00010`	`11011`
(**Bc 1**) XOR (**Bc 2**)	`11101`	`11011`	`00010`	`11011`

En los renglones etiquetados **Baudot 1** y **Baudot 2** están los textos claros en código de Baudot, en **Bc 1** y **Bc 2** los códigos del texto cifrado que les corresponden. Los dos últimos renglones son exactamente iguales y se obtienen de hacer el XOR entre los renglones del código de texto claro y de hacer el XOR de los renglones del código de texto cifrado.

En síntesis, hacer el XOR de dos textos claros es equivalente a hacer el XOR de los textos cifrados con la misma clave obtenidos de ellos. Así que podemos recuperar la información proporcionada por el índice de coincidencias de ambos textos claros si tenemos la de los textos cifrados; si tenemos suficiente texto, el criptoanálisis tiene buenas posibilidades de tener éxito y descifrar el mensaje..

En síntesis lo recomendable en el sistema de Vernam es que la clave no se repita nunca y que no sea inteligible, es decir claves de un sólo uso y lo más aleatorias posible. Esto es un problema serio, porque si la secuencia que constituye la clave es *realmente* aleatoria (alguien se pone a jugar con unos dados y genera números aleatorios), entonces es, por definición, irreproducible; esto significa que si el emisor la genera, el receptor no puede reconstruirla porque *no hay método determinístico que la genere*. Así que hay dos opciones que, por supuesto, debilitan el sistema: una vez generada la secuencia por alguna de las partes, una copia de ésta es transportada al otro extremo; o bien, la secuencia es generada por un método determinístico y aparenta ser aleatoria. Uno de los problemas con esta última opción es que nuestros generadores de números pseudoaleatorios generan realmente un ciclo de números que aparentan ser aleatorios, pero que tarde o temprano, se repite; así que luego de un tiempo, estaríamos usando realmente la misma clave una y otra vez.

Pero, lejos de los problemas prácticos para implementar este sistema de cifrado, si suponemos que realmente poseemos la capacidad de generar secuencias aleatorias de bits y que la clave puede intercambiarse con toda seguridad, ¿qué tan seguro es el sistema? Intuitivamente, si suponemos que la distribución de los bits en la clave es completamente aleatoria y uniforme entonces no importa que tanta estructura tenga el texto claro, sin importar la distribución de frecuencias de cada bit en él, el resultado de cifrarlo tendrá una distribución uniforme, y todo rasgo distintivo y posible fuente de información se pierden.

De hecho si suponemos que la clave es generada aleatoriamente con distribución uniforme entonces ni siquiera un ataque por fuerza bruta daría resultado. Supongamos por ejemplo que A envía a B el criptograma del renglón seis de la tabla 4.1.2, obtenido a partir del renglón 4 de la misma tabla. Escuchando el canal está C, un intruso que pretende violar la privacía de la comunicación. C sabe el tipo de cifrado que utilizan A y B y decide buscar una clave que le dé como resultado algo inteligible como texto descifrado. Luego de buscar un tiempo encuentra que si a la cadena: $shift^{-1}$ `B L S M`, codificada en Baudot se le hace un XOR con el criptograma se obtiene lo siguiente:

Cifrado	F	V	W	$shift^{-1}$	*espacio*
Baudot cifrado	01101	11110	10011	11111	00100
Clave posible	$shift^{-1}$	B	L	S	M
Baudot clave	11111	11001	10010	00101	11100
Baudot descifrado	10010	00111	00001	11010	11000
Descifrado	*l*	*u*	*e*	*g*	*o*

Como *luego* es un buen candidato para ser el texto claro, C podría pensar que ha encontrado la clave aunque nosotros sabemos que no es así. De hecho cualquier palabra o conjunto de palabras con 5 letras de longitud es un posible candidato a texto claro, pues para cualquiera de ellos es posible encontrar una llave que al cifrarlo de como resultado el criptograma capturado por C. Dado cualquier texto claro $\mathbf{P}$ y cualquier texto cifrado $\mathbf{M}$, la clave que transforma $\mathbf{P}$ en $\mathbf{M}$ es $\mathbf{P} \operatorname{XOR} \mathbf{M}$, la suma booleana bit a bit de $\mathbf{P}$ con $\mathbf{M}$.

Es decir, en el esquema de cifrado de Vernam, para cualquier criptograma $\mathbf{M}$ de longitud m, y cualquier texto claro $\mathbf{P}$ de la misma longitud, existe una clave $\mathbf{C}$, también de longitud m, tal que: $\mathbf{M} = \mathbf{P} \operatorname{XOR} \mathbf{C}$. ¡Es posible obtener cualquier texto dado a partir de cualquier otro eligiendo apropiadamente la clave!

El cifrado de Vernam luce, en teoría, bastante seguro.

4.1.3 Seguridad perfecta

Vamos a analizar más a fondo el tema de la seguridad en relación al cifrado de Vernam.

Hemos dicho que, sin importar cuál sea la distribución de los bits, de hecho de las letras, en el texto claro P; si la distribución de las letras en la clave es uniforme entonces la distribución en el criptograma será también uniforme. Intuitivamente: consideremos que e es la letra más frecuente en español y por tanto la más frecuente en un texto claro escrito en ese idioma; como la clave tiene letras distribuidas uniformemente, la probabilidad de que al aparear la clave con el texto claro, la e se aparee con una letra cualquiera del alfabeto es la misma para todas la letras, la e "aporta frecuencia" en la misma proporción a todas las letras del alfabeto en el criptograma. No hay entonces, información que pueda explotarse en el criptograma, no hay patrón alguno, ni repeticiones.

Supongamos que tenemos un sistema de cifrado simétrico [Beu94]: para cifrar y para descifrar se usa la misma clave. Sean $\mathcal{M}$ el conjunto de todos los posibles mensajes, $\mathcal{K}$ el conjunto de todas las posibles claves, y $\mathcal{C}$ el conjunto de todos los posibles textos cifrados a partir de elementos de $\mathcal{M}$ usando llaves de $\mathcal{K}$. Un algoritmo de cifrado es, de hecho, una función: $f : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$. Denotaremos con f_k la instancia particular del algoritmo de cifrado

cuando se utiliza la clave $k \in \mathcal{K}$. Cifrar es entonces aplicar la transformación: $f_k(M) = C$, con $M \in \mathcal{M}$ y $C \in \mathcal{C}$. Descifrar es aplicar la transformación inversa: $f_k^{-1}(C) = M$.

Un sistema simétrico de cifrado $\mathcal{S}$ consta del conjunto de posibles textos claros $\mathcal{M}$, el conjunto de posibles textos cifrados $\mathcal{C}$ y un conjunto de posibles transformaciones $\mathcal{F}$, una para cada clave.

La función f_k es invertible, lo que significa que cada criptograma puede provenir de, a lo más, un texto claro. Puede ser que haya dos claves k y k' tales que $f_k(M) = f_{k'}(M)$; pero no puede ocurrir que, si fijamos una clave, podamos encontrar dos textos claros que, al cifrarlos con esa clave vayan a dar al mismo criptograma. Eso significa que hay al menos tantos criptogramas como textos claros, es decir:

$$|\mathcal{M}| \leq |\mathcal{C}|$$

Pensemos ahora en el cifrado de César en dos casos diferentes:

1. Los mensajes son letras individuales tomadas al azar de un texto cualquiera en español. Podríamos pensar en tomar el Quijote de Cervantes, abrirlo en una página al azar, cerrar los ojos y poner el índice sobre alguna parte de la página y enviar como mensaje la letra sobre la que quedó el dedo. En este caso $|\mathcal{M}| = 26$.

   Imaginemos ahora que somos el criptoanalista que intercepta mensajes entre A y B. A toma una letra al azar, lo que constituye el mensaje claro μ, lo cifra usando un cifrado de César y obtiene el mensaje cifrado γ que envía a B.

   Nosotros capturamos el mensaje γ y, por supuesto, no sabemos quien es μ, pero sabemos que proviene de un texto en español y conocemos la tabla de frecuencias del español (véase el Apéndice A). Así que sabemos que la probabilidad de que A haya escogido la e es de 0.13 aproximadamente, la de que haya escogido a es 0.11, etcétera.

   ¿Cuál es la probabilidad de que la γ que capturamos provenga de cifrar una μ en particular? Esta probabilidad la denotaremos con

$$\begin{aligned} P_\gamma(\mu) &= P(A \text{ haya escogido } \mu \mid A \text{ envía } \gamma) \\ &= P(\mu \mid \gamma). \end{aligned}$$

   Lo único que sabemos es: $P_\gamma(e) = 0.13$, $P_\gamma(a) = 0.11$, etcétera. Es decir, no sabemos más que lo evidente, lo que todo mundo conoce consultando el Apéndice A: $P(\mu)$. En notación, la situación en la que estamos está caracterizada por:

$$P_\gamma(\mu) = P(\mu) \tag{4.1.1}$$

   para toda $\gamma \in \mathcal{C}$ y para toda $\mu \in \mathcal{M}$

2. Ahora digamos que nuestros mensajes son textos de 440 letras en español. Hay un texto en español en particular, la *Carta a Sor Filotea* de Sor Juana, que tiene justamente 440 letras, llamemos t a ese texto. En este caso tenemos que: $|\mathcal{M}| =$ número de posibles textos en español de 440 letras.

   Supongamos nuevamente que somos el criptoanalista y que capturamos un mensaje que envió A a B. Sabemos que A tomó algún texto en español de 440 letras μ, lo cifró a la César y envió a B el mensaje cifrado γ. Supongamos que A eligió la *Carta a Sor Filotea*, pero nosotros no lo sabemos.

   La probabilidad de que el mensaje sea alguno de los millones de textos posibles en español de 440 letras es:

$$P(\mu) = \frac{1}{|\mathcal{M}|} \approx 0 \tag{4.1.2}$$

   Pero nosotros emprendemos el criptoanálisis, analizamos las frecuencias de las letras en el texto y donde en el criptograma aparecía, digamos `V` sospechamos que debe ir e.

   ¿Cuántos textos en español de 440 letras hay, que tengan la misma letra en todas las posiciones donde γ tiene `V`? Digamos que N_1. Indudablemente $N_1 \ll |\mathcal{M}|$, pues hay muchos más en el conjunto de aquellos que *no* tienen la misma letra en las posiciones donde γ tiene `V`. Hemos restringido nuestra atención a un conjunto que tiene N_1 elementos y hemos descartado un conjunto mucho más grande, con $|\mathcal{M}| - N_1$ elementos. Un progreso notable comparado con lo que sabemos sin hacer nada (expresión 4.1.2). Si pensamos en cuántos textos en español no sólo tienen la misma letra en las posiciones donde γ tiene `V`, sino que además tienen e en esas posiciones, habremos restringido aún más nuestro conjunto de posibilidades y habremos descartado una gran parte del conjunto $\mathcal{M}$. Habrá muchos más textos que tengan e en esas posiciones, que textos que tengan a en ellas.

   Cada vez que deducimos una letra más en el criptograma restringimos aún más los posibles textos de los que proviene. Al final nos quedamos con un texto en particular μ tal que:

$$P_\gamma(\mu) \gg P(\mu) \tag{4.1.3}$$

   En contraste hay otras muchas μ que vamos descartando, para ellas tenemos:

$$P_\gamma(\mu) < P(\mu) \tag{4.1.4}$$

   Estas desigualdades son las que nos permiten descifrar. La igualdad (4.1.1) impide hacerlo.

Esto lo podemos formular formalmente [Beu94].

Definición 4.1 Un sistema de cifrado simétrico $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{F})$ es *perfectamente seguro* si para todo criptograma $\gamma \in \mathcal{C}$ tenemos: $P_\gamma(\mu) = P(\mu)$, para todo texto claro $\mu \in \mathcal{M}$

Pensemos en un texto cifrado cualquiera $\gamma \in \mathcal{C}$ y un texto claro cualquiera $\mu \in \mathcal{M}$. Tenemos $P(\mu) > 0$ porque de hecho $\mathcal{M}$ es el conjunto de todos los posibles textos claros y μ es uno de ellos. Si suponemos que $P_\gamma(\mu) = P(\mu)$ entonces también $P_\gamma(\mu) > 0$. Es decir, la probabilidad de que un criptograma cualquiera provenga de un texto claro dado es mayor que cero. eso significa que hay al menos una transformación tal que:

$$f_k(\mu) = \gamma;$$

en otras palabras, hay al menos una llave que lleva μ a γ. Algo que ya habíamos notado en el cifrado de Vernam. De hecho demostramos el siguiente teorema [Beu94, Sch96].

Teorema 4.1 *Si $\mathcal{S}$ es un sistema perfectamente seguro entonces cualquier texto claro puede ser llevado a cualquier texto cifrado usando alguna clave de $\mathcal{S}$.*

Podemos decir, equivalentemente que dados un criptograma y un texto claro cualquiera, en un sistema perfectamente seguro existe una clave que al usarla para cifrar el texto claro hace que se obtenga el criptograma como resultado.

Ya hemos visto que $|\mathcal{C}| \geq |\mathcal{M}|$ en cualquier sistema simétrico. Si el sistema es perfectamente seguro entonces es posible encontrar una transformación que lleve cualquier texto claro posible a cualquier criptograma posible. Además sabemos que una transformación no puede llevar un mismo texto claro a dos diferentes criptogramas, asi que hay al menos tantas transformaciones (llaves de hecho) como criptogramas posibles: $|\mathcal{F}| \geq |\mathcal{C}|$ Si el sistema es perfectamente seguro entonces:

$$|\mathcal{F}| \geq |\mathcal{C}| \geq |\mathcal{M}|.$$

Estas son condiciones necesarias para que un sistema sea perfectamente seguro. Investiguemos ahora las condiciones suficientes.

Sabemos que cuando $P_\gamma(\mu) = P(\mu)$ tenemos un sistema perfectamente seguro. También sabemos que $P_\gamma(\mu) = P(\mu \mid \gamma)$. Ésta es una probabilidad condicional que podemos reescribir de acuerdo al teorema de Bayes:

$$P(\mu \mid \gamma) = \frac{P(\gamma \mid \mu)\ P(\mu)}{\sum_{\mathcal{M}}(P(\gamma \mid \mu_i)\ P(\mu_i))},$$

donde $P(\mu) = \frac{1}{|\mathcal{M}|}$. Así que:

$$P(\mu \mid \gamma) = \frac{P(\gamma \mid \mu)\ \frac{1}{|\mathcal{M}|}}{\frac{1}{|\mathcal{M}|}\sum_{\mathcal{M}} P(\gamma \mid \mu_i)} = \frac{P(\gamma \mid \mu)}{\sum_{\mathcal{M}} P(\gamma \mid \mu_i)}.$$

También sabemos que:

$$\sum_{\mathcal{M}} P(\gamma \mid \mu_i) = 1$$

porque la suma representa la probabilidad de que se obtenga γ como criptograma, a partir de alguna μ y eso es seguro, sabemos que existe alguna clave que cifra μ como γ.

Así que, en síntesis: $P(\mu \mid \gamma) = P(\gamma \mid \mu)$, lo que podemos reescribir, junto con la condición suficiente $P_\gamma(\mu) = P(\mu)$, de la siguiente manera:

$$P_\mu(\gamma) = P_\gamma(\mu) = P(\mu).$$

Resumimos nuestras condiciones suficientes en el siguiente Teorema:

Teorema 4.2 *Si $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{F})$ es un sistema criptográfico simétrico en el que:*

- *Todas las claves son equiprobables;*
- *Para toda pareja (μ, γ), con $\mu \in \mathcal{M}$ y $\gamma \in \mathcal{C}$, existe $f_k \in \mathcal{F}$ tal que $f_k(\mu) = \gamma$; y*
- $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{F}|$;

entonces $\mathcal{S}$ es perfectamente seguro.

En el cifrado de Vernam los tres conjuntos son el mismo. El cifrado de Vernam, también conocido en la literatura como *one-time pad* o *"llaves de un solo uso"*, es entonces perfectamente seguro.

4.1.4 Registros de desplazamiento con retroalimentación lineal

Ya mencionamos entre las limitaciones prácticas del sistema de Vernam que la clave debe ser tan larga como el mensaje, aleatoria, y debe usarse una sola vez. Mencionamos también el hecho de que transportarla, una vez que ha sido generada por alguna de las partes, implica un debilitamiento de la seguridad. Una alternativa es que ambas partes sean capaces de generar una secuencia pseudoaleatoria muy larga como clave.

Con esto en mente se diseñaron unos dispositivos físicos capaces de generar secuencias aparentemente aleatorias de bits llamados: registros de desplazamiento con retroalimentación lineal (LFSR o *Linear Feedback Shift Registers* en inglés) [Sch96]. En la Figura 4.2

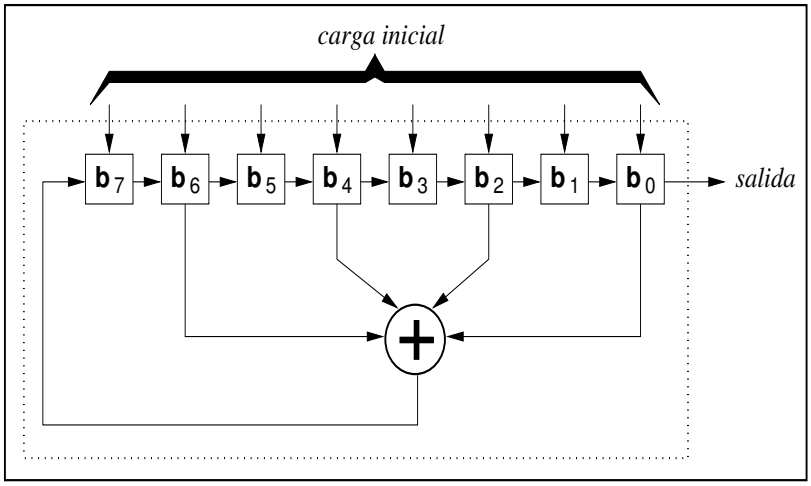

Figura 4.2: Registro de desplazamiento con retroalimentación lineal (LFSR) de ocho bits. El registro es inicializado con 8 bits cualesquiera. En cada paso subsecuente el registro es desplazado un lugar a la derecha, el bit del extremo derecho sale del registro para ser usado como bit de la secuencia aleatoria, el nuevo bit del extremo izquierdo es el resultado de la función $\oplus$ (XOR) sobre los valores de los bits 0, 2, 4 y 6 del paso inmediato anterior.

se muestra esquemáticamente uno de estos dispositivos. El símbolo $\oplus$ denota nuestra conocida función booleana XOR o suma módulo 2; se puede pensar en registros más generales en los que la función $\oplus$ es reemplazada por alguna otra diferente, pero igualmente lineal. Cuando esta función es tan compleja que deja de ser una función lineal, el resultado ya no es un LFSR, por supuesto; en este caso el dispositivo suele llamarse NLSFR por su nombre en inglés (*Non-Linear Feedback Shift Register*). En la figura 4.2 el XOR central recibe sólo los bits 0, 2, 4 y 6 del registro para calcular el nuevo valor de b_7, pero cuales y cuantos bits entran al XOR es arbitrario, y se puede fijar de distintas maneras. Algunas combinaciones son mejores que otras.

Pongamos un ejemplo con un registro de 4 bits. Inicializamos el registro con puros unos, y en cada paso el nuevo bit más significativo es el resultado de un XOR con los bits mas y menos significativos del registro en el paso inmediato anterior. Numeramos los bits de izquierda a derecha, empezando en 0. La columna $b_0 \oplus b_3$ nos dice cuál será el bit más significativo en el siguiente estado:

Paso	Estado del registro	Bit de salida	$b_0 \oplus b_3$	Paso	Estado del registro	Bit de salida	$b_0 \oplus b_3$
1	1111	**1**	0	9	1001	**1**	0
2	0111	**1**	1	10	0100	**0**	0
3	1011	**1**	0	11	0010	**0**	0
4	0101	**1**	1	12	0001	**1**	1
5	1010	**0**	1	13	1000	**0**	1
6	1101	**1**	0	14	1100	**0**	1
7	0110	**0**	0	15	1110	**0**	1
8	0011	**1**	1	16	1111	**1**	0

El estado en el último paso mostrado en la tabla es igual al estado en el primer paso de la tabla, de manera que el ciclo de estados de los registros se repite. El bit menos significativo de los valores de la tabla constituye la secuencia de valores entregados por el registro. Esto significa que hay un periodo de 15 bits entre repeticiones de la secuencia, muy corta para ser útil como clave de un sistema de Vernam. De hecho el tamaño máximo del ciclo (el periodo) de un registro lineal de n bits es $2^n - 1$, que se obtiene después de agotar todas las posibles combinaciones de n bits excepto por la que tiene puros ceros (que siempre produce la misma secuencia en el siguiente paso). Algo que notamos con nuestro ejemplo, es que no importa el valor inicial del registro, pues el valor inicial sólo determina donde empezamos el ciclo.

Tenemos pues que lo mejor que podemos tener es que el tamaño del ciclo de un registro de n bits sea $2^n - 1$, pero dependiendo de que bits sean conectados al XOR, el ciclo puede ser menor; surge entonces la siguiente pregunta: ¿Cuáles y cuántos bits del registro hay que conectar para obtener el máximo ciclo?

La clave de la respuesta a la pregunta consiste en considerar los bits del registro como los coeficientes de un polinomio de grado menor o igual a n. Si por ejemplo tenemos conectados los bits de índice 0 y 3, como en el ejemplo de nuestra tabla, diremos que el polinomio es $x^4 + x + 1$; el 1 es añadido y el resto del polinomio indica que bits intervienen en el XOR en nuestro ejemplo son el primero y el cuarto, por lo que se se pone coeficiente 1 a la potencia 1 y 4. Si consideramos el ejemplo de la figura 4.2, dado que se conectan el primero, tercero, quinto y séptimo bit al XOR, el polinomio correspondiente a esa conexión es: $x^7 + x^5 + x^3 + x + 1$.

Para que el ciclo del LFSR sea máximo debe ocurrir que el polinomio asociado con las conexiones al XOR sea:

- Irreducible sobre $\mathbb{Z}_2[x]$; esto es, que no sea posible factorizarlo como el producto de polinomios de grado menor, módulo 2; y

- Primitivo; un polinomio primitivo es aquel para el que el entero positivo más pequeño m, tal que el polinomio divide a $x^m - 1$ es $m = 2^n - 1$.

El término *primitivo* se usa también en Teoría de Números algebráica para referirse a un polinomio $a_n x^n + \cdots + a_0 \in \mathbb{Z}[x]$ tal que $\text{mcd}(a_0, \ldots, a_n) = 1$. Es quizás desarotunado que el mismo término se use para dos nociones distintas, pero ambas relacionadas con polinomios. Cuando hablemos de polinomios con coeficientes en $\mathbb{Z}_2$, siempre nos referimos a la definición que aparece arriba. En otras ocasiones, especificaremos el significado para evitar confusiones.

En general no es sencillo encontrar polinomios con estas características, pero se conocen muchos de ellos, como los mostrados en la página 376 de [Sch96]. Por ejemplo $x^8 + x^4 + x^3 + x^2 + 1$ es un polinomio de grado 8 que si se utiliza en un LSFR hace que éste genere una secuencia de 255 bits pseudoaleatorios. Se suele elegir un polinomio al azar y hacer pruebas que, de ser pasadas por el polinomio propuesto, proporcionan un cierto grado de certidumbre de que el polinomio en cuestión es primitivo. Nunca se puede estar 100% seguro con estas pruebas, pero se puede estar arbitrariamente seguro, como con las pruebas de primalidad que mencionaremos más adelante en criptografía de llave pública.

Hasta ahora todo se luce más o menos bien. Podemos pensar en utilizar un par de registros LSFR idénticos, colocar uno en el receptor y otro en el emisor y utilizar su salida como la clave para usar un cifrado de llave de un solo uso y así tener seguridad perfecta.

Pero en 1969 se descubrió un algoritmo que fue usado en 1972 para demostrar que, dada una secuencia pseudoaleatoria generada por un LSFR, es posible recuperar el polinomio asociado con la conexión del XOR (que se le suele llamar el *polinomio de alambrado*). El algoritmo es el *algoritmo de Berlekamp-Massey* [Sch96, MvOV96]; es un algoritmo de complejidad $O(n^2)$, donde n es el tamaño, en bits, de la secuencia; para un LSFR de k bits se requiere tener, al menos, una muestra de $n = 2k$ bits de su ciclo completo.

El algoritmo de Berlekamp-Massey calcula la *complejidad lineal* de la secuencia (en esencia la longitud en bits del LSFR que la genera). Es un algoritmo de aproximaciones sucesivas que calcula la complejidad lineal hasta el bit i-ésimo de la secuencia y con esto el polinomio que puede generar hasta ese bit. Si en algún momento el polinomio hallado no es capaz de generar el $(i + 1)$-ésimo bit entonces se calcula un nuevo polinomio en función del anterior.

Es posible entonces usar el algoritmo de Berlekamp-Massey para montar un ataque de texto conocido. Si se tiene un fragmento suficientemente largo (el doble del tamaño del registro) de texto claro y se posee el criptograma correspondiente, entonces se puede usar el algoritmo para recuperar el alambrado del LFSR y con eso obtener la clave completa, lo que rompe completamente la seguridad del sistema.

Se ha propuesto el uso de registros en los que la función que determina el nuevo bit de la izquierda sea no lineal, involucrando el producto de algunos bits y la suma de otros y algunas constantes, por ejemplo. Pero también se han montado ataques basados en correlaciones estadísticas entre los bits producidos. Además se sabe poco de este tipo de registros, suficientemente poco como para no saber que tan seguros son, de manera que su uso no es recomendado.

4.2 ENIGMA

Esta sección está basada en la presentación que se encuentra en [Kah99, Mil96, Wil01].

4.2.1 Antecedentes históricos

En agosto de 1914, el crucero alemán MAGDEBURG se encalló en el Golfo de Finlandia. El capitán quemó tres de las cuatro copias del código[2] naval alemán, pero olvidó la copia que estaba en su cabina. Esta copia fue capturada por soldados rusos; el gobierno de Rusia, tras hacer una copia para ellos, entregó el original a Gran Bretaña. Con él, la Marina Británica bajo el mando de Winston Churchill, entonces Primer Lord del Almirantazgo, logró detener todos los intentos de Alemania de utilizar su marina en el Mar del Norte durante la Primera Guerra Mundial.

En 1923, Winston Churchill reveló la historia del MAGDENBURG con su inimitable estilo en la Cámara de los Comunes. Los alemanes de pronto entendieron por qué todas sus maniobras marítimas habían fracasado, y decidieron que necesitaban un sistema criptográfico que no sufriera de la principal debilidad de los códigos: la captura de cualquier copia compromete toda la edición (se trata simplemente de la tercer máxima de Kerckhoffs).

Afortunadamente, la idea básica necesaria para una máquina criptográfica moderna ya había surgido independientemente en tres lugares, con tres inventores: Edward Hugh Hebern, en Estados Unidos en 1917; Hugo Alexander Koch, en Holanda en 1919; y Arvid Gerhard Damm, en Suecia, en 1919 (de hecho, las patentes de Koch y Damm se solicitaron durante la misma semana de 1919).

Ese concepto es el concepto de un *rotor*.

El *rotor* es un disco de material aislante. En cada cara hay 26 contactos eléctricos. Cada contacto está conectado aleatoriamente a un contacto de la otra cara. Los contactos

[2] Vésae el Capítulo chotroshitoricos, Sección 5.1 para una breve discusión sobre códigos.

representan una letra, de manera que el rotor es una sustitución monoalfabética. La idea es poner al rotor entre dos placas fijas de material aislante, cada una con 26 contactos. De un lado, lo conectamos a un teclado que representa el texto original. El otro lado se conecta a un aparato que produzca el texto cifrado: un teletipo, o un teclado con lámparas que se vayan iluminando. Cuando oprimimos una letra en el texto original, la corriente pasa por el rotor y se produce otra letra en el texto cifrado.

Si esto fuera todo, la invención no sería importante. Pero el rotor no se queda estacionario mientras vamos tecleando el mensaje: después de cada paso, rota, poniendo otra sustitución monoalfabética en juego. Esto resulta en una serie de 26 monoalfabetos relacionados (no exactamente tipo Alberti, pero puesto que el alambrado dentro del rotor es fijo, hay relaciones entre las sustituciones).

Esto ya lo vuelve más interesante, pero todavía no tenemos un sistema demasiado importante. Pero si ahora ponemos *otro* rotor entre el primero y la salida, estamos componiendo dos sustituciones monoalfabéticas. Si hacemos que el segundo rotor se mueva un lugar sólo después d que el primero dio toda la vuelta, en vez de un sistema polialfabético con periodo 26, tenemos un sistema con periodo $26 \times 26 = 676$; y los alfabetos, aunque relacionados, están relacionados de una manera mucho más complicada. Un tercer rotor da 17576 alfabetos, y un cuarto y quinto dan $456,975$ y $11,881,376$ alfabetos, respectivamente. Es fácil entonces construir un sistema donde el periodo es suficientemente largo para escribir las obras completas de Shakespeare, *La Guerra y la Paz* de Tolstoi, *Don Quijote*, *La Iliada*, *La Odisea*, y *El Paraíso Perdido* de Milton, uno detrás del otro, sin repetir ningún alfabeto. Eso evitaría el análisis de frecuencia, que requiere entre 20 y 50 letras por alfabeto utilizado.

Ninguno de los tres inventores obtuvo mucho beneficio económico de sus inventos. Koch le vendió sus patentes a un alemáan en 1923, y murió en 1924. El alemán, Arthur Scherbius, lo utilizó para construir varios modelos de una máquina cifradora a la que llamo ENIGMA.

En 1926, la Kriegsmarine (Marina de Guerra) alemana compró ENIGMAS. Los ENIGMAS se convirtieron en el principal aparato criptográfico de alto nivel para el gobierno y fuerzas armadas alemanes hasta 1945.

4.2.2 Descripción del ENIGMA

El ENIGMA básico de 1926 venía en tres modelos, A, B, y C; la diferencia era el tamaño. En cada caso, tenía un teclado tipo máquina de escribir, y un "teclado" hecho de focos que se prendían indicando el resultado. Tenía tres rotores y un reflector. El reflector es como un rotor, pero sólo tiene contactos de un lado, y regresa la corriente por los rotores en la dirección inversa.

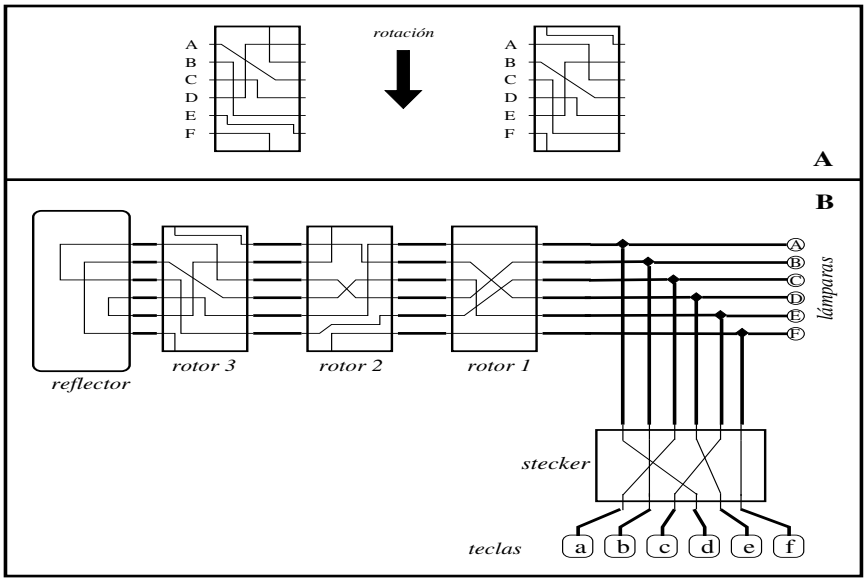

Figura 4.3: Estructura esquemática de ENIGMA. En (A) se muestra un rotor en una posición dada y luego de rotar un lugar. En (B) se muestra la relación entre los rotores, el reflector, el teclado y las lámparas de la máquina.

El uso del reflector tiene como resultado que los alfabetos resultantes sean todos recíprocos. Es decir, si $e \mapsto \texttt{X}$, entonces $x \mapsto \texttt{E}$. De esa manera, uno puede usar la máquina tal cual para cifrar, y para descifrar. En la Figura 4.3 se muestra, esquemáticamente, la estructura general de ENIGMA.

Originalmente la rotación de los rotores era tipo odómetro: el rotor de la derecha se movía cada paso; el rotor central cuando el rotor de la derecha terminaba una vuelta; el rotor de la izquierda cuando el rotor central terminaba una vuelta. Pero debido a la relación entre los alfabetos, en vez de eso Scherbius puso un sistema de engranes y palancas para que el movimiento fuera más irregular.

El orden de los rotores, y su posición original, eran la llave.

El mecanismo de rotación funcionaba gracias al movimiento de oprimir una tecla; eso quiere decir que el rotor derecho se movía *antes* de cifrar la letra correspondiente.

Cada rotor tenía un anillo con 26 ranuras del lado derecho; el lado izquierdo tenía un anillo movible que tenía una sóla ranura. Para mover los otros rotores, había una palanca en forma de 'T' entre los rotores vecinos, que se movía hacia adelante y atrás en cada teclazo. Cuando el rotor de la derecha de la palanca llega a la posición en la cuál tiene la ranura en el

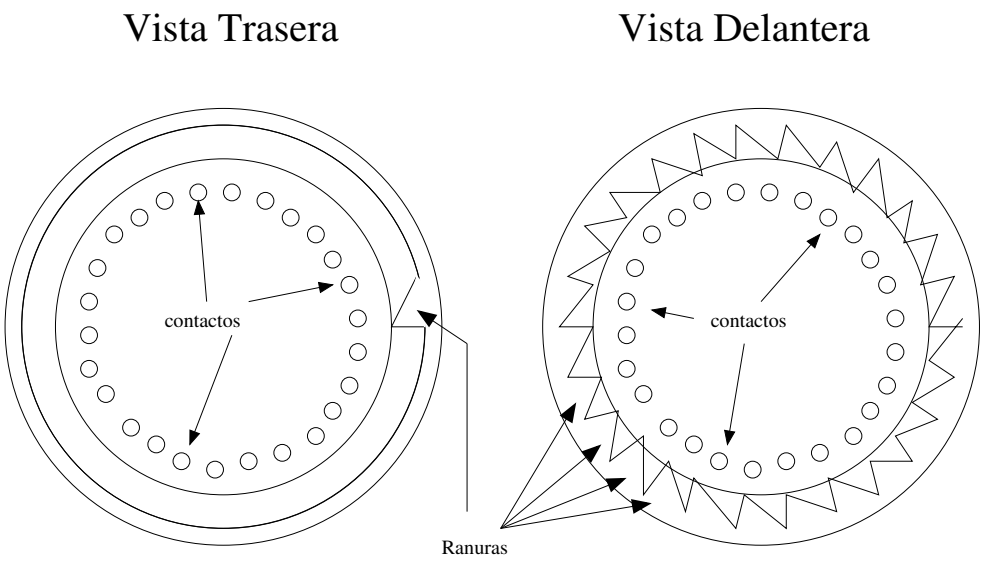

Figura 4.4: Diagrama esquemático de la cara delantera y trasera de un rotor de ENIGMA.

anillo de su lado izquierdo, la porción de la palanca `T` de ese lado entra al hueco. Entonces la porción izquierda de la `T` entra en contacto con una de las 26 ranuras del rotor a su izquierda. En el siguiente teclazo, la palanca se mueve hacia adelante, empujando los dos rotores. Cuando el rotor a la derecha no está en la posición que corresponde a la ranura, esto evita que la palanca entre, y entonces tampoco entra en los huecos del rotor de la izquierda, que se mantiene inmóvil. Un diagrama esquemático se muestra en la Figura 4.4.

El efecto de esta palanca es que entre los rotores derechos y centrales, el mecanismo funciona simplemente como un odómetro; pero el efecto entre los rotores centrales e izquierdo es un poco sorpresivo. Supongamos, por ejemplo, que el rotor de central está a una posición de que la ranura de su lado izquierdo se ajuste a la palanca, y el rotor derecho está en la posición de ajuste. Cuando oprimimos una tecla, la palanca entre el rotor derecho y central mueve tanto el rotor derecho como el rotor central, que es lo que esperamos que suceda. En ese momento, se cifra la letra. Ahora, el rotor derecho está en la posición de ajuste, de manera que la palanca entre el rotor central y el derecho está en posición de mover ambos. Al siguiente teclazo, tanto el rotor izquierdo como el central se mueven, por el movimiento de la palanca. El rotor derecho se mueve en cada teclazo, de manera que todos los rotores se mueven en ese momento. Eso quiere decir que el rotor central se ha movido dos veces en pasos sucesivos, lo cuál es muy distinto al funcionamiento de un odómetro.

La anomalía existe entre los rotores central e izquierdo, y también entre el derecho y central; pero como el rotor derecho se mueve en cada teclazo, no se observa ninguna

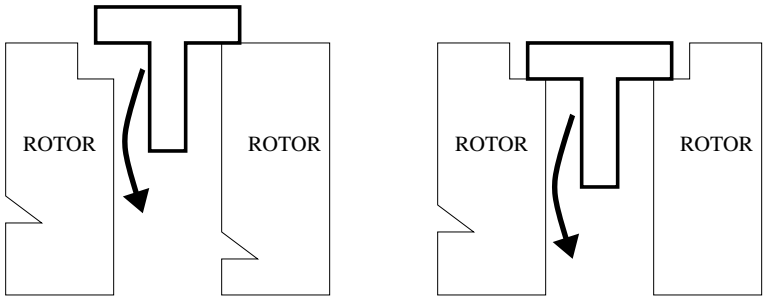

Figura 4.5: Diagrama esquemático de la acción de la palanca `T`. En el diagrama (a), la ranura del rotor de la derecha no se encuentra alineada con la palanca, de manera que al mover la palanca, el rotor de la izquierda no se mueve. En el diagrama (b), la ranura del rotor de la derecha está alineada. Cuando la palanca se mueva, se moverán ambos rotores. Nótese que la ranura del lado derecho de un rotor siempre está alineada con la palanca, pues hay una ranura en cada posición.

diferencia en el movimiento del mecanismo.

Después, la Kriegsmarine agregó una segunda ranura en el anillo movible de los rotores, en posición opuesta a la primera ranura. Esto hacía que el movimiento fuera más irregular, y que los rotores centrales e izquierdos se movieran más frecuentemente.

La operación estándar del Enigma requería de tres operadores: uno leía el mensaje (ya sea por cifrar, o cifrado) y lo tecleaba en la máquina. Una dictaba las letras que se encendían en el tablero; y el tercero escribía lo dictado.

Originalmente, el orden de los rotores se cambiaba cada tres meses. Luego cada mes, cada semana, y cada día. Hacia final de la guerra, había un cambio cada seis horas.

En general, la Kriegsmarine fue quien mejor protocolo y mejor sistema de cambio de llaves tenía, así como los operadores mejores entrenados. Le seguía la Wehrmacht (ejército), y finalmente, al Luftwaffe (fuerza aérea), que tenía muy mal manejo de sus sistemas criptográficos.

En 1938, la Kriegsmarine agregó un cuarto rotor, y luego un quinto. Sin tomar en cuenta la posición inicial de los rotores, el número de posibilidades subió de 6 (tres rotores, escoger tres, ponerlos en algún orden), a 60 (cinco rotores, escoger 3, ponerlos en algún orden). En 1941, agregó la segunda ranura a los anillos movibles de los rotores. En 1942, introdujo al mecanismo una cuarta posición para un rotor delgado entre el rotor izquierdo y el reflector. Había dos rotores, llamados α y β, que se podían poner en posición neutral para mandar y recibir mensajes compatibles con el Enigma de tres rotores.

La Kriegsmarine también agregó en 1940 un *stecker*; consistía de 26 enchufes, que eran conectados con alambres; esto producía una permutación autoinversa de la letras antes de pasar la corriente a los rotores. Originalmente, se usaban de 6 a 11 cables (la permutación tenía entonces entre 6 y 11 biciclos); en 1941, se estandarizó a 10 cables.

4.2.3 Complejidad combinatoria del Enigma

Los datos de ésta sección se obtuvieron de [Mil96].

Las componentes variables del Enigma son las siguiente:

1. El *stecker*, con 0 a 13 cables.
2. Tres rotores ordenados (de derecha a izquierda).
3. El alambrado de cada rotor.
4. La posición inicial de cada rotor.
5. La posición del anillo que indica la rotación del rotor a la izquierda.
6. El cableado del reflector.

Vamos calculando la complejidad de cada componente.

1. El *stecker*. Tenemos p cables, $0 \leq p \leq 13$. El primer cable tiene $\binom{26}{2} = \frac{26\times 25}{2}$ posibilidades. El siguiente tiene $\binom{24}{2}$, etc., y no importa el orden en que los elegimos. Entonces, dado el número p de cables, hay

$$\frac{26!}{(26-2p)! \times p! \times 2^p}$$

posibilidades. Dependiendo del número de cables, los valores son:

p	combinaciones	p	combinaciones
0	1	7	1,305,093,289,500
1	325	8	10,767,019,638,375
2	44,850	9	53,835,098,191,875
3	3,453,450	10	150,738,274,937,250
4	164,038,875	11	205,552,193,096,250
5	5,019,589,575	12	102,776,096,548,125
6	100,391,791,500	13	7,905,853,580,625

El número posible de posiciones del *stecker* es entonces la suma sobre los valores de $p = 0, 1, \ldots, 13$. El resultado es:

$$\sum_{p=0}^{13} \frac{26!}{(26-2p)! \times p! \times 2^p} = 532,985,208,200,576.$$

2 y 3. Los rotores. Tenemos, originalmente, tres rotores puestos en un orden específico. Hay 26! posibles rotores (uno por cada posible sustitución monoalfabética); cualquiera de ellos puede estar en la posición derecha; cualquiera de los $26! - 1$ restantes en la central, y cualquiera de los $26! - 2$ restantes en la posición izquierda. Puesto que el orden importan, el total de posibilidades es $(26!)(26! - 1)(26! - 2)$, es decir,

$$65,592,937,459,144,468,297,405,473,480,371,753,615,896,$$
$$841,298,988,710,328,553,805,190,043,271,168,000,000.$$

4. La siguiente variable es la posición inicial de los rotores. Cada uno de los tres rotores puede estar en cualquiera de la 26 posiciones iniciales originales, lo cual nos dá un total de $26^3 = 17,576$ posibilidades.

5. La siguiente variable era el anillo movible que indica el momento de rotación del rotor a su izquierda. La posición del anillo en el rotor de hasta la izquierda no importa, pues no afecta la operación del Enigma. Eso nos deja 26 posibles posiciones para cada uno de los rotores derecho y central, lo cual nos da $26^2 = 676$ combinaciones posibles.

6. La última variable es el reflector. El reflector tiene 26 contactos, como los rotores, pero sólo en una de sus caras. Tiene trece cables que conectan internamente los 26 contactos, en una serie de trece parejas. Esto representa, matemáticamente, la misma situación que teníamos cuando buscábamos calcular el número de maneras de conectar trece cables en el *stecker*; esto nos da $7,905,853,580,625$ posibilidades.

Ahora tenemos todo lo que necesitamos para calcular la complejidad teórica del Enigma. Puesto que el cálculo en cada inciso arriba es independiente, tenemos que multiplicar todos los números resultantes. El resultado es:

$$3\,,283,883,513,796,974,198,700,882,069,882,752,878,379,955,261,095,623,685,444,$$
$$055,315,226,006,433,615,627,409,666,933,182,371,154,802,769,920,000,000,000$$

o aproximadamente 3×10^{114}.

4.2.4 Complejidad práctica del Enigma

En realidad, el número de arriba es una sobre-estimación sobre consideraciones prácticas; la fabricación de los rotores no permite tener 26! rotores disponibles, por ejemplo.

En la práctica, el número de cables en el *stecker* estaba fijo con $p = 10$, lo cual nos da un valor de

$$150,738,274,937,250$$

posibilidades para la posición del *stecker*. Para la selección de rotores y su orden había cinco posibles rotores, a escoger tres en un orden específico (60 posibilidades); suponemos que el cableado de los rotores era conocido. La posición inicial de los rotores continúa libre, dando $17,576$ posibilidades; la posición de los anillos también sigue igual, lo cual nos da 676 posibilidades; y si asumimos que el cableado del reflector es fijo y conocido, en vez de casi ocho billones de posibilidades tenemos únicamente una.

Entonces, la típica complejidad a la que se enfrentaba un criptoanalista estudiando los mensajes del Enigma es el producto de estas cantidades. Esto es "simplemente"

$$107,458,687,327,250,619,360,000$$

o aproximadamente 10^{23}. Aunque el número es mucho más pequeño que el teórico, aún representa un número impresionantemente grande, y sería imposible montar un ataque de fuerza bruta.

Estas consideraciones convencieron a los altos mandos alemanes que la seguridad que daba el Enigma era mas que suficiente.

4.2.5 Rejewski, Turing, y Bletchley Park: el criptoanálisis de Enigma y la Batalla del Atlántico

En 1920, Rusia invadió Polonia en la guerra Ruso-Polaca. El ejército polaco creó una sección criptoanalítica con la que Polonia bloqué el avance ruso antes de que llegara a Varsovia. Cuando los rusos retrocedieron, el Biuro Szyfrów (Buró de Cifrado) empezó a concentrarse en Alemania. Tuvieron mucho éxito hasta que, en 1928, empezaron a interceptar mensajes con un carácter criptográfico completamente distinto al que las fuerzas armadas alemanas habían usado hasta entonces. Usando espías y análisis, descubrieron que se trataba de un sistema tipo Enigma (Scherbius había puesto a la venta al público los modelos A, B, y C del Enigma; estos tenían un sistema de rotación mucho más simple que el de las fuerzas armadas, carecían de reflector y *stecker*, y el cableado de los rotores era totalmente distinto al de los Enigmas de las fuerzas armadas).

El jefe del Biuro Szifrów se dió cuenta que el aumento de volumen de comunicaciones estaba mecanizando a la criptografía, y que los códigos estaban siendo totalmente abandonados. Compró varias versiones comerciales del ENIGMA para saber cómo funcionaban en términos generales, y contrató a veinte jóvenes matemáticos de la Universidad de Poznán. Hasta ese entonces, los burós de criptoanálisis solían tener filólogos y traductores, no matemáticos. La mayoría de estos jóvenes se salió del Biuro después de poco timepo, pero tres de los matemáticos se quedaron: Marian Rejewski, Henryk Zygalski, y Jerzy Rózycki. Cuando terminaron la carrera (en el caso de Rejewski, una maestría en Göttingen), se incorporaron al Biuro Szygrów en Varsovia tiempo completo, y empezaron a atacar el ENIGMA.

La Kriegsmarine había hecho demasiados cambios a la máquina comercial antes de que fuera adoptada por las fuerzas armadas. De manera que las máquinas comerciales sólo les sirvieron para conocer el mecanismo general de cifrado. Con mucho volumen y mucho trabajo, en quizás uno de los logros criptoanalíticos más impresionantes del siglo XX, Rejewski logró determinar una ecuación matemática que le permitiría descubrir el cableado de los rotores. Pero la ecuación tenía demasiadas incógnitas, y no pudo avanzar más.

En 1932, Hans-Thilo Schmidt, de 44 años, un empleado del Buró de Cifrados de la Wehrmacht, quería más dinero y le ofreció a un agente francés el manual de operaciones del ENIGMA. Francia los compró, pero como el manual no tenía información sobre la llave (las conexiones del *stecker*, la posición de los rotores, el cableado de los rotores), los criptoanalistas franceses no lograron gran cosa. Los franceses le pasaron una copia del manual al Jefe de Inteligencia de Radio del ejército polaco, bajo un tratado de ayuda militar mutua firmado en 1921. El Buró de Inteligencia de Radio no se lo proporcionó al Biuro Szyfrów hasta que su propia gente no logró hacer nada con el. Pero el material que Schmidt les había proporcionado incluía las instrucciones de uso.

Además de la llave del día, cada mensaje tenía que ser enviado con una posición inicial aleatoria de los rotores. El remitente elegía una llave, digamos `PDQ`. Utilizando la llave del día, enviaba `PDQ`, ponía los rotores en esa posición, y entonces enviaba el mensaje. Para evitar problemas por errores de transmisión, la llave del mensaje se repetía al enviarla: `PDQPDQ`. Esto producía seis letras, digamos `MKFXRC`, llamadas el *indicador del mensaje.* Se transmitían, y el destinatario usaba la llave del día para descifrar el indicador, obtener `PDQPDQ`, y luego usaba la posición de los rotores `PDQ` para descifrar el mensaje.

La repetición aumentaba la confiabilidad, pero era en un grave error de seguridad. Representaba un pequeño compromiso criptotexto-criptotexto (es decir, dos criptotextos que sabemos corresponden al mismo texto original). La `M` y la `X`, la `K` y la `R`, la `F` y la `C` corresponden, cada pareja, a la misma letra. Si otro indicador del mismo día fuera `MRAXTT`,

entonces sabemos que el rotor de la izquierda en ambos mensajes está en la misma posición.

Con esta información y 60 mensajes, Rejewski muchas veces lograba encontrar suficientes correspondencias; con estas correspondencias, reducía el problema del cableado de los rotores a seis ecuaciones para el rotor derecho; pero todavía tenía demasiadas incógnitas.

Schmidt siguió vendiéndole información a los franceses. Les vendió las llaves que habían sido utilizadas en agosto y septiembre de 1932. Con ellas, Rejewski redujo las ecuaciones, pero aún no lograba resolverlas. De repente, tuvo una inspiración: hasta entonces, había asumido que el cableado del teclado a los rotores iba de la primera letra del teclado, `Q`, al primer contacto del rotor (correspondiente a la `A`). ¿Qué tal si iba de `Q` a `Q`, `B` a `B`, etc.? Ajustó las ecuaciones, y de pronto todo cobró sentido y obtuvo la solución. Con ellas, Rejewski obtuvo el cableado del rotor derecho. Con el volumen de mensajes acumulado, utilizó la misma técnica para deducir el cableado de todos los rotores. Hacia finales de diciembre de 1932, entregó las primeras soluciones de ENIGMA al jefe del Biuro Szifrów.

Por supuesto, esto no terminaba con el problema. Rejewski había, de hecho, "capturado" el sistema al encontrar los cableados de los rotores y del reflector. Pero la complejidad del sistema, aún con estos datos conocidos, seguía siendo demasiado grande. El problema era recuperar las llaves del día de manera rápida, y eso aún no era resuelto.

Las reglas de operación alemanas prohibían el uso de palabras como indicadores, o claves como `ABC` ó `QWE`. La Kriegsmarine seguía las reglas, pero la Wehrmacht y la Luftwaffe eran mucho más laxas. Un oficial de la Luftwaffe utilizaba mucho los indicadores `CIL` y `LIE`, pues *Cillie* era el nombre de su novia. Con eso, el Biuro Szyfrów encontraba la llave del día. A tal grado fue esto que los criptoanalistas polacos le llamaban "cillies" a los indicadores fáciles de adivinar. Pero por supuesto, no siempre venían los mensajes con cillies.

Rejewski y Zygalski usaron otra curiosidad técnica del ENIGMA: en ninguna posición el ENIGMA cifra una letra a sí misma. Con este detalle, criptoanálisis puro, y el compromiso criptotexto-criptotexto proporcionado por el indicador, Zygalski desarrolló un método que utilizaba una especie de tarjeta perforada que permitían "rápidamente" calcular el indicador, o al menos las dos primeras letras en la mayoría de los casos (normalmente, ninguno de los rotores centrales o izquierdos se movían durante el envío del indicador, lo cual hacía esto posible). Después de esto, había sólo que probar unas cuantas posibilidades. Rejewski diseñó una máquina que podía buscar las aproximadamente $17,000$ combinaciones en un par de horas. La máquina parecía seis enigmas interconectados, cada uno con los rotores un un orden específico; e iban probando las distintas posibilidades que dieran lugar al indicador. La máquina se llamaba *Bomba*.

Hay varias versiones sobre de dónde viene el nombre *Bomba*. La palabra es una palabra polaca que quiere decir bomba; el Coronel Tadeusz Lisicki dijo alguna vez que Jerzy Rozycki

le puso el nombre por un postre de helado que estaban comiendo cuando A Rejewski se le ocurrió la idea. El postre era una bola de helado cubierta de chocolate, que parecía una bomba de caricatura. Por otro lado, Rejewski dijo que era una *bomba* por falta de otro nombre mejor; un documento del ejército americano decía que al llegar a una solución, la máquina producía un ruido mecánico como de una explosión, y por eso el nombre.

El hecho es que las *bombas* funcionaban probando la posición del rotor izquierdo, descartando aquellas que no permitían una conexión que hiciera que la primer y cuarta letra del indicador correspondieran a la misma letra; una vez encontradas las posiciones posibles, se probaban las del rotor central; y finalmente las del rotor derecho. Es decir, movían primero el rotor más lento, y movía al final el rotor más rápido.

En 1938, los alemanes agregaron dos nuevos rotores. Usando sus ecuaciones, Rejewski rápidamente pudo determinar sus cableados, pero las *bombas* ya no funcionaban: en vez de seis ENIGMAS interconectados, se necesitarían 60 (una para cada posible elección y orden de rotores). Regresaron entonces al viejo sistema de usar "hojas de Zygalski" (las tarjetas perforadas) y tratar de adivinar indicadores.

El 24 de julio de 1939, temiendo una invasión en cualquier momento, los polacos le ofrecieron sus resultados a Francia e Inglaterra, pues el volumen de tráfico era demasiado grande para el pequeño Biuro Szyfrów. Al principio, Francia e Inglaterra no les creyeron que habían logrado descifrar el ENIGMA. Después de ver los resultados, quedaron fascinados y se llevaron todo el material bajo sello diplomático. Polonia fue invadida; la buena inteligencia no bastó contra las huestes alemanas. Francia cayó unos meses después, e Inglaterra quedó sola en Europa.

En septiembre de 1939, el Government Code and Cipher School (GC & CS) del gobierno británico, se mudó de Londres a una mansión victoriana en Bletchley Park. Contrató a matemáticos para continuar el trabajo de Rejewski y los criptoanalistas polacos. Entre ellos estaban Alan Turing y Gordon Welchman, que fueron encargados con el trabajo de encontrar cómo resolver los mensajes de ENIGMA más rápidamente.

Los alemanes iniciaron una serie de cambios al manual de operaciones y al uso de los indicadores, buscando fortalecer la seguridad del ENIGMA. Cada cambio requería de nuevas hojas de Zygalski, que eran cortadas a mano y eran tardadas de producir. Y las *bombas* polacas ya no eran suficientes. Turing diseñó una máquina que trabajara no sobre el indicador, sino sobre una palabra probable. Entre más larga, mejor funcionaba la máquina. Ésta también funcionaba a base de probar todas las posibles posiciones de rotores que eran compatibles con la palabra. Las máquinas las llamaron *bombes*, y la primera fue instalada el 18 de marzo de 1940. Al principio, se concentraron en la Luftwaffe, y para mayo ya estaban leyendo casi todo el material con 6 a 48 horas de retraso.

Pero lo más importante no era la Luftwaffe, sino la Batalla del Atlántico, y el ENIGMA de la Kriegsmarine. La Kriegsmarine estableció su propio libro de llaves, y no utilizaba ya indicadores con repeticiones. Las bombes de Turing y el criptoanálisis analítico, a principios de 1941, tardaba entre 38 horas y 11 días para descifrar un mensaje, si es que lo lograban descifrar.

Döenitz, Almirante en Jefe de la flota de submarinos, y posteriormente de la Kriegsmarine, había introducido tres rotores más para un total de ocho. Bletchley Park no conocía el cableado de los nuevos rotores, así que cuando eran usados no podían leer los mensajes. Rejewski estaba en Inglaterra, y sus ecuaciones hubieran podido resolver el problema, pero como los criptoanalistas polacos habían estado en Francia de Vichy en ruta a Inglaterra, no tenían permiso suficiente de seguridad y nunca fueron contactados. El proyecto de Bletchley Park tenía una clasificación de seguridad especial llamada ULTRA; era tan secreta, que estaba prohibido tomar acciones basadas en información de ULTRA que no pudieran ser explicadas en términos de otros canales, para que los alemanes no supieran que se estaban leyendo los mensajes del ENIGMA. A tal grado, que en 1941, Bletchley Park interceptó planes de la Luftwaffe para un bombardeo masivo en Coventry, y Churchill no permitió la evacuación de la ciudad. Así que aunque Rejewski les hubiera podido resolver el problema, no fue contactado debido a las medidas de seguridad que rodeaban Bletchley Park.

Un joven oficial de inteligencia británico, Ian Fleming (que luego escribió los libros de James Bond) sugirió capturar un ENIGMA y su libro de llaves. Su idea era capturar un bombardero alemán, estrellarlo en el Mar del Norte, y capturar al barco de rescate, pero no se pudo llevar a cabo por falta de un bombardero alemán capturado. Un estudiante de licenciatura de Cambridge que trabajaba en Bletchley Park, F. Harry Hinsley, sabía gracias a su estudio de mensajes alemanes que la Kriegsmarine tenía barcos de pesca convertidos al noreste de Islandia, los que enviaban reportes de clima necesarios para los bombardeos durante el Blitz; y que estos barcos patrullaban sin escolta. Hinsley propuso capturar uno de ellos. El Almirantazgo aceptó, y como a las 5 de la tarde del 17 de mayo de 1941, el barco MÜCHEN fue capturado. La tripulación logró tirar el Enigma por la borda, pero no lograron destruir el libro con las llaves de junio. Poco después, el destructor H.M.S. BULLDOG abordó al submarino U-110, a punto de hundirse, y recuperó su ENIGMA intacto. Con el ENIGMA y las llaves de junio, Bletchely Park recuperó los cableados de los nuevos rotores, y empezando el primero de junio, lograba descifrar mensajes en sólo cuatro a seis horas.

Cuando se estaban acabando las llave, el Almirantazgo autorizó otra expedición a Islandia, y el 28 de junio fue capturado en barco LAUENBURG, con las llaves de julio. Llegaron a Bletchley Park el 2 de julio, y el tiempo de descifrado bajó de cuarenta horas (durante el primero y dos de julio) a menos de tres por mensaje. Con esta información mejoraron las bombes, y para agosto de 1941, el tiempo promedio de una solución analítica (sin conocer la llave con anticipación) pasó a 12–40 horas.

Dönitz notó el cambio. En la primavera de 1941, sus submarinos hundieron la mayoría de los barcos mercantes aliados en el Atlántico. De repente, dejaron de poder hundir, y a cambio de esos les estaban hundiendo sus submarinos. (Los aliados descubrían la posición de los submarinos a base de descifrar los mensajes de ENIGMA. Luego enviaban un avión de búsqueda antes de ir a hundirlos para que pareciera que los habían encontrado por casualidad). Aunque los criptoanalistas de la Kriegsmarine le aseguraron a Dönitz que no podía ser que los aliados estuvieran leyendo el ENIGMA, Dönitz decidió agregar un cuarto rotor y cambiar de proceso a un código especial.

Gracias a los mensajes descifrados, Bletchley Park sabía que venía un cambio inminente en los ENIGMAS. Pero no podían hacer nada. Afortunadamente, los alemanes cometieron un error. En diciembre de 1941, dos meses antes del cambio oficial, un submarino alemán envió un mensaje por error utilizando los cuatro rotores. Para empeorar todavía más las cosas, volvió a transmitir el mensaje con tres rotores. Gracias al error, Bletchley Park obtuvo el cableado del nuevo rotor α. (Los rotores α y β eran rotores delgados y fijos pues el mecanismo del ENIGMA no se podía cambiar, de manera que el cuarto rotor no rotaba durante la transmisión). En febrero de 1942, Dönitz hizo oficial el cambio. La bombe de Turing se volvió obsoleta de la noche a la mañana, y Bletchley Park tuvo que comenzar de nuevo. Durante casi todo 1942, no se pudieron leer los mensajes. Durante febrero y marzo de 1942, los submarinos alemanes hundieron 216 barcos aliados.

Bletchley Park no había sido totalmente franco sobre sus logros con Estados Unidos, por temor a que Estados Unidos los dejara filtrar. Pero en abril de 1942 no lograban avanzar con la modificación del a bombe. Ese mes, el Coronel John Tilman de GC & CS visitó OP-20-G, la oficina de criptoanálisis de la marina de Estados Unidos; y en julio permitieron a oficiales del ejército de Estados Unidos visitar Bletchley Park. El 3 de septiembre de 1942, el Comandante Wenger, subdirector de OP-20-G, le pidió oficialmente al Almirante Redman, jefe de OP-20-G, presupuesto para construir un prototipo de Bombe de Estados Unidos para descifrar el Enigma. Estas eran llamadas "Navy Bombes" o "Bombes de la Marina" para diferenciar de la Bombes de Bletchley Park.

El 30 de octubre de 1942, el H.M.S. PETARD capturó al submarino U-559; dos marinos murieron recuperando documentos criptográficos en el interior del submarino. Uno de ellos era el código naval alemán de cifrado de clima, que dieron la entrada que necesitaban los aliados: Las estaciones climatológicas de los alemanes transmitían reportes de clima, que eran necesarios para los submarinos y los aviones; pero como no tenían el ENIGMA de cuatro rotores, mandaban la información usando sólo tres rotores. Sin embargo, usaban la misma llave del día que los submarinos, omitiendo la posición del cuarto rotor. Una vez que se dieron cuenta de ésto, Bletchley Park podía utilizar su viejo sistema para determinar la llave del día, excepto por cuál de los dos rotores α y β se estaba usando, y su posición inicial. Con eso, podían nuevamente a intentar ataques de fuerza bruta. En noviembre de

1942, lograron romper mensajes del ENIGMA de la Kriegsmarine de cuatro rotores en un promedio de 36 horas, el 70% de los días; pero a veces tardaban hasta 10 días, si tenían problemas con los mensajes climatológicos.

Los aliados también descubrieron otro error en la operación de los submarinos alemanes. Si un submarino había estado sumergido varios días, al salir a flota pedía por radio todos los mensajes que no había podido recibir mientras estaba debajo del agua. Estos mensajes se mandaban usando la clave del día de la solicitud, y no la clave del día en que se habían enviado originalmente. Esto permitía a Bletchley Park usar superposición de Kerckhoffs para encontrar llaves de días anteriores y del día actual.

La marina americana continuó su proyecto de bombes. En mayo de 1943, los primeros dos prototipos, *Adán* y *Eva*, fueron puestos en marcha. En 20 minutos, descifraron el primer mensaje, que identificaba la posición de uno de los submarinos de abasto alemanes en el Atlántico. Durante el siguiente mes, los aliados hundieron 41 submarinos, el 25% de la flota de Dönitz, y 9 de los 12 submarinos de abasto. El 22 de mayo, Döntiz concedió el Atlántico Norte a los aliados y comenzó a retirar a sus submarinos.

Para finales de 1943, había 77 bombes de la Marina. Bletchley Park había construido 18 bombes especiales para 4 rotores, pero nunca logró suficiente volumen de trabajo para descifrar todos los mensajes que se recibían. Los británicos empezaron a dejarle el trabajo del descifrado a la Marina de Estados Unidos, y se concentraron en la Luftwaffe y la Wehrmacht. Para la primavera de 1944, entre Bletchley Park y las bombes de la Marina de Estados Unidos, los mensajes alemanes que utilizaban ENIGMA eran descifrados en alrededor de 20 minutos por mensaje.

El gran éxito de Bletchley Park y la Marina de Estados Unidos en descifrar mensajes que utilizaban la ENIGMA se mantuvo secreto hasta mediados de los setentas; en parte, se trató de razones de seguridad e inteligencia. Muchos países africanos y asiáticos continuaron usando ENIGMAS y máquinas similares para su criptografía diplomática, y representaban una fuente importante de inteligencia para Gran Bretaña y Estados Unidos. Para 1972, la mayoría de los ENIGMAS se habían vuelto demasiado viejos y obsoletos, y habían sido abandonados por los gobiernos que los utilizaban.

En 1974, F.W. Winterbotham, un ex-capitán de la Real Fuerza Aérea, escribió sobre el trabajo de Bletchley Park. Esto fue mucho antes de la fecha en que Estados Unidos tenía planeado hacer pública la historia del ENIGMA, pero la información salió a la luz. Poco a poco, los Estados Unidos empezaron a revelar su papel en el éxito criptoanalista de los Aliados. En 1978 una exhibición en el Museo Nacional de Historia del Smithsonian en Washington hizo pública la historia completa.

4.3 PÚRPURA

La presentación de ésta sección está basada en [Kah99]

4.3.1 Antecedentes históricos

En 1912, Herbert Osborne Yardley consiguió trabajo como secretario de códigos del Departamento de Estado de Estados Unidos. Una noche, un mensaje de 500 palabras en código del Coronel House al Presidente Woodrow Wilson pasó por su escritorio, y Yardley decidió ver si lo podía descifrar. Para su gran sorpresa, lo logró (los códigos diplomáticos de Estados Unidos eran bastante anticuados y malos). Envió su solución y observaciones sobre la seguridad del código a su jefe. En 1917, Estados Unidos declaró la guerra a Alemania, y Yardley convenció al Departamento de Estado y al Departamento de Guerra que pusieran el dinero para una sección de criptoanálisis de códigos diplomáticos.

Durante los siguientes años, Yardley continuó trabajando y resolviendo códigos diplomáticos de países enemigos, neutrales, y aliados. En 1919, se le encargó a su organización resolver los códigos japoneses. Yardley prometió resolverlos en un año o renunciar. Casi de inmediato, se arrepintió de su promesa: el texto claro japonés era bastante más complicado de lo que esperaba, aun sin tomar en cuenta el cifrado.

La única gran civilización antigua que parece no haber inventado la idea de la criptografía de algún tipo fue China. Probablemente ésto se debió a que leer y escribir era algo tan raro (hasta hoy en día, una medida popular de inteligencia y estatus social es cuántos ideogramas sabe uno; se considera al número como directamente relacionado con la educación de la persona), que el sólo hecho de escribir un mensaje lo ponía en una especia de código o cifrado, y no era necesario hacer más. La cultura japonesa está fuertemente influenciada por la china, de manera que Japón tampoco desarrolló códigos ni cifrados hasta principios del siglo XX, después de su contacto con el occidente.

El japonés tiene tres "alfabetos". El Kanji, que es el alfabeto de ideogramas, es el primero. Para considerarse que uno domina el japonés escrito, el gobierno exige un vocabulario de 2000 kanji (ideogramas); para leer un periódico, es necesario saber entre 2000 y 2500.

Otro alfabeto es el Hiragana; este es un alfabeto cursivo, que es en realidad un silabario: cada símbolo representa una sílaba, (e.g. *'ka'*, *'ke'*, *'la'*, *'mu'*, etc.) con excepción de uno que representa el sonido *'n'*. En total, hay 46 símbolos, y dos signos diacríticos que modifican algunos de los sonidos.

El tercer y último alfabeto es el Katakana; el Katakana también es un silabario, con una correspondencia uno a uno con el Hiragana; pero mientras que el Hiragana es cursivo, el Katakana es tipo imprenta. Para transliterar palabras en un idioma que no sea el japonés, siempre se usa el Katakana (nunca el Hiragana o el Kanji); y se utiliza como silabario de imprenta.

El código japonés de 1921 era un código basado en el Katakana, que era usado para las transmisiones telegráficas. Se basaba en una sustitución monoalfabética compuesta con un código comercial chino. Tras varios tropiezos, Yardley con la ayuda de Livesey, su secretario (que tenía una gran facilidad para idiomas), fueron encontrando tablas de frecuencia y finalmente lograron descifrar el código en febrero de 1920.

En 1920, Japón contrató a un experto polaco en criptografía quien les recomendó varios cambios: partir un mensaje en pedazos, revolver las partes, y cifrar, para esconder las frases estereotipadas típicas de las comunicaciones oficiales; y también les enseño cómo construir códigos para no tener que utilizar un código comercial. Pero aún con esos cambios la seguridad de los sistemas criptográficos japoneses era mediocre.

En 1921, hubo una gran conferencia de desarme naval entre Estados Unidos, el Reino Unido, Francia, Italia, y Japón. En ella, se estaban negociando los límites relativos de cada potencia. Yardley estaba descifrando los telegramas a los negociadores japoneses, y con ello Estados Unidos logró negociar al máximo el tratado. Japón pidió originalmente una relación diez a ocho con Estados Unidos; las instrucciones a los negociadores era que aceptaran una relación diez a siete, y en el peor caso, que aceptaran una relación diez a seis si era la única posibilidad. Armados con esa información, los Estados Unidos presionaron hasta que se firmó el tratado con la última relación.

En 1924, el nuevo presidente de Estados Unidos, Herbert Hoover, redujo el presupuesto al Departamento de Guerra. Yardley mandó un memorándum al presidente, describiendo la historia y actividades de su unidad. Después de un par de meses, Yardley supuso que el nuevo Secretario de Estado, Henry Stimson, se habría dado cuenta de las realidades diplomáticas, y le envió la solución de varios mensajes diplomáticos importantes. Este gambito había funcionado de maravilla con administraciones anteriores, y Yardley supuso que Stimson, al reconocer la utilidad de los mensajes, le regresaría el presupuesto a su organización. Pero la estrategia no funcionó con Stimson. Cuando éste se enteró de cómo se habían leído los mensajes, se escandalizó. Cerró la oficina y despidió a Yardley, diciendo "Los caballeros educados no leen las cartas ajenas."

En 1931, en plena Depresión, Yardley estaba prácticamente quebrado y decidió escribir sus memoria. El libro, *"The American Black Chamber"* fue un éxito inmediato. El libro incluía la historia de los códigos japoneses y la conferencia de desarme naval de 1921.

Los japoneses decidieron que no iban a permitir que algo parecido ocurriera en el futuro, y empezaron a rediseñar completamente sus sistemas criptográficos. Como no tenían tradición criptográfica propia, se pusieron a estudiar los modelos comerciales que exstían, como el ENIGMA.

4.3.2 Los sistemas japoneses: ROJO, PÚRPURA, CORAL

Para su sistema militar, los japoneses utilizaron en el *ango kikai taipu A* ("máquina de cifrado A"), que era llamada ROJO por los estadunidenses.[3]

La ango kikai taipu A funcionaba mediante una permutación fija inicial, y luego un "medio rotor" con 26 salidas. Un medio rotor consiste de un disco con 26 entradas, igual que el rotor, pero del otro lado en vez de otras 26 salidas, tiene un largo tubo de material aislante con una serie de contactos; a medida que va girando el medio rotor, el tubo gira con él, y los contactos se van moviendo. El alambrado del medio rotor de ROJO permutaba las seis vocales (*a*, *e*, *i*, *o*, *u*, *y*) entre sí, y las 20 consonantes entre sí. Se trataba de un sistema polialfabético, con un periodo de 60, con alfabetos relacionados, donde además las vocales iban a vocales y las consonantes a consonantes. Esto permitía romanizar[4] el katakana, y que el resultado fuera la romanización de otro katakana.

En 1935, ya bajo la administración de Franklin Delano Roosevelt, el código fue atacado for Solomon Kullback y Frank Rowlett, alumnos de Friedman que estaban trabajando para 20-OP-G, el departamento de criptoanálisis y criptografía de la Marina de Estados Unidos. Kullback y Rowlett lograron reconstruir el código totalmente para 1936, resolviendo el sistema por completo.

En 1937, Japón empezó a desarrollar un sistema más seguro, que entró en operación en 1939; el *ango kikai taipu B* ("máquina de cifrado B"), llamado PÚRPURA por Estados Unidos. Aún mantenía una separación en la permutación de seis letras por un lado y veinte por otro, pero ya no tenían que ser vocales y consonantes. El número de alfabetos bajó de 60 a 25, pero la selección de alfabetos era muy irregular. El sistema estaba basado en engranes, y en cableados, pero el corazón de la máquina eran unos "uniselectores". Los uniselectores se utilizaban en centrales telefónicas para elegir circuitos libres cuando había varias llamadas al mismo conmutador, y son muy irregulares (aunque determinísticos) en su operación.

[3] La tradición del Departamento de Estado era ponerle nombre de colores a los códigos diplomáticos; el código diplomático estadounidense de los 20s se distribuía a los oficiales de las embajadas en unas pastas grises, y era conocido como GRIS. Bajo el código GRIS, la palabra clave para *Japón* era `ROJO`; de manera que el código diplomático japonés se conoció como ROJO. A medida que fueron cambiando, se le fueron poniendo nombres de otros colores asociados, como NARANJA, PÚRPURA, etc.

[4] Romanizar significa transliterar del alfabeto japonés al alfabeto latín

Las características del criptotexto indicaban claramente que se trataba de un sistema polialfabético. El Departamento de la Marina encargó el trabajo de romper PÚRPURA a Friedman, y él y Frank Rowlett, Robert Ferner, Albert Small, Samuel Snyder, Genevieve Feinstein, y Mary Jo Dunning, trabajaron durante 18 meses en el problema. Rápidamente determinaron el número de alfabetos, y los dos grupos de las permutaciones, pero no lograban encontrar ningún método electromecánico conocido que reprodujera la variación entre los alfabetos. En el verano de 1940, un recluta recién llegado de M.I.T., Leo Rosen, propuso la idea de los uniselectores telefónicos, y rápidamente OP-20-G había reconstruido una máquina PURPURA. Con una máquina a la mano, la solución analítica se volvió muy sencilla.

La verdad es que la calidad criptográfica de PURPURA era bastante mediocre, comparada con otros sistemas que existían y estaban en uso en ese entonces (por ejemplo, ENIGMA). Los japoneses parecen haber subestimado el ingenio de los criptoanalistas, y sobreestimado la dificultad del japonés[5], violando la primera máxima de Kerckhoffs.

La siguiente máquina, llamada CORAL por OP-20-G, fue introducida en diciembre de 1943. CORAL abandonó la separación del alfabeto en dos grupos de veinte y seis letras cada uno para la permutación, pero su sistema de selección de alfabetos era demasiado regular, y fue roto a principios de 1944. En general, la capacidad criptográfica japonesa fue bastante mediocre durante toda la guerra. En 1941, por ejemplo, Frank Raven descubrió que la llave del día se generaba mediante una permutación: se empezaba por una llave aleatoria, y se aplicaba una permutación de periodo diez a la llave; cada paso daba la llave del día siguiente, hasta que se llegaba a la llave original, que entonces era reemplazada por una nueva llave aleatoria. Esto facilitaba enormemente la solución de criptogramas japoneses.

Una de las principales fuentes de información sobre los movimientos alemanes que tenían los rusos, por ejemplo, era la Embajada de Japón en Moscú, quien se comunicaba con Tokyo sobre la información de sus aliados alemanes. Aparentemente, los únicos que tenían dificultades para leer los códigos japoneses durante la Segunda Guerra Mundial eran los japoneses.

4.3.3 PÚRPURA y la Guerra en el Pacífico

El criptoanálisis de PÚRPURA estaba clasificado como MAGIC, más allá de TOP SECRET, por la Marina de Estados Unidos. OP-20-G se volvió tan bueno descifrando que el 7 de diciembre de 1941, el texto proveniente de OP-20-G llegó al Departamento de Estado dos horas antes de que la Embajada Japonesa terminara de descifrarlo y entregara el

[5]Hasta hoy en día, muchos japoneses dicen que ningún extranjero puede en realidad aprender japonés bien.

Declaración de Guerra al Departamento de Estado[6].

Hubo dos grandes éxitos de Estados Unidos basados en el criptoanálisis de PURPURA y los sistemas relacionados. Posiblemente el más famoso es la Batalla de Midway.

Del 7 de diciembre de 1941 (el ataque a Pearl Harbor), hasta mayo de 1942, la marina japonesa había ganado todas y cada una de sus batallas. Los almirantes japoneses decidieron ir más allá de su plan original, que era conquistar el Pacífico Oriental, y sacar a Estados Unidos completamente del Océano Pacífico. La clave era Midway, una pequeña isleta a la mitad del camino entre Estados Unidos y Asia, un poco al noroeste de Hawaii.

Estados Unidos tiene una base militar importante en Midway. Con esa base ahí, Japón no podía invadir Hawaii, pues se arriesgaba a un ataque en su retaguardia.

El Almirante Isoroku Yamamoto, comandante en jefe de la Marina Japonesa, preparó un plan mediante el cual haría varias fintas hacía las Islas Aleutianas para sacar de posición a la marina de Estados Unidos, y luego atacar Midway. Japón contaba con 6 portaaviones, contra 4 de Estados Unidos (uno de los cuáles había sido dañado recientemente). Lo que Yamamoto no sabía es que Estados Unidos estaba leyendo todas sus ordenes.

Las ordenes, por supuesto, no decían "Midway", sino que usaban una palabra código y coordenadas de mapa cifradas: `AF`. Había varias posibilidades, incluyendo un ataque a las bases navales y aéreas en la Islas Aleutianas, y Nimitz, el comandante en jefe de la Marina de Estados Unidos en el Pacífico, no tenía suficientes recursos para proteger todos los posibles blancos. Joseph John Rochefort, jefe de OP-20-GY (la estación del 20-OP-G en el Pacífico), estaba convencido que el blanco final era Midway, pero Nimitz no estaba dispuesto a arriesgarse sin pruebas.

Rochefort decidió engañar a los japoneses para que ellos identificaran el blanco. Mandó por mensajero un mensaje a Midway, pidiéndoles que mandaran un mensaje sin cifrar solicitando una entrega especial de emergencia de agua, diciendo que su planta desalinizadora se había descompuesto. Midway envió el mensaje. Dos días después, un mensaje de un puesto de espionaje de radio japonés informó al comando de la marina que en `AF` les faltaba agua, pues la planta desalinizadora se había descompuesto.

Con esta información en mano, Nimitz preparó su flota el 27 de mayo de 1942. El 3 de junio, comenzó la operación de Yamamoto. En el primer combate naval de la historia en el cual los barcos en conflicto no se llegaron a ver, los portaaviones USS Enterprise, USS Hornet, y USS Yorktown lanzaron sus bombarderos. Los primeros torpederos cayeron ante el fuego antiaéreo de los portaaviones japoneses. Poco después, mientras los

[6]El mensaje, sin embargo, no tenía información específica y llegó demasiado tarde para prevenir el ataque a Pearl Harbor.

japoneses preparaban sus aviones para el bombardeo terrestre de la base en Midway, los "dive bombers" atacaron por sorpresa y sin oposición. Yamamoto había traído cuatro de sus seis portaaviones a la operación: AKAGI, KAGA, SORYU, y HIRYU. Los bombarderos estadounidenses hundieron AKAGI, KAGA, y SORYU tras unas cuantas horas de combate. En la tarde, hundieron HIRYU, mientras que los japoneses hundieron USS YORKTOWN, el portaaviones que había estado dañado. Yamamoto decidió que sin el soporte aéreo no podía proseguir con la invasión de Midway, y ordenó la retirada. Más aun, los japoneses pasaron de una ventaja a una desventaja en portaaviones y tonelaje total en el Pacífico, pasando de seis contra cuatro a dos contra tres.

Midway fue el principio del fin de la Guerra en el Pacífico. Nunca más volvió a avanzar la Marina Japonesa, y se tuvo que contentar con luchas de defensa y battalas de retaguardia. Como dijera después Nimitz, "La victoria de Midway fue una victoria de Inteligencia." El General George Marshall, Jefe del Estado Mayor, fue más explícito: "Gracias al criptoanálisis, pudimos concentrar nuestras limitadas fuerzas donde eran necesarias; sin el criptoanálisis, hubiéramos estado tres mil millas fuera de lugar."

El segundo gran éxito obtenido gracias a MAGIC fue el asesinato de Yamamoto. En 1943, Yamamoto decidió hacer un tour de inspección por las Islas Salomón, en un intento de subir la moral a las tropas japonesas que estaban ahí atrincheradas esperando el ataque de la Marina de Estados Unidos. Yamamoto era una figura casi legendaria, carismático, un gran estratega y táctico naval. Era adorado por la tropa, quienes en general opinaban que mientras Yamamoto estuviera seguro de la eventual victoria, la victoria estaba segura.

Yamamoto preparó su tour, y envió su itinerario para preparar su llegada a las bases. Desafortunadamente, envió su itinerario cifrado, lo cual por supuesto no evitó que los criptoanalistas de Estados Unidos pudieran leer con todo lujo de detalle dónde y cuándo iba a estar Yamamoto.

Nimitz notó que había un lugar, entre la visita a dos islas, en que se podía tratar de emboscar al avión de Yamamoto. Era un lugar remoto, apenas dentro del límite de los caza bombarderos estadunidenses. Era poco probable que una emboscada ahí fuera considerada casual. Montarla pondría en peligro el secreto de MAGIC.

Pero Yamamoto era demasiado importante, demasiado clave en la mente japonesa, y Nimitz decidió que un pájaro en mano era más importante, y ordenó a dos cazas bombarderos que montaran la emboscada. El 8 de abril, derribaron el avión que traía a Yamamoto, matándolo. El 21 de mayo, el Imperio Japonés anunció la muerte de Yamamoto "durante un tour de inspección" y sin dar más detalles. El hombre que le siguió al mando de la Marina Japonesa estaba tan desmoralizado como el resto; en su discurso al tomar el mando dijo: "Sólo había un Yamamoto. Nadie lo puede reemplazar. Su pérdida es un golpe del que nunca nos recuperaremos."

Pero los japoneses estaban tan convencidos de la seguridad de sus códigos y la inherente impenetrabilidad de su idioma, que atribuyeron el ataque a simple mala suerte; ni siquiera cambiaron su código hasta agosto, cuando estaba previsto el cambio usual.

La historia de MAGIC se supo mucho más rápido que la de ULTRA, posiblemente porque los sistemas japoneses no eran considerados seguros y nadie los estaba imitando. Para 1960, ya se sabían los detalles del asesinato de Yamamoto y la Batalla de Midway. En gran medida, MAGIC ganó la Guerra en el Pacífico, así como ULTRA ganó la Batalla del Atlántico.

Para terminar con los sistemas que podríamos llamar "históricos", vamos a mencionar brevemente tres tipos más de cifrado: los códigos, los nomenclátores, y los cifrados por transposición.

5.1 Códigos

Un **código** consiste de, normalmente, miles de frases, palabras, sílabas, y letras (unidades lingüisticas), y sus *palabras código*, *números código*, o con mayor generalidad, *grupo de código*. Un ejemplo típico se vería de la siguiente manera:

3964	Capitán
2103	listo para
0010	proseguir
1842	detener
9201	hombre a la deriva

Para enviar el mensaje *"El capitán está listo para proseguir,"* se enviaría

```
3965 2103 0010
```

Los códigos se volvieron muy comunes en la segunda mitad del siglo XIX; eran utilizados sobre todo para ahorrarse cuotas de telegrafía, permitiendo el envío de un mensaje largo utilizando únicamente unos cuantos grupos de código. Aunque también proporcionan un cierto grado de seguridad, su seguridad no es mucha debido al respeto de las unidades lingüisticas. Un código modernos, a mediados del siglo XX, contendría aproximadamente 50,000 grupos de códigos.

5.2 Nomenclátores

Entre 1400 y 1850, un sistema que era mitad cifrado y mitad código dominó la criptografía. El sistema normalmente consistía de un alfabeto de cifrado (monoalfabético), con homófonos, y una lista tipo código con nombres, palabras, y sílabas. Orginalmente, la lista tenía sólo nombres, lo cual le dio el nombre al sistema: **nomenclátores.**

Un nomenclátor es, según la Real Academia Española (22a. edición), un *"catálogo de nombres, ya sea de pueblos, de sujetos, o de voces técnicas de una ciencia o disciplina."* Más tarde, los nomenclátores llegaron a crecer hasta ser comparables con los códigos modernos; sin embargo, cualquier sistema tipo código utilizado entre 1400 y 1850 es llamado un nomenclátor.

Una característica histórica curiosa es que los nomenclátores siempre se escribían en grandes hojas sueltas que eran dobladas, mientras que los códigos, una invención moderna, casi invariablemente se escriben en formato de libro.

Originalmente, los nomenclátores listaban tanto el texto original como los elementos codificados en orden lexicográfico. Esto se debía a que era necesario poder codificar y decodificar rápidamente. La única variación ocurría en nomenclátores pequeños, donde los nombres estaban en un orden más aleatorio.

Esto, por supuesto, facilitaba el trabajo del criptoanalista. Si alguién sabía que `137` representaba *de*, y `168` representaba *en*, entonces `21` no podía ser *para*, pues los códigos de palabras que empiezan con *p* serían después; y que *día* tendría que estar entre `138` y `167`.

En 1600 nació Antoine Rossignol, quien se convertiría en el primer criptólogo de tiempo completo en Francia. En 1628, decodificó un mensaje muy importante para el Rey de Francia, y con ello inició su carrera. Alrededor de 1650, Rossignol introdujo la innovación más importante que tuvieron los nomenclátores durante los 400 años en que reinaron supremos en la criptografía mundial.

Lo que hizo Rossignol fue revolver los códigos respecto al texto claro, para que no estuvieran ambas listas en orden lexicográfico. Para facilitar la codificación y decodificación, en vez de escribir el nomenclátor en una sóla tabla, se escribía en dos; una en la que el texto original estuviera en orden lexicográfico, llamado la *"tables à chiffrer"*; y uno en el que los códigos estuvieran listados en orden lexicográfico, llamado la *"tables à dechiffrer."* Debido a ello, estos son conocidos como *nomenclátores de dos partes*, y los anteriores como *nomenclátores de una parte*.

La idea de Rossignol fue adoptada rápidamente, y para 1700 la mayoría de los países usaban nomenclátores de dos partes con 2000 ó 3000 elementos (grupos de códigos).

Los nomenclátores perdieron su importancia con el advenimiento del telégrafo, que estimuló el cifrado y los códigos modernos. Además, hacia 1800 era claro que la seguridad ofrecida por un nomenclátor, aún uno en dos partes y bien hecho, era mediocre y directamente proporcional al tamaño y costo de producción. Paulatinamente fueron abandonados, y la popularización del telégrafo en 1850 los volvió obsoletos.

5.3 Transposición

Un cifrado por transposición no cambia la identidad de las letras, sino únicamente su órden. Por ejemplo, la frace *vine, ví, vencí* se puede cifrar listando alternadamente en dos renglones:

```
V   N   V   V   N   I
  I   E   I   E   C
```

y luego leído por renglones, para producir el criptotexto `VNVVNIIEIEC`. O bien podemos escribirlo en un cuadrado,

```
V  I  N  E
V  I  V  E
N  C  I  X
```

y leído primer la primer columna de arriba hacia abajo, luego la segunda de abajo hacia arriba, etc., para producir el criptotexto `VVN CII NVI EEX`.

Es fácil reconocer el resultado de un sistema de transposición, pues la distribución de letras es la distribución usual del idioma en el que está escrito el mensaje.

El tipo más común de transposición es la transposición de columnas. Uno empieza con una palabra clave, por ejemplo, *QUIJOTE*; numeramos las letras en su orden alfabetico:

5	7	2	3	4	6	1
Q	*U*	*I*	*J*	*O*	*T*	*E*

Después, escribimos el mensaje deseado debajo de la palabra clave:

5	7	2	3	4	6	1
Q	*U*	*I*	*J*	*O*	*T*	*E*
Y	A	C	E	A	Q	U
I	E	L	H	I	D	A
L	G	O	F	U	E	R
T	E	Q	U	E	A	T
A	N	T	O	E	X	T
R	E	M	O	L	L	E
G	O	D	E	V	A	L
I	E	N	T	E	Q	U
E	S	E	A	D	V	I
E	R	T	E	Q	U	E
L	A	M	U	E	R	T
E	N	O	T	R	I	U
N	F	O	D	E	S	U
V	I	D	A	C	O	N
S	U	M	U	E	R	T
E	R	O	S	A	P	Q

donde la `P` y `Q` finales son relleno. Luego, listamos cada columna en el orden indicado por la palabra clave (en nuestros tradicionales bloques de cinco letras):

```
UARTT ELUIE TUUNT QCLOQ TMDNE TMOOD MQEHF UOOET AEUTD AUSAI UEELV EDQER
```

```
ECEAY ILTAR GIEEL ENVSE QDEAX LAQVU RISOR PAEGE NEOES RANFI UR
```

También se pueden dejar lugares vacíos al final, o en lugares preacordados entre los corresponsales.

El caso de rectangulos con lugares vacíos tiene buena seguridad para un mensaje relativamente corto. Pero con varios mensajes en la misma clave, se vuelve muy sencillo de criptoanalizar. Para datos sobre el criptoanálisis de sistemas de transposición, dirigimos al lector al último capítulo del libro de Sinkov [Sin66]. Si componemos la transposición columnar (es decir, hacemos una segunda transposición columnar al terminar la primera), el sistema se conoce como un sistema de *doble transposición*, y la seguridad aumenta en mucho. Pero nuevamente, dos o más mensajes con la misma clave debilitan en mucho el sistema.

En general, los métodos de transposición no fueron muy populares hasta épocas recientes, probablemente porque es dificil contruir aparatos mecánicos que realizen transposición, mientras que es relativamente sencillo construír los que hacen sustitución. Otro problema es que requieren trabajar con el mensaje completo en vez de con pedazos aislados (como por ejemplo, letra por letra), lo cual también reduce su eficiencia.

Parte II

Sistemas Modernos

6.1 El origen de DES

En 1959 se inventó el circuito integrado, y como consecuencia inmediata los sistemas de cómputo redujeron su costo de producción, de mantenimiento, y su tamaño. Las computadoras se propagaron en ámbitos distintos de las aplicaciones científicas y militares a las que habían estado consagradas. A fines de la década de los cincuentas, había ya muchas compañías dedicadas a producir equipo de cómputo: IBM, Honeywell, General Electric, RCA, NCR, Burroughs, y DEC, entre otros. El lenguaje Fortran fue inventado a finales de esa década y poco después surgieron Cobol y Algol. Durante la década de los sesentas (siglo XX) las computadoras comenzaron a ser omnipresentes en oficinas gubernamentales y en los bancos; se creó el germen de lo que hoy día es Internet y surge el sistema operativo Unix. Éste era panorama a principios de la década de los setentas: uso cada vez más extenso de los equipos de cómputo en aplicaciones comerciales e interconexión de diversos equipos con el propósito de intercambiar datos y compartir recursos.

Surgió entonces la necesidad de comunicaciones confidenciales entre equipos de cómputo diversos. Los fabricantes de estos equipos proveían de software (programas) para tal fin, pero no había garantía alguna del nivel de seguridad proporcionado por dichos programas, y además no existía compatibilidad entre programas hechos por diferente fabricante. Muchos de los mecanismos de cifrado de datos eran hechos a la medida del cliente, poco generales en su rango de aplicación y basados en mecanismos obscuros.

Ante esta situación la oficina nacional de estándares de los Estados Unidos (*National Bureau of Standards* o NBS, actualmente al *National Institute of Standards and Technology* o NIST) estableció un programa de protección de datos. Como parte de este programa se pretende establecer un algoritmo criptográfico estándar para ser utilizado por las dependencias gubernamentales. Así el 15 de mayo de 1973 el NBS emitió una convocatoria pública para recibir propuestas para un algoritmo criptográfico. Dicho algoritmo debía satisfacer los siguientes requisitos [Sch96]:

1. Proveer un alto nivel de seguridad.

2. Estar completamente especificado y ser fácil de entender.

3. La seguridad del algoritmo debe residir enteramente en la clave y no en el supuesto de mantener secreto el algoritmo mismo.

4. El algoritmo debe poder ser puesto a disposición del público sin restricciones (i.e. no debe haber licencias de uso sobre él).

Nadie se presentó a esta convocatoria, por lo que se emitió una nueva el 27 de agosto de 1974. En atención a ésta última, se presentó un algoritmo diseñado por un grupo de trabajo de los laboratorios de investigación de IBM; el nombre del algoritmo era *Lucifer*.

La idea central de *Lucifer* fue de Horst Feistel, un inmigrante alemán que arribo a los Estados Unidos en 1932 según algunas fuentes [Kah99], o en 1934 según otras [Sin99]. Feistel siempre quiso trabajar en criptografía, pero debido a su origen durante la segunda guerra mundial estuvo en arresto domiciliario. Luego de la guerra, trabajó un tiempo como investigador en el área de criptografía para la Fuerza Aérea y más tarde trabajó en la misma área para una compañía privada (Maitre Corporation). Al parecer [Sin99] los proyectos de Feistel en ambos empleos fueron cancelados por intervención de la NSA (*National Security Agency*), dependencia gubernamental que tiene a su cargo todo lo relacionado con la seguridad de comunicaciones, la criptografía, y el criptoanálisis en los Estados Unidos. Finalmente Feistel se incorporó en 1967 al IBM Thomas J. Watson Reserch Laboratory, de donde surgió *Lucifer*, a principios de los setenta.

La oficina de estándares había solicitado a la NSA la evaluación de las propuestas que recibiera, así que la NSA se dió a la tarea de evaluar a *Lucifer* y hacerle algunas modificaciones.

En el esquema original de *Lucifer* el mensaje era un bloque datos de 128 bits de longitud al igual que la clave usada para cifrarlo. De los 128 bits de clave realmente sólo se usaban 112, porque el octavo bit de cada grupo de ocho bits consecutivos (que en adelante llamaremos bytes) se consideraba un bit de verificación de paridad de los otros siete. Luego de las modificaciones de NSA el mensaje era de 64 bits de longitud al igual que la clave, de la que sólo se usaban realmente 56 bits, siguiendo el mismo esquema de verificación de paridad descrito. Otras de las modificaciones a *Lucifer* fueron en las llamadas *cajas S* (*S-box*), el corazón criptográfico de *Lucifer* que serán descritas más adelante y añadir una permutación inicial al texto de entrada, misma que se deshace (aplicando la permutación inversa) al final; esta permutación no tiene propósito criptográfico alguno, es sólo para que al implementar en hardware el algoritmo fuera eficiente la carga del texto a cifrar.

Por supuesto muchos expertos del área (como Whitfield Diffie) sospecharon de las modificaciones de NSA al algoritmo original. Las modificaciones se habían hecho sin justificación técnica alguna, se redujo el tamaño de la clave, lo que aparenta debilitar el algoritmo. Algunos sospechaban que NSA había puesto "puertas traseras" en el algoritmo que le permitieran a sus expertos y sus máquinas descifrar fácilmente las comunicaciones cifradas con el algoritmo modificado. De hecho en 1978 un comité especial del Senado de los Estados Unidos investigó la controversia y la parte no clasificada del resumen se exonera a NSA de haber hecho modificaciones maliciosas. Más adelante serán claras las razones por las que NSA modificó *Lucifer* y también será claro que la NSA sabía, evidentemente, más de lo que todos sospechaban (que, paradójicamente, era lo que algunos sospechaban).

Finalmente el 23 de noviembre de 1976 el algoritmo propuesto, con las modificaciones de la NSA, se adoptó como estándar para comunicaciones gubernamentales no clasificadas, el famoso *estándar de cifrado de datos* o DES (*Data Encryption Standard*). En 1981 se adoptó como estándar para el sector privado bajo el nombre de DEA (*Data Encryption Algorithm*). La especificación del estándar hacía obligatoria la revisión del mismo periódicamente: en 1983 DES fue certificado nuevamente; igual que en 1987, a pesar de las reticencias de NSA que sospechaba que podría romperse pronto. En 1993, a pesar de que ya había tenido problemas la re-certificación en 1987, se volvió a confirmar, en buena medida porque no había otras alternativas. En enero de 1997 se anunció el proyecto del *Advanced Encryption Standard* o AES y en septiembre de ese año se abrió una nueva convocatoria para recibir propuestas que reemplazaran a DES. DES fue certificado por última vez en enero de 1999 [oST99] ya que el 2 de octubre de 2000 se anunció oficialmente al algoritmo ganador de la convocatoria de 1997: Rijndael, un algoritmo propuesto por los belgas Joan Daemen y Vincent Rijmen. En noviembre de 2001 se publicó el nuevo estándar AES [oST01] y entró

en vigor el 26 de mayo de 2002.

6.2 Descripción de DES

Como todos los algoritmos de su tipo (redes de Feistel), DES trabaja en rondas. Cada ronda es casi idéntica a las demás. Nos avocaremos a describir lo que ocurre en una ronda de DES y luego procederemos a señalar las diferencias y el algoritmo que determina la clave que entra a cada ronda (algoritmo de planificación de llave o *key scheduling* en inglés).

Llamaremos M al texto claro completo escrito en binario que se cifrará. Con K denotaremos la llave que se le proporciona a DES para cifrarlo. Tanto M como K miden 64 bits originalmente, pero el bit del extremo izquierdo (el más significativo) de cada byte (octeto de bits) de la clave se considera como el bit de verificación de paridad de los restantes siete bits en el byte; así que realmente la clave tiene $64 - 8 = 56$ bits útiles.

Identificaremos los bits del mensaje a cifrar como $m_{64}, m_{63}, \ldots, m_1$, cuanto mayor sea el índice más a la izquierda está el bit.

Lo primero que se le hace a M es una permutación sin propósito criptográfico. La permutación, descrita en [oST99] (pag. 10) es:

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

los números en la tabla indican al índice del bit de M y su posición en la tabla es la que adquieren luego de la permutación; así el bit 58 de M queda en la primera posición, el bit 50 en la segunda y así sucesivamente. Puesta así la permutación se ve complicada, pero en realidad no lo és: si nos fijamos en los últimos números de los primeros cuatro renglones tenemos la sucesión: 2, 4, 6 y 8; en general los bits pares de M van a dar a la primera mitad y los impares a la segunda. Éste es un truco para que en una máquina con un canal de datos (bus) de 8 bits la carga de M en el registro del chip DES sea eficiente, y sólo se requiera que sea un registro de carga-desplazamiento. Este registro está dividido en dos mitades por lo que veremos a continuación y que es una característica de las redes de Feistel. Al final,

a la salida de la última ronda de DES se le aplica la permutación inversa: lo que se hizo al principio se deshace, es por eso que la permutación inicial no tiene ninguna implicación criptográfica, y su objetivo es puramente operativo para hacer eficiente la carga y descarga del texto a cifrar y del texto cifrado respectivamente.

El algoritmo completo de DES tiene Digamos que estamos en la ronda i de DES, el algoritmo completo tiene 16 de estas rondas. A esta ronda llega una clave k_i, que de hecho está determinada por la clave K que se da como entrada al algoritmo. El algoritmo que determina k_i a partir de K se abordará más adelante. Sean R_i y L_i las mitades derecha e izquierda del texto de entrada a esta ronda. Al entrar a la primera ronda R_0 y L_0 son las mitades derecha e izquierda del texto de entrada a DES luego de pasar por la permutación inicial. Las longitudes de ambas mitades son iguales a 32 bits y la clave k_i mide 48 bits. La salida de esta ronda son nuevamente dos mitades que constituyen la entrada a la siguiente ronda, llamemos R_{i+1} y L_{i+1} a las mitades derecha e izquierda de salida, respectivamente. Los valores de estas están determinados por las siguientes expresiones:

$$L_{i+1} = R_i \tag{6.2.1}$$

$$R_{i+1} = L_I \oplus f(R_i, k_i) \tag{6.2.2}$$

donde f es la función de cifrado y $\oplus$ denota la disyunción exclusiva (XOR) de sus dos operandos. Para comprender mejor la estructura general de DES es recomendable acudir a la Figura 6.1.

Lo importante aquí es la función f, que recibe dos argumentos: k_i de 48 bits y R_i de 32 bits. La función consiste de varios pasos (véase la Figura 6.2):

1. Permutación expansiva. Los 32 bits de R_i pasan por una permutación expansiva (así que no es una permutación formalmente hablando), que entrega 48 bits a partir de los 32 de R_i. La permutación ([oST99], pag. 13) es:

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

   de donde se obtienen 48 bits a partir de los 32 de R_i.

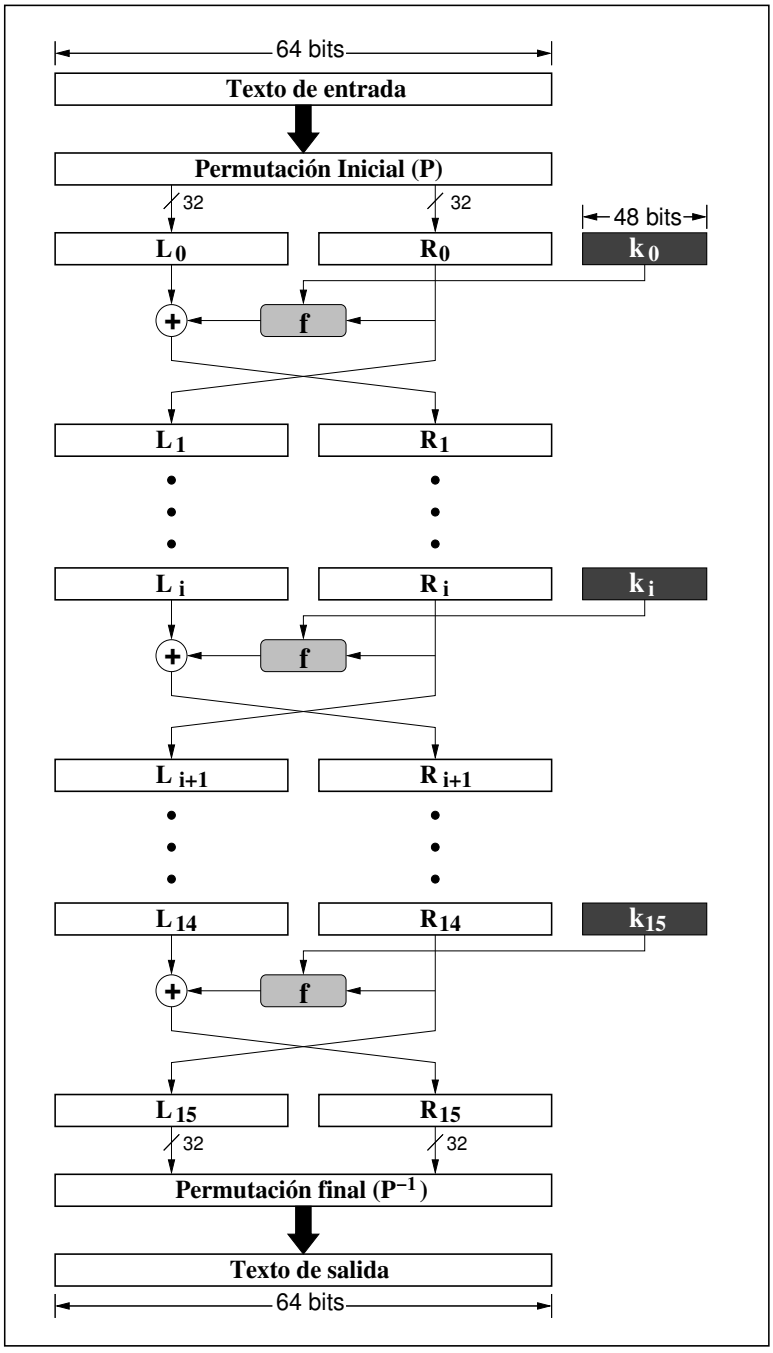

Figura 6.1: Estructura general de DES.

2. A salida de la permutación expansiva se le aplica un XOR bit a bit con los 48 bits de k_i.

3. El resultado del XOR se introduce al dispositivo de substitución constituido por 8 diferentes cajas S. Cada caja recibe 6 bits de entrada. El primero y el último se usan como los bits de un número binario que se interpreta como el índice del renglón r de una matriz, los otros cuatro se usan como un el índice de la columna c. Cada caja S tiene definida una matriz diferente con $4 \times 16 = 64$ entradas, cada entrada es un número de cuatro bits. Cada caja S entrega como salida el número binario de cuatro bits contenido en la entrada (r, c). En total las 8 cajas S entregan entonces $4 \times 8 = 32$ bits. Las ocho cajas son diferentes, pero se usan las mismas ocho y en el mismo orden en cada ronda DES. Esto está ilustrado en la figura 6.3.

4. A la salida de las cajas S se le aplica la siguiente permutación ([oST99], pag. 15):

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

   La salida de esta permutación es la salida de f.

Las entradas de las matrices de las ocho cajas S se encuentran especificadas en el Apéndice 1 de [oST99] y se muestran a continuación.

Caja S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Caja S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

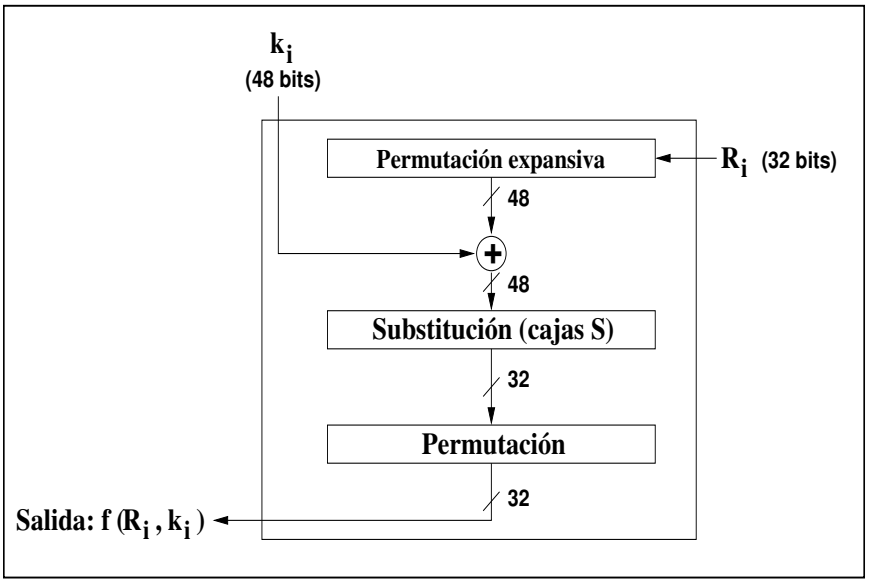

Figura 6.2: Estructura general de la función de cifrado de DES.

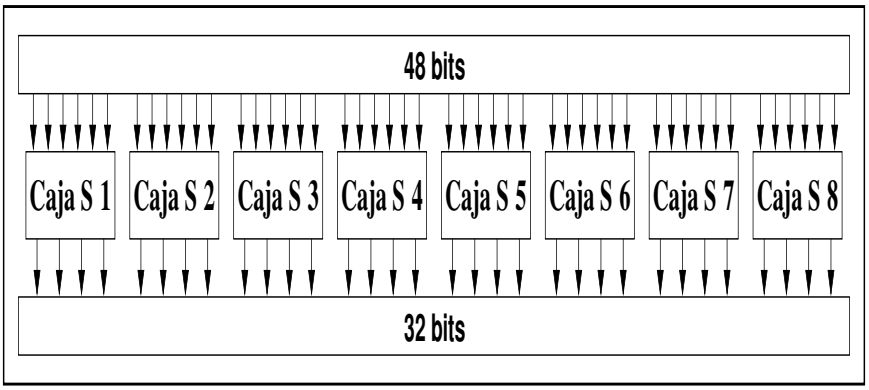

Figura 6.3: Esquema de la función de substitución de DES, el corazón criptográfico del algoritmo. En cada ronda hay 8 *cajas S* (las mismas 8 en cada ronda, en el mismo orden), cada *caja S* recibe como entrada seis bits y entrega como salida cuatro.

Caja S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Caja S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Caja S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Caja S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Caja S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Caja S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Sólo falta por aclarar cómo se determina la llave k_i de entrada a la ronda i a partir de la clave originalmente dada como entrada a DES y que hemos denotado como K. Esto

constituye lo que en el estándar se denomina *planificación de llave* (*key schedule*). El esquema general de éste, para una sola ronda, se muestra en la Figura 6.4.

Una vez verificada la paridad de cada byte de la clave quedan, como dijimos, 56 bits útiles en la clave K. Sea $k'_0 = K$ (56 bits). El algoritmo de planificación de clave procede como sigue:

1. Se divide k'_i en dos bloques de 28 bits cada uno.
2. Cada mitad es rotada uno o dos bits a la izquierda, ambas mitades el mismo número de bits. El número de bits a rotar está determinado por la siguiente tabla [oST99, Sch96]:

Ronda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Desp.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3. La salida de este desplazamiento constituye k'_{i+1}.
4. La salida de este desplazamiento es también pasada por una permutación de compresión a la que entran $28 \times 2 = 56$ bits y de la que salen 48 bits. Esta permutación es [oST99, Sch96]:

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

   Algunos bits se omiten: 9, 18, 22, 25, 35, 38, 43 y 54. Pero como antes la clave se desplazo algunos bits a la izquierda, los omitidos no siempre son los mismos bits de K.

La razón de todo este lío es lograr los dos objetivos señalados por Claude Shannon y que trataremos con mayor detalle más delante: confusión y difusión. Hacer que el texto de salida no se parezca nada al de entrada (confusión) y que cada pequeña parte de la información de entrada tenga impacto en la información de salida (difusión). En DES cada bit de salida en el texto cifrado es función de TODOS los bits del texto de entrada y de

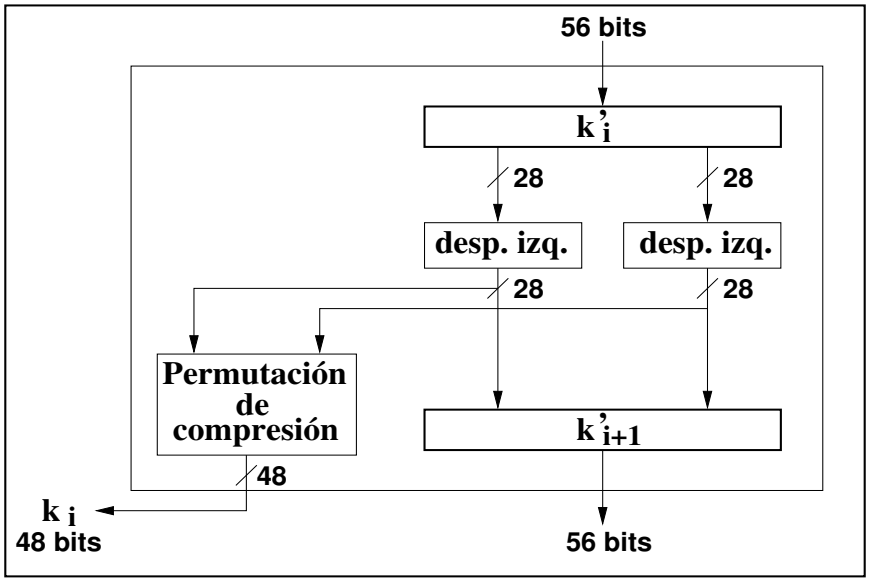

Figura 6.4: Determinación de las llaves de cada ronda de DES. Para la primera ronda k'_0 son los 56 bits de la clave luego de la verificación de paridad.

TODOS los bits de la clave: un cambio en uno solo de los bits de entrada genera cambios en el 50% de los bits de salida, aproximadamente ([Sch96, FdlGH$^+$01]).

Una cualidad de DES es que si se reintroduce a DES un texto previamente cifrado con él y se invierte el orden de las claves de cada ronda (k_{15} entra como k_0, k_{14} como k_1 etcétera), entonces el resultado es el texto claro originalmente cifrado.

Una variante común de DES es el llamado **triple DES** o 3DES. Consiste en elegir tres llaves, K_1, K_2, y K_3, y tomar

$$E_{K_3}\Big(E^{-1}_{K_2}(E_{K_1}(P))\Big)$$

donde P es el mensaje, E_{K_i} consiste en cifrar usando DES con la llava K_i, y E^{-1} es el algoritmo de descifrado de DES. La razón por la cual se usa el descifrado en el segundo paso es para hacer 3DES compatible con DES: para usar el algoritmo para un cifrado DES, simplemente se elige $K_1 = K_2 = K_3$, lo cual resulta en un cifrado utilizando la llave K_1.

6.3 Redes de Feistel, Cifrado de Bloques, y Teoría de Información

Tanto DES como su antecesor directo, *Lucifer*, forman parte de dos grandes familias de algoritmos de cifrado: el cifrado de bloques por rondas, y las redes de Feistel.

La idea del cifrado por rondas es tomar una operación criptográfica relativamente sencilla, y repetirla varias veces para aumentar su seguridad.

Las redes de Feistel son una idea un poco más complicada, pensada en términos de aplicación a computadoras. La idea fue propuesta por Horst Feistel en un artículo de la revista *Scientific American* en 1973 [Fei73].

En ambos casos, las ideas teóricas que informan el diseño de los sistemas criptográficos se deben a Claude Elmwood Shannon, que en 1948 publicó dos artículos que inauguraron la Teoría de Información [Sha48, Sha49].

Las dos técnicas básicas que se utilizan en criptografía son *confusión* y *difusión*.

La *confusión* busca esconder la relación entre el texto original y el texto cifrado. La manera más sencilla de obtener confusión es mediante substituciones. En el caso de cifrado por bloques, se substituye una secuencia larga de caracteres (un bloque), por otra a base de manipular sub-bloques del texto como unidades completas. El resultado final es simplemente una substitución poligráfica con n grande. Esto dificulta el análisis de frecuencia y el análisis de contactos, por ejemplo.

La *difusión* busca disipar la información, buscando que el efecto de un pedazo del texto original se haga sentir a lo largo de todo (o al menos, una parte grande) del texto cifrado. Por ejemplo, la substituciones monoalfabéticas y polialfabéticas carecen de difusión, pues cada caracter del texto original afecta únicamente un caracter del texto cifrado, a saber el que le corresponde. Las substituciones poligráficas tienen más difusión, y entre mayor sea el tamaño de las gráficas, mayor difusión. En general, la manera más sencilla de obtener difusión es mediante permutaciones (o transposiciones).

Repetir confusión normalmente no mejora la seguridad de manera significativa; componer dos substituciones monoalfabéticas no es mejor que una sola substitución monoalfabética; repetir substituciones polialfabéticas ayudan a agrandar el periodo de la llave, pero no representan un aumento significativo en la dificultad final de criptoanálisis. Pero, por lo general, repetir confusión **sí** mejora la seguridad total. Por ejemplo, la doble transposición de columnas es mucho más segura que la transposición de columnas simple. O por poner otro ejemplo: si después de una ronda logramos que un bit de la entrada afecte dos de la salida, repetir el proceso hace que el bit de entrada original afecte cuatro de salida,

luego ocho, etc.

En general, difusión por sí misma, o confusión por sí misma, no dan suficiente seguridad. Los cifrados modernos (post-ENIGMA) buscan utilizar tanto difusión como confusión.

Las redes de Feistel son un tipo especial de cifrado por bloques, que se presta a implementación con computadoras, y que utiliza el cifrado por rondas. La idea es empezar con un bloque de tamaño $2t$ bits, y partirlo en dos pedazos, el pedazo izquierdo I_0, y el derecho, D_0, cada uno de t bits.

En la i-ésima ronda, con I_{i-1} y D_{i-1} definidos, definimos:

$$\begin{aligned} I_i &= D_i; \\ D_i &= I_{i-1} \oplus f(D_{i-1}, K_i) \end{aligned}$$

donde $\oplus$ representa el XOR (suma booleana bit a bit), K_i es la i-ésima llave, y f es una función que depende de la llave y el bloque, y cuyo valor es un bloque de t bits.

En otras palabras, en la i-ésima ronda, primero hacemos que la antigua mitad derecha se vuelva la nueva mitad izquierda; y la nueva mitad derecha es el XOR de la vieja mitad izquierda, y el resultado de una función criptográfica de la vieja mitad derecha y la llave correspondiente.

Hay dos razones principales por las cuáles las redes de Feistel son muy útiles: primero, la función f no tiene que ser demasiado fuerte (desde el punto de vista criptográfico), pues al ser un cifrado de rondas, la composición de rondas fortalece el algoritmo completo. Pero más importante: tenemos una garantía de que, *independientemente de quién sea la función f, las redes de Feistel siempre son invertibles, y son invertibles mediante el mismo algoritmo.*

Como $D_i = I_{i-1} \oplus f(D_{i-1}, K_i)$, podemos recuperar I_{i-1} a partir de D_i, D_{i-1}, y K_i, simplemente:

$$I_{i-1} = D_i \oplus f(D_{i-1}, K_i).$$

Y para recuperar D_{i-1}, sólo necesitamos conocer I_i, pues $D_{i-1} = I_i$. Entonces, dados I_i y D_i, podemos recuperar I_{i-1} y D_{i-1}.

No importa quien sea f, no importa que tan complicada, no importa *que no sea invertible*, la red de Feistel es invertible, y mediante el mismo algoritmo:

$$\begin{aligned} D_{i-1} &= I_i; \\ I_{i-1} &= D_i \oplus f(D_{i-1}, K_i). \end{aligned}$$

Si no tomamos en cuenta la permutación inicial y la permutación terminal (que no tienen

Llave con bits de paridad	Llave
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

Tabla 6.1: Llaves débiles de DES

contribución criptográfica alguna), DES es simplemente una red de Feistel, con:

$$f(D_{i-1}, K_i) = P(S(E(D_{i-1}) \oplus K_i)),$$

donde E es la expansión, S es la Caja S, y P es la permutación.

El objetivo de la expansión E es dar difusión; la S es la única parte no lineal de DES; la seguridad principal de una red de Feistel reside en la falta de linealidad de la función f, y por ello la seguridad de DES reside principalmente en sus cajas S. Una f lineal produce un criptosistema catastróficamente inseguro.

Otra parte de la seguridad de una red de Feistel reside en el manejo de uso de llaves, aunque en términos generales se trata de una contribución mucho menor a la seguridad total del algoritmo.

6.4 Algunas Propiedades de DES

6.4.1 Llaves débiles, semi-débiles, y posiblemente débiles

Por cómo se maneja la distribución de llaves de DES, ciertas llaves son débiles: la llave se parte a la mitad, y cada mitad se maneja independientemente. Si la mitad tiene puros ceros o puros unos, entonces siempre se usa la misma llave en cada ronda. Estas se llaman **llaves débiles**. Las cuatro llaves débiles aparecen en la Tabla 6.1, escritas en hexadecimal (cada dígito represnta un byte).

Algunos pares de llaves cifran texto de manera idéntica; es decir, una llave del par puede usarse para descifrar mensajes que se cifraron usando la otra llave del par. Estas llaves generan únicamente dos subllaves, en vez de dieciséis. Cada una de estas subllaves se utiliza ocho veces en el algoritmo. Estas son llamadas **llaves semi-débiles**, y vienen en seis parejas de dos llaves cada una. Aparecen en la Table 6.2, en su versión con bit de paridad.

01FE	01FE	01FE	01FE	y	FE01	FE01	FE01	FE01
1FE0	1FE0	0EF1	0EF1	y	E01F	E01F	F10E	F10E
01E0	01E0	01F1	01F1	y	E001	E001	F101	F101
1FFE	1FFE	0EFE	0EFE	y	FE1F	FE1F	FE0E	FE0E
011F	011F	010E	010E	y	1F01	1F01	0E01	0E01
E0FE	E0FE	F1FE	F1FE	y	FEE0	FEE0	FEF1	FEF1

Tabla 6.2: Parejas de Llaves semi-débiles de DES

Algunas llaves producen sólo cuatro subllaves, cada una de las cuales es utilizada cuatro veces por el algoritmo. Estas son **llaves posiblemente débiles**, y hay 48 de ellas. Están listadas, incluyendo los bits de paridad, en la Tabla 6.3.

En total, DES tiene 64 llaves que pueden dar problemas, de un total de

$$72,057,594,037,927,936 \text{ llaves.}$$

Otra propiedad importante de las llaves de DES es la **complementación.** Sean P, C, y K el mensaje, mensaje cifrado, y llave, respectivamente, y sean P', C', y K' los complementos bit a bit de P, C, y K, respectivamente. Entonces, si $E_K(P) = C$ (es decir, si el resultado de aplicar el algoritmo de cifrado E a P con la llave K, es C) entonces $E_{K'}(P') = C'$. Es decir, el cifrado del complemento del mensaje con el complemento de la llave es igual al complemento del cifrado del mensaje con la llave.

Esto quiere decir que podemos reducir el espacio de búsqueda a la mitad, pues basta con encontrar el mensaje o su complemento.

6.4.2 Propiedades algebraicas de DES

DES actúa sobre bloques de 64 bits, produciendo bloques de 64 bits. El número de maneras en que podemos mandar cada bloque de 64 bits a cada posible bloque de 64 bits es $(2^{64})!$ (el número de permutaciones de los bloques). Pero el algoritmo de DES usa una llave de 56 bits, lo cuál sólo produce 2^{56} de esas permutaciones. Si usamos cifrado múltiple (es decir, aplacamos DES varias veces, una después de la otra, posiblemente con llaves distintas), parece posible cubrir un mayor número de esas posibles permutaciones. Pero eso sólo es cierto si DES no tiene ciertas propiedades algebraicas.

Un algoritmo de cifrado E es **cerrado** si y sólo si el resultado de aplicarlo dos veces se puede obtener aplicándolo una sola vez; es decir, si para todo mensaje P, y llaves K_1 y K_2,

1F1F 0101 0E0E 0101	E001 01E0 F101 01F1
011F 1F01 010E 0E01	FE1F 01E0 FE0E 01F1
1F01 011F 0E01 010E	F101 1FE0 FE01 0EF1
0101 1F1F 0101 0E0E	E01F 1FE0 F10E 0EF1
E0E0 0101 F1F1 0101	FE01 01FE FE01 01FE
FEFE 0101 FEFE 0101	E01F 01FE F10E 01FE
FEE0 1F0E FEF1 0E01	E001 1FFE F101 0EFE
E0FE 1F01 F1FE 0E01	FE1F 1FFE FE0E 0EFE
FEE0 011F FEF1 010E	1FFE 01E0 0EFE 01F1
E0FE 011F F1FE 010E	01FE 1FE0 01FE 0EF1
E0E0 1F1F F1F1 0E0E	1FE0 01FE 0EF1 01FE
FEFE 1F1F FEFE 0E0E	010E 1FFE 01F1 0EFE
FE1F E001 FE0E F101	0101 E0E0 0101 F1F1
E01F FE01 F10E FE01	1F1F E0E0 0E0E F1F1
FE01 E01F FE01 F10E	1F01 FEE0 0E01 FEF1
E001 FE1F F101 FE0E	011F FEE0 010E FEF1
01E0 E001 01F1 F101	1F01 E0FE 0E01 F1FE
1FFE E001 0EFE F001	011F E0FE 010E F1FE
1FE0 FE01 0EF1 FE01	0101 FEFE 0101 FEFE
01FE FE01 01FE FE01	1F1F FEFE 0E0E FEFE
1FE0 E01F 0EF1 F103	FEFE E0E0 FEFE F1F1
01FE E01F 01FE F10E	E0FE FEE0 F1FE FEF1
01E0 FE1F 01F1 FE0E	FEE0 E0FE FEF1 F1FE
1FFE FE1F 0EFE FE0E	E0E0 FEFE F1F1 FEFE

Tabla 6.3: Llaves posiblemente débiles de DES

existe una llave K_3 tal que

$$E_{K_2}(E_{K_1}(P)) = E_{K_3}(P).$$

Un ejemplo de un sistema cerrado es la substitución monoalfabética, donde la llave es la permutación del alfabeto.

En el case de DES, sabemos que el mismo algoritmo E se puede usar para descifrar un mensaje cifrado con DES. De manera que si DES fuese cerrado, entonces sería un **grupo**: el resultado de aplicar DES dos veces es aplicar DES una vez (con cierta llave), y toda aplicación de DES se puede invertir con una aplicación de DES.

Si DES fuera un grupo, entonces DES sería vulnerable a ciertos tipos de ataque, como el ataque de Encuentro a la Mitad de Merkle y Hellman, que se tardaría "sólo" 2^{28} pasos (en vez de los 2^{55} que requiere el ataque de fuerza bruta); describiremos el ataque de Encuentro a la Mitad en la siguiente sección.

Pero en 1992, K.W. Campbell y M.J. Wiener [CW93] probaron que DES no es un grupo, siguiendo el trabajo de varios otros criptoanalistas.

Otra propiedad que puede tener DES es la de **pureza**. Decimos que un algoritmo de cifrado E es **puro** si y sólo si, para todo mensaje P, y llaves K_1, K_2, y K_3, existe una llave K_4 tal que

$$E_{K_3}(E_{K_2}(E_{K_1}(P))) = E_{K_4}(P).$$

Nótese que todo sistema que sea cerrado es puro, pero un sistema puede ser puro sin ser cerrado.

Si DES fuese puro, entonces el triple cifrado de DES sería igual de sencillo de atacar que DES, y no ganaríamos nada. No se sabe si DES es puro o no, pero se sospecha que no lo es.

6.4.3 Ataque de Encuentro a la Mitad

El ataque de Encuentro a la Mitad fue sugerido por Merkle y Hellman, como mencionamos arriba.

Supongamos que tenemos $C = E_{K_2}(E_{K_1}(P))$. Dada la pareja P y C (es decir, se trata de un ataque de texto claro conocido), calculamos los 2^{55} posibles resultados de $E_{K_i}(P)$. Después calculamos todos los $E^{-1}_{K_j}(C)$, y los comparamos con los $E_{K_i}(P)$ que calculamos en el paso anterior. Cuando encontramos una coincidencia, obtenemos una pareja (K_i, K_j) que corresponde a $E^{-1}_{K_j}(C) = E_{K_i}(P)$; luego usamos esa pareja con otra pareja de texto-criptotexto (P_2, C_2), y eso detecta si nuestra pareja de llaves es la correcta. El tiempo de

proceso es el mismo que el necesario para romper una sola aplicación de DES.

El cifrado triple no parece padecer del defecto del cifrado doble. El estándar de cifrado triple consiste en elegir tres llaves, K_1, K_2, y K_3, y calcular

$$E_{K_3}(E_{K_2}^{-1}(E_{K_1}(P))).$$

La razón por la que se aplica el inverso en la segunda aplicación de DES es para que el algoritmo sea compatible con el cifrado simple de DES: si tomamos $K_1 = K_2$, el resultado es simplemente $E_{K_3}(P)$.

6.4.4 Número de Rondas

Después de cinco rondas de DES, cada bit de la salida depende de cada bit de la entrada. Después de ocho rondas, cada bit de la salida es esencialmente una función aleatoria de cada bit de entrada y cada bit de la llave. ¿Entonces, por qué se usan 16 rondas? La respuesta a esa pregunta no se conoció públicamente hasta que se conoció el criptoanálisis diferencial, que estudiaremos en el siguiente capítulo.

Hay principalmente dos ataques de criptoanálisis analítico contra DES: el criptoanálisis diferencial, y el criptoanálisis lineal.

En este capítulo vamos a describir ambos ataques. Puesto que son algo complicados, vamos a introducir una red de Feistel simplificada primero, sobre la cual vamos a aplicar ambos criptoanálisis de manera ilustrativa.

7.1 Una red de Feistel simplificada: RFS

Para poder explicar más fácilmente el criptoanálisis de DES, vamos a usar una red de Feistel inventada para dicho objetivo. No se trata de un sistema criptográficamente seguro, sino de un sistema elegido para explicar el criptoanálisis. El sistema y la aplicación del criptoanálisis están tomados del artículo de Howard Heys [Hey]. Le vamos a llamar RFS

("Red de Feistel Simplificada").

Nuestra red es una red de 4 rondas; las primeras tres rondas operan primero sumando la llave correspondiente, mediante una suma booleana bit a bit; y antes de terminar también sumamos una llave para que el criptoanalista no pueda simplemente "deshacer" la última ronda. Asumimos que los bits de cada llave son generados independientemente. Luego aplicamos la porción no lineal de la ronda (las cajas S), y finalmente una permutación. La última ronda no tiene la permutación.

L RFS trabaja sobre bloques de 16 bits; cada bloque se divide en cuatro sub-bloques de cuatro bits cada uno, y cada sub-bloque es la entrada de una caja S que no es lineal; es decir, los bits de salida no son una operación lineal de los bits de entrada.

Vamos a usar la misma función para todas nuestras cajas S (en DES, cada caja S es diferente, aunque todas las rondas usan las mismas cajas S). La función que define la caja S es la siguiente, que es una cada S parcial de las usadas en DES:

Entrada	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Salida	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Al final de las rondas 1, 2, y 3, tenemos una permutación de los bits. La permutación va a ser la misma en cada ronda. Si 1 representa el bit de hasta la izquierda, y 16 el bit de hasta la derecha, la permutación está dada explícitamente por:

Entrada	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Salida	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

La permutación se puede también dar de una manera un poco más sencilla: la salida i de la j-ésima caja S se conecta a la entrada j de la i-ésima caja S de la siguiente ronda. Un diagrama de la Red de Feistel Simplificada está en la Figura 7.1.

7.2 Criptoanálisis Diferencial

El primer avance importante en el criptoanálisis de DES fue el trabajo de Biham y Shamir [BS91], publicado por primera vez en 1990. Se trata del criptoanálisis diferencial, que es un ataque útil no sólo contra DES, sino contra cualquier red de Feistel. De hecho, el ataque fue originalmente presentado como un ataque contra "sistemas tipo DES" pues en su encarnación original no era práctico contra DES. La idea, sin embargo, era claramente importante.

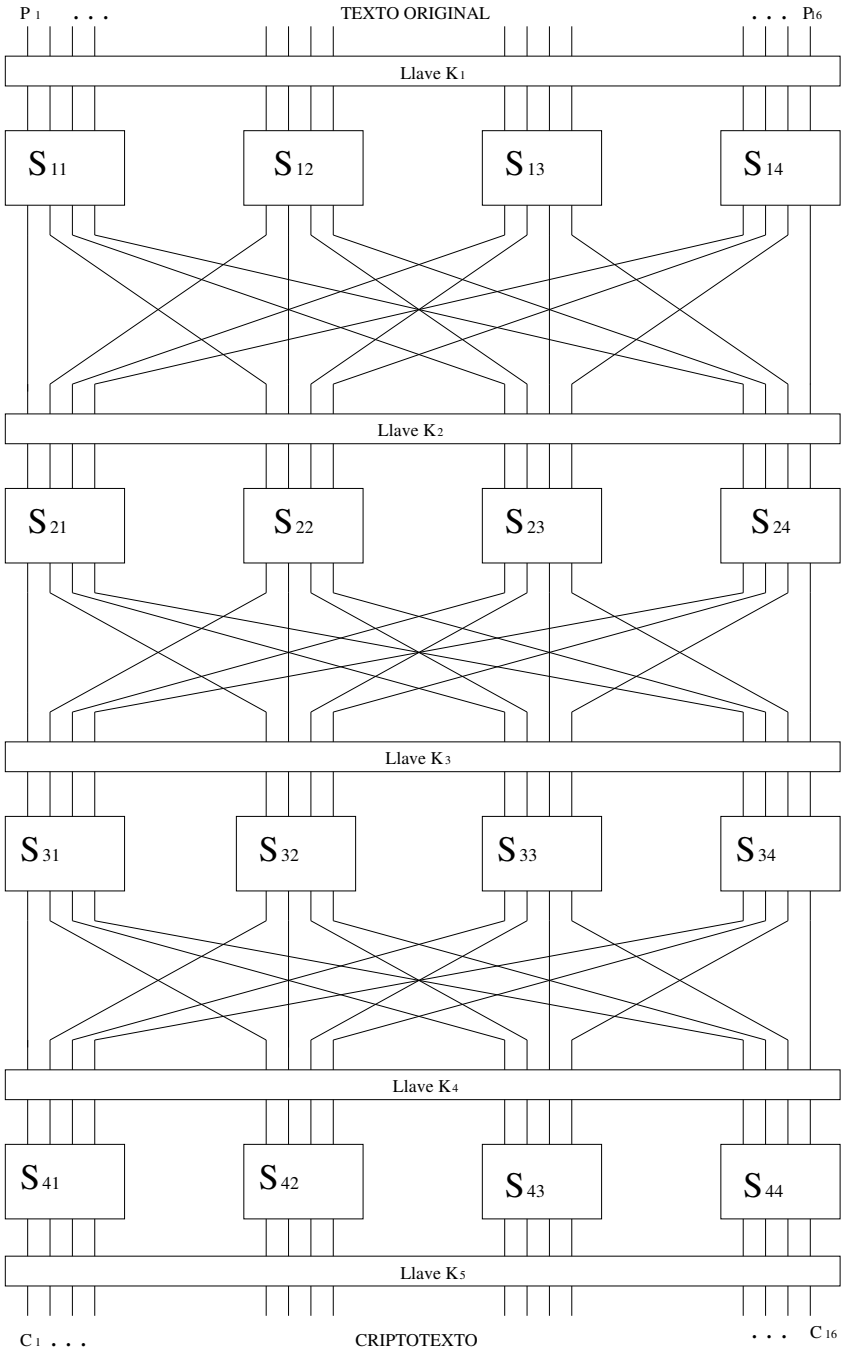

Figura 7.1: Una red de Feistel simplificada

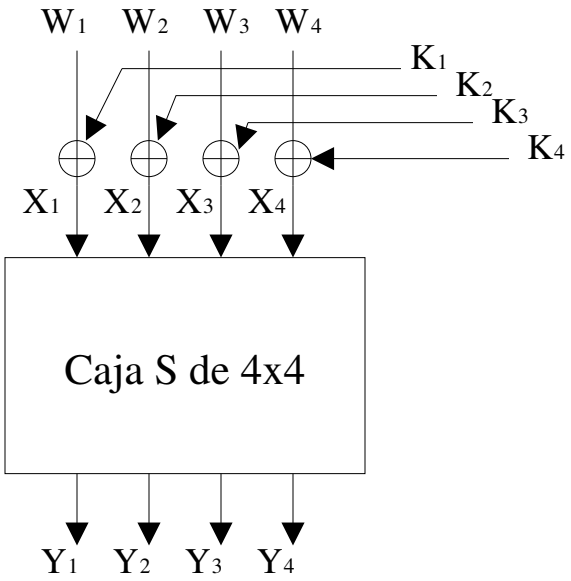

Figura 7.2: Caja S con el efecto de la llave en la entrada

Posteriormente, el ataque fue mejorado, y en 1993, Biham y Shamir publicaron un libro [BS93b] donde el ataque fue mejorado significativamente, al grado en que ya era aplicable (al menos en teoría) a DES.

El criptoanálisis diferencial es un ataque de texto claro elegido; es decir, podemos escoger el texto a cifrar,y tenemos acceso al texto cifrado resultante (e.g. tenemos un chip resistente que cifra). El objetivo es recuperar la llave que se está utilizando para cifrar.

El criptoanálisis diferencial busca explotar la correlación entre diferencias en el texto original, y diferencias entre las salidas. Para ser más explícitos, supongamos que tenemos una red de Feistel de bloques de tamaño $2t = n$, y como en el caso de DES y la mayoría de las redes, el efecto de la llave es un XOR con el texto antes de aplicar la parte no lineal de la función f.

Supongamos que tenemos dos entradas a la ronda,

$$\begin{aligned} X &= [X_1, \ldots, X_n] \\ X' &= [X'_1, \ldots, X'_n] \end{aligned}$$

(es decir, X_i es el i-ésimo bit de X; X'_i es el i-ésimo bit de X')[1]. La "diferencia" entre X y

[1]Nótese bien que la numeración es la opuesta a la que especificamos para DES.

X' se define mediante la suma booleana:

$$\Delta X = \Delta(X, X') = X \oplus X'.$$

Si el resultado de cifrar X es $Y = [Y_1, \ldots, Y_n]$, y el resultado de cifrar X' es $Y' = [Y'_1, \ldots, Y'_n]$, entonces la diferencia entre las salidas es:

$$\Delta Y = \Delta(Y, Y') = Y \oplus Y'.$$

Si nuestro cifrado fuera ideal, la probabilidad de que una diferencia de salidas $\Delta(Y)$ corresponda a una diferencia particular de entradas $\Delta(X)$, debería ser $\frac{1}{2^n}$ (excepto cuando $\Delta(X, X') = 0$, en cuyo caso debe ser que la probabilidad de $\Delta(Y, Y') = 0$ sea igual a 1). El criptoanálisis diferencial busca utilizar los casos en que una cierta $\Delta(Y, Y')$ particular ocurre respecto a una $\Delta(X, X') \neq 0$ particular con una probabilidad p, con

$$p \gg \frac{1}{2^n}.$$

El símbolo $\gg$ se lee "mucho mayor que". Una pareja $(\Delta X, \Delta Y)$ que satisface esta condición se llama un **diferencial** de la ronda en cuestión. Un diferencial es un *diferencial de k rondas* si funciona después de la aplicación de k rondas.

¿Cómo se usa un diferencial para atacar una red de Feistel como DES? Se usa para encontrar parte de la llave usada en la red. En el caso de DES, si logramos recuperar la última llave usada, una búsqueda exhaustiva nos encontraría los bits faltantes de la llave completa.

Supongamos que tenemos una red de Feistel de r rondas, y un diferencial $(\Delta X, \Delta Y)$ de $r - 1$ rondas para esta red.

Lo que hacemos es tomar mucha parejas X, X' tales que $\Delta(X, X') = \Delta X$, el diferencial buscado; con ellas obtenemos parejas Y, Y', y las correspondientes mitades (I_r, D_r) y (I'_r, D'_r). En teoría, conocemos con alta probabilidad la diferencia de los resultados de la ronda anterior; es decir, conocemos con alta probabilidad (dada por el diferencial),

$$\Delta(I_{r-1}, I'_{r-1}) \text{ y } \Delta(D_{r-1}, D'_{r-1});$$

nos fijamos a ver si la mitad izquierda de Y y de Y' corresponden al $\Delta(D_{i-1}, D'_{i-1})$ que "conocemos." En caso afirmativo, decimos que X y X' (y también Y y Y') son un *buen par,* y que son un *mal par* en caso contrario.

Tomamos un "buen par", y lo tratamos de descifrar a traves de la última ronda; es decir, buscamos las posibles llaves K_r tales que nuestra salida hubiera provenido de salidas

(I_{r-1}, D_{r-1}) e (I'_{r-1}, D_{r-1}) que tengan la diferencia esperada. Sólo ciertas llaves pueden dar nuestras D_r y D'_r, al tiempo que guardan la diferencia buscada en la ronda $r-1$. Mantenemos una cuenta de cuáles llaves funcionan, y repetimos con otros buenos pares.

Lo que esperamos es que la llave correcta resulte posible en la mayoría de los casos, pues la probabilidad de la característica es alta. Las otras posiblidades de llaves ocurren con probabilidad aleatoria, distribuidas uniformemente. Entonces, una vez que hayamos obtenido suficientes buenas parejas, la verdadera llave debe sobresalir de nuestra cuenta por encima del ruido de el resto de las posibles llaves. Finalmente, una búsqueda exhaustiva demuestra y encuentra los bits faltantes de la llave.

El ataque es un ataque de "umbral." Si no examinamos suficientes pares buenos, el ruido de las llaves falsas es demasiado para detectar la verdadera llave. En general, el número de parejas que se requieren examinar es inversamente proporcional a la probabilidad asociada a la característica: entre mayor probabilidad, menos parejas.

Las preguntas que tenemos que contestar ahora son: ¿Existen las características? Y en caso afirmativo, ¿Cómo las encontramos? Otra pregunta importante es: ¿Qué tan efectivo es el ataque contra DES?

Debido al efecto de difusión, lo que busca uno para construír características es trabajar de ronda en ronda, y luego componer los resultados. La ventaja es que es mucho más fácil trabajar con una sóla ronda que con varias. La desventaja es que si tenemos una característica con probabilidad p_1 en la primer ronda, y una con probabilidad p_2 en la segunda que se puede componer con la primera, entonces la composición tiene probabilidad $p_1 p_2 < p_1, p_2$ (a menos que una de ellas tenga probailidad 1). En el caso de DES y otras redes de Feistel, tenemos que empezar con un ΔX que sea muy "chico", es decir, muy pocos unos y muchos ceros. La idea es que nos tenemos que concentrar en el menor número posible de cajas S para minimizar los problemas. El artículo original de Biham y Shamir, por ejemplo, sugería una característica para DES de tres rondas que tenía únicamente tres unos del lado izquierdo y dos del lado derecho (una diferencia de 5 bits, de un total de 64).

La razón por la cual queremos una red de Feistel donde la acción de la llave sea un XOR es que dicha red "respeta" diferenciales: la diferencia entre los mensajes al entrar a la ronda es igual a la diferencia después de haber sumado las llaves: si tenemos X y X', y llave K, entonces

$$\begin{aligned} \Delta(X, X') &= X \oplus X' \\ &= (X \oplus K) \oplus (X' \oplus K) \\ &= \Delta(X \oplus K, X' \oplus K), \end{aligned}$$

y podemos analizar las diferencias sin preocuparnos del efecto de la llave.

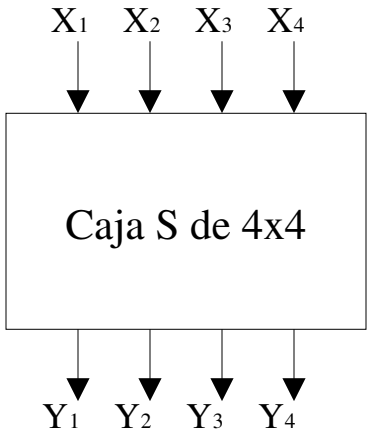

Figura 7.3: La caja S con los bits de entrada y salida numerados.

7.2.1 Criptoanalisis Diferencial de RFS

Para ilustrar cómo se encuentran características, por qué deben existir, y cómo se usan para hacer criptoanálisis a DES, vamos aplicar el criptoanálisis diferencial a nuestra RFS.

Para poder construir características con probabilidad alta, tenemos que examinar las propiedades de nuestras cajas S, y usar esas propiedades para determinar las propiedades diferenciales totales. Específicamente, consideramos todas las posibles diferencias de entradas y salidas de nuestras cajas S, para determinar un par con alta probabilidad. Después combinamos pares de ronda en ronda para que un diferencia distinta de cero en la salida de una ronda corresponda a una diferencia en la entrada de la siguiente.

Primero examinamos las parejas $(\Delta X, \Delta Y)$ relacionadas con nuestra caja S. Numeramos los bits de entrada como $X = [X_1\, X_2\, X_3\, X_4]$ de izquierda a derecha, y los de salida como $Y = [Y_1\, Y_2\, Y_3\, Y_4]$, también de izquierda a derecha. Véase la Figura 7.3.

Podemos examinar todas las parejas de diferencias $(\Delta X, \Delta Y)$, y calcular la probabilidad de que ΔY resulte dado que ΔX es la diferencia de entrada; es decir, consideramos todas las posibles parejas (X', X'') de entradas tales que $\Delta(X', X'') = X' \oplus X'' = \Delta X$, y analizamos cuántas veces resulta que $\Delta(Y', Y'') = \Delta Y$. Puesto que el orden en que consideramos la pareja (X', X'') no importa, para cada valor de ΔX basta considerar 16 pares: por cada posible valor de X', simplemente escogemos $X'' = X' \oplus \Delta X$.

Por ejemplo, consideremos algunos valores de X y Y, y los correspondientes valores de ΔY dada la pareja $(X, X \oplus \Delta X)$, en la Tabla 7.1. La primer columna tiene el valor de X, pasando por las 16 posibilidades. La segunda columna tiene el resultado de aplicar la caja

X	Y	ΔY		
		$\Delta X = 1011$	$\Delta X = 1000$	$\Delta X = 0100$
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

Tabla 7.1: Ejemplos de pares de diferencias para la Caja S

S a X; las últimas tres columnas tienen tres valores de ΔY, dependiendo de si ΔX vale 1011, 1000, ó 0100, respectivamente.

De la Tabla 7.1 podemos ver, por ejemplo, que $\Delta Y = 0010$ occurre 8 de 16 veces cuando $\Delta X = 1011$; es decir, la probabilidad de que $\Delta Y = 0010$ dado que $\Delta X = 1011$ es de $8/16$, o un medio, mucho más alto que $1/2^4 = 1/16$, que es la probabilidad esperada en un sistema completamente aleatorio. Por otro lado, la probabilidad de obtener $\Delta Y = 1010$ dado un $\Delta X = 0100$ es 0.

Podemos tabular la información completa para la caja S en una *Tabla de Distribución de Diferencias*, donde los renglones corresponden a los valores de ΔX (en hexadecimal, en nuestro ejemplo), y las columnas los valores de ΔY. Cada elemento de la tabla representa el número de veces que la diferencia de salidas ΔY corresponde a la diferencia de entradas ΔX; de manera que la probabilidad de una ΔY dada ΔX es el número en el renglón de ΔX y la columna de ΔY, dividida entre el número total de parejas (en este caso, 16). La tabla para nuestra caja S está en la Tabla 7.2

Notemos que la suma de cada entrada en un mismo renglón debe ser 16, pues hay 16 parejas. De manera similar, la suma de las entradas en cada columna debe ser 16 también.

		Diferencia de salida ΔY															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
	2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
	3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
	4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
	5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
	6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
Δ	7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
X	8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
	9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
	A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
	B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
	C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
	D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
	E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
	F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Tabla 7.2: Tabla de Distribución de Diferencias

Además, puesto que la caja S es una función, el primer renglón, correspondiente a $\Delta X = 0$, debe tener un 16 en en la columna de $\Delta Y = 0$, y 0 en el resto de las entradas; lo mismo con la primer columna. Pensemos en el resto de las entradas (fuera del primer renglón y primera columan) como la "parte principal de la tabla."

Una "distribución ideal" sería aquella en la cual para cualquier ΔX distinto de 0 y cualquier ΔY distinta de cero, la probabilidad de que ΔY ocurra dado ΔX fuese 1/16; es decir, que la tabla tuviera puros unos en la parte principal.

Pero puesto que la suma de cada renglón y de cada columna debe ser igual a 16, y el primer renglón y la primer columna tienen ceros excepto en la primer entrada, alguna entrada en la parte principal de cada renglón y de cada columna debe ser mayor que 1. Más aún, cada entrada debe ser par, pues ΔX corresponde en realidad a dos parejas distintas, $(X, X \oplus \Delta X)$ y $(X \oplus \Delta X, X)$. De manera que una "caja S ideal" es imposible.

En nuestro ejemplo, la probabilidad más alta no trivial (es decir, en la parte principal de la tabla) ocurre con $\Delta X =$ B, y $\Delta Y = 2$, con probabilidad $8/16 = 1/2$.

Supongamos entonces que elegimos dos mensajes cuya diferencia de entrada sea $\Delta P =$ [`0000 1011 0000 0000`] (véase la Figura 7.4 para referencia). En las cajas S_{11}, S_{13}, y S_{14},

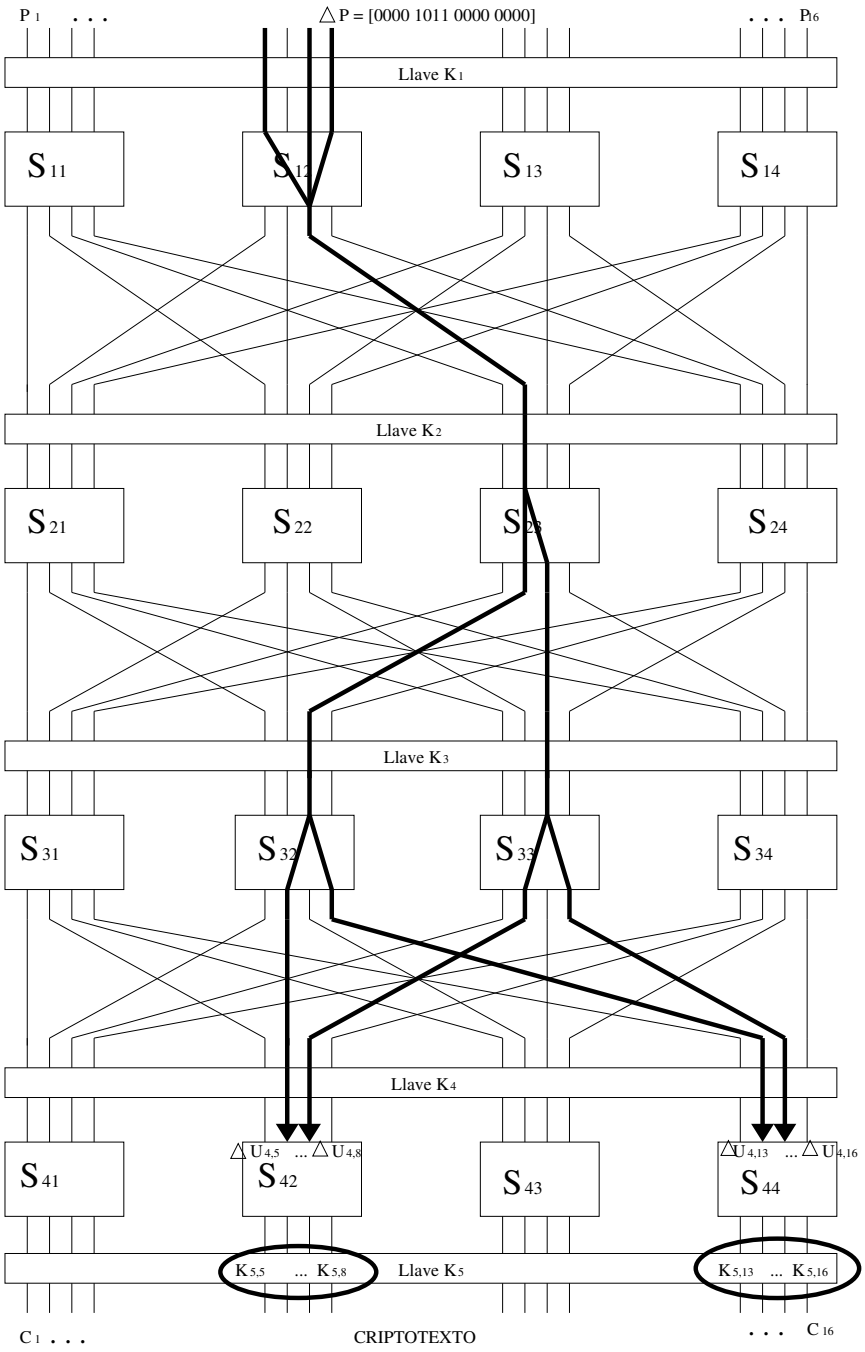

Figura 7.4: Ejemplo de criptoanálisis diferencial con la RFS

no hay diferencia en la salida; y en la caja S_{12}, con probabilidad 1/2 la diferencia de salida es 0010. Es decir, al iniciar la segunda ronda, con probabilidad 1/2 la única diferencia entre los dos mensajes está en el segundo bit de entrada de S_{23}; es decir, la diferencia en la entrada de esa caja es 0100. Buscando en el renglón de $\Delta X = 4$, vemos que con probabilidad 6/16 la diferencia de salida es $\Delta Y = 0110$. Es decir, con probabilidad $(1/2) * (6/16) = 3/16$, dado que la diferencia en la entrada era ΔP, al empezar la tercer ronda la diferencia es [0000 0010 0010 0000] (después de la permutación).

Ahora tenemos la misma diferencia en la entrada de dos cajas, S_{32} y S_{33}; en ambos casos, $\Delta X = 2$. Buscando en el renglón de 2, vemos que la mayor probabilidad se alcanza con $\Delta Y = 5$, con probabilidad 6/16 en cada caja. Entonces, con probabilidad

$$\left(\frac{3}{16}\right)\left(\frac{6}{16}\right)\left(\frac{6}{16}\right) = \frac{27}{1024}$$

la diferencia al empezar la cuarta ronda es

$$\Delta C = [0000\ 0110\ 0000\ 0110].$$

Esto nos da un diferencial de tres rondas, a saber, la pareja:

$$(\Delta P, \Delta C) = (0B00, 0606)$$

que corresponden a la diferencia de entrada de mensajes, y la diferencia al empezar la cuarta ronda (es decir, al terminar la tercera); este diferencial de tres rondas tiene probabilidad $27/1024 \gg 1/2^{16}$.

Para hacer criptoanálisis, elegimos muchas parejas de textos para los cuales $\Delta P = 0B00$; aquellos pares en las cuales la diferencia al terminar la tercer ronda sea 0606 son los "pares buenos", y el resto son los "pares malos."

Por la construcción de nuestra RFS, estas diferencias afectan sólo los bits 5–8 y 13–16 de la salida; es decir, nos dan posible información sobre los bits 5–8 y 13–16 de la última llave aplicada: K_5.

Para recuperar esos bits de la llave K_5, procedemos de la siguiente manera: tomamos un par de textos con diferencia ΔP, y obtenemos el resultado de aplicar la RFS completa. Luego hacemos un desciframiento parcial de la última ronda para ver si es probable que lo que teníamos es un "par bueno." Para ello, probamos, para cada bit influenciado por una caja S afectada por la diferencia esperada en la tercera ronda, una llave posible que, si "hacemos pasar" nuestro criptotexto resultante hacia atrás por la última llave y por las cajas S, nos resulte en la diferencia ΔC esperada.

Por cada pareja de textos cuya diferencia es ΔP, obtenemos de esta manera varios posibles valores para los bits involucrados de la llave K_5. Cada vez que un valor específico de esos bits hace que el resultado corresponda a la diferencia esperada ΔC al final de la tercer ronda, aumentamos un contador asociado a ese valor. Repetimos para un número grande de parejas de textos con diferencia ΔP. Al final, el valor asociado a cada posible subllave es el número de veces en que las diferencias obtenidas son consistentes con pares buenos (asumiendo que la subllave elegida sea correcta).

Se espera que la subllave correcta tenga una alta probabilidad de dar pares buenos, mientras que subllaves incorrectas dan resultados uniformemente distribuidos. Una vez que se tenga suficientes datos, la verdadera subllave sobresale por encima del ruido.

Por ejemplo, si tomamos 5000 parejas de textos con diferencia ΔP (es decir, producimos 10000 cifrados), la subllave correcta se espera sobresalga con probabilidad casi el doble que cualquier otra (y en algunos casos, mucho más).

Hay un par de consideraciones que se deben hacer: nuestro cálculo de probabilidad asume que el proceso de cada ronda es independiente uno de otro; en realidad no es independiente, pues las rondas están relacionadas unas con otras. Sin embargo, la relación es suficientemente débil que afecta la probabilidad sólo de manera marginal. Por ejemplo, el valor de probabilidad calculado arriba es de $27/1024 = 0.0264$, y un experimento obtuvo un valor muy cercano: 0.0244.

En general, esta es la idea del criptoanálisis diferencial: para atacar una red de Feistel de R rondas, se busca una diferencial de $R-1$ rondas y se utiliza para encontrar los valores de la última llave. En el caso de DES, la última llave proporciona casi todos los bits de la llave, y el resto se buscan mediante fuerza bruta.

Particularmente útiles son características de una ronda que se pueden componer consigo mismas para obtener características de varias rondas.

Entre más rondas tiene una red de Feistel, más complicado es el análisis. Entre mayor difusión rápida tenga, también más complicado es el ataque, pues rápidamente se involucran varias cajas S en las consideraciones de diferencias.

7.2.2 Resistencia al Criptoanálisis Diferencial

Es difícil decir si un sistema es resistente al criptoanálisis diferencial. Lo que se busca para fortalecer el sistema contra éste ataque es, en términos generales, que el número de cajas que están "activas" dada una diferencia específica sea grande (es decir, el número de cajas S involucradas, dada una diferencia, sea grande); y se busca diseñar las cajas S de

manera que se minimize la probabilidad de los pares de diferencia; es decir, que la Tabla de Distribución de Diferencias tenga entradas relativamente chicas en su parte principal.

Cuando el criptoanálisis diferencial salió a la luz pública, Coppersmith hizo públicos por primera vez los criterios que se habían utlizado para el diseño de las cajas S de *Lucifer* y de DES. Los criterios eran:

1. Ningún bit de salida debe estar muy cerca de una función lineal de los bits de entrada.
2. Si fijamos los bits de la derecha e izquierda de la entrada, y variamos los cuatro centrales, cada posible salida ocurre exactamente una vez.
3. Si dos entradas a la caja difieren exactamente en un bit, entonces las salidas difieren en al menos dos bits.
4. Si dos entradas difieren en exactamente los dos bits del centro, las salidas difieren en al menos dos bits.
5. Si dos entradas difieren en los primeros dos bits y son idénticas en los últimos dos bits, las salidas no deben ser iguales.
6. Para cualquier diferencia distinta de cero entre las entradas, no más de ocho de los treinta y dos pares con esa diferencia pueden tener la misma diferencia de salida.
7. Semejante al inciso anterior, para el caso de tres cajas S activas.

Como se podrá ver, varios de los criterios (por ejemplo, los incisos 3, 4, 5, y 6) están pensados precisamente para dificultar el criptoanálisis diferencial. Esto ya lo habían sospechado Biham y Shamir, que experimentaron con cambiar el orden de las cajas, o reemplazarlas por cajas generadas de manera aleatoria, y verificaron que la red de Feistel resultante era en general mucho más sensible y suceptible al criptoanálisis diferencial que DES.

De pronto, se entendió por qué la NSA cambió las cajas S: las de *Lucifer* eran sustancialmente más débiles ante el criptoanálisis diferencial. La evidencia indica que la NSA conocía el criptoanálisis diferencial varios años antes de que fuera re-descubierto por Biham y Shamir, y que se buscó optimizar DES contra él.

La resistencia de DES contra el criptoanálisis diferencial se puede mejorar aumentando el número de rondas. A medida que aumenta el número de rondas, aumenta el número de textos que se requieren para el análisis. Para DES de 17 ó 18 rondas, se require más o menos el mismo número de textos que lo que se requiere para un ataque de fuerza bruta. Con 19 o más rondas, el criptoanálisis diferencial se vuelve imposible, pues requiere más de

2^{64} textos elegidos; pero DES tiene un tamaño de bloque de 64 bits, de manera que sólo tiene 2^{64} textos posibles. En general, uno puede probar que un algoritmo es resistente al criptoanálisis diferencial si se puede demostrar que la cantidad de textos elegidos necesarios para montar el ataque es mayor que el número de textos posibles.

7.3 Criptoanálisis Lineal

Si bien parece que el diseño de DES tenía muy claro y presente el riesgo del criptoanálisis diferencial, es más o menos evidente que el criptoanálisis lineal de Mitsuru Matsui no estaba contemplado. En un artículo [Mat94], Matsui propuso un nuevo ataque contra DES y otras redes de Feistel.

A diferencia del criptoanálisis diferencial, que es un ataque de texto elegido, el criptoanálisis lineal es un ataque de texto conocido, que se puede adaptar a un ataque de criptotexto conocido a cambio de un aumento en la complejidad.

La idea básica del criptoanálisis lineal consiste en aproximar partes del criptosistema mediante expresiones lineales. Si $X = [X_1 \dots X_n]$ es la entrada y $Y = [Y_1 \dots Y_n]$ es la salida, bajo *expresiones lineales* nos referimos a expresiones de la forma

$$X_{i_1} \oplus \cdots \oplus X_{i_u} \oplus Y_{j_1} \oplus \cdots \oplus Y_{j_v} = 0.$$

El criptoanalista busca expresiones como éstas que tengan una alta o baja probabilidad de ser ciertas para el criptosistema en cuestión. Si nuestro sistema fuera completamente aleatorio, se esperaría que la ecuación fuese válida con una probabilidad de exactamente $1/2$. La desviación de la probabilidad real de $1/2$ es el *sesgo lineal*. Es decir, si la expresión es cierta con probabilidad p_L, entonces el sesgo lineal de la expresión es $|(1/2) - p_L|$. Entre más grande sea el sesgo lineal, mejor para el criptoanálisis lineal.

Otra manera de expresar la ecuación sería con los bits de la llave involucrados todos del lado derecho de la igualdad. Pero si estos bits suman 0, no pasa nada; si suman 1, la probabilidad de la expresión es simplemente el complemente de la probabilidad de antes (es decir, $1 - p_L$), y en ambos casos el sesgo lineal no cambia. De manera que nuevamente podemos ignorar el efecto de la llave para calcular el sesgo de una expresión.

Si $p_L = 1$, entonces la expresión es lineal, y es muy fácil atacar el sistema. Si, por el contrario, tenemos que $p_L = 0$, entonces la expresión es el complemento de una expresión lineal. Cuando trabajamos módulo dos, el complemente de una expresión lineal se llama una *expresión afín*. Un sistema que tiene un expresión afín nuevamente sufre de una debilidad catastrófica, exactamente equivalente a la debilidad de una expresión lineal.

Si $p_L > 1/2$, lo que tenemos es una aproximación lineal; si $p_L < 1/2$, tenemos una aproximación afín. En ambos casos, el sesgo lineal produce una suceptibilidad al criptoanálisis lineal. Puesto que en las redes de Feistel la no linealidad depende exclusivamente de las cajas S, nos concentramos en las cajas S para buscar aproximaciones.

Hay muchas maneras de montar el ataque; nos vamos a concentrar en una de ellas, que Matsui llama el Algoritmo 2.

7.3.1 Usando las expresiones lineales

¿Cómo usamos una expresión lineal, suponiendo que ya la tenemos, para el criptoanálisis lineal?

Consideremos el caso de nuestra RFS. Supongamos que hemos encontrado una expresión

$$X_{i_1} \oplus \cdots \oplus X_{i_u} \oplus Y_{j_1} \oplus \cdots \oplus Y_{i_v} = 0$$

con sesgo lineal ε para las primeras tres rondas. Lo que hacemos es tomar varios textos $P_1, \ldots, P_r$ y sus criptotextos correspondientes, $C_1, \ldots, C_r$.

La idea es nuevamente muy parecida a lo que hicimos con el criptoanálisis diferencial. Tomamos los criptotextos C_i, y hacemos un desciframiento parcial por la última ronda, buscando aquellos valores de la llave K_5 que nos corresponden a resultados en los cuales la expresión es válida (si se trata de una aproximación lineal) o inválida (si se trata de una proximación afín). Llevamos la cuenta de cuáles llaves nos dan el resultado buscado. Lo que esperamos es que la llave correcta (o más exactamente, los bits involucrados en la llave correcta) sobresalgan, pues aparecerán como correctos en una mayoría de los casos, mientras que otras posibilidades apareceran de manera uniforme y aleatoria. Es decir, esperamos que la llave correcta aparezca con probabilidad significativamente distinta de $1/2$ (cerca de 0 si se trata de una aproximación afín; cerca de 1 en el caso de una aproximación lineal). Una llave incorrecta, por otro lado, funciona de manera relativamente aleatoria, y esperamos que la probabilidad sea muy cercana a $1/2$. Dados suficientes textos P_i y sus criptotextos C_i, la verdadera llave se distingue de las falsas por su frecuencia.

7.3.2 Complejidad del ataque

Matsui demostró que en general, si ε corresponde al sesgo lineal de la expresión, el número de textos conocidos necesarios para montar el ataque es inversamente proporcional a ε^2; es decir, $N \approx 1/\varepsilon^2$.

Para montar el ataque con criptotexto conocido, necesitamos expresiones que que involucren sólo una X_i; normalmente corresponden a valores mucho más pequeños de sesgo lineal, y por tanto aumenta la N considerablemente.

7.3.3 Tabla de Aproximación Lineal

Igual que con el criptoanálisis diferencial, normalmente estudiamos las propiedades lineales de las cajas S por separado, y luego tomamos composiciones de nuestros resultados a lo largo de las ronda.

Para estudiar la caja S de la RFS, contruimos una *Tabla de Aproximación Lineal*, que aparece en la Tabla 7.4. Esta tabla contiene la información sobre los sesgos lineales de las distintas aproximaciones a la caja S.

Por ejemplo, para la caja S de nuestra RSF, consideremos la expresión lineal

$$X_2 \oplus X_3 \oplus Y_1 \oplus Y_3 \oplus Y_4 = 0$$

o equivalentemente

$$X_2 \oplus X_3 = Y_1 \oplus Y_3 \oplus Y_4.$$

Calculamos los 16 posibles valores de $X = [X_1X_2X_3X_4]$ y examinamos los valores de salida para ver si la expresión es cierta. Como se ve en la Tabla 7.3, la relación es cierta para 12 de 16 casos, y por ello el sesgo lineal es $12/16 - 1/2 = 1/4$. De manera similar, la ecuación

$$X_1 \oplus X_4 = Y_2$$

tiene sesgo lineal 0, y la ecuación

$$X_3 \oplus X_4 = Y_1 \oplus Y_4$$

tiene sesgo lineal $2/16 - 1/2 = -3/8$. En el último caso, la mejor aproximación es una aproximación afín, como lo indica el signo.

Para construír la Tabla de Aproximación Lineal, consideramos una expresión de la forma:

$$a_1X_1 \oplus a_2X_2 \oplus a_3X_3 \oplus a_4X_4 \oplus b_1Y_1 \oplus b_2Y_2 \oplus b_3Y_3 \oplus b_4Y_4 = 0$$

con $a_i, b_j \in \{0, 1\}$. Estos coeficientes corresponden a parejas de números entre 0 y 15, escritos en binario, $(a_1a_2a_3a_4, b_1b_2b_3b_4)$. Por ejemplo, la expresión

$$X_1 \oplus X_4 \oplus Y_2 \oplus Y_3 \oplus Y_4 = 0$$

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4	$X_2 \oplus X_3$	$Y_1 \oplus Y_3 \oplus Y_4$	$X_1 \oplus X_4$	Y_2	$X_3 \oplus X_4$	$Y_1 \oplus Y_4$
0	0	0	0	1	1	1	0	0	0	0	1	0	1
0	0	0	1	0	1	0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	1	1	0	0	1
0	1	0	0	0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	1	1	1	1	1	1	1	1	0
0	1	1	0	1	0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	0	0	0	1	1	0	0	1
1	0	0	0	0	0	1	1	0	0	1	0	0	1
1	0	0	1	1	0	1	0	0	0	0	0	1	1
1	0	1	0	0	1	1	0	1	1	1	1	1	0
1	0	1	1	1	1	0	0	1	1	0	1	0	1
1	1	0	0	0	1	0	1	1	1	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1	0	1	0
1	1	1	1	0	1	1	1	0	0	0	1	0	1

Tabla 7.3: Aproximaciones Lineales para la Caja S de la RSF

corresponde a la pareja $(1001, 0111)$.

En la tabla, los renglones corresponden al primer número, las columnas al segundo número. Por ejemplo, la entrada en el renglón 2, columna A, corresponde a la pareja $(0010, 1010)$, es decir a la expresión

$$X_3 \oplus Y_1 \oplus Y_3 = 0.$$

En la entrada en el renglón 2, columna A, escribimos el número de entradas a la caja S $[X_1X_2X_3X_4]$, con su correspondiente salida $[Y_1Y_2Y_3Y_4]$ tales que la expresión correspondiente es cierta, **menos** 8 (la mitad de las posibilidades). La idea es que si hay 16 posibles entradas, el número en la tabla entre 16 es el valor, con signo, del *sesgo lineal* (y **no** la probabilidad de que la expresión sea válida).

Entonces tenemos que la expresión

$$X_2 \oplus X_3 \oplus Y_1 \oplus Y_3 \oplus Y_4 = 0,$$

que corresponde al renglón 6 columna B de la tabla, tiene un sesgo lineal de $4/16$; es decir, está más cerca de ser lineal que afín, y la probabilidad de que la expresión sea válida para una entrada aleatoria es $1/2+4/16 = 12/16$; esto lo vemos en la Tabla 7.3, pues la expresión es cierta en doce de los 16 casos posibles.

La Tabla de Aproximación Lineal tiene varias propiedades importantes. Entre ellas, mencionamos las siguientes:

1. El renglón correspondiente al 0 tiene un 8 en la columna que corresponde al cero, y ceros en el resto. Esto se debe a que la suma vacía siempre vale cero, de manera que la suma vacía de las entradas siempre es igual a la suma vacía de las salidas; y puesto que cada posible número aparece como salida para alguna entrada, cualquier combinación lineal no vacía de las Y_i toma el valor 1 ocho veces y el valor 0 ocho veces; eso quiere decir que una expresión que involucra únicamente a las Y_i, e involucra al menos a una Y_i, no tiene sesgo lineal. Esto da que el resto de el renglón deba valer cero.

2. La columna correspondiente al 0 tiene un ocho en el primer renglón, y ceros en el resto. Nuevamente, esto se debe a que cada combinación lineal no trivial de las X_i toma el valor 0 ocho veces, y el valor 1 ocho veces, de manera que una expresión que involucra únicamente a las X_i, e involucra al menos a una X_i, tiene un sesgo lineal de cero.

3. La suma de cada renglón y de cada columna es 8 ó -8.

		Suma de la Salida															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u	1	0	0	-2	-2	0	0	-2	6	2	2	0	0	2	2	0	0
m	2	0	0	-2	-2	0	0	-2	-2	0	0	2	2	0	0	-6	2
a	3	0	0	0	0	0	0	0	0	2	-6	-2	-2	2	2	-2	-2
	4	0	2	0	-2	-2	-4	-2	0	0	-2	0	2	2	-4	2	0
d	5	0	-2	-2	0	-2	0	4	2	-2	0	-4	2	0	-2	-2	0
e	6	0	2	-2	4	2	0	0	2	0	-2	2	4	-2	0	0	-2
	7	0	-2	0	2	2	-4	2	0	-2	0	2	0	4	2	0	2
E	8	0	0	0	0	0	0	0	0	-2	2	2	-2	2	-2	-2	-6
n	9	0	0	-2	-2	0	0	-2	-2	-4	0	-2	2	0	4	2	-2
t	A	0	4	-2	2	-4	0	2	-2	2	2	0	0	2	2	0	0
r	B	0	4	0	-4	4	0	4	0	0	0	0	0	0	0	0	0
a	C	0	-2	4	-2	-2	0	2	0	2	0	2	4	0	2	0	-2
d	D	0	2	2	0	-2	4	0	2	-4	-2	2	0	2	0	0	2
a	E	0	2	2	0	-2	-4	0	2	-2	0	0	-2	-4	2	-2	0
	F	0	-2	-4	-2	-2	0	2	0	0	-2	4	-2	-2	0	2	0

Tabla 7.4: Tabla de Aproximación Lineal

La última propiedad es importante, y quizás no sea tan claro por qué es cierta: Tomemos un renglón cualquiera. Si la entrada a la caja S no es E en hexadecimal (en cuyo caso la salida es 0), entonces la mitad de las expresiones lineales en las Y_i toman el valor 0, y la otra mitad toma el valor 1; si la entrada es E, entonces toda expresión lineal en las salidas toma el valor 0. Si estamos en un renglón que corresponde a una cierta expresión lineal de las X_i cuyo valor es 0 en la entrada, los que van a contar en ese renglón (en las distintas columnas) son los que dan 0, y los que dan 1 no cuentan; en todos los casos excepto E, esto corresponde a 8 entradas en el renglón; en el caso de E, a 16. Si al evaluar la expresión lineal en la entrada nos da 1, entonces las que cuentan son las que dan 1, y no las que dan 0. En los casos distintos de E, estamos sumando 8 entradas en el renglón; en el caso de E, sumamos cero.

Si evaluamos la expresión lineal en 1110 y nos da 0, entonces en el renglón tuvimos 8 "aciertos" por cada entrada distinta de 1110, y 16 por la entrada 1110. En total, tenemos entonces $8 * 15 + 16 = 136$ "aciertos." Después restamos 8 de cada entrada, lo cual nos deja exactamente 8 puntos en el renglón. De manera que la suma de las entradas del renglón debe ser igual a 8.

Si al evaluar la expresión lineal en 1110 nos da 1, entonces en el renglón tuvimos ocho aciertos por cada entrada distinta de E, y cero con E, para un total de $8 * 15 = 120$ aciertos. Después restamos 8 de cada entrada, y eso nos deja -8 puntos en el renglón, de manera que la suma de las entradas en el renglón debe ser igual a -8.

Un análisis similar ocurre con las columnas.

Puesto que ningún renglón y ninguna columna suma 0, cada renglón y cada columna debe tener alguna divergencia de la linealidad; es decir, es imposible construir una caja S que sea perfectamente resistente al criptoanálisis lineal, y siempre hay sesgo lineal que explotar. Es decir, siempre podemos encontrar alguna expresión lineal que tenga sesgo lineal distinto de cero para la caja S que estamos investigando.

7.3.4 Lema de Amontonamiento de Matsui

Una vez que hemos encontrado aproximaciones lineales (o afines) a las cajas S, es necesario juntarlas para encontrar el sesgo lineal del sistema completo en varias rondas; en el caso de el criptoanálisis diferencial, puesto que estabamos hablando de probabilidades, asumimos la independencia y simplemente multiplicamos las probabilidades en cada ronda para obtener el total.

En el caso del criptoanálisis lineal, el dato que tenemos es el sesgo, y el dato que buscamos es el sesgo, de manera que no es cierto que simplemente multipliquemos los resultados. El

Lema de Amontonamiento de Matsui nos proporciona una fórmula sencilla para evaluar el sesgo del criptosistema total basado en el sesgo de cada ronda.

Nuevamente, para simplificar, suponemos que las expresiones son independientes de ronda en ronda. Al igual que en el criptoanálisis diferencial, ésta suposición no es del todo cierta, pero los experimentos parecen indicar que es lo suficientemente cerca de ser cierta para que no afecte de manera perceptible los resultados.

Supongamos que tenemos dos expresiones E_1 y E_2, que puede tomar el valor 0 o el valor 1, con distinta probabilidad:

$$\begin{aligned} p(E_1 = i) &= \begin{cases} p_1 & \text{si } i = 0 \\ 1 - p_1 & \text{si } i = 1 \end{cases} \\ p(E_2 = i) &= \begin{cases} p_2 & \text{si } i = 0 \\ 1 - p_2 & \text{si } i = 1 \end{cases} \end{aligned}$$

Si E_1 y E_2 son independientes, entonces:

$$p(E_1 = i, E_2 = j) = \begin{cases} p_1 p_2 & \text{si } i = 0, j = 0 \\ p_1(1 - p_2) & \text{si } i = 0, j = 1 \\ (1 - p_1)p_2 & \text{si } i = 1, j = 0 \\ (1 - p_1)(1 - p_2) & \text{si } i = 1, j = 1 \end{cases}$$

Entonces, tenemos que

$$\begin{aligned} p(E_1 \oplus E_2 = 0) &= p(E_1 = E_2) \\ &= p(E_1 = 0, E_2 = 0) + p(E_1 = 1, E_2 = 1) \\ &= p_1 p_2 + (1 - p_1)(1 - p_2). \end{aligned}$$

Si ahora escribimos $p_1 = \frac{1}{2} + \varepsilon_1$, $p_2 = \frac{1}{2} + \varepsilon_2$, donde ε_i es el sesgo lineal, con

$$-\frac{1}{2} \leq \varepsilon_i \leq \frac{1}{2},$$

entonces, sustituyendo, tenemos:

$$\begin{aligned} p(E_1 \oplus E_2 = 0) &= ((1/2) + \varepsilon_1)((1/2) + \varepsilon_2) + ((1/2) - \varepsilon_1)((1/2) - \varepsilon_2) \\ &= (1/2) + 2\varepsilon_1\varepsilon_2. \end{aligned}$$

Es decir, el sesgo lineal de la composición $E_1 \oplus E_2 = 0$ es dos veces el producto de los sesgos lineales de las expresiones por separado. Esto nos lleva al siguiente resultado:

Lema 7.1 (Lema de Amontonamiento de Matsui) *Si tenemos n variables binarias independientes, $X_1, \ldots, X_n$, cada una con sesgo lineal ε_i (es decir, $p(X_i = 0) = (1/2) + \varepsilon_i$), entonces*

$$p(X_1 \oplus \cdots \oplus X_n = 0) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i.$$

Es decir, el sesgo lineal $\varepsilon_{1,\ldots,n}$ de la expresión compuesta es:

$$\varepsilon_{1,\ldots,n} = 2^{n-1} \prod_{i=1}^{n} \varepsilon_i.$$

Dem.: La prueba es por inducción sobre n. El caso $n = 1$ y $n = 2$ ya lo hicimos. Suponiendo válido el resultado para n, tenemos que

$$\begin{aligned}
p(X_1 \oplus \cdots \oplus X_n \oplus X_{n+1} = 0) &= p(X_1 \oplus \cdots \oplus X_n = 0, X_{n+1} = 0) + \\
&\quad\; p(X_1 \oplus \cdots \oplus X_n = 1, X_{n+1} = 1) \\
&= \left(\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i\right)\left(\frac{1}{2} + \varepsilon_{n+1}\right) + \\
&\quad \left(\frac{1}{2} - 2^{n-1} \prod_{i=1}^{n} \varepsilon_i\right)\left(\frac{1}{2} - \varepsilon_{n+1}\right) \\
&= \frac{1}{2} + 2\left(2^{n-1} \prod_{i=1}^{n} \varepsilon_i\right)\varepsilon_{n+1} \\
&= \frac{1}{2} + 2^n \prod_{i=1}^{n+1} \varepsilon_i.
\end{aligned}$$

□

Esto nos dice que componer expresiones con alto sesgo lineal tiende a producir expresiones con alto sesgo lineal, que es precisamente lo que buscamos.

7.3.5 Aproximación lineal de la RSF

Una vez que hemos analizado el sesgo lineal de las componentes de nuestra red de Feistel, procedemos a construír aproximaciones lineales de nuestro criptosistema para poder efectuar un criptoanálisis lineal de éste. La Figura 7.5 presenta lo que hacemos.

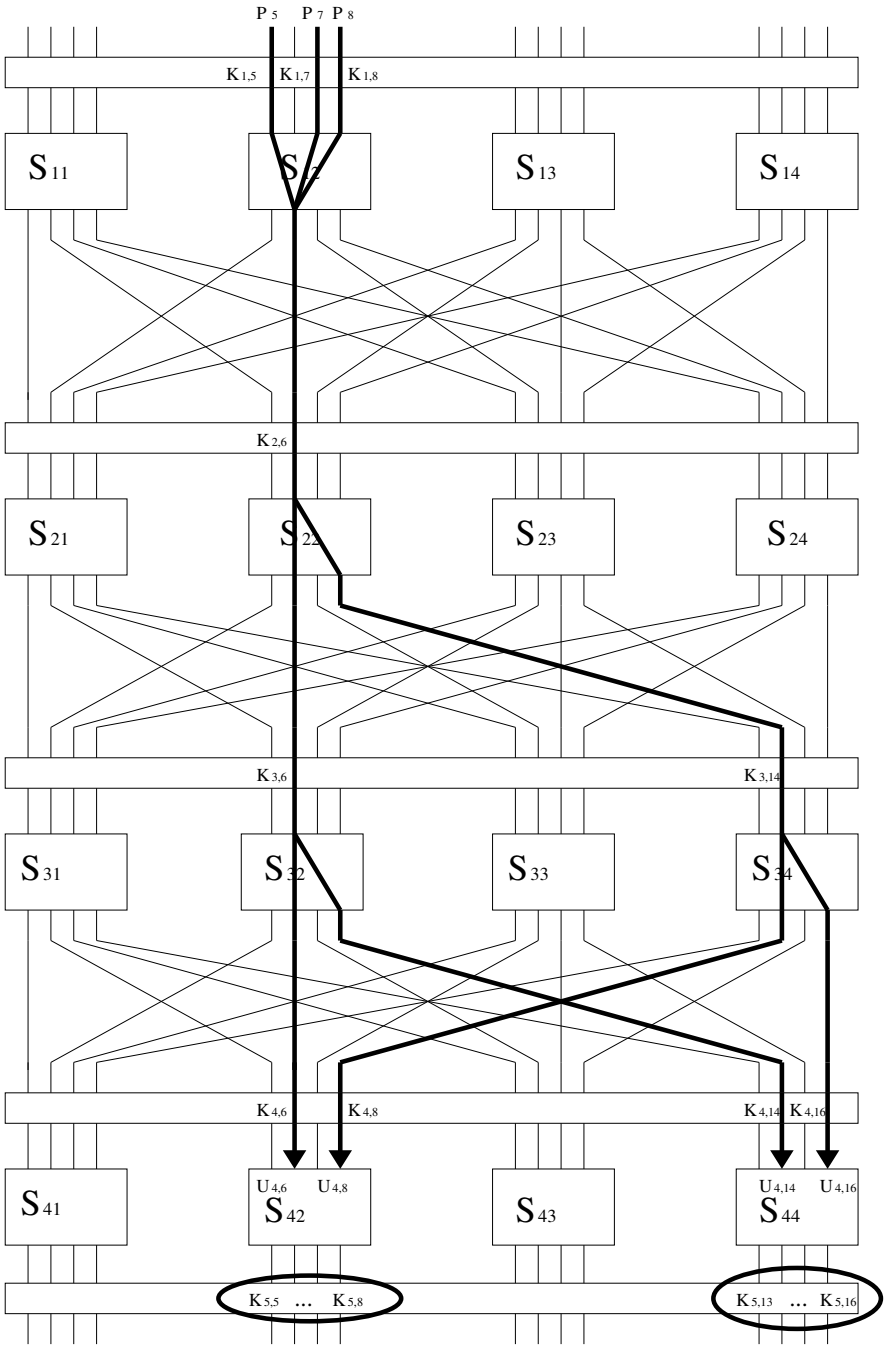

Figura 7.5: Ejemplo de criptoanálisis lineal con la RFS

Primero, buscamos en la Tabla de Aproximación Lineal entradas con alto sesgo, que se combinen bien unas con otras de ronda en ronda. Por ejemplo, una entrada con alto sesgo lineal en nuestra tabla es la correspondiente a $(1, 7)$; sin embargo, si buscamos los bits de salida involucrados (los bits 2, 3, y 4 de la caja), vemos que en la siguiente ronda se involucrarían tres cajas S. Buscamos simplificar la expresión, de manera que queremos minimizar, dentro de lo posible, el número de cajas S involucradas en la expresión.

Escogemos, pues, una expresión que no involucre tantas cajas en la siguiente ronda, a saber:

$$X_1 \oplus X_3 \oplus X_4 = Y_2$$

con sesgo lineal 4/16 (corresponde a la entrada (B,4) de la Tabla de Aproximación Lineal). Esto nos dá una expresión lineal que sólo usa un bit de la salida, de manera que sólo afecta una caja de la siguiente ronda. Ahora hay que decidir a cuál caja se lo aplicamos.

Si se la aplicamos a S_{11}, eso nos daría en la siguiente ronda una entrada con valor 8 para S_{22}. Pero el renglón de 8 en la tabla es muy plano (excepto la entrada correspondiente a salida `F`, pero entonces vamos a involucrar las cuatro cajas en la tercer ronda). Si se la aplicamos a S_{12}, en la siguiente ronda vamos a tener una entrada de 4 en S_{22}; el renglón de 4 tiene una buena salida con alto sesgo y pocos bits: salida 5, con sesgo $-(1/4)$.

Si se lo aplicamos a S_{13}, tendríamos en la siguiente ronda a S_{22} con entrada 2, lo cual nos da a E con buen sesgo; pero E consiste de tres bits prendidos. Si se lo aplicamos a S_{14}, entonces en la siguient ronda vamos a tener a S_{22} con entrada 1. Podríamos tomar salida de 7 con alto sesgo, pero entonces tenemos tres cajas en la siguiente ronda.

En resumen, lo que minimiza el trabajo para la siguiente ronda es aplicarlo a S_{12}, con una salida de valor 0100.

En la segunda ronda, vamos a tener a S_{22} con entrada 4; el sesgo que conviene es la expresión

$$X_2 = Y_2 \oplus Y_4,$$

que corresponde a la entrada $(4, 5)$ de nuestra Tabla de Aproximación Linal, con sesgo $-(1/4)$. Esto resulta en que en la tercer ronda, vamos a tener involucradas a las cajas S_{32} y S_{34}, ambas con entrada 4. Repetimos la elección de expresión para cada una de ellas, y al iniciar la cuarta ronda, vamos a tener involucradas las cajas S_{42} y S_{44}, cada una con entrada 0101.

Terminamos, pues, con las siguientes expresiones:

Caja	Expresión	Prob.	Sesgo
S_{12}	$X_1 \oplus X_3 \oplus X_4 = Y_2$	$\frac{12}{16}$	$\frac{1}{4}$
S_{22}	$X_2 = Y_2 \oplus Y_4$	$\frac{4}{16}$	$-\frac{1}{4}$
S_{32}	$X_2 = Y_2 \oplus Y_4$	$\frac{4}{16}$	$-\frac{1}{4}$
S_{34}	$X_2 = Y_2 \oplus Y_4$	$\frac{4}{16}$	$-\frac{1}{4}$

que aproximan las cajas involucradas. Sea P el texto de entrada, P_j el j-ésimo bit de P, contado de izquierda a derecha; análogo con C_j, donde C es la salida del criptosistema. Sean K_i la llave de la i-ésima ronda, con K_{ij} el j-ésimo bit, también contado de izquierda a derecha, de K_i. Sea U_i la entrada a las cajas en la ronda i, con U_{ij} el j-ésimo bit (contado de izquierda a derecha) de U_i; y sea V_i la salida de las cajas en la i-ésima ronda, V_{ij} el j-ésimo bit de V_i.

Tenemos que $U_1 = P \oplus K_1$. También tenemos que

$$\begin{aligned} V_{1,6} &= U_{1,5} \oplus U_{1,7} \oplus U_{1,8} \\ &= (P_5 \oplus K_{1,5}) \oplus (P_7 \oplus K_{1,7}) \oplus (P_8 \oplus K_{1,8}), \end{aligned}$$

con sesgo de $(1/4)$. Esto aproxima la primera ronda.

Para aproximar la segunda ronda, tenemos

$$V_{2,6} \oplus V_{2,8} = U_{2,6}$$

con sesgo $-(1/4)$; como $U_{2,6} = V_{1,6} \oplus K_{2,6}$, nuestra aproximación con sesgo $-(1/4)$ es

$$V_{2,6} \oplus V_{2,8} = V_{1,6} \oplus K_{2,6}.$$

Tomando las expresiones para la primer y segunda ronda, tenemos que

$$V_{2,6} \oplus V_{2,8} \oplus P_5 \oplus P_7 \oplus P_8 \oplus K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} = 0$$

con sesgo $2(1/4)(-1/4) = -(1/8)$ (aplicando el Lema de Amontonamiento).

Para la tercer ronda, tenemos $V_{3,6} \oplus V_{3,8} = U_{3,6}$, con sesgo $-(1/4)$, y $V_{3,14} \oplus V_{3,16} = U_{3,14}$ con sesgo $-(1/4)$. Puesto que $U_{3,6} = V_{2,6} \oplus K_{3,6}$, y $U_{3,14} = V_{2,8} \oplus K_{3,14}$, tenemos la expresión:

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus V_{2,6} \oplus K_{3,6} \oplus V_{2,8} \oplus K_{3,14} = 0$$

con sesgo $2(-1/4)(-1/4) = (1/8)$. Nuevamente usamos el Lema de Amontonamiento.

Ahora combinamos nuestra expresión para las primeras dos rondas, con la expresión para la tercera, y obtenemos:

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus P_5 \oplus P_7 \oplus P_8 \\ \oplus K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus K_{3,14} = 0.$$

Notamos que $U_{4,6} = V_{3,6} \oplus K_{4,6}$, $U_{4,8} = V_{3,14} \oplus K_{4,8}$, $U_{4,14} = V_{3,8} \oplus K_{4,14}$, y $U_{4,16} = V_{3,16} \oplus K_{4,16}$, podemos reescribirlo como:

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus \sum\nolimits_K = 0$$

donde

$$\sum\nolimits_K = K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus K_{3,14} \oplus K_{4,6} \oplus K_{4,8} \oplus K_{4,14} \oplus K_{4,16}$$

y por ende $\sum_K$ tiene un valor fijo, ya sea 0 ó 1 dependiendo de la llave. Aplicando el Lema de amontonamiento, le expresión de arriba tiene un sesgo de:

$$2^3 \left(\frac{1}{4}\right)\left(-\frac{1}{4}\right)\left(\frac{1}{8}\right) = -\frac{1}{32}.$$

Como $\sum_K$ tiene un valor fijo, la expresión

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$$

es cierta con probabilidad 15/32 o con probabilidad 17/32, dependiendo de si $\sum_K = 0$ ó $\sum_K = 1$, respectivamente.

En otras palabras, hemos obtenido una aproximación lineal a las primeras tres rondas del criptosistema RSF, con un sesgo lineal de magnitud 1/32. Ahora queremos usar este sesgo para determinar algunos de los bits de la llave K_5.

7.3.6 Criptoanálisis lineal de la RSF

Una vez que hemos obtenido una aproximación lineal para $R-1$ rondas de nuestra red de Feistel de R rondas, con sesgo lineal relativamente grande, se puede atacar el criptosistema recuperando bits de la última llave utilizada. En el caso de RSF, tenemos una aproximación lineal de tres rondas, y la podemos usar para extraer bits de la llave K_5. Específicamente, buscamos extraer los bits de la llave asociados con las cajas S que están involucradas en la última ronda de nuestra aproximación lineal (en este caso, S_{42} y S_{44}, lo cual nos da acceso a los bits 5–8 y 13–16 de K_5.

Para ello, se procede de manera similar al caso de criptoanálisis diferencial. Es decir, hacemos un desciframiento parcial de la última ronda del criptosistema. Explícitamente, para todos los posibles valores de los bits involucrados en la última llave, tomamos los bits correspondientes del criptotexto, hacemos la suma booleana correspondiente, y vemos de dónde hubiese provenido el resultado con respecto a las caja S. Esto lo hacemos para cada pareja de texto/criptotexto que tenemos, y llevamos la cuenta para cada valor de la subllave involucrada. La cuenta para una subllave específica se incrementa si la expresión lineal que tenemos es cierta para la última ronda de cajas S (lo cuál se verifica usando el desciframiento parcial) y los bits del texto conocido. Al final, se asume que la subllave cuya cuenta difiere en mayor medida de un medio del número de muestras de texto/criptotexto usadas es la subllave correcta. Esto se debe a que la probabilidad de que esto suceda para la llave correcta difiere significativamente de 1/2 (debido al sesgo lineal de la expresión), mientras que esperamos que con llaves aleatorias la expresión sea cierta aproximadamente la mitad de las veces.

7.3.7 DES y el criptoanálisis lineal

El criptoanálisis lineal es fuertemente dependiente de la estructura de la cajas S, y las cajas S de DES *no* están optimizadas contra el ataque. De hecho, el orden de las cajas S elegido para DES está entre el 9% y 16% más débil contra el criptoanálisis lineal, a diferencia de su resistencia contral el criptoanálisis diferencial, donde se encuentra entre el 15% más resistente.

No es dificil encontrar otro orden para las cajas S de DES que aumentaría significativamente la resistencia contra el criptoanálisis lineal, sin debilitar la resistencia contra el criptoanálisis diferencial. Según Don Coppersmith, la resistencia al criptoanálisis lineal no fue parte de los criterios de diseño de DES. O bien no sabían sobre el criptoanálisis lineal, o bien sabían sobre un ataque mucho más poderoso, tal que la resistencia contra él tomó precedencia y que aún no conocemos.

Hay una cierta dualidad entre el criptoanálisis lineal y el criptoanálisis diferencial. No es dificil probar, por ejemplo, que entre más grandes sean las cajas S, mayor es la resistencia contra el criptoanálisis diferencial, pero menor contra el lineal: si aumenta el número de entradas y salidas, incrementa la probabilidad de una expresión lineal con muy alto sesgo.

Una interesante avenida de ataque a DES combina los criptoanálisis lineal y diferencial, en el llamado *criptoanálisis lineal-diferencial*, de Susan Langford y Hellman [LH94]. El ataque no es más rápido que los ataques por separado, pero requieren de mucho menos texto. Sin embargo, Langford y Hellman sólo pudieron aplicarlo a DES de ocho rondas.

8.1 Funciones de un solo sentido

Probablemente todos hemos armado algún rompecabezas durante nuestra vida y luego de armarlo, generalmente, con todo el dolor de nuestro corazón, lo hemos desarmado para guardarlo en su caja porque estorbaba en la mesa. Desarmarlo es cosa fácil, pero armarlo no lo es en general. Este es un ejemplo de un proceso que es fácil de llevar a cabo en una dirección pero difícil de invertir. Podríamos pensar en la función que mapea cada pieza a su posición en el rompecabezas y estaríamos hablando entonces de una función biyectiva, por tanto invertible, pero cuya inversa es difícil de obtener. Esta noción es de gran utilidad en criptografía. A las funciones que son fáciles de calcular en un sentido pero cuyas inversas son difíciles de calcular se les suele llamar *funciones de un solo sentido* o *one-way functions*.

Hablando en términos estrictamente matemáticos tales cosas no existen: una función biyectiva es invertible y ya. Sin embargo, hablando en términos computacionales, donde

generalmente no basta con decir si algo existe o no, sino que además hay que calcularlo, sí es posible pensar en tales funciones y, más aún, usarlas. Podríamos pensar, por ejemplo, en escribir un mensaje coherente en las piezas de un rompecabezas armado y luego dárselo desarmado a alguien. Aunque por supuesto que descifrar el mensaje en tales circunstancias no será fácil, éste es un ejemplo trivial de que las funciones de un solo sentido pueden ser útiles en criptografía.

Pero pordríamos numerar las piezas del rompecabezas y hacemos un patrón de como deben ir acomodadas. Entonces será trivial armar el rompecabezas en poco tiempo, muy poco tiempo comparando con el que tomaría armarlo sin ayuda del patrón. Ahora estamos hablando de funciones de un solo sentido en las que, si se da una clave, obtener la inversa se vuelve (relativamente) fácil. A este tipo particular de funciones de un solo sentido se les llama funciones con puerta de trampa (*trapdoor functions*). Una puerta de trampa es fácil de abrir si se conoce el secreto para abrirla.

Tenemos buenos ejemplos de funciones de un solo sentido. Sabemos, por ejemplo, que es fácil generar números primos, sabemos también que es fácil multiplicar un par (o más) de números primos, pero si se nos da un número cualquiera es difícil, computacionalmente hablando, obtener los factores primos que lo determinan[1]. Otro caso es el de la exponenciación en un campo finito[2]: es fácil elevar un número a una potencia, pero dados un par de números a y b no es fácil determinar a qué exponente x hay que elevar a para obtener b (i.e. determinar x tal que $a^x = b$). Cabe hacer énfasis en que esta última función es de un solo sentido en campos finitos. Todos sabemos que en $\mathbb{R}$ es fácil en ambos sentidos, pues el inverso de la exponenciación es el logaritmo. Pero cuando se trabaja en un campo finito el problema se complica. En ese caso se habla del *logaritmo discreto*, y trataremos sobre él más adelante.

8.2 Factorización

¿Cuáles son los factores primos de 1386? Buscamos números primos que dividen sin residuo a 1386, así que el algoritmo más ingenuo que podemos plantear es el mostrado en la Figura 8.1, que consiste en probar con cada número primo a ver si divide al número en cuestión, comenzando por 2. Tratamos entonces de dividir entre cada primo hasta que ya no sea posible dividir entre él y entonces nos pasamos al siguiente. Nos detenemos

[1] El teorema fundamental de la aritmética establece que cualquier número natural puede escribirse de manera única (salvo el orden) como el producto de números primos.

[2] Recuérdese nuestros conocidos $\mathbb{Z}_p$ con p un número primo, que de hecho son casos especiales de campos finitos. En general para todo primo p y todo entero positivo n existe un campo finito de tamaño p^n y es único salvo isomorfismo. p es llamada la *característica* del campo.

```
FACTORIZACIÓN(N)
 1  p ← 2
 2  c ← N
 3  while ¬(esPrimo(c)) do
 4      while ¬(p|c) do
 5          p ← siguientePrimo(p)
 6      endwhile
 7      p → factores
 8      c ← c/p
 9  endwhile
10  c → factores
11  end
```

Figura 8.1: Algoritmo simple para factorizar. La función *esPrimo*(x) regresa *verdadero* si su argumento es primo, *siguientePrimo*(x) regresa el primer número primo mayor que su argumento, *factores* es una lista donde se van guardando los factores primos del número N, $p|c$ significa p divide a c, y el símbolo $\neg$ denota la negación lógica.

cuando obtenemos de cociente un número primo. En la Tabla 8.1 se muestra el proceso. La expresión buscada es: $1386 = 2 \times 3^2 \times 7 \times 11$

Cabe mencionar que tras la función booleana *esPrimo*(x) usada en el algoritmo de la Figura 8.1 hay un proceso bastante tardado. La función debe decidir si su argumento es un número primo o no, es decir si a su vez tiene factores primos (es compuesto) o no. Parece que regresamos al punto de partida. Pero hay diversos mecanismos que permiten decidir si un número n es primo o no. Si es par, por ejemplo, ya sabemos que no es (porque lo divide 2); si no lo divide ningún número (de hecho ningún número impar, usando lo anterior) menor que $\sqrt{n}$ sabemos que sí lo es.[3]

En general lo que suele hacerse en criptografía cuando se desea obtener un número primo, es utilizar *pruebas de primalidad* que proporcionan un cierto grado de certidumbre, arbitrariamente decidido por el usuario, acerca de si un número dado n es primo o no, aprovechándose de que los primos poseen propiedades que, en general, es raro que posean el resto de los números. Por ejemplo todo primo p satisface que, para toda b tal que $\text{mcd}(b, p) = 1$ ocurre:

$$\text{b}^{p-1} \equiv 1 \pmod{p} \tag{8.2.1}$$

[3]Si todos los factores primos de n fueran mayores que $\sqrt{n}$, y n no es primo, entonces sería el producto de dos números, r y s, ambos mayores que $\sqrt{n}$. Pero entonces $n = rs > \sqrt{n}\sqrt{n} = n$, lo cuál es una contradicción.

Número	Divisor	Cociente
1386	2	693
693	2	No se puede
693	3	231
231	3	77
77	3	No se puede
77	5	No se puede
77	7	11 (primo)

Tabla 8.1: Factorización de 1386.

Si n no es primo puede ser que satisfaga 8.2.1 pero es raro[4]. Si aplicamos repetidamente la prueba a un cierto número n con diferentes valores de b entonces incrementamos nuestra certeza de que n es primo.

Existen diversas pruebas de primalidad, como la de *Solovay-Strassen*, *Lehmann*, *Rabin-Miller* o *Cohen-Lenstra*. Referimos al lector a [Sch96, Kob94, MvOV96] para más detalles.

Evidentemente existen algoritmos mejores para factorizar enteros, pero no mucho mejores. Los algoritmos con complejidad probada son exponenciales y de los que parecen tener complejidad menor, ésta aún no se ha demostrado. Hablaremos más delante de algunos de estos algoritmos, pero en síntesis, de todas maneras se tardan mucho para enteros "grandes", donde "grande" significa de cientos de dígitos. Hay circunstancias para las que existen algoritmos muy buenos para factorizar si el número de entrada tiene factores primos pequeños, por ejemplo, cosa que no ocurre con los que se utilizan en criptografía.

En 1977 se publicó en la columna de Martin Gardner en *Scientific American* un número de 129 dígitos (uno de los conocidos retos de RSA) y en 1994 un equipo de voluntarios coordinado por Michael Graff y Paul Leyland intercambiando información por correo electrónico logró factorizarlo utilizando el algoritmo de la criba cuadrática con polinomios múltiples (MPQS) de A.K. Lenstra y Manasse tardándose ocho meses. En el proyecto participaron un total de 1600 máquinas trabajando autónomamente. Ésto da una buena idea de lo tardado que puede ser factorizar un número grande.

[4] A los números que satisfacen 8.2.1 sin ser primos se les denomina números de Carmichael, en 1994 se probó que hay una infinidad de ellos.

8.3 El logaritmo discreto

En un campo finito es fácil elevar un número a una potencia, por ejemplo $2^8 = 256$ en $\mathbb{Z}_{11}$ sería:

$$2^8 = 3 \pmod{11},$$

porque $256 \equiv 3 \pmod{11}$.

Pero si se nos pregunta

¿A qué potencia hay que elevar 2 para obtener 3 en $\mathbb{Z}_{11}$?

ya no es tan fácil responder. Tendríamos que probar varias potencias para encontrar la respuesta correcta. El problema es tanto más difícil cuanto más grande sea el tamaño del campo finito[5].

En $\mathbb{R}$ el problema además de ser fácil tiene solución siempre que el argumento y la base del logaritmo sean positivos. Pero en un campo finito puede ser que el problema ni siquiera tenga solución. Por ejemplo: no existe x tal que $3^x = 7$ en $\mathbb{Z}_{13}$.

Igual que en el caso del problema de factorización (en el cuál hay ciertos números para los cuales el problema es sencillo), existen algoritmos más o menos rápidos para calcular el logaritmo discreto en $\mathbb{Z}_p$ si $p-1$ sólo tiene factores primos pequeños, así que en criptografía se usan campos en los que $p-1$ tenga al menos un factor primo grande.

8.4 Mensajes y números

Sabemos que hay procesos que es fácil hacer en un sentido pero que es difícil invertir y estos procesos son funciones que operan sobre números. Pero si lo que deseamos es cifrar mensajes hechos de letras y números ¿cómo hacemos para transformar un mensaje en un número y así poder usar esas funciones de un solo sentido para cifrarlo?

Dado que en particular en una computadora el mensaje debe ser almacenado carácter por carácter mediante el código binario usado por la computadora para representar información, sea ASCII o algún otro, es posible pensar en utilizar esta representación como el número que

[5]De hecho el problema del logaritmo discreto no requiere que la estructura del conjunto sea un campo, basta con que sea un grupo. En criptografía es donde se le refiere a un campo. Por supuesto en un grupo G, cuya operación se denota por $*$ y se escribe multiplicativamente, la potencia n-ésima de $a \in C$, a^n, se define como $a * \ldots * a$, n factores.

representa al mensaje. Así por ejemplo el mensaje "HOLA" podría verse como el número `484F4C41` en hexadecimal ya que el código ASCII de la "H" es 48, el de la "O" es 4F, etc.

Otra posibilidad es la utilizada en [Kob94]: ver a cada carácter de un mensaje como un dígito en base 26 (si sólo utilizamos letras y sólo utilizamos el alfabeto de 26 letras) y ver entonces un mensaje como un número escrito en esa base. Así que la "A" es nuestro cero, luego "B" vale 1, "C" vale 2 y así sucesivamente hasta llegar a "Z" que vale 25. En este esquema el mensaje "HOLA" sería el número:

$$7 \times 26^3 + 14 \times 26^2 + 11 \times 26^1 + 0 \times 26^0 = 132782$$

dado que "H" = 7, "O" = 14, "L" = 11, y "A" = 0. Nótese que el valor de nuestros caracteres en este esquema es la diferencia entre el valor que poseían en el esquema anterior y el valor de la "A" también en el esquema anterior.

Dado un mensaje largo, digamos de varias cuartillas, si usamos cualquiera de los esquemas mencionados obtendremos un número gigantesco. ¿Cómo hacemos para manipularlo? Para responder esta pregunta hay que hacer varias consideraciones. Por lo mencionado en las secciones anteriores y lo que veremos más adelante las funciones de un solo sentido que se suelen usar en criptografía involucran números grandes, de cientos de dígitos, así que nuestros mensajes pueden ser también muy grandes. Aun así en varias cuartillas hay muchos cientos de caracteres, podríamos rebasar fácilmente cualquier límite que se nos imponga. Lo que se hace en este caso es fragmentar el mensaje en varios números del tamaño que se esté utilizando.

Una vez superado el problema de la representación queda aun el de la manipulación de estos grandes números y para resolverlo se echa mano de las computadoras y de bibliotecas de software especializado en la manipulación de grandes números.

8.5 Idea de la criptografía de llave pública

Como ya dijimos, el hecho de que se requiera un sistema criptográfico presupone la existencia de tres cosas, a saber:

- El enemigo, el que no deseamos que se entere de ciertas cosas.
- Los amigos, aquella persona o personas con las que queremos comunicarnos sin que nadie se entere de lo que decimos.
- Un medio de comunicación inseguro, susceptible de ser observado por el enemigo.

Los mecanismos clásicos de cifrado de mensajes, basan su seguridad en mantener oculta la llave usada para cifrar los datos. Las partes a comunicarse no deben darse el lujo de decirse la llave a través del canal así que deben ponerse de acuerdo por adelantado. En sistemas como Vigenère, por ejemplo, el enemigo puede conocer el algoritmo, pero no debe conocer la palabra clave usada para cifrar, que es la misma que se usa para descifrar (sistema simétrico). Aún en un sistema asimétrico, como el de Hill, decirse la clave a través del canal destruye la seguridad del sistema, porque conocer la clave de cifrado proporciona todo lo necesario para conocer rápidamente la de descifrado.

El problema de ponerse de acuerdo en una clave se resolvería si hubiera de hecho dos claves, una que cualquiera puede conocer y que por tanto puede decirse a través del canal de comunicación, y otra que sólo sirve para descifrar, que sólo conoce su propietario y, algo muy importante, que no puede ser obtenida fácilmente a partir de la clave para cifrar. Podemos entonces imaginar un esquema como este: A quiere comunicarse secretamente con B y le avisa a B de su deseo; B le envía a A una clave que puede usar para cifrar aquellos mensajes secretos que quiere que sólo conozca B, pues sólo B conoce una segunda clave capaz de descifrar los mensajes dirigidos a él. Nadie, incluyendo a A mismo, es capaz de encontrar en un tiempo razonable la clave secreta que tiene B para descifrar mensajes, a pesar de conocer la clave para cifrarlos.

La existencia de un sistema asimétrico es condición necesaria (pero no suficiente) para lograr lo expuesto en el párrafo anterior. Si el sistema, además de ser asimétrico, posee la cualidad de que una de las claves no proporciona información suficiente para obtener la otra, entonces ya reunimos la condición necesaria y suficiente para lograrlo, y el problema de distribuir la clave entre los usuarios queda resuelto.

En 1976 dos ingenieros eléctricos de la Universidad de Stanford, Whitfield Diffie y Martin Hellman, junto con Ralph Merkle (un estudiante de licenciatura en la Universidad de California en Berkeley), publicaron un artículo titulado *New directions in cryptography* [DH76] en el que exponían un mecanismo mediante el cual es posible que dos personas pueden ponerse de acuerdo en una palabra clave secreta comunicándose a gritos entre una multitud que no debe conocer su secreto (es decir comunicándose mediante un canal inseguro) y que puede conocer el algoritmo utilizado. Le llamaron *"criptosistema de llave pública."*

La idea general de los sistemas criptográficos de llave pública es, como ya se mencionó, que existen dos claves: una pública para cifrar, y otra secreta, para descifrar. Es posible pensar en un directorio donde se encuentran las claves de cifrado para varias personas, esas claves están relacionadas íntimamente con las claves para descifrar, cada clave para descifrar es conocida únicamente por la persona a la que le pertenece. Con este esquema cualquier persona A puede enviar un mensaje cifrado a otra persona, B, usando la clave pública de cifrado para B, pero sólo B puede descifrar el mensaje. El hecho de conocer la clave de

cifrado no facilita (al menos no grandemente) la obtención de la clave de descifrado.

En síntesis un sistema de llave pública posee dos propiedades [Kob94, Sch96]:

1. Cada persona en el sistema posee los medios necesarios para cifrar y descifrar mensajes. Cifrar los destinados a cualquier otra persona y descifrar los que vayan dirigidos a ella.

2. Los medios para descifrar los mensajes de una persona no son obtenibles en un tiempo razonable por cualquier otra persona.

Al parecer la Agencia de Seguridad Nacional (NSA por sus siglas en inglés) estadunidense ya tenia conocimiento de sistemas criptográficos de llave pública desde 1966, diez años antes de la publicación del artículo de Diffie y Hellman [Sch96]. También el servicio criptográfico británico descubrió por su cuenta la criptografía de llave pública antes de su descubrimiento oficial, lo que se mantuvo en secreto hasta hace poco [Sin99].

Adicionalmente los sistemas criptográficos de llave pública pueden utilizarse para certificar que un mensaje dado haya sido enviado por un usuario particular del sistema. Supóngase que alguien, A, desea enviar un mensaje de tal forma que, una vez recibido, al destinatario no le quepa duda de que fue A quien lo envió, y además que si alguien intercepta el mensaje pueda obtener información suficiente como para hacerse pasar por A y enviar mensajes a su nombre en el futuro. En un esquema de llave pública bastaría que A añadiera al mensaje información que sólo puede ser obtenida usando su llave privada y que, procesándola usando la llave pública de A, haga evidente que solo A pudo enviar el mensaje. A un esquema de certificación de este tipo se le denomina *firma digital.* Más adelante veremos como pueden usarse algunos esquemas criptográficos de llave pública para implementar firmas digitales.

8.6 Intercambio de llaves de Diffie-Hellman

Imaginemos que queremos usar un sistema criptográfico simétrico. Hay una sola palabra clave que es utilizada para cifrar y descifrar, y ésta no debe conocerla el enemigo. Así que ambas partes deben acordar una clave sin que nadie más se entere de ella. El canal de comunicación es inseguro, es escuchado por todo mundo, así que las partes tienen una de dos opciones:

1. Ponerse de acuerdo por adelantado: enviarse la clave a través de un mensajero autorizado y seguro, o citarse en algún lugar para ponerse de acuerdo sin que nadie escuche.

2. Ponerse de acuerdo usando el canal: de tal manera que nadie pueda obtener la clave (o al menos obtenerla en un tiempo razonable) a partir de la información que es enviada a través del canal por ambas partes.

El esquema de intercambio de llaves de Diffie-Hellman hace posible la segunda alternativa. Para explicar el esquema, presentamos la siguiente analogía:

Imaginemos que tanto A como B tienen sendos botes con capacidad para tres litros de pintura. A y B se ponen de acuerdo en un color básico que se dicen a gritos; C, un intruso que pretende enterarse de las conversaciones privadas de A y B también sabe ese color. A y B ponen un litro de pintura de color básico en sus botes. Además A añade un litro de su color favorito, que nadie, salvo ella conoce. B hace lo propio, añade también un litro exacto de su color secreto y luego, ante la vista de todos, incluyendo a C, se intercambian los botes. Ni A sabe el color secreto de B, ni B sabe el de A, ambos sólo reciben la pintura mezclada. C tampoco puede saber ninguno de los dos colores secretos. Luego A añade al bote que recibió de B un litro de su propio color secreto, y B añade al bote que recibió de A un litro de su propio color. Ahora tanto A como B tienen exactamente el mismo color en sus botes y nadie más puede conocer su composición exacta. Este ejemplo es una buena ilustración del esquema de intercambio de llave de Diffie-Hellman [Sin99].

En un campo finito $\mathbb{Z}_p$ (p primo por supuesto) decimos que un elemento g es un *generador*, una *raíz primitiva*, o *primitivo* módulo p si para cualquier elemento $x \in \mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ existe un número n tal que $x = g^n$, es decir $g^n \equiv x \ (mod\ p)$. Por ejemplo en $\mathbb{Z}_{11}$, 2 es un generador módulo 11 porque todo elemento de $\mathbb{Z}_{11}^* = \{1, \ldots, 10\}$ puede escribirse como una potencia de 2 (véase Tabla 8.2). Si un elemento a no es raíz primitiva, entonces al elevarlo a diferentes potencias se obtienen *algunos* elementos de $\mathbb{Z}_p^*$, es decir el conjunto de posibles resultados de a^n es un subconjunto propio de $\mathbb{Z}_p^*$, pero si es raíz primitiva entonces el conjunto de posibles resultados de a^n es exactamente todo $\mathbb{Z}_p^*$.

Sean Alicia y Bartolo[6] (A y B respectivamente) las dos partes que desean obtener una clave secreta. El algoritmo de intercambio de llaves de Diffie-Hellman es el siguiente:

1. Alicia y Bartolo se ponen de acuerdo en un par de números: un primo grande p y una raíz primitiva $g \in \mathbb{Z}_p^*$. Se pueden poner de acuerdo en un canal inseguro.
2. Alicia elige un entero aleatorio grande α, con $0 < \alpha < p - 1$, y le envía a Bartolo
$$X = g^\alpha \pmod{p}.$$
3. Bartolo elige un entero aleatorio grande β, con $0 < \beta < p - 1$, y le envía a Alicia
$$Y = g^\beta \pmod{p}.$$

[6]Generalmente en los libros y artículos de criptografía se usan *Alice* y *Bob.*

$x \in \mathbb{Z}_{11}^*$	Expresión	Valor nominal
1	2^{10}	1024
2	2^1	2
3	2^8	256
4	2^2	4
5	2^4	16
6	2^9	512
7	2^7	128
8	2^3	8
9	2^6	64
10	2^5	32

Tabla 8.2: Los elementos de $\mathbb{Z}_{11}$ expresados como potencias de 2. Los elementos de la primera columna son los de la última módulo 11.

4. Alicia calcula $K = Y^{\alpha} \pmod{p}$.
5. Bartolo calcula $K' = X^{\beta} \pmod{p}$.

Tenemos que $K = K' = g^{\alpha\beta} \pmod{p}$. Nadie que haya estado escuchando la conversación entre Alicia y Bartolo puede calcular K. Sólo conocería p, g, X, y Y, y para calcular K tendría que poder hacer alguna de las siguientes cosas:

1. Obtener el logaritmo discreto de X ó Y en base g módulo p para obtener α ó β, respectivamente, y poder calcular $g^{\alpha\beta} \pmod{p}$.
2. Calcular $g^{\alpha\beta} \pmod{p}$ de alguna otra manera diferente a la opción anterior.

La conjetura de Diffie-Hellman es que la segunda opción no puede hacerse sin haber llevado a cabo la primera[7], es decir, forzosamente se tiene que optar por 1. Como veremos, ésto es en general difícil. Para asegurarse de que lo sea realmente debe elegirse p grande y de tal forma que $\frac{p-1}{2}$ sea también primo, g puede elegirse arbitrariamente.

8.7 Algunos preliminares

Sabemos del capítulo de sistemas poligráficos, que en $\mathbb{Z}_m$ un elemento cualquiera tiene inverso multiplicativo si y sólo si es primo relativo con m. Por supuesto si estamos en $\mathbb{Z}_p$,

[7]Hasta la fecha no se ha probado que la segunda opción no es posible sin haber resuelto antes la primera, pero no hya nada que asegure que en efecto sea imposible.

con p primo, entonces todos los elementos distintos de cero tienen inverso multiplicativo, porque cualquier número $a \in \mathbb{Z}_p^*$ distinto de cero está en el conjunto $\{1, \ldots, p-1\}$ y es primo relativo a p.

También sabemos como calcular el inverso. En el mismo capítulo de sistemas poligráficos vimos que el algoritmo de Euclides extendido nos proporciona un método fácil para encontrar el inverso.

Otros dos resultados que nos serán útiles son los siguientes.

Teorema 8.1 (Pequeño Teorema de Fermat) *Sea p un número primo y a un entero. Si* $\text{mcd}(p, a) = 1$ *(es decir, si p y a son primos relativos) entonces:*

$$a^{p-1} \equiv 1 \pmod{p}$$

Otra manera de enunciarlo restringiendo el conjunto de posibles valores de a es: Si p es primo entonces $a^{p-1} \equiv 1 \pmod{p}$ para toda a en $\mathbb{Z}_p^*$

Recordemos $\mathbb{Z}_n^*$ denota el subconjunto de elementos en $\mathbb{Z}_n$ relativamente primos con n. Por ejemplo $\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$. Como en el teorema p es primo entonces de hecho $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$.

Teorema 8.2 *Si $n_1, n_2, \ldots, n_k$ son primos relativos dos a dos y $n = n_1 n_2 \ldots n_k$ entonces, para cualesquiera enteros x y a, y para toda $i \in \{1, 2, \ldots, k\}$ ocurre:*

$$x \equiv a \pmod{n_i} \quad \textit{si y sólo si} \quad x \equiv a \pmod{n}$$

Este teorema es una consecuencia del conocido Teorema Chino del Residuo.

8.8 Criptosistema de envío de mensajes Massey-Omura

Un mecanismo de cifrado de mensajes similar al intercambio de llaves de Diffie-Hellman es el de Massey-Omura.

Imaginemos que tenemos un portafolios con información confidencial que debemos hacerle llegar a otra persona. Cerramos el portafolios con un candado muy seguro, del que

sólo nosotros tenemos la llave, ni siquiera el destinatario de los documentos tiene copia de la llave. Así lo enviamos al destinatario, que, por supuesto, no puede abrirlo; en vez de eso añade al portafolios su propio candado super-seguro del que sólo él tiene llave y nos lo manda de regreso. Ahora nosotros procedemos a quitar nuestro candado usando nuestra muy exclusiva llave y, claro está, dejamos en el portafolio el candado del destinatario que no podemos abrir. El portafolios viaja de regreso al destinatario que con sólo quitar su propio candado tiene acceso a los documentos confidenciales de su interior.

Formalmente hablando el algoritmo de Massey-Omura es el siguiente:

1. Alicia y Bartolo se ponen de acuerdo en un entero primo grande p.

2. Alicia selecciona aleatoriamente un entero e_A entre 1 y $p-1$ tal que $\text{mcd}(e_A, p-1) = 1$ y usando el algoritmo de Euclides calcula su inverso d_A, es decir, d_A es un entero tal que:
$$\text{e}_A d_A \equiv 1 \pmod{p-1} \tag{8.8.2}$$
y $0 < d_A < p$.

3. Bartolo selecciona aleatoriamente un entero e_B entre 0 y $p-1$ tal que $\text{mcd}(e_B, p-1) = 1$ y usando el algoritmo de Euclides calcula su inverso d_B, de manera que:
$$\text{e}_B d_B \equiv 1 \pmod{p-1}. \tag{8.8.3}$$

4. Alicia, quien posee el mensaje a cifrar M, envía a Bartolo:
$$M_A = M^{e_A} \pmod{p}. \tag{8.8.4}$$

5. Bartolo recibe el mensaje cifrado y calcula
$$M_{A,B} = M_A^{e_B} \pmod{p}, \tag{8.8.5}$$
y se lo envía de vuelta a Alicia.

6. Alicia recibe $M_{A,B}$ y calcula:
$$M' = M_{A,B}^{d_A} \pmod{p}, \tag{8.8.6}$$
y lo envía a Bartolo.

7. Bartolo calcula:
$$M'' = M'^{\,d_B} \pmod{p}, \tag{8.8.7}$$
que resulta ser el mensaje original M.

Veamos por qué funciona. De las expresiones (8.8.6) y (8.8.7), tenemos que:

$$M'' \equiv (M_{A,B})^{d_A d_B} \pmod{p},$$

y usando (8.8.5) y (8.8.3):

$$M'' \equiv (M_A^{e_B})^{d_A d_B} \equiv (M_A^{e_B d_B})^{d_A} \equiv M_A^{d_A} \pmod{p}.$$

Finalmente, usamos (8.8.4) y (8.8.2) para obtener:

$$M'' \equiv (M^{e_A})^{d_A} \equiv M^{e_A d_A} \equiv M \pmod{p}.$$

Puesto que tanto $1 < M, M'' < p$, deben ser iguales.

8.9 Criptosistema de ElGamal

En 1984 Taher ElGamal[8] propuso un esquema que permite tanto el cifrado de mensajes como la implementación de firmas digitales. El procedimiento común a ambos usos es el siguiente:

1. Elegir los siguientes números:
   - Un primo p (grande). Esto determina el campo finito donde se efectuarán las operaciones.
   - Un número aleatorio $g < p$, generador en $\mathbb{Z}_p$.
   - Cada usuario u elige un número aleatorio x_u, con $0 < x_u < p$.
2. Calcular
$$y_u = g^{x_u} \pmod{p}. \tag{8.9.8}$$
3. Pueden hacerse públicos y_u, g y p. Debe mantenerse en secreto x_u. De hecho p y g son públicos y comunes para todos los usuarios del sistema de llave pública (todo el directorio de personas y llaves), pero y_u es particular de cada usuario.

[8] Es frecuente encontrar su nombre escrito como *Elgamal* (corrompiendo la sintaxis árabe). Fue consultor de Netscape en el desarrollo del protocolo `SSL`; antes trabajó en RSA Data Security, y actualmente tiene su propia compañía llamada *Securify*.

8.9.1 Cifrado de mensajes

Imaginemos que Alicia desea enviar un mensaje a Bartolo usando el criptosistema de ElGamal. Alicia conoce g, p, y y_B, y procede a hacer lo siguiente:

1. Elige un entero k aleatoriamente, primo relativo a $p-1$. Alicia mantiene secreto el valor de k.

2. Calcula
$$a \equiv g^k \pmod{p}, \tag{8.9.9}$$
y
$$b \equiv y_B^k M \pmod{p} \tag{8.9.10}$$

3. El mensaje cifrado es la pareja ordenada (a, b).

La expresión (8.9.10) implica que

$$\frac{b}{a^{x_B}} \equiv \frac{y_B^k M}{a^{x_B}} \pmod{p}.$$

Usando (8.9.8) obtenemos:
$$\frac{y_B^k M}{a^{x_B}} \equiv \frac{g^{kx_B} M}{a^{x_B}} \pmod{p}. \tag{8.9.11}$$

Por otra parte, de la expresión (8.9.9) tenemos:

$$\mathrm{a}^{x_B} \equiv g^{kx_B} \pmod{p}.$$

Sustituyendo en (8.9.11) tenemos finalmente:

$$\frac{y_B^k M}{a^{x_B}} \equiv \frac{g^{kx_B} M}{a^{x_B}} \equiv \frac{g^{kx_B} M}{g^{kx_B}} \equiv M \pmod{p}.$$

Para descifrar el mensaje se calcula

$$M = \frac{b}{a^{x_B}} \pmod{p},$$

que sólo lo puede efectuar quien posea la clave secreta x_B. Así que este esquema de cifrado se basa en la hipótesis de que el logaritmo discreto es difícil de calcular.

8.9.2 Firma digital

Sea M el mensaje que se desea firmar y supongamos que Alicia desea firmarlo para enviarlo a Bartolo, de tal manera que Bartolo quede convencido que el mensaje viene de Alicia y no de otra persona. Para hacer esto utilizando el criptosistema de ElGamal, Alicia hace lo siguiente:

1. Alicia elige un entero aleatorio k, primo relativo a $p-1$, que mantendrá en secreto.

2. Calcula $a \equiv g^k \pmod{p}$.

3. Encuentra, usando el algoritmo de Euclides extendido, un entero positivo b tal que $M \equiv (x_A a + kb) \pmod{(p-1)}$.

4. La "firma" del mensaje M es la pareja ordenada (a, b).

Para confirmar que la firma es válida, Bartolo calcula: $y_A^a a^b \pmod{p}$ cuyo valor debe coincidir con: $g^M \pmod{p}$. Notemos que Bartolo conoce a y b, pues fueron enviados como la firma del mensaje M, conoce M (pues el mensaje fue enviado); y el valor de g y y_A son públicos y por ende conocidos.

Para verificar que en efecto $y_A^a a^b \equiv g^M \pmod{p}$, recordemos que $y_A \equiv g^{x_A} \pmod{p}$, y que $a \equiv g^k \pmod{p}$. De manera que

$$\begin{aligned} y_A^a a^b &\equiv (g^{x_A})^a \left(g^k\right)^b \pmod{p} \\ &\equiv g^{a x_A} g^{kb} \pmod{p} \\ &\equiv g^{a x_A + kb} \pmod{p} \\ &\equiv g^M \pmod{p} \end{aligned}$$

por la elección de b.

¿Quien posee la información necesaria para calcular a y b de tal manera que $g^M \equiv y_A^a a^b \pmod{p}$? El valor de b depende del valor de x_A, que suponemos es conocido sólo por Alicia. De manera que al recibir (M, a, b), estamos tan seguros de que el mensaje fue envíado por Alicia como estamos de la seguridad del criptosistema mismo.

8.10 Criptosistema RSA

En 1977 tres científicos de la computación adscritos a M.I.T., Ron Rivest, Adi Shamir, y Leonard Adleman, desarrollaron un algoritmo de cifrado y firma digital conocido como RSA, las iniciales de sus apellidos.

El algoritmo se basa en una generalización del Pequeño Teorema de Fermat:

Teorema 8.3 (Teorema de Euler) *Sea n un entero positivo, y sea a un entero primo relativo con n. Entonces*

$$a^{\varphi(n)} \equiv 1 \pmod{n},$$

donde $\varphi(n)$ es la función φ de Euler, cuyo valor en n es el número de enteros positivos menores que n que son primos relativos con n.

Para calcular el valor de $\varphi(n)$, se pueden usar las siguientes propiedades:

Teorema 8.4 *La función φ de Euler tiene las siguientes propiedades:*

1. *Si p es un primo, entonces $\varphi(p) = p - 1$.*
2. *Si p es un primo, y $a > 0$, entonces $\varphi(p^a) = (p-1)p^{a-1}$.*
3. *Si a y b son primos relativos, entonces $\varphi(ab) = \varphi(a)\varphi(b)$.*

Las llaves públicas y privadas de RSA se calculan de la siguiente manera:

1. Se eligen dos números primos grandes p y q aleatoriamente, con p distinto de q.
2. Se calcula $n = pq$.
3. Después, elegimos una llave de cifrado e, que es un entero positivo que es primo relativo con con $(p-1)(q-1)$. Notemos que e debe ser impar, pues tanto $p-1$ como $q-1$ son pares.
4. Una vez electo e, calculamos la llave de descifrado privada, que es un entero positivo d electo de tal forma que

$$ed \equiv 1 \pmod{(p-1)(q-1)}, \tag{8.10.12}$$

lo que se hace usando el algoritmo extendido de Euclides. Es decir, d es el inverso multiplicativo de e módulo $(p-1)(q-1)$. Notemos que

$$\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1).$$

5. La llave pública es la pareja (n, e). La llave privada, que se mantiene secreta, es d. Los valores de los primos p y q no deben revelarse, pero no es necesario recordarlos[9].

Aunque técnicamente la llave pública es la pareja (n, e) (n es llamado el *módulo RSA*, y e el *exponente público*; d es llamado el *exponente privado*), a veces se refiere uno simplemente a e como la llave o el exponente público, cuando el módulo n se entiende por contexto.

8.10.1 Cifrado de mensajes

Para cifrar un mensaje M (que asumimos es un entero positivo que satisface $M < n$[10] y tal que $\text{mcd}(M, n) = 1$), cuyo destinatario tiene llave pública (n, e), calculamos el valor del mensaje cifrado C mediante la fórmula:

$$C \equiv M^e \pmod{n}. \tag{8.10.13}$$

Para descifrar, el destinatario del mensaje calcula

$$M' \equiv C^d \pmod{n}, \tag{8.10.14}$$

donde d es su llave privada.

Para ver por qué funciona el sistema, notemos que (8.10.12) es equivalente a la existencia de un entero k tal que

$$ed = 1 + k(p-1)(q-1).$$

Puesto que $C \equiv M^e \pmod{n}$, entonces

$$M' \equiv C^d \equiv M^{ed} = M^{1+k(p-1)(q-1)} = M \cdot \left(M^k\right)^{(p-1)(q-1)} \pmod{n}. \tag{8.10.15}$$

[9] Aunque, como veremos, el algoritmo de RSA no requiere los valores de p y q, sino únicamente los valores de n, d, y e, es relativamente común guardar los valores de p y q, manteniendolos secretos, para agilizar la *implementación* del algoritmo.

[10] Cuando el mensaje M es mayor que n, se tiene partir el mensaje en bloques, cada bloque con valor menor que n, y enviarlos por separado.

Como $\text{mcd}(M, n) = 1$, y por lo tanto M^k y n son primos relativos, el Teorema de Euler nos dice que

$$(M^k)^{(p-1)(q-1)} = (M^k)^{\varphi(n)} \equiv 1 \pmod{n},$$

de manera que sustituyendo en (8.10.15) tenemos que

$$M' \equiv M \cdot (M^k)^{\varphi(n)} \equiv M \cdot 1 \equiv M \pmod{n}.$$

Puesto que M' y M son menores que n, tenemos que $M' = M$, que es lo que deseabamos.

Se solía recomendar que los primos p y q fueran *primos fuertes*, lo que se refiere a primos p con la propiedad de que tanto $p - 1$ como $p + 1$ tengan factores primos grandes. Esto debido a ciertos algoritmos de factorización tenían altas probabilidades de éxito si $p - 1$ ó $p + 1$ tenían sólo factores primos pequeños. Sin embargo, métodos de factorización más recientes tienen la misma probabilidad de éxito con primos fuertes que con "primos débiles", de manera que la relevancia de los primos fuertes ya no es tan importante como antes. Sin embargo, se sigue recomendando utilizar primos fuertes de ser posible.

8.10.2 Firmas digitales

RSA puede usarse también para firmas digitales. Si Alicia desea enviar un mensaje firmado a Bartolo, entonces luego de enviar el mensaje M (cifrado o no) a Bartolo, envía $M_f = M^d$. Una vez recibido tanto M como M_f, Bartolo calcula $M_f^e \bmod n$. Notemos que tanto e como n son conocidos. Puesto que $M^{ed} \equiv M \pmod{n}$, Bartolo puede ahora comparar el valor de M con el valor de M_f^e; la firma es correcta si estos dos valores son iguales.

Puesto que calcular M_f requiere conocer el valor de d, que suponemos es conocido sólo por Alicia, la firma garantiza que el mensaje fue envíado por Alicia.

8.10.3 Seguridad de RSA

Recordemos que la llave pública de RSA es la pareja (n, e), donde n es el producto de dos primos p y q. La llave privada es el entero d que es el inverso multiplicativo de e módulo $\varphi(n) = (p - 1)(q - 1)$.

Dado que n y e son conocidos, resulta que conocer el valor de d es equivalente a conocer la factorización de n. Es decir, tenemos el siguiente resultado:

Teorema 8.5 *Sea (n, e) una llave pública de RSA. Dada la llave privada d, uno puede factorizar el módulo $n = pq$ de manera eficiente. Conversamente, dada la factorización $n = pq$, uno puede obtener el valor de d de manera eficiente.*

Dem.: Si tenemos la factorización $n = pq$ de n, calculamos $\varphi(n) = (p-1)(q-1)$. Dado el valor de $\varphi(n)$ y de e, podemos calcular d utlizando el algoritmo de Euclides extendido.

Conversamente, supogamos que conocemos d, y buscamos obtener una factorización de n. Primero calculamos $k = de - 1$. Puesto que $de \equiv 1 \pmod{(p-1)(q-1)}$, tenemos que k es múltiplo de $\varphi(n) = (p-1)(q-1)$. Puesto que $\varphi(n)$ es par, podemos escribir $k = 2^t r$, donde r is impar y $t \geq 1$.

Tenemos que $g^k \equiv 1 \pmod{n}$ para toda $g \in \mathbb{Z}_n^*$, por el Teorema de Euler, de manera que $g^{k/2}$ es una raíz cuadrado de la unidad módulo n. Por el Teorema Chino del Residuo, puesto que n es el producto de dos primos, la unidad tiene cuatro raíces cuadradas módulo n; éstas son 1, -1, y los enteros x y $-x$, donde x satisface:

$$\begin{aligned} x &\equiv 1 \pmod{p}, \\ x &\equiv -1 \pmod{q}. \end{aligned}$$

Si tomamos x ó $-x$, entonces $\mathrm{mcd}(x-1, n)$ vale p ó q, lo cual proporciona la factorización de n. Es fácil verificar que si g es elegido al azar de los elementos de $\mathbb{Z}_n^*$, entonces con probabilidad al menos $\frac{1}{2}$ alguno de los elementos de la sucesión $g^{k/2}, g^{k/4}, \ldots, g^{k/2^t} \bmod n$ es una raíz cuadrada de la unidad con la que obtenemos la factorización de n. Todos los elementos de la sucesión se pueden calcular eficientemente, en tiempo $O(\log^3(n))$. Entonces, para factorizar n, tomamos elementos $g \in \mathbb{Z}_n$; si $\mathrm{mcd}(g, n) > 1$, esto proporciona un factor propio de n, y por ende su factorización. Si $\mathrm{mcd}(g, n) = 1$, entonces $g \in \mathbb{Z}_n^*$, y procedemos a ver si algún elemento de la sucesión

$$g^{k/2}, g^{k/4}, \ldots, g^{k/2^t}$$

proporciona un factor propio de n. Si no, escogemos otra $g \in \mathbb{Z}_n$. Esperamos obtener una factorización despueés de dos intentos, en promedio. □

Este teorema nos dice que la seguridad de RSA es equivalente a la dificultad del problema de factorización del módulo[11].

Nótese que hablamos de factorizar el módulo de RSA, no de factorizar en general. El problema de factorizar un módulo de RSA es un caso particular de el problema general

[11] Como veremos en el capítulo siguiente, nos referimos a la seguridad contra una ruptura total. No se sabe si es posible obtener el valor de M sin obtener el valor de d o una factorización de n.

de factorización, y en general es más fácil factorizar cuando sabemos que el número tiene sólo dos factores primos de tamaños comparables, que cuando no conocemos nada sobre el número.

Vale la pena notar también que hay una simetría entre e y d; si bien nosotros elegimos el valor de e y de ahí calculamos el valor de d, también es posible elegir una llave privada d y utilizarla para calcular la llave pública e. Esto se suele hacer cuando queremos algún valor específico que permita que el descifrado se haga de manera rápida. Por el contrario, si nos interesa que el cifrado se haga de manera rápida, elegimos e.

8.11 Criptosistemas de llave pública vs. simétricos

Los criptosistemas de llave pública tienen varias ventajas por encima de los criptosistemas simétricos como DES: resuelven totalmente el problema de distribución de llaves, y como beneficio adicional proporcionan (en su mayoría) un método de verificación de identidad y de firma electrónica que los sistemas simétricos no proporcionan.

Podemos entonces preguntarnos por qué siguen utlizandose criptosistemas simétricos como DES o AES. La respuesta es simplemente que los criptosistemas de llave pública son extremadamente lentos comparados con los sistemas simétricos. Esto provoca que el costo de enviar mensajes largos utilizando criptosistemas de llave pública sea muy superior al costo de enviarlos con criptosistemas simétrcos. Esta diferencia es suficientemente importante para justificar la existencia continuada de los criptosistemas simétricos.

En este capítulo vamos a iniciar el criptoanálisis de algunos sistemas de llave pública. Para ello, vamos a analizar dos problemas matemáticos, que son la base de la mayoría de los criptosistemas de llave pública que tenemos: el problema del logaritmo discreto, y el problema de factorización. En el siguiente capítulo, estudiaremos algunos ataques contra RSA que no se basan directamente en la factorización del módulo.

Vamos a comenzar repasando algunos conceptos, e indicando la complejidad de algunos algoritmos básicos que se utilizan en Teoría de Números.

9.1 Ruptura contra Ruptura Total

Recordemos que en un criptosistema de Llave Pública, cada usuario A tiene asociada una llave privada d_A, que a veces es llamada la llave de descifrado; y una llave pública, e_A,

a veces llamada la llave de cifrado. El valor de e_A es conocido y público. Para envíar un mensaje M a A, enviamos $E(M, e_A)$, es decir, el resultado de cifrar el mensaje M usando la llave e_A.

A diferencia de los métodos criptoanalíticos que vimos en los capítulos anteriores, en el caso de los sistemas de llave pública es posible resolver un problema de criptotexto conocido sin recuperar la llave privada d_A. Eso nos lleva a dos posibles "éxitos" por parte del criptoanálista:

1. **Ruptura del sistema.** Decimos que el criptoanalista ha *roto el sistema* si, conociendo el algoritmo E, y dados $E(M, e_A)$ y e_A específicos, le es posible recuperar M.

2. **Ruptura total del sistema.** Decimos que el criptoanalista ha obtenido una *ruptura total del sistema* si, conociendo el algoritmo de cifrado E y la llave pública e_A, puede recuperar la llave privada d_A (mediante cualquier tipo de ataque; por ejemplo, texto claro elegido, criptotexto conocido, texto claro conocido, etc.).

Notemos que una ruptura total del sistema implica una ruptura, pero el converso no es necesariamente cierto.

9.2 Los problemas asociados a los criptosistemas de llave pública

En 1988, Whitfield Diffie notó que la mayoría de los algoritmos de llave pública están basados en uno de los siguientes tres problemas:

1. **Knapsack.** Dado un conjunto de enteros, encontrar un subconjunto cuya suma sea un número N dado.

2. **Logaritmo Discreto.** Si p es un primo, y g y M son enteros, encontrar un entero x tal que $g^x \equiv M \pmod{p}$; y sus variantes para campos finitos y grupos abelianos.

3. **Factorización.** Si N es el producto de dos primos, entonces se busca resolver alguno de los siguientes problemas:

   (a) Factorizar N (Problema de Factorización).

   (b) Dados M y C, encontrar d tal que $M^d \equiv C \pmod{N}$ (Problema de RSA).

(c) Dados enteros e y C, encontrar M tal que $M^e \equiv C \pmod{N}$ (Problema inverso de RSA).

(d) Dado un entero x, decidir si existe un entero y tal que $x \equiv y^2 \pmod{N}$ (Problema de Residuosidad Cuadrática).

Los incisos que aparecen en el problema de Factorización reflejan que en el caso de sistemas que se basan en la factorización, a veces un problema más sencillo basta para romper el sistema, mientras que la factorización de N proporciona una ruptura total.

El problema de Knapsack no lo vamos a estudiar, pues los criptosistemas que originalmente se basaban en él son inseguro; aunque todavía hay variantes que son consideradas seguras, nos vamos a concentrar en los otros dos problemas.

9.3 Complejidad de algunas operaciones sencillas en Teoría de Números

9.3.1 La notación O

Es común expresar la complejidad de un algoritmo en términos de la notación O, *"orden de."* La definición formal es la siguiente:

Definición 9.1 Sean $f, g: D \to \mathbb{R}$, donde $\mathbb{N} \subset D \subset \mathbb{R}$. Decimos que $f(x)$ *es del orden de* $g(x)$, $f(x) = O(g(x))$, si y sólo si existen constantes $C, n_0 > 0$ tales que

$$\forall x > n_0, \qquad |f(x)| \leq C|g(x)|.$$

Es fácil verificar que si $f(x) = a_n x^n + \cdots + a_1 x + a_0$, con $a_n \neq 0$, entonces $f(x) = O(x^n)$.

Si $f(x) = O(1)$, entonces decimos que $f(x)$ está acotada. Si $f(x) = O(x)$, entonces decimos que $f(x)$ es de orden lineal; $f(x) = O(x^2)$ es de orden cuadrático, etc. En general, si $f(x) = O(x^n)$ para algún entero positivo n, decimos que $f(x)$ es de *orden polinomial*.

Si $f(x) = O(\log^\alpha x)$, con $\alpha > 0$, decimos que $f(x)$ es de orden logarítmico; si $f(x) = O(\exp(g(x)))$, donde $g(x)$ es una función polinomial de x, entonces decimos que $f(x)$ es de orden exponencial.

En el caso de algoritmos de teoría de números, nuestra función f es una función cuyo valor en x es el tiempo que tarda el algoritmo en terminar, dado como entrada x. En

términos generales, suelen ser funciones de la *longitud* de x, es decir, de $\log(x)$. Esto suele provocar confusiones; por ejemplo, el algoritmo de Euclides es de orden polinomial, puesto que es polinomial en el logaritmo de su argumento. En cambio, factorizar n por división es un algoritmo exponencial, pues es $O(\sqrt{n}) = O(\exp(\frac{1}{2}\log\, n))$.

En general, pensamos que algoritmos que son polinomiales en $\log(n)$ son "rápidos" y "eficientes", mientras que algoritmos que son exponenciales en $\log(n)$ son "difíciles."

9.3.2 Complejidad de operaciones

Para poder estimar el tiempo que tardan diversos algoritmos complejos, es necesario saber cuánto tiempo tardan ciertas operaciones sencillas. En esta sección, listamos, sin demostración, la complejidad computacional de varias de estas operaciones. Las demostraciones se pueden encontrar en varios libros, por ejemplo en [Kob94].

1. Sumar n y m, con $n < m$, tarda $O(\log(m))$.

2. Multiplicar n y m tarda $O(\log(n)\log(m))$. Para cálculos aproximados, con $n < m$, tiene complejidad $O(\log^2(m))$.

3. Dividir con residuo el entero a entre el entero b también tarda $O(\log(a)\log(b))$.

4. Calcular el máximo común divisor de a y b mediante el algoritmo de Euclides extendido tiene complejidad $O(\log(a)\log(b))$.

5. Teorema Chino del Residuo: dados $a_1, \ldots, a_k$, y enteros positivos primos relativos dos a dos $n_1, \ldots, n_k$, encontrar la única x módulo $N = \prod n_i$ tal que $x \equiv a_i \pmod{n_i}$ tarda $O(\log^2(N))$, usando el método de Gauss.

6. La suma y resta modular, $a \pm b \pmod{n}$, dados $a \pmod{n}$ y $b \pmod{n}$, tiene complejidad $O(\log(n))$.

7. Multiplicación modular, es decir, dados $a \pmod{n}$ y $b \pmod{n}$, calcular $ab \pmod{n}$ tarda $O(\log^2(n))$.

8. Invertir modularmente, es decir, dado $a \pmod{n}$ calcular $a^{-1} \pmod{n}$ si es que existe tiene complejidad $O(\log^2(n))$.

9. Exponenciación modular: dado $a \pmod{n}$, y un entero positivo $k < n$, calcular $a^k \pmod{n}$ tarda $O(\log^3(n))$.

9.3.3 "Eleva al cuadrado y multiplica"

El método más usado para calcular exponenciación modular es muy importante y vale la pena mencionarlo; se le llama el método de *eleva al cuadrado y multiplica* (*"square and multiply"* en inglés).

Si queremos calcular, por ejemplo, $3^{10} \pmod{13}$ de manera inocente, podemos calcular

$$((((((((3 \times 3) \times 3) \times 3) \times 3) \times 3) \times 3) \times 3) \times 3)$$

y ya sea ir reduciendo módulo 13, o reducir al final. Esto requiere nueve multiplicaciones.

O bien, podemos notar que $10 = 8 + 2 = 2^3 + 2^1$, y por lo tanto,

$$3^{10} = 3^{8+2} = ((3^2)^2)^2 \times 3^2.$$

Podemos calcular $3^2 \pmod{13}$, y guardar el resultado en un registro. Luego calcular $(3^2)^2 \pmod{13}$, y luego $((3^2)^2)^2 \pmod{13}$, y multiplicar el resultado por el total guardado. En total, hicimos cuatro multiplicaciones modulares, lo cual representa un ahorro significativo.

El procedimiento general es el siguiente: para calcular $a^b \pmod{N}$, primero escribimos b en binario:

$$b = a_0 + a_1 2 + a_2 2^2 + \cdots + a_m 2^m, \qquad a_i \in \{0, 1\}, \quad a_m = 1,$$

y luego inicializamos dos variable $c = 1$, que es el acumulador, y $t = b$, que es el resultado parcial de elevar al cuadrado. En cada paso, $i = 0, \ldots, m-1$, si $a_i = 1$, entonces multiplicamos c por p módulo N y guardamos el resultado en c; si $a_i = 0$, dejamos c como estaba; y en cualquiera de los dos casos, calculamos $t^2 \pmod{N}$ y guardamos el resultado en t. En el m-ésimo paso, como $a_m = 1$, multiplicamos c por t módulo N, y el resultado es igual a $a^b \pmod{N}$.

9.4 El Problema del Logaritmo Discreto

Hay dos variantes del problema del logaritmo discreto:

Problema del Logaritmo Discreto: Dado un primo p, un generador α de $\mathbb{Z}_p^*$, y un elemento $\beta \in \mathbb{Z}_p^*$, encontrar el único entero x, $1 \leq x \leq p-1$ tal que $\alpha^x \equiv \beta \pmod{p}$; es decir, calcular $\log_\alpha(\beta) = x$.

Problema Generalizado del Logaritmo Discreto: Dado un grupo cíclico G de orden n, un generador α, y un elemento $\beta \in G$, encontrar el único entero x, $1 \leq x \leq n-1$ tal que $\alpha^x = \beta$ en G.

En algunos casos, por ejemplo, el grupo cíclico $\mathbb{Z}_n$, el problema generalizado del logaritmo discreto es muy sencillo. En otros, no parece serlo. Los ejemplos más usados son el grupo multiplicativo de un campo finito, $\mathbb{F}_q$, donde $q = p^n$ es la potencia de un primo. Se sabe que para cada primo p, y cada entero positivo n, existe un único campo con $\mathbb{F}_q$ elementos (salvo isomorfismos), y que los únicos campos finitos que existen tienen p^n elementos, para algún primo p y algún entero positivo n. Además, el grupo multiplicativo de $\mathbb{F}_q$ (es decir, el grupo cuyos elementos son los elementos distintos de cero de $\mathbb{F}_q$, y cuya operación es la multiplicación en $\mathbb{F}_q$) siempre es cíclico:

Teorema 9.1 *Sea $\mathbb{F}$ un campo finito. Entonces $\mathbb{F}^*$, el grupo multiplicativo de $\mathbb{F}$, es cíclico.*

Dem.: Sea m el número de elementos de $\mathbb{F}$. Entonces, $\mathbb{F}^*$ tiene $m-1$ elementos.

Supongamos que $\mathbb{F}^*$ no es cíclico. De acuerdo con el Teorema Fundamental de Grupos Abelianos Finitamente Generados, se puede escribir como una suma de grupos cíclicos:

$$\mathbb{F}^* \cong \mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_r},$$

donde n_1 divide a n_2, n_2 divide a n_3, ..., y n_{r-1} divide a n_r. Como $\mathbb{F}^*$ no es cíclico, $r > 1$ y $n_r < m-1$.

Sin embargo, es claro entonces que $x^{n_r} = 1$ para toda $x \in \mathbb{F}^*$; y por lo tanto, $x^{n_r+1} = x$ para toda $x \in \mathbb{F}$ (incluyendo al cero). Eso quiere decir que el polinomio

$$f(x) = x^{n_r+1} - x$$

tiene m soluciones distintas en $\mathbb{F}$; pero un polinomio de grado k en un campo no puede tener más de k soluciones distintas en el campo, y en este caso, $n_r + 1 < m$, lo cual es una contradicción.

La contradicción surgió de suponer que $\mathbb{F}^*$ no era cíclico, de manera que concluimos que $\mathbb{F}^*$ es un grupo cíclico. □

El problema del Logaritmo Discreto está relacionado con el algoritmo de intercambio de llaves de Diffie-Hellman, y con los criptosistemas de ElGamal y Massey-Omura.

En realidad, en esos casos el problema del Logaritmo Discreto es un poco más de lo que parece necesitarse; en todos ellos, si logramos resolver el problema del Logaritmo Discreto

asociado al grupo que se está usando, eso nos proporciona una ruptura (en algunos casos total) del sistema. Sin embargo, el problema que forma la base de esos algoritmos es el siguiente:

Problema de Diffie-Hellman: Dado un grupo cíclico G, un generador α, y dados los elementos α^a, α^b, con $1 < a, b < n$, encontrar α^{ab}.

Claramente, si podemos resolver el problema del Logaritmo Discreto para G, entonces podemos resolver el Problema de Diffie-Hellman para G; simplemente calculamos el logaritmo discreto de α^a para obtener el valor de a, y luego calculamos $(\alpha^b)^a$ para obtener α^{ab}. Esto es: el problema de Diffie-Hellman no es más dificil que el Problema del Logaritmo Discreto.

Sin embargo, nadie sabe si el Problema de Diffie-Hellman es equivalente al Problema del Logaritmo Discreto; hasta ahora, a nadie se le ha ocurrido ninguna manera de resolver el Problema de Diffie-Hellman que no utilice de alguna manera un logaritmo discreto, pero tampoco nadie a probado que si pudieramos resolver el Problema de Diffie-Hellman entonces podríamos resolver el Problema del Logaritmo Discreto. Esta proposición es equivalente a la Hipótesis de Diffie-Hellman:

Hipótesis de Diffie-Hellman: El Problema de Diffie-Hellman para un grupo G es igual de dificil que el Problema del Logaritmo Discreto para G.

Hasta ahora, todos los ataques contra los criptosistemas que se basan en la dificultad de resolver el Problema de Diffie-Hellman consisten en calcular el Logaritmo Discreto. De manera que en ésta sección nos concentraremos en los algoritmos que tenemos para calcular logaritmos discretos.

Hay principalmente tres categorías de algoritmos para este problema:

1. Algoritmos que funcionan para grupos arbitrarios, y cuya velocidad no depende del grupo.
2. Algoritmos que funcionan para grupos arbitrarios, pero que son muy buenos para grupos que satisfacen ciertas propiedades matemáticas.
3. Cálculo de Índices, un algoritmo muy bueno que sólo funciona para cierto tipo de grupos.

Vamos a cubrirlos en orden. En todos los casos, suponemos que tenemos dado el grupo cíclico G, el orden del grupo n, un generador α de G, y un elemento β de G. Buscamos calcular $\log_\alpha(\beta)$. A menos que digamos explícitamente lo contrario, escribimos el grupo multiplicativamente, y el elemento neutro se denota por 1.

Vamos a hablar de "multiplicaciones modulares"; nos referimos a multiplicar elementos dentro de G. En la mayoría de los casos, donde G es $\mathbb{Z}_p^*$ para algún primo p, se tratan en realidad de multiplicaciones modulares. En otros casos similares, la complejidad es comparable.

9.4.1 Algoritmos generales que no dependen del grupo

Búsqueda exhaustiva

Este es el algoritmo de fuerza bruta. Consiste en simplemente calcular α, α^2, α^3, etc., hasta encontrar el valor que es igual a β.

El tiempo esperado para este algoritmo es $O(n)$ multiplicaciones modulares.

Paso grande, paso chico

El algoritmo paso grande, paso chico (*"Baby step, giant step"* en ingés) es un cambio de tiempo por memoria. Hay un precálculo extensivo, guardamos una tabla en memoria, y luego la usamos para reducir el tiempo de búsqueda. En el caso de un solo cáculo de logaritmo discreto, no hay ganancia comparado con el método de búsqueda exhaustiva, pero para varios cálculos, parte del problema sólo se tiene que hacer una vez.

La observación crucial del método es la siguiente: sea $m = \lceil \sqrt{n} \rceil$. Si $\beta = \alpha^x$, entonces existen i, j, $0 \leq i, j < m$ tales que $x = im + j$; y tenemos que $\alpha^x = \alpha^{im}\alpha^j$.

El algoritmo de paso grande, paso chico, comienza construyendo una tabla, cuyas entradas son (j, α^j), con $0 \leq j \leq m$; estos son los "pasos chicos." Después, reordenamos la tabla para que estén ordenadas de acuerdo a la segunda componente. Finalmente, calculamos α^{-m}, el "paso grande." Esto se hace durante la fase de precálculo; sólo hay que hacerlo una vez, y no depende del elemento β.

Dado β, y ya hecho el precálculo, primero buscamos β en la tabla. Si β es la segunda entrada de algún (j, α^j), con $0 \leq j < m$, es decir, si $\beta = \alpha^j$ para alguna de estas j, entonces simplemente $\log_\alpha(\beta) = j$ y ya terminamos.

Si β no está en la tabla, entonces calculamos $\beta\alpha^{-m}$, y lo buscamos en la tabla. La tabla está ordenada según la segunda coordenada para facilitar ésta búsqueda; en cuanto lleguemos a un elemento de la tabla cuya segunda coordenada es mayor que el número que buscamos, podemos terminar la búsqueda. Continuamos de esta manera, calculando $\beta\alpha^{-im}$, hasta encontrarlo en la tabla; es decir, hasta encontrar una j tal que $\beta\alpha^{-im} = \alpha^j$.

Entonces tenemos que $\beta = \alpha^{im}\alpha^j$, de manera que $\log_\alpha(\beta) = im + j$. Claramente, encontraremos tales valores para alguna i, $0 \leq i \leq m$.

Para utilizar el algoritmo de paso grande, paso chico, tenemos que guardar $O(\sqrt{n})$ términos; calcular la tabla requiere $O(\sqrt{n})$ mutliplicaciones modulares, y $O(\log(n)\sqrt{n})$ comparaciones para ordenarla. La búsqueda requiere aproximadamente $O(\sqrt{n})$ multiplicaciones y $O(\sqrt{n})$ búsquedas en la tabla.

Dada la tabla, el algoritmo tiene una complejidad aproximada de $O(\sqrt{n})$ mutliplicaciones.

Ejemplo 9.1 $G = \mathbb{Z}_{113}^*$, $\alpha = 3$. El orden del grupo es 112.

Para el precálculo, tomamos $m = \lceil\sqrt{112}\rceil = 11$, y calculamos:

j	0	1	2	3	4	5	6	7	8	9	10
$3^j \bmod 113$	1	3	9	27	81	17	51	40	7	21	63

Después ordenamos la tabla según su segunda entrada:

j	0	1	8	2	5	9	3	7	6	10	4
$3^j \bmod 113$	1	3	7	9	17	21	27	40	51	63	81

Finalmente, calculamos $\alpha^{-1} \equiv 3^{-1} \equiv 38 \pmod{113}$.

Sea $\beta = 57$. Primero lo buscamos en la tabla. Como no está, calculamos $\beta\alpha^{-11} \equiv 29 \pmod{113}$, y lo buscamos en la tabla. No está. Después calculamos $\beta\alpha^{-22} \equiv 100$, y lo buscamos en la tabla. No está. Calculamos $\beta\alpha^{-33} \equiv 37$, y lo buscamos en la tabla; no está. Continuamos de ésta manera calculando $\beta\alpha^{-44} \equiv 112$, $\beta\alpha^{-55} \equiv 55$, $\beta\alpha^{-66} \equiv 26$, $\beta\alpha^{-77} \equiv 39$, $\beta\alpha^{-88} \equiv 2$, hasta $\beta\alpha^{-99} \equiv 3 = \alpha^1$, que está en la tabla.

Tenemos entonces que $\beta\alpha^{-99} = \alpha$, de manera que $\beta = \alpha^{100}$, y $\log_\alpha^\beta = 100$. $\lhd$

Algoritmo ro de Pollard

El algoritmo ro de Pollard es un algoritmo tipo "Monte Carlo" (toma decisiones probabilísticas, y existe la posibilidad de que termine sin encontrar el logaritmo discreto); tiene la misma complejidad esperada que el algoritmo de paso grande, paso chico, pero no requiere de espacio en memoria.

Primero partimos al grupo G en tres conjuntos, S_1, S_2, S_3, de más o menos el mismo tamaño, usando alguna propiedad que sea fácil de verificar. No importa mucho cómo lo partimos, pero por ejemplo, debemos especificar que $1 \notin S_2$.

Después definimos una sucesión de elementos de G, $x_0, x_1, \ldots, x_n, \ldots$, con $x_0 = 1$, y

$$x_{i+1} = f(x_i) = \begin{cases} \beta \cdot x_i & \text{si } x_i \in S_1 \\ x_i^2 & \text{si } x_i \in S_2 \\ \alpha \cdot x_i & \text{si } x_i \in S_3 \end{cases}$$

para toda $i \geq 0$. La sucesión de elementos define a su vez dos sucesiones de enteros módulo n: $a_0, a_1, \ldots$ y $b_0, b_1 \ldots$, tales que $x_i = \alpha^{a_i}\beta^{b_i}$ para cada $i \geq 0$. Es decir, $a_0 = b_0 = 0$, y para toda $i \geq 0$,

$$a_{i+1} = \begin{cases} a_i & \text{si } x_i \in S_1 \\ 2a_i \bmod n & \text{si } x_i \in S_2 \\ a_i + 1 \bmod n & \text{si } x_i \in S_3 \end{cases}$$

$$b_{i+1} = \begin{cases} b_i + 1 \bmod n & \text{si } x_i \in S_1 \\ 2b_i \bmod n & \text{si } x_i \in S_2 \\ b_i & \text{si } x_i \in S_3 \end{cases}$$

La sucesión de x_i es una sucesión en un conjunto finito, así que eventualmente tenemos una repetición. Vamos a tener algo de la forma que aparece en la Figura 9.1, lo cual explica el nombre del algoritmo.

Buscamos indices i, j, $i \neq j$, tales que $x_i = x_j$. Si los encontramos, puesto que $x_i = \alpha^{a_i}\beta^{b_i}$, y $x_j = \alpha^{a_j}\beta^{b_j}$, entonces

$$\log_\alpha(x_i) = a_i + b_i \log_\alpha(\beta) \pmod{n}$$

y también

$$\log_\alpha(x_j) = a_j + b_j \log_\alpha(\beta) \pmod{n}.$$

Puesto que a_i, a_j, b_i, b_j son conocidos, obtenemos

$$(b_i - b_j)\log_\alpha(\beta) \equiv (a_j - a_i) \pmod{n}.$$

Si $b_i \not\equiv b_j \pmod{n}$, entonces podemos despejar el valor de $\log_\alpha(\beta)$ de la ecuación.

Se puede probar que en la mayoría de los casos, vamos a estar en el caso "bueno" en que se puede despejar el valor. En caso que $b_i \equiv b_j \pmod{n}$, el algoritmo termina anunciando su fracaso.

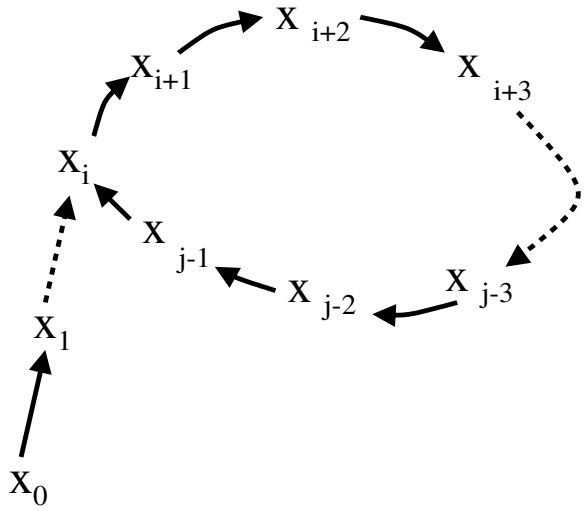

Figura 9.1: La sucesión d x_i del algoritmo ro de Pollard.

La manera inocente de proceder para encontrar los indices i y j sería ir calculando los valores (x_i, a_i, b_i) para cada i, guardarlos en memoria, e ir comparando hasta encontrar una repetición. Pero entonces usamos más memoria que en el de paso grande, paso chico, y se pierde el tiempo comparando. Por ello, se utiliza mejor el Algoritmo de Floyd para Búsqueda de Ciclos.

Algoritmo de Floyd para Búsqueda de Ciclos. Uno empieza con la pareja (x_1, x_2), y calcula de manera iterativa las parejas (x_i, x_{2i}) a partir de la pareja anterior (x_{i-1}, x_{2i-2}), hasta que $x_m = x_{2m}$ para alguna m. Si la cola de la sucesión tiene longitud λ y el ciclo tiene longitud μ (se puede calcular los valores esperados de λ y μ), entonces la primera vez que $x_m = x_{2m}$ ocurren con

$$m = \mu(1 + \lfloor \lambda/\mu \rfloor).$$

Entonces en cada paso sólo tenemos en memoria los valores de x_i y de x_{2i}, y no guardamos la tabla. Puesto que el valor de m arriba satisface $\lambda < m \leq \lambda + \mu$, se puede probar que la complejidad esperada del Algoritmo de Floyd es $O(\sqrt{n})$. Por ello, el tiempo esperado para el algoritmo ro de Pollard es $O(\sqrt{n})$ operaciones del grupo.

En el raro caso en que el algoritmo termine con un fracaso, podemos repetir el proceso eligiendo enteros arbitrarios a_0, b_0 entre 1 y $n-1$, y empezando con $x_0 = \alpha^{a_0}\beta^{b_0}$.

Ejemplo 9.2 $G = \mathbb{Z}^*_{383}$, $\alpha = 2$, $n = 191$, $\beta = 228$.

Partimos el conjunto G en tres subconjuntos de acuerdo a la siguiente regla: $x \in S_1$ si

i	x_i	a_i	b_i	x_{2i}	a_{2i}	b_{2i}
1	228	0	1	279	0	2
2	279	0	2	184	1	4
3	92	0	4	14	1	6
4	184	1	4	256	2	7
5	205	1	5	304	3	8
6	14	1	6	121	6	18
7	28	2	6	144	12	38
8	256	2	7	235	48	152
9	152	2	8	72	47	54
10	304	3	8	14	96	118
11	372	3	9	256	97	119
12	121	6	18	304	98	120
13	12	6	19	121	5	51
14	144	12	38	144	10	104

Tabla 9.1: Pasos intermedios del algoritmo ro de Pollard

$x \equiv 1 \pmod 3$; $x \in S_2$ si $x \equiv 0 \pmod 3$; y $x \in S_3$ si $x \equiv 2 \pmod 3$. La Tabla 9.1 da los valores de x_i, a_i, b_i, x_{2i}, a_{2i}, y b_{2i} al final de cada paso del algoritmo. Entonces tenemos que $x_{14} = x_{28} = 144$. Calculamos $r = b_{14} - b_{28} \bmod 191 = 125$, $r^{-1} = 136 \bmod 191$, y entonces

$$\log_\alpha(\beta) = r^{-1}(a_{28} - a_{14}) \bmod 191 = 110.$$

◁

9.4.2 Algoritmos generales, buenos para grupos particulares

Algoritmo de Pohlig-Hellman

El algoritmo de Pohlig-Hellman asume que además de conocer el orden n del grupo G, también conocemos la factorización de n. El algoritmo es particularmente bueno cuando todos los factores primos de n son "pequeños." Para formalizar el concepto, damos algunas definiciones:

Definición 9.2 *Sea $B > 0$ un entero positivo. Decimos que un entero n es B-homogéneo o B-liso (en inglés, "B-smooth") si y sólo si para todo número primo p, si $p|n$, entonces $p < B$.*

Definición 9.3 *Decimos informalmente que un entero n es homogéneo si es B-homogéneo para alguna B suficientemente chica.*

Que tan chica es "suficientemente chica" depende de nuestras aplicaciones.

Supongamos que $n = p_1^{e_1} \cdots p_r^{e_r}$, donde cada p_i es un primo, $p_1 < p_2 < \cdots < p_r$, y $e_i > 0$ para toda i.

Si $x = \log_\alpha(\beta)$, podemos calcular los valores

$$x_i \equiv x \pmod{p_i^{e_i}}.$$

Conversamente, si conocemos todos los valores $x_i \bmod p_i^{e_i}$, entonces podemos usar el Teorema Chino del Residuo para obtener el valor de $x \bmod n$.

Claro que el problema es que no sabemos quién es x_i sin saber quien es x, o al menos, no se vé ninguna manera clara de cómo encontrarlo.

El algorimo de Pohlig-Hellman determina el valor de x_i. Esto se logra calculando los dígitos $\ell_0, \ell_1, \ldots, \ell_{e_i-1}$ de la representación base p_i:

$$x_i = \ell_0 + \ell_1 p_i + \cdots + \ell_{e_i} p_i^{e_i-1},$$

con $0 \leq \ell_j \leq p_i - 1$.

Para calcular los valores de ℓ_j, primero calculamos el valor de $\overline{\alpha} = \alpha^{n/p_i}$, e inicializamos $\gamma = 1$, $\ell_{-1} = 0$. Si ya tenemos el valor de ℓ_{j-1}, entonces calculamos el valor de ℓ_j de la siguiente manera: Primero, multiplicamos γ por $\alpha^{\ell_{j-1} p_i^{j-1}}$, y guardamos el resultado en γ. Después calculamos $\overline{\beta} = (\beta\gamma^{-1})^{n/q^{j+1}}$; y finalmente, el valor de ℓ_j está dado por:

$$\ell_j = \log_{\overline{\alpha}}(\overline{\beta}).$$

Ejemplo 9.3 Vamos a usar el Algoritmo de Pohlig-Hellman para calcular el logaritmo discreto de $\beta = 210$ en el grupo $\mathbb{Z}_{251}^*$, con generador $\alpha = 71$. La factorización de $n = 250$ está dada por

$$n = 250 = 2 \times 5^3.$$

Primero calculamos $x_1 = x \bmod 2$. Para ella, tomamos $\overline{\alpha} = a^{n/2} \bmod 251 = 250$, y $\overline{\beta} = \beta^{n/2} \bmod 251 = 250$. Entonces $x_1 = \log_{250}(250) = 1$; por lo tanto, $x_1 = 1$.

Después calculamos $x_2 = x \bmod 5^3 = \ell_0 + \ell_1 5 + \ell_2 5^2$. Para ello, calculamos $\overline{\alpha} = \alpha^{n/5} \bmod 251 = 20$. Después tomamos $\gamma = 1$, $\overline{\beta} = (\beta\gamma^{-1})^{n/5} \bmod p = 149$. Como el orden de $\overline{\alpha}$ es pequeño, usamos búsqueda exhaustiva para obtener que

$$\ell_0 = \log_{20}(149) = 2.$$

Para calcular ℓ_1, tomamos $\gamma = 1\alpha^2 \bmod 251 = 21$, y $\overline{\beta} = (\beta\gamma^{-1})^{n/25} \bmod 251 = 113$. Nuevamente, usando búsqueda exhaustiva, tenemos que

$$\ell_1 = \log_{20}(113) = 4.$$

Para calcular ℓ_2, tomamos $\gamma = 21 \times \alpha^{4\cdot 5} \bmod 251 = 115$, y $\overline{\beta} = (\beta\gamma^{-1})^{n/125} \bmod 251 = 149$. Usamos búsqueda exhaustiva para encontrar que:

$$\ell_2 = \log_{20}(149) = 2.$$

Por lo tanto $x_2 = \ell_0 + \ell_1 5 + \ell_2 5^2 = 172$.

Finalmente, resolvemos el sistema de congruencias

$$\begin{aligned} x &\equiv 72 \pmod{125} \\ x &\equiv 1 \pmod{2} \end{aligned}$$

para obtener que $x = \log_{71}(21) = 197$. ◁

Dada la factorización de n, la complejidad del algoritmo de Pohlig-Hellman es

$$O(\sum_{i=1}^{r} e_i(\log(n) + \sqrt{p_i}))$$

operaciones del grupo. En particular, el algoritmo es eficiente cuando cada divisor primo p_i de n es relativamente chico, es decir, cuando n es un número homogéneo.

Si n es primo, entonces el algoritmo de Pohlig-Hellman degenera al algoritmo de paso grande, paso chico.

9.4.3 Cálculo de Índices

La Cálculo de Índices es le métodos más poderoso que se conoce para calcular logaritmos discretos. Desgraciadamente, sólo se sabe cómo hacerlo funcionar bien en el caso de $\mathbb{F}_q^*$, y es particularmente bueno dentro de ésta subclase para el caso en que $q = 2^m$, un caso favorito para programadores.

La idea del Cálculo de Índices es buscar una colección $S \subseteq G$, llamada una *base de factores*, tal que:

1. S es relativamente chica.
2. Una proporción relativamente grande de los elementos de G se pueden expresar **eficientemente** como un producto de elementos de S.

Las palabras claves son "relativamente" y "eficientemente."

El algoritmo tiene una fase de precálculo, que sólo se tiene que hacer una vez para cada grupo G, y una fase para calcular el valor de $\log_\alpha(\beta)$.

Precálculo

1. Escogemos nuestra base de factores
$$S = \{p_1, p_2, \ldots, p_t\} \subseteq G.$$
2. Escogemos un entero aleatorio k, $0 \leq k \leq n-1$; calculamos α^k, y de ser posible, lo expresamos en términos de S:
$$\alpha^k = \prod_{i=1}^{t} p_i^{c_i}, \qquad c_i \geq 0;$$
si α^k no puede ser expresado de esta manera, elegimos otro entero k y repetimos.
3. Tomando logaritmos de ambos lados, obtenemos una relación lineal entre los $\log_{\alpha_i}(p_i)$:
$$k \equiv \sum_{i=1}^{t} c_i \log_\alpha(p_i) \pmod{n}.$$
4. Repetimos hasta tener $t + c$ ecuaciones, donde c es un número pequeño elegido de tal manera que el sistema de ecuaciones lineales tenga una alta probabilidad de tener solución única.
5. Resolvemos el sistema de ecuaciones módulo n utilizando eliminación gaussiana, y obtenemos los valores $\log_\alpha(p_i)$, $1 \leq i \leq t$.

Si después de obtener $t + c$ ecuaciones el sistema no tiene solución única, entonces repetimos el proceso para obtener más ecuaciones, hasta obtener solución única.

En la fase de precálculo ya se ven algunas de las razones por las que S debe ser "relativamente chico" y debe satisfacer la condición de "eficiencia." Queremos que S sea chico, para

no tener que encontrar demasiadas ecuaciones, y para que resolver el sistema de ecuaciones sea más rápido. Entre más elementos tiene S, más ecuaciones necesitamos, y más tiempo lleva encontrar la solución del sistema.

Queremos que un número grande de elementos de G sean expresables en términos de S para no tener que tratar muchos valores de k antes de poder expresar α^k en términos de S; y queremos que la expresión se encuentre de manera eficiente para que no nos tardemos mucho en encontrar los valores c_i en el precálculo.

Cálculo

Dado β, y ya hecho el precálculo, procedemos al cálculo de $\log_\alpha(\beta)$.

1. Escogemos un entero aleatorio k, $0 \leq k \leq n-1$, y calculamos $\beta\alpha^k$.
2. Tratamos de expresar $\beta\alpha^k$ en términos de S:

$$\beta\alpha^k = \prod_{i=1}^{t} p_i^{d_i}, \qquad d_i \geq 0.$$

3. Si no es posible, escogemos otra k y repetimos.
4. Si podemos encontrar la expresión, tomamos logaritmos de ambos lados y obtenemos:

$$\begin{aligned} \log_\alpha(\beta) + k &= \sum_{i=1}^{t} d_i \log_\alpha(p_i) \\ \log_\alpha(\beta) &= \left(\sum_{i=1}^{t} d_i \log_\alpha(p_i)\right) - k. \end{aligned}$$

Puesto que los valores de $\log_\alpha(p_i)$ ya son conocidos, esto nos da el valor de $\log_\alpha(\beta)$.

Nuevamente, vemos que queremos que sea relativamente fácil expresar elementos de G como productos de elementos de S para que no tengamos que tratar con muchas k, y para que sea fácil determinar si podemos expresarlo, y en ese caso cuál es el valor de las d_i.

El principal problema para el Cálculo de Índices es entonces escoger S. En el caso en que $G = \mathbb{Z}_p^*$ con p un primo, se suele usar:

$$S = \{\text{primeros } t \text{ primos}\} \cup \{-1\},$$

donde t es no muy grande.

En el caso en que $G = \mathbb{F}_{p^n}^*$, con $n > 0$, y p chico, representamos $\mathbb{F}_{p^n}$ como un anillo de polinomios con coeficientes en $\mathbb{F}_p$ módulo ciertas relaciones (la construcción usual de extensiones de campos), y tomamos para S todos los $f(x) \in \mathbb{F}_p[x]$ irreducibles y de grado menor o igual a $n-1$, y le agregamos todas las constantes.

Cuando $p = 2$, hay una variante al algoritmo que se debe a Coppersmith, que hace al Cálculo de Índices mucho más eficaz. En términos generales, para tener la misma seguridad que el problema del logaritmo discreto para $\mathbb{F}_{p^n}$ con $p \geq 3$, se requiere cuando trabajamos con $p = 2$un exponente del doble de tamaño; es decir, $\mathbb{F}_{2^m}$ con $\log_2(m) \approx 2\log_2(n)$.

La complejidad del Cálculo de Índices es aproximadamente:

$$O\Big(\exp\Big(c\sqrt{\log(q)}\sqrt{\log(\log(q))}\Big)\Big)$$

multiplicaciones modulares, donde $c \approx 2$, y estamos trabajando en $\mathbb{F}_q$. El valor de c es fijo y constante, y depende de la elección exacta de S. La variante de Coppersmith reduce c a $c < 1.587$.

Este tipo de complejidad se conoce como *tiempo subexponencial*, pues es menor que el tiempo exponencial, pero asintóticamente mayor que cualquier tiempo polinomial.

Ejemplo 9.4 Vamos a usar Cálculo de Índices para calcular logaritmo discreto en $\mathbb{Z}_{229}^*$, con generador $\alpha = 6$, de orden $n = 228$. Sea $\beta = 13$, y buscamos $\log_{16}(13)$.

Primero escogemos nuestra base de factores; en este caso, tomamos $S = \{2, 3, 5, 7, 11\}$, los primeros once primos. (El grupo es suficientemente chico para que no valga la pena incluír a -1 en nuestra base de factores).

Después buscamos relaciones que involucren los elementos de la base de factores. No vamos a poner los intentos fallidos, pero se encuentran las siguientes relationes:

$$\begin{aligned}
6^{100} \bmod 229 = 180 &= 2^2 \cdot 3^2 \cdot 5 \\
6^{18} \bmod 229 = 176 &= 2^4 \cdot 11 \\
6^{12} \bmod 229 = 165 &= 3 \cdot 5 \cdot 11 \\
6^{62} \bmod 229 = 154 &= 2 \cdot 7 \cdot 11 \\
6^{143} \bmod 229 = 198 &= 2 \cdot 3^2 \cdot 11 \\
6^{206} \bmod 229 = 210 &= 2 \cdot 3 \cdot 5 \cdot 7.
\end{aligned}$$

Con estas ecuaciones obtenemos un sistema de seis ecuaciones en cinco incógnitas, a saber:

$$\begin{array}{rcl}
100 &\equiv& 2\log_6(2) + 2\log_6(3) + \log_6(5) \\
18 &\equiv& 4\log_6(2) + \log_6(11) \\
12 &\equiv& \log_6(3) + \log_6(5) + \log_6(11) \\
62 &\equiv& \log_6(2) + \log_6(7) + \log_6(11) \\
143 &\equiv& \log_6(2) + 2\log_6(3) + \log_6(11) \\
206 &\equiv& \log_6(2) + \log_6(3) + \log_6(5) + \log_6(7)
\end{array}$$

donde todas las congruncias son módulo 228, y son ecuaciones en las incógnitas $\log_6(2)$, $\log_6(3)$, $\log_6(5)$, $\log_6(7)$, y $\log_6(11)$.

Resolvemos el sistema usando eliminación gaussiana, y obtenemos las siguientes soluciones:

$$\begin{aligned}
\log_6(2) &= 21, \\
\log_6(3) &= 208, \\
\log_6(5) &= 98, \\
\log_6(7) &= 107, \\
\log_6(11) &= 162.
\end{aligned}$$

Esto completa la fase de precálculo. Ahora procedemos al cálculo. Tomamos $k = 77$ de manera aleatoria. Calculamos

$$\beta\alpha^k = 13 \cdot 6^{77} \bmod 229 = 147 = 3 \cdot 7^2,$$

de manera que

$$\log_6(13) = (\log_6(3) + 2\log_6(7) - 77) \bmod 228 = 117.$$

◁

9.4.4 Diffie-Hellman vs. Logaritmo Discreto

Cabe mencionar que, utilizando curvas elípticas, Marer, Wolf, y Boneh han establecido la equivalencia del Problema de Diffie-Hellman y el Problema del Logaritmo Discreto en ciertos casos especiales, a saber:

1. Si p es primo, $p-1$ tiene factorización conocida, y $\phi(p-1)$ (ϕ de Euler) es B-homogéneo con $B \approx O((\log(p))^c)$, cuando trabajamos en $\mathbb{Z}_p^*$.

2. Si G es de orden n, la factorización de n es conocida, y $\phi(n)$ es B-homogéneo, con $B \approx O((\log(p))^c)$.

3. Si G es de orden n, y para cada divisor primo p de n, $p-1$ ó $p+1$ son B-homogéneos, con $B \approx O((\log(p))^c)$.

9.5 El Problema de Factorización

La factorización de enteros está íntimamente ligada con el criptosistema RSA. Igual que en el caso de la relación entre el Problema del Logaritmo Discreto y los criptosistemas que se basan en Diffie-Hellman, resolver el Problema de Factorización nos da una ruptura total de RSA, pero el problema de criptoanálisis de RSA requiere de un poco menos.

Recordemos cuál es el Problema de Factorización:

Problema de Factorización. Dado un entero n, encontrar su factorización como producto de primos.

En general, el Problema de Factorización se resuelve de manera iterativa, utilizando el Problema de Descomposición:

Problema de Descomposición. Dado un entero n que no es primo, encontrar enteros a y b, tales que $1 < a, b < n$ y $n = ab$.

Para resolver el Problema de Factorización a partir del de Descomposición, simplemente le aplicamos el Problema de Descomposición a a y b, y repetimos hasta llegar a factores primos. En general, es mucho más fácil saber si un entero es primo o no que factorizar o descomponer.

Por otro lado, la ruptura de RSA requiere de resolver el siguiente problema:

Problema de RSA. Dados un entero n que es el producto de dos primos distintos, un entero e que satisface $(e, \phi(n)) = 1$, y el valor $M^e \bmod n$, recuperar M.

Nuevamente, no se sabe si el Problema de RSA es equivalente al Problema de Factorización (o incluso, al Problema de Descomposición aplicado al producto de dos primos distintos). Pero a diferencia del caso de los criptosistemas que se basan en el Logaritmo Discreto, existen métodos criptoanalíticos para atacar RSA que no dependen directamente de factorizar el módulo n. Algunos de estos métodos los vamos a ver en el capítulo siguiente.

En esta sección, sin embargo, nos vamos a concentrar en los algoritmos que existen para factorizar enteros.

Algunos algoritmos para factorizar están especializados para que funcionen muy bien cuando el entero n tiene alguna forma especial; estos son llamados *algoritmos de factorización de propósito especial*. Ejemplos de algoritmos de propósito especial son el método de división, el algoritmo ro de Pollard, el algoritmo $p-1$ de Pollard, la factorización por curvas elípticas, y la criba especial de campos numéricos. Los algoritmos que tienen una complejidad que depende exclusivamente del tamaño de n, por otro lado, son *algoritmos de factorización generales.* Ejemplos de algoritmos generales son la criba cuadrática y la criba general de campos numéricos. En general, cuando un método de factorización de propósito especial es aplicable, conviene usarlo. Por ello, la estrategia más usual para factorizar un número arbitrario es intentar aplicar algoritmos que encuentran factores pequeños primero, y capitalizar la forma especial que el número tendría como consecuencia; y si todo falla, entonces sacar los algoritmos generales y usarlos.

Un ejemplo de una estrategia general sería:

1. Buscar factores primos pequeños mediante división, con primos $p < b_1$ para alguna cota b_1.
2. Después, aplicar el algoritmo ro de Pollard, buscando factores primos p con $b_1 \leq p < b_2$ para alguna cota b_2.
3. Utilizar el algoritmo de factorización con curvas elípticas para buscar factores primos p, $b_2 \leq p < b_3$ para alguna cota b_3.
4. Finalmente, si todo lo anterior falla, aplicar un algoritmo general como la criba cuadrática o la criba de campos numéricos.

Para estudiar los algoritmos, vamos a suponer que nos dan un entero impar positivo n (los factores de 2 los podemos extraer fácilmente), y que sabemos que n no es primo, y que no es una potencia perfecta (es decir, que $n \neq a^b$ para enteros $a > 0$, y $b > 1$; esto es fácil de verificar también).

9.5.1 Algoritmos para buscar factores pequeños de n

División

Simplemento, hacemos división entera de n entre los primeros primos $p_1, \ldots, p_t, \ldots$ hasta encontrar uno que deje residuo 0. El caso extremo requiere que probemos todos los primos hasta $\lfloor \sqrt{n} \rfloor$; éste es el caso cuando n es el producto de dos primos de más o menos el

mismo tamaño. Pero el caso general tarda aproximadamente $O(p+\log(n))$ divisiones hasta encontrar un factor, donde p es el segundo factor primo más grande de n.

En general, si usamos división para un número grande elegido de manera aleatoria, podemos esperar que el algoritmo encuentre factores pequeños de manera relativamente rápida, pero que se tarde mucho tiempo en encontrar el segundo factor primo más grande de n. Expícitamente:

Teorema 9.2 *Sea n un entero elegido de manera aleatoria en el intervalo $[1, x]$.*

(i) Si $\frac{1}{2} \leq \alpha \leq 1$, entonces la probabilidad de que el factor primo más grande de n sea menor o igual a x^α es aproximadamente $1 + \log(\alpha)$. Por ejemplo, la probabilidad de que n tiene un factor primo mayor que $\sqrt{x}$ es aproximadamente $\log(2) \approx 0.69$.

(ii) La probabilidad de que el segundo factor primo más grande de n sea menor o igual a $x^{0.2117}$ es aproximadamente $\frac{1}{2}$.

(iii) El número esperado de factores primos de n (contando multiplicidad) es aproximadamente $\log\log(n) + O(1)$.

Algoritmo ro de factorización de Pollard

Este algoritmo tiene varias ideas en común con el algoritmo de nombre similar para encontrar logaritmos discretos. En general es bueno para encontrar factores primos pequeños de enteros que son compuestos.

Sea $f: S \to S$ una función arbitraria de un conjunto de tamaño n. Sea $x_0 \in S$, y definimos $x_{i+1} = f(x_i)$. Tarde o temprano la sucesión se repite. Buscamos $i \neq j$ con $x_i = x_j$.

El algoritmo ro de Pollard busca repeticiones con $x_0 = 2$, $x_{i+1} = x_i^2 + 1 \pmod{p}$, donde p es un factor primo de n. Si logramos encontrar $x_i \equiv x_j \pmod{p}$, entonces $p|(x_i - x_j, n)$.

Pero como no concemos los factores primos de n, lo que hacemos es calcular $x_{i+1} \equiv x_i^2 + 1 \pmod{n}$, y calculamos $(x_i - x_{2i}, n)$ en cada paso.

Si $1 < (x_i - x_{2i}, n) < n$, entonces hemos encontrado un factor propio de n y terminamos.

Si $(x_i - x_{2i}, n) = 1$, entonces pasamos a x_{i+1} y x_{2i+2} y repetimos.

Si $(x_i - x_{2i}, n) = n$, entonces terminamos con fracaso. La probabilidad es baja, pero si algoritmo termina con fracaso, podemos tratar de nuevo con una nueva función, por ejemplo $f(x) = x^2 + c$, con $c \neq 0, -2$.

El tiempo esperado es $O(n^{1/4})$ multiplicaciones modulares.

Ejemplo 9.5 Vamos a usar el algoritmo ro de Pollard para encontrar un factor no trivial de $n = 455459$.

Empezamos con $x_0 = 2$; luego tenemos los siguientes valores, donde $x_{i+1} \equiv x_i^2 + 1 \pmod{455459}$, y $d = (x_i - x_{2i}, 455459)$:

i	x_i	x_{2i}	d
1	5	26	1
2	26	2871	1
3	677	179685	1
4	2871	155260	1
5	44380	416250	1
6	179685	43670	1
7	121634	164403	1
8	155260	247944	1
9	44567	68343	743

Por lo tanto, un factor no trivial de 455459 es 743, y el cofactor es $455459/743 = 613$. $\lhd$

Algortimo $p-1$ de Pollard

El algoritmo $p-1$ de Pollard es también un algoritmo de propósito especial de factorización que se usa para encontrar de manera eficiente un factor primo p de un entero compuesto n, con la propiedad de que $p-1$ es homogéneo con respecto a alguna cota B relativamente pequeña.

La idea detrás del algoritmo es la siguiente: sea Q el mínimo común múltiplo de todos los números p^a, donde:

1. p es primo, $a > 0$;
2. $p \leq B$;
3. $p^a \leq n$.

Si $p^a \leq n$, entonces $a \log(p) \leq \log(n)$, y por lo tanto,

$$a \leq \left\lfloor \frac{\log(n)}{\log(p)} \right\rfloor .$$

Es decir, tenemos que

$$Q = \prod_{\substack{q \leq B \\ q \text{ primo}}} q^{\lfloor \log(n)/\log(q) \rfloor}.$$

Sea entonces p un factor primo de n, tal que $p-1$ es B-homogéneo. Entonces $p-1|Q$. Si a es tal que $(a,p)=1$, entonces tenemos que $a^{p-1} \equiv 1 \pmod{p}$, por el Pequeño Teorema de Fermat, y por lo tanto, que $a^Q \equiv 1 \pmod{p}$. Sea entonces $d = (a^Q - 1, n)$. Puesto que p divide a ambos términos, $p|d$ y ésto nos da un factor $d > 1$ de n.

Es posible que $d = n$, y entonces el algoritmo termina con fracaso. Pero eso casi no sucede si n tiene al menos dos factores primos que sean "grandes" comparados con B, de manera que el algoritmo es eficiente si ya descartamos la posibilidad de que n sea B-homogéneo (usando, por ejemplo, división y el algoritmo ro de factorización de Pollard).

El algoritmo, explícitamente, es el siguiente: primero, elegimos una cota B de homogeneidad.

Después escogemos a, $2 \leq a \leq n-1$, de manera arbitraria, y calculamos $d = (a, n)$.

Si $d \geq 2$, entonces d es un factor propio de n, y hemos terminado. Si $d = 1$, entonces para cada primo $q \leq B$, calculamos $\ell_q = \lfloor \log(n)/\log(q) \rfloor$, y luego tomamos

$$\begin{aligned} Q &= \prod_{\substack{q \leq B \\ q \text{ primo}}} q^{\ell_q} \\ \alpha &= a^Q. \end{aligned}$$

Finalmente, calculamos $d = (\alpha - 1, n)$. Si $d = 1$ ó $d = n$, terminamos el algoritmo con fracaso. Si $1 < d < n$, entonces d es un factor propio de n.

Si n tiene un factor primo p tal que $p-1$ es B-homogéneo, entonces el algoritmo $p-1$ de Pollard tiene un tiempo esperado de $O(B \log(n)/\log(B))$ multiplicaciones modulares para encontrar el factor p. En la práctica, se suele elegir B con $10^5 \leq B \leq 10^6$.

Si el algoritmo termina en fracaso con $d = 1$, entonces podemos buscar primos $q_1, \ldots, q_r$ mayores que B, tomando α y elevando a la q_i para cada i sucesivamente, y al final calcular nuevamente el máximo común divisor del resultado menos uno, y n.

Ejemplo 9.6 Usamos el algoritmo $p-1$ de Pollard para encontrar un factor propio de $n = 19048567$.

Elegimos como cota de homogeneidad $B = 19$, y tomamos el entero $a = 3$; verificamos que $(3, n) = 1$.

Los valores de q y ℓ_q están dados en la siguiente tabla:

q	ℓ_q
2	24
3	15
5	10
7	8
11	6
13	6
17	5
19	5

Entonces $a^Q = 554506 \bmod 19048567$, y al calcular el máximo común divisor, obtenemos $(554506 - 1, 19048567) = 5281$. En efecto, $p = 5281$ es factor de 19048567, con cofactor $q = 3607$.

Notemos que $p - 1 = 5280 = 2^5 \times 3 \times 5 \times 11$, es decir, $p - 1$ es 19-homogéneo; por otro lado, $q - 1 = 3606 = 2 \times 3 \times 601$, de manera que $q - 1$ no es 19-homogéneo. $\lhd$

9.5.2 Factorización con curvas elípticas

La factorización por curvas elípticas fue inventada por Hendrik W. Lenstra, Jr., basándose en el algoritmo $p - 1$ de Pollard. Cómo funciona explícitamente es algo complicado, y por ende no se considera dentro de éste capítulo. El lector interesado puede encontrar más información sobre curvas elípticas y sobre este algoritmo en el Apéndice B.

Si pensamos en el algoritmo $p - 1$ de Pollard, podemos ver que el $p - 1$ es "en realidad" el orden del grupo $\mathbb{Z}_p^*$, donde p es un factor primo de n. El algoritmo falla cuando este grupo no tiene elementos de orden "adecuado," a saber B-homogéneo.

La idea de Lenstra fue reemplazar el grupo $\mathbb{Z}_p^*$ con el grupo de puntos de una curva elíptica sobre el campo $\mathbb{Z}_p$. Como hay muchas curvas, y el orden de estos grupos está distribuido uniformemente en el intervalo $[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$, hay más opciones con mayor probabilidad de que el orden del grupo sea B-homogéneo.

Si encontramos una curva tal que el grupo asociado tiene orden sea B-homogéneo, entonces el algoritmo tiene alta probabilidad de encontrar el factor p; si la curva que elegimos tiene grupo cuyo orden no es B-homogéneo, el algoritmo falla. En el caso del algorimo $p-1$ de Pollard, si el grupo no tiene orden B-homogéneo no hay nada que hacer (fuera de agrandar B); pero en el caso de la factorización por curvas elípticas, simplemente podemos

elegir otra curva elíptica y volver a tratar de aplicar el algoritmo.

Si estamos buscando un factor p de n, entonces el tiempo esperado para el algoritmo de factorización por curvas elípticas es aproximadamente

$$O\Big(\exp\Big(\sqrt{2}\sqrt{\log(p)}\sqrt{\log\log(p)}\Big)\Big)$$

operaciones con curvas elípticas. El principal problema del algoritmo consisten en codificar estas operaciones, pues a diferencia de operaciones modulares, no suele haber funciones ya hechas que las lleven a cabo, y "sumar" dos puntos del grupo de puntos de una curva elíptica tiende a ser más tardado que sumar dos enteros módulo n.

El algoritmo de factorización con curvas elípticas tiende a encontrar los factores primos chicos primero, y es el mejor algoritmo (en la práctica) para buscar factores primos con aproximadamente 50 dígitos decimales.

Cuando n es el producto de dos primos de más o menos el mismo tamaño (es decir, cuando estamos tratando el caso de factorizar un módulo de RSA), el algoritmo se encuentra en su peor caso posible, y el tiempo esperado es

$$O\Big(\exp\Big(\sqrt{\log(n)}\sqrt{\log\log(n)}\Big)\Big)$$

operaciones de curvas elípticas.

Puesto que la implementación del algoritmo es menos eficiente que el de otros algoritmos con la misma complejidad, este algoritmo no suele usarse para atacar módulos de RSA.

Sin embargo, el algoritmo tiene una gran importancia tanto teórica como psicológica: el algoritmo fue extremadamente novedoso en utilizar una estructura (las curvas elípticas) que no se sospechaba que tuviera que ver con el problema de factorización. También pone a disposición de el criptoanalista una rica teoría matemática asociada a las curvas elípticas. Desde el punto de vista psicológico, el algoritmo es un fuerte recordatorio que nuevas técnicas pueden aparecer desde direcciones inesperadas.

9.5.3 Métodos de Fermat, Kraitchik, y Fracciones Continuas

La idea detrás de los tres algoritmos más avanzados para factorización que se conocen (la Criba Cuadrática, la Criba Especial de Campos Numéricos, y la Criba General de Campos Numéricos) data de Fermat. En esta sección vamos a explicar cuál es el método de Fermat, y las mejoras que se hicieron al método antes de la creación de las cribas.

Nuestra presentación está basada en el artículo de Carl Pomerance [Pom96].

Método de diferencia de cuadrados de Fermat

Supongamos que queremos factorizar $n = 8051$. Podemos simplemente probar algunos primos chicos, 2, 3, ..., 17, pero encontramos que ninguno lo divide. Podemos continuar hasta $\sqrt{n}$, pero en vez de eso podemos hacer la misma observación que en su momento hizo Fermat: notamos que

$$\begin{aligned} 8051 &= 8100 - 49 \\ &= 90^2 - 7^2 \\ &= (90+7)(90-7) \\ &= 97 \times 83, \end{aligned}$$

y después podemos verificar que tanto 97 como 83 son números primos, y hemos terminado.

Es decir, si podemos expresar n como una diferencia de cuadrados, $n = a^2 - b^2$, entonces $n = (a+b)(a-b)$. Siempre y cuando $a - b \neq 1$, eso nos dá una descomposición no trivial de n (asumimos que tanto a como b son enteros positivos, de manera que $a + b$ nunca vale 1).

En general, si $n = ab$, con $1 < b < a < n$, entonces podemos escribir

$$ab = \left(\frac{1}{2}(a+b)\right)^2 - \left(\frac{1}{2}(a-b)\right)^2$$

(recuerde que asumimos que n es impar, de manera que a y b son también impares, y por lo tanto $a+b$ y $a-b$ son ambos pares). Eso nos da una factorización con factores propios como diferencia de cuadrados. Es decir: si podemos expresar n como una diferencia de cuadrados, $n = a^2 - b^2$, con $a - b > 1$, entonces de esta diferencia podemos encontrar factores propios de n; conversamente, si tenemos factores propios de n, entonces es posible utilizarlos para encontrar una expresión de n como diferencia de cuadrados.

El método de Fermat es entonces buscar cuadrados u y v tales que $n = u^2 - v^2$; para ello, tomamos $u = \lceil \sqrt{n} \rceil$, y vemos is $u^2 - n$ es un cuadrado; si lo es, ya terminamos; si no lo es, sumamos 1 a u y volvemos a tratar. Continuamos hasta encontrar la diferencia de cuadrados que nos den una descomposición no trivial de n (recuerde que también estamos asumiendo que ya sabemos que n no es un primo, ni una potencia perfecta).

Método de Kraitchik

En los años veinte, Maurice Kraitchik propuso una mejora técnica interesante al método de diferencias de cuadrados de Fermat. Primero, en vez de buscar u y v tales que $n = u^2 - v^2$, buscamos u y v tales que $u^2 - v^2 = kn$, es decir, tales que $u^2 \equiv v^2 \pmod{n}$.

La congruencia tiene soluciones triviales si $u \equiv \pm v \pmod{n}$, y soluciones no triviales cuando $u \not\equiv \pm v \pmod{n}$. Si n es impar, divisible entre al menos dos primos distintos, entonces se puede demostrar que al menos la mitad de las soluciones a la congruencia

$$u^2 \equiv v^2 \pmod{n}$$

con $(uv, n) = 1$ son no triviales. Si encontramos una tal que $u \not\equiv \pm v \pmod{n}$, entonces tenemos que $1 < (u - v, n) < n$; pues $n|(u + v)(u - v)$, pero $n \not| u + v$, $n \not| u - v$. Esto nos daría entonces un factor propio de n.

¿Cómo utilizamos la idea de Kraitchik para factorizar $n = 2041$?

Buscamos el primer cuadrado mayor que n; puesto que $\lceil \sqrt{n} \rceil = 46$, el primero es $46^2 = 2116$. Sea $Q(x) = x^2 - n$. Entonces evaluamos el polinomio $Q(x)$ en $47, 48, \ldots$, y obtenemos la sucesión

$$75, 168, 263, 360, 459, 560, \ldots$$

Fermat seguiría probando hasta encontrar un cuadrado, pero aquí entra la segunda parte de la idea de Kraitchik. Buscamos varios valores de x tales que el producto de los valores correspondientes de $Q(x)$ sean un cuadrado. Pues si tuvieramos que

$$Q(x_1)Q(x_2)\cdots Q(x_k) = v^2$$

para algún entero v, y si definimos $u = x_1 \cdots x_k$, entonces

$$\begin{aligned} u^2 &\equiv (x_1 \cdots x_k)^2 \pmod{n} \\ &\equiv x_1^2 \cdots x_k^2 \pmod{n} \\ &\equiv (x_1^2 - n)(x_2^2 - n)\cdots(x_k^2 - n) \pmod{n} \\ &\equiv Q(x_1)\cdots Q(x_k) \pmod{n} \\ &\equiv v^2 \pmod{n}. \end{aligned}$$

Si tenemos suerte, vamos a tener que $u \not\equiv \pm v \pmod{n}$ (esperamos que esto sea cierto en al menos la mitad de los casos), y con ello obtenemos un factor propio de n. En realidad, hay también que verificar que $(uv, n) = 1$, pero si $(u, n) > 1$ ó $(v, n) > 1$, entonces eso nos da un factor propio de n y ya terminamos.

La pregunta es cómo encontramos los valores $x_1, \ldots, x_k$ con esta propiedad. Kratichik notó que algunos de los valores de $Q(x)$ se factorizan muy facilmente:

$$\begin{aligned} Q(46) = 75 &= 3 \times 5^2 \\ Q(47) = 168 &= 2^3 \times 3 \times 7 \\ Q(49) = 360 &= 2^3 \times 3^2 \times 5 \\ Q(51) = 560 &= 2^4 \times 5 \times 7 \end{aligned}$$

y su producto es entonces igual a $2^{10} \times 3^4 \times 5^4 \times 7^2$, que es un cuadrado. Entonces tomamos

$$\begin{array}{rcll} u & = & 46 \times 47 \times 49 \times 51 \mod 2041 & = & 311 \\ v & = & 2^5 \times 3^2 \times 5^2 \times 7 \mod 2041 & = & 1416 \end{array}$$

y tenemos que $u^2 \equiv v^2 \pmod{n}$. Verificamos fácilmente que $311 \not\equiv 1416 \pmod{2041}$, de manera que

$$(1416 - 311, 2041) = 13$$

es un factor propio de 2041; en efecto, $2041 = 13 \times 157$.

La idea central es entonces usar la sucesión $x^2 - n$, empezando en $\lceil \sqrt{n} \rceil$, y encontrar una subsucesión cuyo producto sea un cuadrado. Si tenemos suerte (y esperamos tener suerte en al menos la mitad de las veces), eso nos proporciona una descomposición propia de n.

Fracciones Continuas

En 1931, D.H. Lehmer y R.E. Powers sugirieron reemplazar la sucesión $x^2 - n$ de Kraitchik con la sucesión $Q_1, Q_2, \ldots$, asociada con la expansión por fracciones continuas de $\sqrt{n}$, y utilizar dicha sucesión de manera análoga a la de Kraitchik para encontrar una descomposición no trivial de n. La expansión por fracciones continuas es simplemente un método de generar dos sucesions, $a_1, a_2, \ldots$, y $b_1, b_2, \ldots$ tales que $(a_i, b_i) = 1$ para cada i, y que la sucesión de fracciones a_i/b_i converge "rápidamente" a $\sqrt{n}$. Entonces, Lehmer y Powers sugirieron como el i-ésimo término $Q_i = a_i^2 - b_i^2 n$. Tenían entonces que $Q_i \equiv a_i^2 \pmod{n}$, y se puede proceder como antes.

Las fracciones continuas han sido objeto de mucho estudio por matemáticos a lo largo de los años. Aquí sólo vamos a presentar la definición de las fracciones y mencionaremos sus propiedades básicas.

Dado $x \in \mathbb{R}$, su *expansión por fracciones continuas* se define de la siguiente manera: definimos dos sucesiones de manera iterativa:

$$\begin{array}{rclcrcl} a_0 & = & \lfloor x \rfloor; & & x_0 & = & x - a_0; \\ a_1 & = & \lfloor 1/x_0 \rfloor; & & x_1 & = & (1/x_0) - a_1; \\ & \vdots & & & & \vdots & \\ a_i & = & \lfloor 1/x_{i-1} \rfloor; & & x_i & = & (1/x_{i-1}) - a_i; \\ & \vdots & & & & \vdots & \end{array}$$

Si para alguna i, $1/x_{i-1}$ es un entero, entonces $x_i = 0$ y truncamos las dos sucesiones.

La i-ésima aproximación a x por fracciones continuas está dada por:

$$\frac{b_i}{c_i} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots + \cfrac{1}{a_i + x_i}}}$$

(es fácil ver que el resultado de la derecha es un racional, al que expresamos en mínimos términos como b_i/c_i). Para faciliar la tipografía, esto se escribe normalmente como:

$$\frac{b_i}{c_i} = a_0 + \frac{1}{a_1+}\,\frac{1}{a_2+}\,\frac{1}{a_3+}\,\cdots\,\frac{1}{a_i + x_i}.$$

Se puede probar que x es racional si y sólo si la sucesión de fracciones continuas termina; es decir, si $x_i = 0$ para alguna $i \geq 0$. Si x es irracional, las fracciones continuas nos dan una sucesión de racionales b_i/c_i, con la propiedad de que $\lim_{i\to\infty} b_i/c_i = x$. Además, las fracciones parciales convergen "rápidamente" a x en el sentido del inciso (ii) de la siguiente proposición:

Proposición 9.1 *Sea $x \in \mathbb{R}$ un real irracional, y supongamos que a_i, x_i, y b_i/c_i están definidas como en los párrafos anteriores. Entonces:*

(i) $\frac{b_0}{c_0} = \frac{a_0}{1}$; $\frac{b_1}{c_1} = \frac{a_0 a_1 + 1}{a_1}$.

(ii) Para toda $i \geq 2$,

$$\frac{b_i}{c_i} = \frac{a_i b_{i-1} + b_{i-2}}{a_i c_{i-1} + c_{i-2}},$$

y la expresión de la derecha ya está escrita en mínimos términos.

(iii) Para toda i, $|b_i^2 - x^2 c_i^2| < 2x$.

Si aplicamos el inciso (iii) al caso que nos interesa, $x = \sqrt{n}$, y donde $Q_i = b_i^2 - n c_i^2$, eso nos dice que $|Q_i| < 2\sqrt{n}$, lo cual dice que los términos de la sucesión $Q_1, Q_2, \ldots$ tienden a ser más pequeños que los de la sucesión de Kraitchik, lo cual facilita su uso.

Notamos que $Q_i \equiv b_i^2 \pmod{n}$, de manera que nuevamente, si encontramos una subsucesión $Q_{i_1}, \ldots, Q_{i_k}$ tal que

$$Q_{i_1} \times \cdots \times Q_{i_k} = v^2$$

con v un entero positivo, entonces $u = b_{i_1} \cdots b_{i_k}$ satisface $u^2 \equiv v^2 \pmod{n}$, y procedemos como antes.

Este método se conoce como el *método de fracciones continuas*, y durante muchos años fue el mejor método para factorizar números grandes.

Cómo encontrar subsucesiones

El principal cuello de botella de los métodos de Kraitchik y de fracciones continuas es encontrar una subsucesión de nuestros términos (ya sea $Q(x_1), Q(x_2), \ldots$ en el método de Kraitchik, o la sucesión $Q_1, Q_2, \ldots$ asociada al método de fracciones continuas) cuyo producto sea un cuadrado.

En 1975, John Brillhart y Michael Morrison propusieron una estrategia sistemática para buscar una subsucesión cuyo producto sea un cuadrado; y se trata simplemente de aplicar álgebra lineal.

A cada entero positivo n le podemos asociar un "vector de exponentes" $v(n)$; es un vector con un número infinito de coordenadas, pero casi nulo (es decir, todas las entradas son cero, excepto quizás por un número finito). Si escribimos

$$m = \prod p_i^{\nu_i}$$

donde p_i es el i-ésimo primo, y $\nu_i \geq 0$, entonces

$$v(m) = (\nu_1, \nu_2, \ldots, \nu_n, \ldots).$$

Por ejemplo,

$$\begin{aligned} v(75) &= (0,1,2,0,0,\ldots) \\ v(168) &= (3,1,0,1,0,\ldots) \\ v(360) &= (3,2,1,0,0,\ldots) \\ v(560) &= (4,0,1,1,0,\ldots). \end{aligned}$$

La multiplicación de números se convierte en suma de vectores; es decir, si

$$\begin{aligned} v(a) &= (\nu_1(a), \nu_2(a), \ldots), \\ v(b) &= (\nu_1(b), \nu_2(b), \ldots), \end{aligned}$$

entonces

$$v(a \times b) = (\nu_1(a) + \nu_1(b), \nu_2(a) + \nu_2(b), \ldots, \nu_n(a) + \nu_n(b), \ldots).$$

Nos interesa poder identificar un número que es un cuadrado. Un entero positivo m es un cuadrado si y sólo si cada entrada de $v(m)$ es par. Puesto que sólo nos interesa saber si cada entrada es par o no, entonces en vez de manejar el vector con entradas en los enteros

no negativos, podemos tomar los vectores con entradas módulo 2; es decir, en vez de los valores de arriba, usamos:

$$\begin{aligned} v(75) &= (0,1,0,0,0,\ldots) \\ v(168) &= (1,1,0,1,0,\ldots) \\ v(360) &= (1,0,1,0,0,\ldots) \\ v(560) &= (0,0,1,1,0,\ldots). \end{aligned}$$

Ahora notamos que la suma de estos vectores es el vector cero; es decir, el producto de 75, 168, 360, y 560 es un cuadrado, como ya lo habíamos notado.

La ventaja de trabajar con entradas módulo 2 es que entonces podemos pensar que estamos en un espacio vectorial sobre el campo de dos elementos, lo cual nos permite utilizar todas las herramientas de álgebra lineal para atacar el problema.

Pero aún así, trabajar con vectores infinitos (aunque sean casi nulos) es problemático. Brillhart y Morrison sugirieron entonces que en vez de trabajar con la factorización total de los números, trabajaramos sólo con las primeras $\pi(B)$ coordenadas (donde $\pi(m)$ es el número de primos menores o iguales a m), y que nos concentraramos en aquellos números que son B-homogéneos, con B relativamente chico. Esto también nos permite factorizar fácilmente los números, lo cual es necesario para obtener los vectores.

Supongamos entonces que nos concentramos en números B-homogéneos, y que tenemos $\pi(B)+1$ tales números. Si pensamos en los vectores de estos números, cada vector tiene $\pi(B)$ coordenadas. Entonces estamos en un espacio vectorial de dimensión $\pi(B)$ sobre $\mathbb{F}_2$; si tenemos $\pi(B)+1$ vectores, entonces sabemos por álgebra lineal que el conjunto es linealmente dependiente. Es decir, hay una combinación lineal no trivial que suma cero; pero los únicos coeficientes en $\mathbb{F}_2$ son 0 y 1; es decir, hay un subconjunto de estos vectores cuya suma es el vector cero. Por lo tanto, hay un subconjunto de estos números B-homogéneos cuyo producto es necesariamente un cuadrado. Y encontrar la combinación lineal que suma cero es un fácil ejercicio de álgebra lineal.

Hay un refinamiento más que se puede hacer: Brillhard y Morrison notaron que sólo aquellos primos p para los cuáles n es un cuadrado módulo p son importantes, de manera que el resto de los primos puede ser deshechado. Pero entonces hay que agregar una entrada al vector que tenga en cuenta el "signo" del elemento, pues vamos a trabajar módulo n; es decir, las entradas de los vectores corresponden a un conjunto S que contiene al -1, y a todos los primos p, $p \leq B$, tales que n es un cuadrado módulo p. Recuerde que el teorema de reciprocidad cuadrática y el símbolo de Legendre nos permiten calcular de manera muy sencilla si n es un cuadrado módulo p o no (véase el Apéndice E).

Tanto en el método de Kraitchik como en el de fracciones continuas, los primos p tales

que n no es un cuadrado módulo p nunca dividen a las Q_i. Por ejemplo, en el método de Kraitchik tenemos $Q_i = r^2 - n$ para un entero r. Entonces $Q_i \equiv r^2 \pmod{n}$; si p divide a Q_i, entonces p divide a $r^2 - n$, y por lo tanto $r^2 \equiv n \pmod{p}$; es decir, n es un cuadrado módulo p.

Esto nos dice que podemos descartar aquellos primos $p \leq B$ tales que n no es un cuadrado módulo p sin ningún problema.

Entonces el proceso se vuelve el siguiente: vamos calculando la sucesión $Q_1, Q_2, \ldots$. En cada paso, vemos si podemos expresar Q_i como un producto de elementos de S (es decir, si es B-homogéneo). En caso afirmativo, guardamos el número y el vector de exponentes correspondiente. En caso negativo, descartamos Q_i y continuamos. Una vez que tengamos $|S| + 1$ vectores de exponentes, buscamos una combinación lineal no trivial que sume cero, y con eso obtenemos u y v tales que $u^2 \equiv v^2 \pmod{n}$, como antes.

El único posible problema es si al terminar nos encontramos con que $u \equiv \pm v \pmod{n}$; es decir, si lo que encontramos es una solución trivial. En ese caso, tenemos que buscar otra combinación lineal no trivial que sume cero, o bien agregar nuevos vectores para buscar otra combinación lineal.

El tiempo esperado para encontrar una solución no trivial a $u^2 \equiv v^2 \pmod{n}$ es aproximadamente de

$$O\Big(\exp\Big(\sqrt{2}\sqrt{\log(n)}\sqrt{\log\log(n)}\Big)\Big)$$

pasos para terminar.

9.5.4 La Criba Cuadrática

La Criba Cuadrática es un método de factorización de propósito general, cuyas ideas básicas están centradas en los algoritmos de Kraitchik y Lehmer-Powers, modificados como lo sugiere Brillhard-Morrison.

El punto donde podemos buscar optimizar los algoritmos de Lehmer-Powers de fracciones continuas y el de Kraitchik está en el momento en que, después de generar los valores $Q_1, Q_2, \ldots$, probamos a ver si son B-homogéneos (en realidad, probamos a ver si se pueden expresar como un producto de elementos de la base de factores S:

$$S = \{-1\} \cup \Big\{p \leq B \;\Big|\; p \text{ es primo, y } \Big(\tfrac{n}{p}\Big) = 1\Big\},$$

y expresarlos de esa manera para calcular el vetor de exponentes). La velocidad de ambos métodos depende íntimamente de dos cosas:

1. La velocidad con la que podemos reconocer que un número es S-factorizable, y la velocidad con la que podemos encontrar su factorización en caso afirmativo; y
2. La distribución de los valores S-factorizables entre los valores de las Q_i.

Con la distribución de los valores S-factorizables no hay mucho que hacer. Se **conjetura** (y esta conjetura está relacionada con la Hipótesis de Riemann) que los valores S-factorizables están distribuídos de manera uniforme, pero no sabemos mucho a ciencia cierta. Donde sí podemos tratar de mejorar el tiempo es en el reconocimiento de los números S-factorizables, y en la velocidad con la que logramos encontrar dicha expresión.

En 1981, Carl Pomerance sugirió una manera mucho mejor de reconocerlos, para el caso del método de Kraitchik. La idea está basada en la criba de Erastóstenes.

En el caso de la criba de Erastóstenes, escribimos todos los números entre 2 y N inclusive; marcamos el 2, y "tachamos" todos los números que sean múltiplos de 2. Luego buscamos el siguiente número que no esté tachado (el 3); lo marcamos, y "tachamos" todos los múltiplos de tres. Al terminar, buscamos el siguiente número que no esté tachado, y continuamos de esta manera. Cuando lleguemos a $\sqrt{N}$, todos los números que estén marcados o no estén tachados son primos.

La criba de Erastóstenes no sólo encuentra primos: también nos dice cuantos factores primos distintos tiene cada número. Si contamos cuantos taches distintos tienen los números tachados, ese es el número de factores primos distintos que tiene.

La idea de Pomerance fue que en vez de ir probando cada número Q_i para ver si es S-factorizable, podemos cribar los números hasta n usando los primos de S (puesto que $-1 \in S$, en realidad cribamos los números de $-n$ a n). Primero, armamos una tabla con $2N+1$ entradas, numeradas de $-n$ a n. Luego, en cada paso, tomamos un primo $p \in S$, y a cada múltiplo i de p (son fáciles de encontrar en un arreglo), dividimos entre la máxima potencia de p que divide a i; y reemplazamos i por el resultado en el arreglo. Al terminar de cribar con todos los primos p de S, aquellas entradas que tienen un 1 son las que corresponden a números S-factorizables. Entonces, cuando calculamos Q_j, simplemente tomamos $Q_j \bmod n$, y buscamos en la tabla a ver si $Q_j \bmod n$ es S-factorizable; si la entrada es 1, sí lo es; si la entrada no es 1, entonces podemos deshechar Q_j y pasar al siguiente número en la sucesión, a saber Q_{j+1}.

La ventaja de cribar es que los números divisibles entre p ocurren en lugares predecibles de nuestra tabla original, de manera que el cribado se puede hacer muy rápidamente.

En la práctica, lo que hacemos es empezar con un arreglo $A[\ ]$, indexado por x con $-M \leq x \leq M$, donde M es alguna cota elegida con anterioridad. La inicializamos a $\lfloor \log |Q(x + \lfloor \sqrt{n} \rfloor)| \rfloor$, donde $Q(y) = y^2 - n$. Notemos que si $p|Q(y)$, entonces $p|Q(y + p\ell)$

para todo entero ℓ, pues Q es un polinomio. De manera que es fácil encontrar las entradas que son múltiplos de p.

Dado p, resolvemos la congruencia $(x + \lfloor\sqrt{n}\rfloor)^2 - n \equiv 0 \pmod{p}$. Esta congruencia tiene dos soluciones módulo p, x_1 y x_2. Entonces, le restamos $\lfloor\log(p)\rfloor$ a todas las entradas $A[x]$ donde $x \equiv x_1 \pmod{p}$ ó $x \equiv x_2 \pmod{p}$, $-M \leq x \leq M$. Esto lo hacemos para cada primo impar p; (el caso de $p = 2$ y de potencias de p se trata de manera similar).

Al terminar este proceso, que es el proceso de cribado, aquellas x para las cuáles $A[x]$ es cercana a cero son las que con mayor probabilidad son S-factorizables. Puede ser que tengamos candidatos de más debido a la naturaleza aproximada del proceso utilizado, pero tiene la ventaja de que el cribado descrito es *extremadamente* rápido, y sólo se tiene que hacer una vez.

Un cálculo de complejidad, asumiendo que los valores S-factorizables están distribuídos de manera uniforme, nos dice que el orden de complejidad de la Criba Cuadrática es de

$$O\Big(\exp\Big(\sqrt{\log(n)}\sqrt{\log\log(n)}\Big)\Big).$$

Es decir, comparado con el método de fracciones continuas, hemos cambiado al constante $\sqrt{2}$ por una constante de 1. Parece ser una mejora mínima, pero en la práctica representa una mejora muy fuerte: el número de dígitos de enteros para los cuales la Criba Cuadrática es un método realísticamente eficiente para factorizar es aproximadamente el doble del número de dígitos de los enteros para los cuales el método de fracciones continuas es razonable.

Otra gran ventaja de la criba cuadrática es que es muy amena a la paralelización del proceso, lo cual también aumenta el tamaño de los enteros que podemos factorizar, a base de dividir labores entre varios procesadores. Esto se logra ya sea utilizando varios polinomios distintos en vez de uno sólo para generar los valores Q_i, o repartiendo los intervalos de valores a probar entre los distintos procesadores; o una combinación de ellos.

9.5.5 La Criba Especial de Campos Numéricos

La Criba Especial de Campos Numéricos es un algoritmo de factorización de propósito especial, para factorizar enteros que se pueden expresar de la forma $a^b \pm 1$, con a chico ($a < 100$), y b relativamente grande.

En 1986, Don Coppersmith, Andrew Odlyzko, y Richard Schroeppel publicaron un algoritmo para calcular logaritmos discretos que utiliza campos numéricos cuadráticos.

Un *campo numérico* es una extensión algebraica finita de $\mathbb{Q}$. Son el principal objeto de estudio de la Teoría de Numéros Algebráica; los campos numéricos cuadráticos son de la

forma $\mathbb{Q}(\sqrt{d})$, donde d es un entero libre de cuadrados.

Inspirado por éste algoritmo, John Pollard circuló en 1988 una carta a varias personas donde esbozaba una idea para factorizar ciertos numéros grandes de la forma $a^b \pm 1$, usando campos numéricos. Pollard ilustró su idea factorizando el séptimo número de Fermat:

$$F_7 = 2^{2^7} = 59,649,589,127,497,217 \times 5,704,689,200,685,129,054,721.$$

El séptimo número de Fermat había sido el primer gran éxito del método de fracciones continuas en 1970.

Mucha gente no tomó en serio la sugerencia de Pollard; la idea tenía varias dificultades técnicas, y aún suponiendo que pudiesen ser resueltas, el algoritmo sólo pretendía servir para números de forma muy especial.

Pero una cuantas gentes sí tomaron en serio a Pollard, y en particular, entre la gente que le prestó atención se encontraban Hendrik W. Lenstra, Jr., su hermano Arjen Lenstra, y Mark Manasse.

Hendrik Lenstra resolvió varias de las dificultades técnicas y teóricas del método, y Arjen Lenstra y Manasse varias de las dificultades de programación. Brian La Macchia y Andrew Odlyzko desarrollaron algoritmos eficientes para tratar con matrices grandes poco densas, que aparecen en el método. Con estas herramientas en mano, los Lenstra y Manasse decidieron probar la idea de Pollard con un verdadero premio, $F_9 = 2^{2^9} + 1$, el noveno primo de Fermat. Se sabía que tenía un factor primo de 7 dígitos, y el cofactor de 148 dígitos se sabía que no era primo, pero no se conocían ningún factor no trivial de éste. En la primavera de 1990, anunciaron el éxito del proyecto. La factorización de F_9 está dada por:

$$\begin{aligned} F_9 &= 2^{2^9} + 1 \\ &= 2,424,833 \\ &\quad \times 208,337,395,836,200,454,918,783,366,342,657 \\ &\quad \times 741,640,062,627,530,801,524,787,141,901,937,474,059,940,781,897,519, \\ &\quad\ \ 023,905,821,316,144,415,759,504,705,008,092,818,711,693,940,737. \end{aligned}$$

Fue un anuncio sensacional, sobre todo porque el trabajo se hizo con mucha paralelización, para la cual el algoritmo es especialmente apto.

¿Y qué pasa con números generales, o con las dificultades técnicas? Si las ignoramos, o asumimos que hay una manera eficiente de darle la vuelta a los problemas, el algoritmo tiene un tiempo esperado sorprendente:

$$O\Big(\exp\Big(c\sqrt[3]{\log(n)}\sqrt[3]{(\log\log(n))^2}\Big)\Big)$$

operaciones, donde la c es una constante (hablaremos un poco más sobre la c más abajo). Era la primera vez que se lograba un método cuyo tiempo esperado tuviera el exponente de $\log(n)$ por abajo de $\frac{1}{2}$; y si bajar la constante en la criba cuadrática había representado un aumento tan dramático con respecto al método de fracciones continuas, ¿qué significaría bajar el *exponente*? Muchísimo.

¿Cuál era la idea de Pollard, y cómo funciona la Criba Especial de Campos Numéricos?

Primero, obtenemos un polinomio mónico con coeficientes enteros $f(x)$ que sea irreducible, y un entero m tal que $f(x) \equiv 0 \pmod{n}$. El polinomio debe tener grado d "moderado"; si $10^{100} \leq n \leq 10^{200}$, normalmente se toma $d = 5$ ó $d = 6$.

Encontrarlos no es dificil: una vez elegido el grado d, tomamos $m = \lfloor n^{1/d} \rfloor$, y escribimos n en "base m":

$$n = m^d + c_{d-1}m^{d-1} + \cdots + c_0; \qquad 0 \leq c_i < m.$$

Tomamos $f(x) = x^d + c_{d-1}x^{d-1} + \cdots + c_0$. Esto nos da un polinomio mónico con coeficientes enteros tales que $f(m) \equiv 0 \pmod{n}$. Si $f(x)$ es irreducible, entonces no hay problema. Pero si tenemos la "mala suerte" de que $f(x)$ sea reducible, entonces tenemos dos polinomios mónicos con coeficientes enteros, $g(x)$, $h(x)$, tales que $f(x) = g(x)h(x)$. Entonces $n = f(m) = g(m)h(m)$, y un resultado de John Brillhart, Michael Filaset, y Andrew Odlyzko nos dice que esta factorización de n es no trivial, y entonces hemos terminado nuestra búsqueda por una descomposición no trivial de n. Así que podemos suponer que $f(x)$ es irreducible.

Sea α una raíz compleja de $f(x)$, y consideramos $\mathbb{Z}[\alpha]$, los polinomios en α con coeficientes enteros. Como $f(\alpha) = 0$ y $f(m) \equiv 0 \pmod{n}$, es en vez de α ponemos m y reducimos módulo n, obtenemos un mapa

$$\phi\colon \mathbb{Z}[\alpha] \to \mathbb{Z}_n$$

que es, de hecho, un homomorfismo de anillos.

Supongamos ahora que S es un conjunto finito de pares de enteros (a, b), que satisfacen las siguientes condiciones:

1. Para cada $(a, b) \in S$, a y b son primos relativos.

2. El producto de los enteros algebráicos[1] $a - \alpha b$, con $(a, b) \in S$ es un cuadrado en $\mathbb{Z}[\alpha]$.

[1] Un entero algebráico es la raíz de un polinomio mónico con coeficientes enteros; los enteros algebráicos forman un subanillo de los números algebráicos, y como α es un entero algebráico, cada elemento de $\mathbb{Z}[\alpha]$ es entero algebráico.

Es decir:

$$\prod_{(a,b)\in S} (a - \alpha b) = \gamma^2$$

para algún $\gamma \in S$.

3. El producto de todos los enteros $a - mb$ con $(a, b) \in S$ es un cuadrado en $\mathbb{Z}$; es decir,

$$\prod_{(a,b)\in S} (a - mb) = v^2$$

con $v \in \mathbb{Z}$.

Entonces γ se puede escribir como un elemento de $\mathbb{Z}[\alpha]$, y aplicamos el morfismo ϕ, y obtenemos:

$$\begin{aligned} u^2 &= \phi(\gamma)^2 \\ &= \phi(\gamma^2) \\ &= \phi\left(\prod_{(a,b)\in S} (a - \alpha b)\right) \\ &= \prod_{(a,b)\in S} \phi(a - \alpha b) \\ &= \prod_{(a,b)\in S} (a - mb) \\ &\equiv v^2 \pmod{n}, \end{aligned}$$

y ya sabemos qué hacer cuando tenemos una congruencia $u^2 \equiv v^2 \pmod{n}$.

¿De dónde sacamos S y las parejas (a, b)? La tercer propiedad que deben satisfacer es fácil: usamos vectores de exponentes y una criba, igual que antes, para encontrar valores que nos den un cuadrado. Si empezamos recorriendo las parejas (a, b) con a y b primos relativos, cumplimos la condición uno también. Tenemos dos variables para la criba, de manera que fijamos b, cribamos sobre los valore de a; al terminar, cambiamos el valor de b y repetimos, etc.

La propiedad difícil es la segunda propiedad. La idea de Pollard es que si $\mathbb{Z}[\alpha]$ es la colección de todos los enteros algebráicos en el campo $\mathbb{Q}(\alpha)$ (su *anillo de enteros*), entonces se trata de un objeto muy estudiado en Teoría de Números Algebráica y para el cual sabemos varias cosas. Si además tenemos la suerte de que sea un dominio de factorización única (los

anillos de enteros de un campo numérico no siempre son dominios de factorización única[2]), entonces es muy facil arreglar las cosas para construír S: cribamos en $\mathbb{Z}[\alpha]$ y en $\mathbb{Z}$ al mismo tiempo, y usamos "exponentes de vectores" que manejen los irreducibles de $\mathbb{Z}[\alpha]$ y los primos de $\mathbb{Z}$ al mismo tiempo, y terminamos.

Los números $n = a^b \pm 1$ que sugería Pollard tienen forma especial para aumentar la probabilidad de que estas condiciones especiales se cumplan.

9.5.6 La Criba General de Campos Numéricos

La Criba General de Campos Numéricos es un refinamiento de la Criba Especial que se puede utilizar para buscar factorizaciones de enteros impares arbitrarios, y no sólo aquellos de forma especial como en la Criba Especial. Es, a la fecha, el algoritmo más poderoso con mejor tiempo esperado asintótico para factorización, y se trata de un algoritmo de propósito general. Igual que en el caso de la Criba Especial, el tiempo esperado de la Criba General es de

$$O\Big(\exp\Big(c\sqrt[3]{\log(n)}\sqrt[3]{(\log\log(n))^2}\Big)\Big)$$

operaciones, y la diferencia entre el tiempo de la Criba General y la Criba Especial está en el valor de la constante c.

Como notamos en la sección anterior, la idea de Pollard para la Criba Especial de Campos Numéricos era trabajar en un anillo $\mathbb{Z}[\alpha]$ y "confiar" en que ciertas propiedades buenas se satisficieran ahí. Para números de la forma especial considerada por Pollard, hay buena probabilidad (pero no hay garantía) que se cumplan, pero para enteros arbitrarios no hay razón para suponer que estaremos en la situación ideal, o cerca de ella.

El proceso de la Criba General es idéntico al de la Criba Especial, y nos enfrentamos entonces al problema de construír el conjunto S de parejas que satisface las condiciones listadas. Y como notamos arriba, el único verdadero obstáculo es la segunda condición.

En 1990, Joe Buhler, Hendrik Lenstra, y Carl Pomerance habían resuelto todas las dificultades teóricas, y Len Adleman había simplificado el algoritmo; éste fué publicado en 1993, con una descripción de la Criba General de Campos Numéricos[LL93].

Presentamos ahora un esbozo de lo que hicieron: la norma $N(a - \alpha b)$ sobre $\mathbb{Q}$ es fácil de calcular, y tenemos que $N(a - \alpha b) = b^d f(a/b)$. Esto es una versión homogeneizada de $f(x)$. Decimos que $a - \alpha b$ es *Y-homogéneo* si $N(a - \alpha b)$ es Y-homogéneo en el sentido

[2]En el siglo XIX, Lamé presentó una "prueba" del Último Teorema de Fermat; el error en el argumento es suponer que el anillo de enteros de un campo numérico siempre (o al menos de cierto tipo especial de campo numérico) es un dominio de factorización única.

usual. Como la norma es multiplicativa, si el producto de varios enteros algebráicos de la forma $a - \alpha b$ es el cuadrado de un entero algbráico, entonces el producto de las normas correspondientes también tiene que ser el cuadrado de un entero. Y es fácil encontrar un conjunto de parejas (a, b) tales que el producto de $N(a - \alpha b)$ sea un cuadrado: lo hacemos utilizando una criba para encontrar los valores Y-homogéneos de $N(a-\alpha b)$, y luego usamos álgebra lineal.

Pero el que el producto de los números $N(a - \alpha b)$ sea un cuadrado es necesario, mas no suficiente para que el producto de las $a - \alpha b$ sea un cuadrado en $\mathbb{Z}[\alpha]$. La razón principal para ello es que la norma manda a varios ideales primos de $\mathbb{Z}[\alpha]$ al mismo ideal de $\mathbb{Z}$; por ejemplo, en $\mathbb{Z}[i]$, tanto $2 + i$ como $2 - i$ tienen norma 5. El producto es $(2 + i)(2 - i) = 5$, y 5 tiene norma $25 = 5^2$, pero 5 es libre de cuadrados en $\mathbb{Z}[i]$. Tenemos pues que trabajar un poco más.

Para cada primo p, sea R_p el conjunto de todas las soluciones de $f(x) \equiv 0 \pmod{p}$. Cuando encontramos una pareja (a, b) tal que p divide a $N(a - \alpha b)$, algún ideal primo arriba[3] de p divide a $a - \alpha b$. Podemos decir exactamente cuál, pues a/b es congruente módulo p a algún miembro de R_p, y esto nos permite distinguir los ideales primos que están arriba de p. Entonces podemos usar vectores de exponentes que tengan $\#R_p$ coordenadas para cada primo p, y con ello llevar la cuenta de la factorización de $a-\alpha b$ en ideales primos; nótese que $\#R_p \leq d$, el grado de $f(x)$.

Con esto hemos sobrepasado el principal obtsáculo a encontrar S, pero todavía hay varias obstrucciones. Se supone que estamos trabajando en $\mathbb{Z}[\alpha]$, pero esto puede no ser el anillo de enteros completo de $\mathbb{Q}(\alpha)$. De hecho, puede que ni siquiera sea un Dominio de Dedekind[4], y entonces no podemos hablar de factorización en ideales primos. Y aunque tengamos tal factorización, el párrafo anterior sólo garantiza que el ideal principal generado por el producto de los enteros algebráicos $a - \alpha b$ es el cuadrado de un ideal, pero no necesariamente el cuadrado de un ideal principal, de manera que no tenemos todavía la condición dos para S. Y aunque sea el cuadrado de un ideal principal, puede que no sea un cuadrado (por ejemplo, el ideal generado por -9 en $\mathbb{Z}$ es el cuadrado de un ideal principal en $\mathbb{Z}$, pero -9 no es un cuadrado). Y aunque el producto de las $a - \alpha b$ sea el cuadrado de un entero algebráico, no sabemos que tal entero algebráico esté en $\mathbb{Z}[\alpha]$.

La última obstrucción es la más fácil de resolver, usando $f'(\alpha)^2$ como un factor multiplicador, donde $f'(x)$ es la derivada formal de $f(x)$. Es un teorema que si f es un polinomio mónico irreducible con coeficientes enteros y con una raíz compleja α, y si γ está en el anillo

[3]Es decir, es un ideal primo $\mathfrak{p}$ tal que $\mathfrak{p} \cap \mathbb{Z} = (p)$.

[4]Un dominio de Dedekind es un dominio entero en el cual todo ideal distinto de cero se puede expresar de manera única como un producto de ideales primos. Los anillos de enteros de un campo numérico siempre son dominios de Dedekind.

de enteros de $\mathbb{Q}(\alpha)$, entonces $f'(\alpha)\gamma$ es un elemento de $\mathbb{Z}[\alpha]$. De manera que si γ^2 es un cuadrado en el anillo de enteros de $\mathbb{Q}(\alpha)$, entonces $f'(\alpha)^2\gamma^2$ es un cuadrado en $\mathbb{Z}[\alpha]$.

Pero las otras obstrucciones parecen más dificiles. Sin embargo, Len Adleman tuvo una simple e ingeniosa idea que de un sólo tiro las resuelve en su totalidad. La idea es que aunque tenemos varias obstrucciones complicadas, forman, módulo cuadrados, un espacio vectorial sobre $\mathbb{F}_2$ de dimensión relativamente pequeña. Lo primero que se nos ocurre entonces es ignorar el problema, pero la dimensión de las obstrucciones no es *tan* pequeña. Lo que Adleman sugirió es elegir de manera aleatoria algunos caracteres cuadráticos, y usar sus valores en los números $a - \alpha b$ como coordenadas nuevas de nuestros vectores de exponentes. Elegimos tal caracter al principio, y lo fijamos por el resto del algoritmo. Entonces, cuando estamos usando los exponentes de vectores para encontrar números $a - \alpha b$ cuyo producto sea un cuadrado, en realidad vamos a obtener números que son un cuadrado "módulo el espacio de obstrucciones," y que ademas tienen alta probabilidad de ser un cuadrado de verdad.

Por ejemplo, considere mosel caso del -9 que mencionamos arriba. Si por alguna razón no logramos "ver" el signo, entonces -9 se ve como un excelente candidato para ser un cuadrado, pues sabemos que todo primo p que divide a -9 tiene exponente par en la factorización de -9. Pero si consideramos un caracter cuadrático evaluado en -9, por ejemplo el símbolo de Legendre $\left(\frac{-9}{p}\right)$, que es 1 si -9 es un cuadrado módulo p y -1 si -9 no es un cuadrado módulo p, y lo evaluamos por ejemplo en $p = 7$, entonces rápidamente obtenemos que -9 no es un cuadrado módulo 7, y por lo tanto no puede ser un cuadrado en $\mathbb{Z}$. Si hacemos varias de estas pruebas, la "probabilidad" de que nuestro producto de $a - \alpha b$ sea un cuadrado es muy alta. Si resulta que no lo es, podemos seguir usando algebra lineal sobre $\mathbb{F}_2$ para encontrar nuevos candidatos.

Todavía hay dificultades, sin lugar a dudas. Una de ellas es el problema de la raíz cuadrada: si tenemos la factorización en primos de varios enteros y su producto es un cuadrado, entonces es fácil encontrar la raíz cuadrada usando la factorización en primos. Pero en $\mathbb{Z}[\alpha]$ el problema no es tan transparente. Afortunadamente, hay mecanismos para resolverlo, aunque éste sigue siendo un paso computacionalmente interesante en el algoritmo.

9.5.7 Complejidad de las cribas

Recordemos que tanto la Criba Especial de Campos Numéricos como la Criba General tienen una complejidad esperada de

$$O\Big(\exp\Big(c\sqrt[3]{\log(n)}\sqrt[3]{(\log\log(n))^2}\Big)\Big),$$

pero no hemos dicho cuánto vale la constante c.

Hay tres valores para la constante c, dependiendo de que versión de la criba se use. La Criba Especial tiene

$$c = \left(\frac{32}{9}\right)^{1/3} \approx 1.523.$$

La Criba General, por otro lado, que sirve para cualquier entero positivo impar que no es primo ni una potencia perfecta tiene una constante con valor

$$c = \left(\frac{64}{9}\right)^{1/3} \approx 1.923.$$

Finalmente, Don Coppersmith ha propuesto una versión de la Criba General que utiliza varios polinomios en vez de uno sólo, y para éste método la constante tiene un valor de

$$c = \frac{1}{3}\left(92 + 26\sqrt{13}\right)^{1/3} \approx 1.902.$$

Este es el mejor método asintótico general que se conoce.

9.5.8 La Criba Cuadrática vs. La Criba General de Campos Numéricos

¿En dónde está el punto de cruce entre la Criba Cuadrática y la Criba General de Campos Numéricos? Si bien la Criba General tiene mejor tiempo asintótico, para números más "pequeños" no es necesariamente el mejor método.

La respuesta a la pregunta depende de a quién le pregunte uno. Pero en algunas cosas todo mundo está de acuerdo. Para número "pequeños", digamos con 100 dígitos o menos, la Criba Cuadrática es mucho mejor; para números más grandes, digamos con más de 130 dígitos, la Criba General es mejor (uponiendo siempre que ninguno de los métodos especiales sea aplicable).

La razón por la cual la pregunta no tiene una mejor respuesta es que la respuesta depende de los detalles de la programación y de la computadora que se esté utlizando. Por ejemplo, se ha reportado que la Criba General de Campos Numéricos es muy sensible a cuánta memoria tiene la computadora que se está utilizando; aunque la Criba Cuadrática también depende de éste factor, depende en mucho menor medida.

En general, sin embargo, ambos métodos son peores que la factorización por curvas elípticas, excepto para cierto tipo muy especial de números compuestos, llamados los *números difíciles.* Los módulos de RSA, sin embargo, son todos números difíciles.

Hay ataques a RSA que no dependen directamente de factorizar el módulo, y éstos son el tema de este capítulo.

Puesto que tenemos un algoritmo para factorizar el módulo con complejidad

$$O\Big(\exp\Big(c\sqrt[3]{\log(n)}\,\sqrt[3]{(\log\log(n))^2}\Big)\Big),$$

(a saber, la Criba General de Campos Numéricos), sólo nos interesan aquellos ataques a RSA cuya complejidad sea menor a la de la Criba General de Campos Numéricos.

En algunos de dichos ataques, el ataque culmina con una ruptura total del sistema, pero eso no siempre es el caso. En su mayoría se tratan de ataques a ciertas implementaciones o abusos de RSA, es decir, a malas elecciones al momento de implementar RSA, y no a defectos inherentes en el algoritmo mismo.

Vamos a usar las siguientes convenciones en este capítulo: el módulo de RSA se denotará N, con $N = pq$ el producto de dos primos distintos. La llave pública consiste de dos

enteros positivos (N, e), y la llave privada de un entero d, tal que $de \equiv 1 \pmod{\varphi(N)}$; $\varphi(N)$ es la función φ de Euler, cuyo valor en N es $(p-1)(q-1)$.

El material de este capítulo está basado en el artículo de Dan Boneh [Bon99].

10.1 Ataque por módulo común

Supongamos que tenemos una red con varios usuarios, y queremos utilizar RSA para comunicación segura entre ellos. Para no generar muchos módulos $N_i = p_i q_i$, uno por usuario, simplemente elegimos un único módulo común para todos, $N = pq$, y le damos a cada usuario una llave pública y privada (e_i, d_i) distinta.

Parecería que esto nos produce una seguridad razonable, pero no es así. Vimos en el Teorema 8.5 que dado el módulo N, y las llaves pública y privadas (e, d) de un usuario, podemos obtener una factorización del módulo N de manera eficiente. Como cada usuario tiene una pareja (e_i, d_i), y conoce N, cada usuario puede usar su llave para factorizar N. Una vez factorizado N, dada la llave pública de cualquier otro usuario e_j, le es fácil encontrar el valor de la llave privada d_j. Es decir, cada usuario tiene una ruptura total del sistema.

Es entonces muy importante nunca usar un módulo para dos o más usuarios.

10.2 Ataques por exponente privado pequeño

Uno de los principales defectos de RSA y la criptografía de llave pública es su lentitud comparada con los sistemas criptográficos simétricos. Muchas veces se busca acelerar los sistemas durante la implementación.

Por ejemplo, para reducir el tiempo de descifrado de RSA, uno puede elegir un valor pequeño de d, y utilizarlo para calcular la llave pública e. Un valor chico bien elegido puede acelerar la velocidad de descifrado en un factor de al menos diez, de manera que se trata de una mejora sustancial. Por ejemplo, si $d = 3$, sólo se requieren dos multiplicaciones módulares para calcular M^d.

Desafortunadamente, si d es demasiado pequeño, se arriesga uno a una ruptura total del sistema.

Teorema 10.1 (M. Wiener) *Sea $N = pq$, $q < p < 2q$ dos primos. Supongamos que $d < \frac{1}{3}N^{1/4}$. Dado (N, e), con $ed \equiv 1 \pmod{\varphi(N)}$, se puede recuperar d eficientemente.*

Dem.: Usamos fracciones continuas. Puesto que $ed \equiv 1 \pmod{\varphi(N)}$, existe $k \in \mathbb{Z}$ tal que $ed - k\varphi(N) = 1$. Tenemos entonces que

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \frac{1}{d\varphi(N)},$$

de manera que $\frac{k}{d}$ aproxima $\frac{e}{\varphi(N)}$. No conocemos el valor de $\varphi(N)$, pero N es buena aproximación a $\varphi(N)$, puesto que

$$\varphi(N) = (p-1)(q-1) = pq - p - q + 1 = N - p - q + 1$$

y $p + q - 1 < 3\sqrt{N}$. Por lo tanto,

$$|N - \varphi(N)| < 3\sqrt{N}.$$

Tenemos entonces que:

$$\begin{aligned}\left|\frac{e}{N} - \frac{k}{d}\right| &= \left|\frac{ed - k\varphi(N) - kN + k\varphi(N)}{Nd}\right| \\ &= \left|\frac{1 - k(N - \varphi(N))}{Nd}\right| \\ &\leq \left|\frac{3k\sqrt{N}}{Nd}\right| \\ &= \frac{3k}{d\sqrt{N}}.\end{aligned}$$

Tenemos también que $k\varphi(N) = ed - 1 < ed$. Como $e < \varphi(N)$, entonces

$$k < d < \frac{1}{3}N^{1/4}.$$

Entonces tenemos, usando la desigualdad de arriba, que

$$\left|\frac{e}{N} - \frac{k}{d}\right| \leq \frac{1}{d\sqrt[4]{N}} < \frac{1}{2d^2}.$$

Ésta es una relación clásica de aproximación diofantina, en la cuál la distancia debe ser más pequeña que una función sencilla del denominador.

El número de fracciones $\frac{k}{d}$ con $d < N$ que aproximan $\frac{e}{N}$ con esta precisión está acotado por $\log_2(N)$, y todas se obtienen como parte de la expansión de $\frac{e}{N}$ en fracciones continuas. Uno calcula las primeras $\log_2(N)$ fracciones continuas de la expansión, y una de ellas es k/d, que ya está escrito en mínimos términos. Probamos todos los denominadores de nuestras fracciones hasta encontrar el valor correcto de d. □

Para evitar el ataque, si N es de 1024 bits, d debe tener al menos 256 bits. Esto es malo para smartcards, pues hacen que el descifrado sea bastante más lento.

Otra manera de evitar el ataque es usando una e más grande. Es decir, en vez de usar el valor de $e \bmod \varphi(N)$, publicar un valor $e' = e + t\varphi(N)$, para algún entero $t \gg 0$. Para efectos de el cifrado y descifrado de RSA, e' funciona igual que e (aunque calcular la e'-ésima potencia es más tardado que la e-ésima); pero entonces la k que aparece en la prueba es muy grande, lo cual evita que el ataque prospere. En general, si $e' > N^{3/2}$, el ataque no se puede montar.

Otra posible manera de intentar reducir el tiempo de descifrado es usar el Teorema Chino del Residuo. Para ello, tenemos que recordar no sólo el valor de N, sino también los valores p y q. Entonces uno define $d_p \equiv d \pmod{p-1}$, $d_q \equiv d \pmod{q-1}$, y para descifrar, dado el mensaje cifrado C, uno calcula dos valores:

$$\begin{aligned} M_p &\equiv C^{d_p} \pmod{p}, \\ M_q &\equiv C^{d_q} \pmod{q}. \end{aligned}$$

Entonces usamos el Teorema Chino del Residuo para encontrar M, que es el único valor módulo $pq = N$ tal que $M \equiv M_p \pmod{p}$ y $M \equiv M_q \pmod{q}$.

Uno puede elegir d_p y d_q chicos, pero de tal manera que d sea grande (lo cual es posible).

Desafortunadamente, hay un ataque tal que dado (N, e), permite factorizar el módulo en tiempo $O\left(\min\left\{\sqrt{d_p}, \sqrt{d_q}\right\}\right)$, de manera que si d_p ó d_q son demasiado pequeños, nuevamente tenemos problemas.

La cota dada por el Teorema de Wiener no es la mejor posible; Boneh y Durfee tienen un ataque modificado que funciona si $d < N^{0.292}$. El consenso entre los expertos es que en general $d < N^{1/2}$ es malo, y probablemente inseguro. Aunque esto aún es un problema abierto, no se recomienda usar exponentes privados menores de $\sqrt{N}$.

10.3 Ataques por exponente público pequeño

En vista de que no conviene intentar reducir el tiempo de descifrado mediante el uso de un exponente privado pequeño, quizás podríamos reducir el tiempo de cifrado (y de firmas electrónicas usando RSA), mediante el uso de un exponente público pequeño.

Como e debe ser impar, el menor posible valor para e es 3, pero hay ciertos ataques que funcionan con e tan pequeña, de manera que el valor recomendado para el exponente público es $e = 2^{16} + 1 = 65537$; en expresión binario, este valor tiene sólo dos unos, lo cual

acelera la exponenciación modular sustancialmente: sólo requiere de 16 multiplicaciones, mientras que un valor arbitrario de $e \leq \varphi(N)$ requiere, en promedio de alrededor de mil multiplicaciones.

A diferencia de los ataques en la sección anterior, los ataques que se basan en n exponente público pequeño normalmente no resultan en una ruptura total del sistema, y únicamente en la recuperación del mensaje M.

10.3.1 El Teorema de Coppersmith

Los ataques más poderosos por exponente público pequeño de RSA se basan en un Teorema de Coppersmith [Cop97]. El teorema de Coppersmith tiene muchas aplicaciones, pero sólo vamos a mencionar algunas. La demostración usa el algoritmo LLL[1] de reducción de base de retículas.

Teorema 10.2 (Teorema de Coppersmith) *Sean N un entero, y sea $f \in \mathbb{Z}[x]$ un polinomio mónico de grado d. Definimos $X = N^{\frac{1}{d}-\epsilon}$ para alguna $\epsilon \geq 0$. Entonces, dados N y f, se pueden encontrar eficientemente todos los enteros x_0, tales que $|x_0| < X$ y $f(x_0) \equiv 0 \pmod{N}$. El tiempo que tarda el algoritmo para encontrarlos es, en el peor de los casos, el mismo que tarda el algoritmo LLL aplicado a una retícula de dimensión $O(w)$, donde $w = \min\{1/\epsilon, \log_2(N)\}$.*

El Teorema de Coppersmith nos da entonces un algoritmo para encontrar de manera eficiente todas las raíces de $f(x)$ módulo N que son menores en valor absoluto que $X = N^{1/d}$. A medida que X se vuelve más pequeña, el tiempo que nos lleva correr el algoritmo también se vuelve menor. La fuerza del teorema es que nos da la abilidad de encontrar raíces pequeñas de polinomios módulo un entero compuesto N; los métodos para encontrar raíces cuando el módulo es un primo son bien conocidos, y mucho más rápidos, pero en general no hay un resultado análogo al Teorema Chino del Resiudo para polinomios de grado $d \geq 2$.

Vamos a esbozar las ideas principales en la prueba del Teorema de Coppersmith, usando una presentación simplificada de Howgrave-Graham.

Dado un polinomio $h(x) = \sum a_i x^i \in \mathbb{Z}[x]$, definimos

$$\|h\|^2 = \sum_i |a_i|^2.$$

La prueba se basa en la siguiente observación:

[1]El algoritmo LLL recibe su nombre de sus creadores: Arjen Lenstra, Hendrik Lenstra, Jr., y Laszlo Lovasz.

Lema 10.1 *Sea $h(x) \in \mathbb{Z}[x]$ un polinomio de grado d, y sea X un entero positivo. Supongamos que $\|h(xX)\| < N/\sqrt{d}$. Si $|x_0| < X$ satisface $h(x_0) \equiv 0 \pmod{N}$, entonces $h(x_0) = 0$ en los enteros.*

Dem.: De la desigualdad de Schwartz tenemos que

$$\begin{aligned} |h(x_0)| &= \left|\sum a_i x_0^i\right| \\ &= \left|\sum a_i X^i \left(\frac{x_0}{X}\right)^i\right| \\ &\leq \sum \left|a_i X^i \left(\frac{x_0}{X}\right)^i\right| \\ &\leq \sum \left|a_i X^i\right| \\ &\leq \sqrt{d}\|h(xX)\| \\ &< N. \end{aligned}$$

Puesto que $h(x_0) \equiv 0 \pmod{N}$, y en valor absoluto es menor que N, concluímos que $h(x_0) = 0$. □

El Lema 10.1 nos dice que si h es un polinomio con norma pequeña, entonces todas las raíces chicas de h módulo N son en realidad raíces de h sobre los enteros. El Lema sugiere entonces que para encontrar una raíz chica x_0 de $f(x)$ mod N, nos conviene buscar un polinomio $h(x) \in \mathbb{Z}[x]$ de norma pequeña que tenga las mismas raíces que $f(x)$ mod N. Entonces x_0 sería una raíz de $h(x)$ sobre los enteros, y las raíces enteras de polinomios con coeficientes enteros son fáciles de encontrar.

Para encontrar $h(x)$, podemos buscar un polinomio $g(x) \in \mathbb{Z}[x]$ tal que $h(x) = g(x)f(x)$ tenga norma pequeña, esto es, menor a N. Esto equivale a buscar una combinación lineal con coeficientes enteros de los polinomios $f(x)$, $xf(x)$, $x^2f(x), \ldots, x^r f(x)$, con norma pequeña. Desgraciadamente, en general no hay una combinación lineal no trivial que tenga norma suficientemente chica.

Coppersmith descubrió un truco para resolver el problema: si $f(x_0) \equiv 0 \pmod{N}$, entonces $f(x_0)^k \equiv 0 \pmod{N^k}$ para toda k. En general, definimos los siguientes polinomios para alguna m determinada:

$$g_{u,v}(x) = N^{m-v} x^u f(x)^v.$$

Entonces x_0 es raíz de $g_{u,v}(x)$ módulo N^m para toda $u \geq 0$ y toda $0 \leq v \leq m$. Para poder usar el Lema 10.1 tenemos que encontrar una combinación lineal con coeficientes enteros de los polinomios $g_{u,v}(x)$, a la que llamaremos $h(x)$, y tal que $h(xX)$ tenga norma menor que

N^m (pues X es una cota superior para las x_0, que satisface $X \leq N^{1/d}$). Puesto que ahora tenemos una cota de N^m en vez de N, uno puede demostrar que para m suficientemente grande siempre existe tal $h(x)$. Una vez que encontramos $h(x)$, aplicamos el Lema 10.1, que implica que el polinomio tiene a x_0 como raíz en los enteros, y podemos encontrar x_0 fácilmente.

Sólo falta ahora ver cómo encontrar $h(x)$ de manera eficiente. Para ello, mencionamos ciertos resultados básicos sobre retículas en $\mathbb{Z}^w$, sin demostración. Una posible referencia es [Lov86].

Sean $u_1, \ldots, u_w \in \mathbb{Z}^w$ vectores linealmente independientes. Una retícula (de rango completo) generado por $\langle u_1, \ldots, u_w \rangle$ es el conjunto de todas las combinaciones lineales con coeficientes enteros de $u_1, \ldots, u_w$. El determinante de L se define como el determinante de la matriz cuadrada de $w \times w$ cuyos renglones son los vectores $u_1, \ldots, u_w$.

En nuestro caso, consideramos los polinomios $g_{u,v}(xX)$ como vectores, y estudiamos la retícula L que generan. Tomamos los valores $v = 0, \ldots, m$ y $u = 0, \ldots, d-1$, de manera que la retícula tiene dimensión $w = d(m+1)$. Por ejemplo, cuando $f(x)$ es un polinomio mónico cuadrático y $m = 3$, la retícula que resulta está generada por los renglones de la siguiente matriz:

$$\begin{pmatrix}
N^3 & & & & & & & \\
 & XN^3 & & & & & & \\
 & * & X^2N^2 & & & & & \\
 & * & * & X^3N^2 & & & & \\
 & * & * & * & X^4N & & & \\
 & * & * & * & * & X^5N & & \\
 & * & * & * & * & * & X^6 & \\
 & * & * & * & * & * & * & X^6
\end{pmatrix}$$

donde los renglones de la matriz corresponden, respectivamente, a $g_{0,0}(xX)$, $g_{1,0}(xX)$, $g_{0,1}(xX)$, $g_{1,1}(xX)$, $g_{0,2}(xX)$, $g_{1,2}(xX)$, $g_{0,3}(xX)$, y $g_{1,3}(xX)$; las columnas respectivamente a 1, x, x^2, x^3, x^4, x^5, x^6, y x^7; y las entradas $*$ corresponden a coeficientes en los polinomios cuyos valores podemos ignorar. Todas las entradas vacías corresponden a ceros. Como al matriz es triangular superior, su determinante el el producto de los elementos de la diagonal. El objetivo es encontrar vectores pequeños en esta retícula.

Un resultado clásico de Hermite dice que cualquier retícula L de dimensión w contiene un punto $v \in L$, $v \neq 0$, cuya norma L_2 satisface $\|v\| \leq \gamma_w \det(L)^{1/w}$, donde γ_w es una constante que depende sólo de w. La cota de Hermite se puede usar para demostrar que para $m \gg 0$, nuestra retícula contiene vectores de norma menor a N^m, que es lo que queremos. Pero la pregunta es cómo podemos construir de manera eficiente un vector chico en L cuya longitud

no sea mucho mayor a la cota de Hermite. El algoritmo LLL es un algoritmo eficiente que hace precisamente eso:

Teorema 10.3 (LLL) *Sea L una retícula generada por $\langle u_1, \ldots, u_w \rangle$. Si $\langle u_1, \ldots, u_w \rangle$ son dados como la entrada, el algoritmo LLL encuentra un punto $v \in L$ que satisface*

$$\|v\| \leq 2^{w/4} \det(L)^{1/w}.$$

El tiempo de proceso del algoritmo LLL es un polinomio de grado 4 en la longitud de la entrada.

El algoritmo LLL tiene muchas aplicaciones en teoría de números computacional y criptografía. Su descubrimiento en 1982 dio un algoritmo eficiente para factorizar polinomios sobre los enteros, y de hecho sobre anillos de enteros de campos numéricos. El algoritmo LLL se usa para muchos ataques a diversos criptosistemas; por ejemplo, muchos criptosistemas basados en el knapsack se rompieron usando el algoritmo LLL.

Con el algoritmo LLL podemos ahora completar la prueba del Teorema de Coppersmith. Para verificar que el vector obtenido mediante LLL satisface la cota del Lema 10.1, necesitamors que:

$$2^{w/4} \det(L)^{1/w} < N^m/\sqrt{w},$$

donde $w = d(m+1)$ es la dimensión de L. En efecto, para $m \gg 0$, la desigualdad se satisface; si $X = N^{\frac{1}{d}-\epsilon}$, basta con tomar $m = O(k/d)$ con $k = \min\{1/\epsilon, \log N\}$. En consecuencia, el tiempo que tarda el algoritmo de Coppersmith está dominado por el tiempo que tarda correr el algoritmo LLL en una retícula de dimensión $O(w)$, como dijimos.

Vale la pena mencionar que aunque la técnica de Coppersmith parece ser aplicable para ciertos polinomios de dos variables, no se tiene una prueba de que siempre se puede usar; hay varios resultados que dependen de una versión de dos variables, de manera que extender el Teorema de Coppersmith a polinomios de varias variables es un problema importante por resolver.

10.3.2 Ataque de Hastad por envío masivo

Nuestra primera aplicación del teorema de Coppersmith es una mejora de Hastad a un viejo ataque. Supongamos que B quiere mandarle un mensaje cifrado M a varias personas, $P_1, P_2, \ldots, P_k$. Cada persona P_i tiene su propia llave RSA (N_i, e_i). Y supongamos que M es menor que todas las N_i.

Inocentemente, B cifra M con cada llave, y le manda el i-ésimo cifrado a P_i. Nuestro criptoanalista obtiene entonces los k textos cifrados, con las distintas llaves.

Por simplicidad, supongamos que todos los exponentes públicos e_i valen 3. Entonces es fácil para el criptoanalista recuperar el mensaje M si $k \geq 3$. En efecto, tenemos tres mensajes cifrados, C_1, C_2, y C_3, donde

$$C_1 \equiv M^3 \pmod{N_1}; \quad C_2 \equiv M^3 \pmod{N_2}; \quad C_3 \equiv M^3 \pmod{N_3}.$$

Podemos asumir que $(N_i, N_j) = 1$, pues si tienen algún factor común, lo podemos usar para factorizar los módulos (puesto que son distintos entre sí). Entonces podemos aplicar el Teorema Chino del Residuo a C_1, C_2, y C_3, para obtener un $C' \in \mathbb{Z}_(N_1N_2N_3)$ tal que $C' \equiv M^3 \pmod{N_1N_2N_3}$. Pero como M es menor que cada N_i, entonces $M^3 < N_1N_2N_3$. Es decir, $C' = M^3$ en los enteros. Para recuperar M, simplemente calculamos la raíz cúbica de C'.

En general, si todos los exponentes públicos son iguales a e, podemos recuperar M en cuanto $k \geq e$.

Este es el viejo ataque, y obviamente sólo es eficaz cuando e es muy pequeña. Pero Hastad describió un ataque mucho más poderoso. Para motivarlo, consideremos una defensa inocente contra el ataque de arriba. En vez de simplemente cifrar M y enviarlo, B puede "rellenar" el mensaje antes de cifrar; por ejemplo, si M tiene m bits de largo, entonces B puede envíar el cifrado de $M_i = i2^m + M$ a la persona P_i. Como el criptoanalista obtiene cifrados de distintos textos, el ataque de arriba no se puede montar. Pero Hasta probó que este tipo de relleno lineal no es seguro; de hecho, si aplicamos un polinomio fijo antes de cifrar, obtenemos una transmisión masiva insegura.

Supongamos que para cada participante $P_1, \ldots, P_k$, B tiene un polinomio fijo, conocido públicamente, $f_i(x) \in \mathbb{Z}_{N_i}[x]$; y para envíar un mensaje M, B envía el cifrado de $f_i(M)$ a P_i. El criptoanalista entonces conoce todas las llaves públicas, los polinomios $f_i(x)$, y los valores $C_i = f_i(M)^{e_i}$ para $i = 1, \ldots, k$. Si k es suficientemente grande, entonces podemos recuperar el texto original:

Teorema 10.4 (Hastad) *Sean $N_1, \ldots, N_k$ enteros positivos primos relativos dos a dos, y sea $N_{min} = \min_i(N_i)$. Sean $g_i \in \mathbb{Z}_{N_i}[x]$, $i = 1, \ldots, k$ polinomios de grado menor o igual a d. Supongamos que existe un único entero $M < N_{min}$ que satisface*

$$g_i(M) \equiv 0 \pmod{N_i} \qquad \text{para cada } i = 1, \ldots, k.$$

Si $k > d$, entonces uno puede recuperar eficientemente M dados los valores (N_i, g_i).

Dem.: Sea $\overline{N} = N_1 \cdots N_k$. Podemos asumir que todos los $g_i(x)$ son mónicos; pues si para alguna i el coeficiente principal de g_i no es invertible en $\mathbb{Z}_{N_i}$, entonces no es primo relativo con N_i y podemos usar su máximo común divisor para factorizar N_i. Si multiplicamos cada g_i por una potencia apropiada de x, podemos asumir que todos los g_i tienen grado exactamente igual a d. Construímos ahora el polinomio

$$g(x) = \sum_{i=1}^{k} T_i g_i(x)$$

donde

$$T_i = \begin{cases} 1 \bmod N_j & \text{si } i = j \\ 0 \bmod N_j & \text{si } i \neq j \end{cases}$$

Las T_i son enteros que se pueden encontrar con el Teorema Chino del Residuo. Entonces $g(x)$ es mónico, pues es mónico módulo todas las N_i y su coeficiente principal es menor $\overline{N}$. El grado de $g(x)$ es d; y sabemos que $g(M) \equiv 0 \pmod{\overline{N}}$. El Teorema es ahora consecuencia del Teorema de Coppersmith, pusto que $M < N_{min} \leq \overline{N}^{1/k} < \overline{N}^{1/d}$. □

El Teorema de Hastad nos dice que un sistema de ecuaciones en una incógnita con módulos primos relativos dos a dos se puede resolver eficientemente, si es que nos dan suficientes ecuaciones. Si tomamos $g_i = f_i^{e_i} - C_i \pmod{N_i}$, el criptoanalista puede recuperar M de los criptotextos que tenemos siempre que el número de personas sea al menos d, donde d es el máximo de $e_i \text{gr}(f_i)$ tomado sobre $i = 1, \ldots, k$. En particular, si todas las e_i son iguales a un mismo valor e, podemos recuperar el texto original M cuando $k > e$.

Para poderse proteger contra el ataque de Hastad, es necesario utilizar $e \gg 0$, o bien usar rellenos aleatorios en vez de rellenos fijos para los mensajes.

10.3.3 Ataque de Franklin-Reiter por mensajes relacionados

M. Franklin y M. Reiter descubrieron un método de ataque cuando B le manda a A varios mensajes cifrados relacionados, usando el mismo módulo. Sea (N, e) la llave pública de RSA de A, y supongamos ue $M_1, M_2 \in \mathbb{Z}_N^*$ son dos mensajes distintos que satisfacen la relación $M_1 = f(M_2) \pmod{N}$ para algún polinomio conocido $f(x) \in \mathbb{Z}_N[x]$. Para enviar M_1 y M_2 a A, B cifra cada mensaje por separado y envía los criptotextos C_1 y C_2. Vamos a ver que, conociendo (N, e) y $f(x)$, dados C_1 y C_2 el criptoanalista puede recuperar fácilmente M_1 y M_2. Aunque el ataque funciona para cualquier valor chico de e, lo hacemos con $e = 3$ para facilitar la exposición.

Lema 10.2 (Franklin-Reiter) *Sean $e = 3$, y (N, e) una llave pública de RSA. Sean $M_1 \neq M_2 \in \mathbb{Z}_N^*$ dos mensajes que satisfacen la relación $M_1 \equiv f(M_2) \pmod N$ para algún polinomio lineal $f(x) = ax + b \in \mathbb{Z}_N[x]$, con $b \neq 0$. Entonces, dados (N, e, C_1, C_2, f), un criptoanalista puede recuperar M_1 y M_2 en tiempo cuadrático en* $\log(N)$.

Dem.: Por lo pronto, tomamos e arbiraria. Como $C_1 \equiv M_1^e \pmod N$, sabemos que M_2 es raíz del polinomio

$$g_1(x) = f(x)^e - C_1 \in \mathbb{Z}_N[x].$$

De manera análoga, M_2 es raíz del polinomio

$$g_2(x) = x^3 - C_2 \in \mathbb{Z}_N[x].$$

El factor lineal $x - M_2$, por lo tanto, divide a ambos polinomios. Podemos usar el algoritmo de Euclides[2] para calcular el máximo común divisor de $g_1(x)$ y $g_2(x)$. Si el máximo común divisor es lineal, entonces hemos encontrado el valor de M_2; el algoritmo de Euclides funciona en tiempo cuadrático con argumentos e y $\log(N)$.

Supongamos ahora que $e = 3$, y vamos a probar que el máximo común divisor *tiene* que ser lineal. El polinomio $x^3 - C_2$ se factoriza módulo p y módulo q como el producto de un factor lineal y un factor irreducible cuadrático, pues como $(e, \varphi(N)) = 1$, $x^3 - C_2$ sólo tiene una raíz en $\mathbb{Z}_N$. Como g_2 no divide a g_1, el máximo común divisor tiene que ser lineal.

Si $e > 3$, el máximo común divisor es casi siempre lineal. Sin embargo, en algunos casos extraños es posible tener M_1, M_2, y f para el cual el máximo común divisor no es lineal, y en ese caso el ataque fracasa. □

Cuando $e > 3$, el ataque tiene complejidad cuadrática en e; de manera que sólo se puede montar cuando el exponente público es chico. Para e grande, la cantidad de trabajo necesaria para calcular el máximo común divisor se vuelve prohibitiva, y el ataque no se puede montar.

10.3.4 Ataque de Coppersmith por relleno chico

En ésta sección utlizaremos el concept del *resultante de dos polinomios.* El lector que no esté familiarizado con resultantes puede encontrar la información necesaria (sin demostraciones) en el Apéndice C.

[2]El anillo $\mathbb{Z}_N[x]$ no es un anillo euclidiano; pero si aplicamos el algoritmo de Euclides usual a sus polinomios, se puede probar que o bien el algoritmo trabaja sin problemas, o bien si el algoritmo tiene algún problema, podemos entonces usar esa información para factorizar N.

El ataque de Franklin-Reiter puede parecer un poco artificial. Después de todo, ¿por qué asumimos que B le va a mandar a A cifrados de mensajes relacionados? Sin embargo, Coppersmith fortaleció el ataque, y de paso probó un resultado muy importante sobre rellenos.

Un algoritmo inocente de relleno aleatorio sería tomar el texto M, y agregarle unos cuantos bits aleatorios al principio o al final. Pero el siguiente ataque demuestra el peligro de rellenos simples como éste. Supongamos que B envía un mensaje M con relleno aleatorio a A. El criptoanalista intercepta el criptotexto C, y evita que llegue a su destino. Como B nota que A no responde, decide volver a enviar el mensaje M a A. Nuevamente le aplica un relleno aleatorio, y transmite el criptotexto correspondiente. El criptoanalista ahora tiene dos criptotextos que corresponden a dos cifrados del mismo texto, pero con rellenos aleatorios distintos. El siguiente teorema nos dice que aunque el criptoanalista no conoce el relleno utilizado, de todos modos puede recuperar el texto original.

Teorema 10.5 *Sea (N, e) una llave pública de RSA, donde N tiene longitud de n-bits. Sea $m = \lfloor n/e^2 \rfloor$. Sea $M \in \mathbb{Z}_N^*$ un mensaje de longitud menor o igual a $n - m$ bits. Definimos $M_1 = 2^m M + r_1$ y $M_2 = 2^m M + r_2$, donde r_1 y r_2 son dos enteros distintos, arbitrarios, tales que $0 \leq r_1, r_2 < 2^m$. Dados (N, e), y los cifrados C_1 y C_2 de M_1 y M_2 (pero sin conocer r_1 ni r_2), se puede recuperar M eficientemente.*

Dem.: Definimos $g_1(x, y) = x^e - C_1$, y $g_2(x, y) = (x + y)^e - C_2$. Sabemos que si $y = r_2 - r_1$, entonces ambos polinomios tienen a M_1 como raíz común. Por lo tanto, $\Delta = r_2 - r_1$ es una raíz del resultante en x de g_1 y g_2, $h(y) = R_x(g_1, g_2) \in \mathbb{Z}_N[y]$. El grado de h es a lo mucho e^2. Además, $|\Delta| < 2^m < N^{1/e^2}$, de manera que Δ es una raíz pequeña de $h(y)$ módulo N, y la podemos encontrar de manera eficiente usando el Teorema de Coppersmith. Una vez que encontramos el valor de Δ, podemos usar el ataque de Franklin-Reiter de la sección anterior y recuperar M_2; con M_2, obtenemos M. □

Si $e = 3$, el ataque se puede montar siempre y cuando el relleno tenga longitud no mayor a $1/9$ de la longitud del mensaje. Esto ya es un resultado importante: nos dice que para exponentes chicos, necesitamos rellenos grandes. Pero si usamos el valor recomendado de $e = 65537$, entonces el ataque se vuelve inútil contra llaves de RSA del tamaño usual.

10.3.5 Ataques por exposición parcial de la llave

Sea (N, d) una llave privada de RSA. Supongamos que de alguna manera, el criptoanalista logra obtener una fracción de los bits de d, digamos, la cuarta parte de los bits. ¿Puede

entonces reconstruír el resto de d? Es algo sorpresivo que la respuesta a esta pregunta es *sí* cuando el exponente público e es chico. En 1998, Boneh, Durfee, y Frankel probaron que si $e < \sqrt{N}$, entonces es posible reconstruir d de una fracción de sus bits. Los resultados ilustran la gran importancia de cuidar **toda** la llave privada de RSA.

Teorema 10.6 (Boneh-Durfee-Frankel) *Sea (N, d) una llave privada de RSA, donde N es de longitud n bits. Dados los $\lceil n/4 \rceil$ bits menos significativos de d, se puede reconstruír toda d mediante un algoritmo cuya complejidad es lineal en $e \log_2(e)$.*

La demostración del Teorema 10.6 depende de otro resultado de Coppersmith:

Teorema 10.7 (Coppersmith, 1997) *Sea $N = pq$ un módulo de RSA de n bits de longitud. Dados los $n/4$ bits menos significativos de p, o los $n/4$ bits más significativos de p, se puede factorizar N de manera eficiente.*

Dem.: Demostramos el Teorema 10.6 asumiendo el Teorema 10.7. Por definición de d y e, existe un entero k tal que

$$1 = ed - k\varphi(N) = ed - k(N - p - q + 1).$$

Como $d < \varphi(N)$, tenemos también que $0 < k \leq e$. Si reducimos la ecuación módulo $2^{n/4}$ (es decir, nos fijamos sólo en los $n/4$ bits menos significativos), usamos la sustitución $q = N/p$, y multiplicamos por p para eliminar denominadores, tenemos:

$$(ed)p - kp(N - p + 1) + kN = p \pmod{2^{n/4}}.$$

Como el criptoanalista conoce los $n/4$ bits menos significativos de d, y conoce el valor de e (que es público), el criptoanalista tiene el valor de $ed \bmod 2^{n/4}$. Por lo tanto, la ecuación de arriba nos dá una ecuación en dos incógnitas, k y p. Para cada uno de los e posibles valores de k, el criptoanalista tiene que resolver una ecuación de grado dos en p para obtener un número de candidatos para el valor de $p \bmod 2^{n/4}$. Para cada uno de estos candidatos, puede usar el algoritmo del Teorema 10.7 para tratar de factorizar N. El número total de candidatos para $p \bmod 2^{n/4}$ es a lo mucho $e \log_2(e)$, de manera que después de a lo mucho $e \log_2(e)$ intentos, habrá factorizado N y obtenido una ruptura total del sistema. □

El Teorema 10.6 se conoce como un ataque de *exposición parcial de la llave.* Existen ataques similares para valores más grandes de e, siempre y cuando $e < \sqrt{N}$. Es interesante notar que los criptosistemas que se basan en el problema del logaritmo discreto, como por ejemplo ElGamal, no parecen ser suceptibles a ataques de exposición parcial de la llave.

10.3.6 Filtración inherente con exponente público chico

Para terminar con los problemas de exponente público chico, vamos a demostrar que el criptosistema RSA filtra la **mitad** de los bits más significativos del exponente privado cuando el exponente público es pequeño.

Recordemos que tenemos la ecuación $ed - k(N - p - q + 1) = 1$ para algún entero k que satisface $0 < k \leq e$. Dado k, el criptoanalista puede calcular

$$\hat{d} = \lfloor (kN + 1)/e \rfloor.$$

Entonces tenemos que

$$|\hat{d} - d| \leq k(p + 1)/e \leq \frac{3k\sqrt{N}}{e} < 3\sqrt{N}.$$

Es decir, $\hat{d}$ es una buena aproximación para d. La cota nos dice que, para la mayoría de los valores de d, la mitad de los bits más significativos de $\hat{d}$ son iguales a los de d. Como sólo hay e posibles valores para k, el criptoanalista puede construír un conjunto pequeño de tamaño e tal que cada elemento del conjunto tiene la mitad de sus bits más significativos iguales a los de d.

El caso en que $e = 3$ es particularmente interesante: en este caso, el valor de k está fijo, y debe ser $k = 2$. De esta manera, no hay opciones, y el sistema ha filtrado completamente la mitad de los bits más significativos de d.

10.4 Ataques de implementación

Vamos ahora a ver un tipo distinto de ataques a RSA. En vez de atacar la estructura matemática de RSA, como en las secciones anteriores, se atacan las implementaciones de RSA.

10.4.1 Ataques por tiempo

Imaginemos una tarjeta inteligente con la llave privada de RSA del usuario. Como la tarjeta es resistente y no se puede examinar directamente, el criptoanalista que obtiene la tarjeta no puede descubrir directamente la llave. Sin embargo, un ataque de P. Kocher, descrito en 1996, muestra cómo a base de medir de manera precisa el tiempo que le lleva a la tarjeta hacer el descifrado (o firma), el criptoanalista puede descubrir el valor de la llave privada d.

Por ejemplo, supongamos que la tarjeta implementa RSA usando el método de eleva al cuadrado y multiplica. Recordemos que si

$$d = d_n d_{n-1} \dots d_0$$

es una representación binaria de d (es decir, $d = \sum 2^i d_i$, con $d_i \in \{0, 1\}$), entonces el método de elevar al cuadrado y multiplicar calcula $c = M^d$ iterativamente de la siguiente manera: empezamos inicializando una variable t y una variable c, poniendo $t = M$ y $c = 1$; para $i = 0, \dots, n$, si $d_i = 1$, multiplicamos c por t módulo N; si $d_i = 0$, dejamos c como estaba; y en ambos casos reemplazamos t por t^2 mod N.

Para montar el ataque, el criptoanalista le pide a la tarjeta inteligente que genere firmas electrónicas para un gran número de mensajes aleatorios, $M_1, \dots, M_k \in \mathbb{Z}_N^*$, y mide el tiempo T_i que le lleva a la tarjeta generar la firma del i-ésimo mensaje.

El ataque recupera los bits de d de uno por uno, empezando por el menos significativo. Puesto que d tiene que ser impar, sabemos que $d_0 = 1$. Consideramos ahora como recuperar d_1. En la segunda iteración del algoritmo de elevar al cuadrado y multiplicar, $t = M^2$ mod N, y $c = M$. Si $d_1 = 1$, entonces la tarjeta inteligente calcula el producto $ct = M \cdot M^2$ mod N; si $d_1 = 0$, la tarjeta no hace nada. Sea t_i el tiempo que le lleva a la tarjeta calcular $M_i \cdot M_i^2$ mod N. Las t_i's son distintas entre sí, pues el tiempo que lleva calcular la multiplicación depende del valor de M_i. El tiempo t_i se calcula antes de montar el ataque, en una máquina separada, usando las especificaciones físicas de la tarjeta (es decir, la implementación física del algoritmo en la tarjeta). Esto le permite aislar el tiempo de ésta iteración, y separarlo del tiempo completo del algoritmo.

Kocher notó que si $d_1 = 1$, entonces los conjuntos $\{t_i\}$, y $\{T_i\}$ están relacionados: si para alguna i tenemos que t_i es mucho más grande que el promedio, entonces la T_i correspondiente es muy probable que también sea más grande que el promedio. Es decir, hay una alta correlación entre el conjunto ordenado $(t_1, \dots, t_k)$, y el conjunto ordenado $(T_1, \dots, T_k)$.

Pero si $d_1 = 0$, entonces los dos conjuntos ordenados $(t_1, \dots, t_k)$ y $(T_1, \dots, T_k)$ se comportan como variables aleatorias independentes, sin ninguna correlación. El criptoanalista puede medir la correlación entre los dos conjuntos, y determinar de esa manera si d_1 vale 0 ó 1.

Una vez determinados los valores de d_0 y d_1, puede repetir el proceso para encontrar los valores de d_2, d_3, etc.

Recordemos además que si se está usando un exponente público e pequeño, entonces el ataque de exposición parcial de la llave de la sección anterior nos dice que el ataque de Kocher sólo tiene que recuperar la cuarta parte de los bits de d para poder recuperar el

valor completo de d.

Hay dos maneras de defenderse del ataque de Kocher. La primera y más sencilla es agregarle a la implementación un tiempo de espera adicional para que la exponenciación modular siempre se tarde el mismo tiempo.

La segunda manera es una idea de Rivest, y se llama *cegado*[3]. Antes de descifrar M (o de firmarlo), la tarjeta inteligente escoge un número aleatorio $r \in \mathbb{Z}_N^*$, y calcula el valor de $M' = M \cdot r^e \pmod{N}$. Después eleva M' a la d módulo N, y obtiene $C' = (M')^d \pmod{N}$. Finalmente, la tarjeta calcula $C = C'/r \pmod{N}$, que es el valor de M. El cegado hace que la tarjeta esté exponenciando un mensaje aleatorio M' que no es conocido por el criptoanalista, de manera que éste último no puede montar el ataque.

Kocher también ha descubierto otro ataque similar, llamado Criptoanálisis de consumo. Kocher demostró que el consumo de energía de la tarjeta inteligente cuando ésta genera una firma electrónica permite determinar si en una iteración se están realizando una o dos multiplicaciones; cuando se realiza una, el bit de la llave es 0, y si el bit de la llave es 1, entonces se realizan dos operacions. Ni agregar tiempo de espera, ni el cegado de Rivest, protegen contra este ataque.

10.4.2 Errores aleatorios

Muchas implementaciones de RSA usan el Teorema Chino del Residuo para acelerar el descifrado y firmado de documentos, facilitanto el cálculo de $M^d \bmod N$. En vez de trabajar módulo N, el usuario calcula las firmas del documento módulo p y módulo q, y luego usa el Teorema Chino del Residuo para obtener el resultado correcto. Explícitamente, si queremos calcular $M^d \bmod N$, primero cálculamos

$$C_p \equiv M^{d_p} \pmod{p}$$

y

$$C_q \equiv M^{d_q} \pmod{q}$$

donde $d_p \equiv d \pmod{(p-1)}$ y $d_q \equiv d \pmod{(q-1)}$; y después obtenemos el valor de $C = M^d \bmod N$ tomando

$$C \equiv T_1 C_p + T_2 C_q \pmod{N},$$

donde

$$T_1 = \left\{ \begin{array}{l} 1 \bmod p \\ 0 \bmod q \end{array} \right\} \qquad \text{y} \qquad T_2 = \left\{ \begin{array}{l} 0 \bmod p \\ 1 \bmod q \end{array} \right\}$$

[3] *Blinding* en inglés.

son los coeficientes del Teorema Chino del Residuo.

El tiempo que lleva hacer este último cálculo es negligible comparado con el tiempo de hacer las exponenciaciones, y éstas son a su vez mucho más rápidas que la exponenciación módulo N, pues p y q son aproximadamente de la mitad del tamaño de N. Esto es, multiplicar módulo p es aproximadamente cuatro veces más rápido que multiplicar módulo N. En general, usar el Teorema Chino del Residuo reduce el tiempo necesario de operación por un factor de cuatro.

Sin embargo, Boneh, DeMillo, y Lipton observaron en 1997 que hay un peligro inherente en usar el método del Teorema Chino del Residuo. Supongamos la computadora o tarjeta inteligente comete un error durante la generación de la firma en una instrucción; por ejemplo, mientras copia un registro de un lugar de memoria a otro, comete una transposición de bits. Este error puede ser causado facilmente por interferencia electromagnética, o quizás incluso por algún error de hardware como el que tenían algunas versiones iniciales del procesador Pentium. Dada una firma calculada con un error como éste, el criptoanalista puede factorizar el módulo N y obtener una ruptura total.

Vamos a explicar cómo utilizar el error para obtener una ruptura total, siguiendo la presentación de Arjen Lenstra. Supongamos que ocurren un solo error al generar la firma. Como resultado, exactamente una de C_p y C_q está calculada incorrectamente, mientras que la otra está calculada correctamente. Sin pérdida de generalidad, supongamos que C_p está calculado correctamente, pero el valor obtenido $\hat{C}_q$ para C_q es incorrecto. La firma resultante,

$$\hat{C} = T_1 C_p + T_2 \hat{C}_q,$$

es por ende incorrecta. Cuando el criptoanalista recibe $\hat{C}$, se da cuenta de inmediate que hay un error, pues $\hat{C}^e \not\equiv M \pmod{N}$. Pero, notemos que

$$\hat{C}^e \equiv M \pmod{p}; \qquad \text{pero} \qquad \hat{C}^e \not\equiv M \pmod{q}.$$

Eso quiere decir que el máximo común divisor de N y $\hat{C}^e - M$ es p, lo cual nos da un factor no trivial de N, y con ello la factorización de N.

Para que el ataque funcione, el criptoanalista tiene que conocer el valor de M; es decir, estamos suponiendo que la persona que está generando las firmas electrónicas no está usando ningún tipo de relleno aleatorio. Agregar relleno aleatorio evita que se monte el ataque, pero una mucho mejor defensa es simplemente verificar la firma generada antes de enviarla. Verificar el resultado es particularmente importante cuando usamos la aplicación del Teorema Chino del Residuo para aumentar la velocidad.

Los errores aleatorios son un verdadero peligro para muchos sistemas criptográficos; varias implementaciones, incluyendo varias de RSA que *no* usan el Teorema Chino del

Residuo, se pueden atacar con éxito utilizando errores aleatorios. Esto es un problema para estos sistemas, pues la verificación suele ser igual de lenta que el cifrado y descifrado, y esta lentitud ya es uno de los problemas de estos sistemas.

11.1 Criptografía simétrica vs. criptografía de llave pública

A lo largo del texto hemos visto diversos mecanismos de cifrado que podemos clasificar, como ya hemos mencionado, en sistemas simétricos (aquellos en los que se utiliza la misma clave para cifrar y para descifrar), y sistemas de llave pública (un tipo particular de los sistemas asimétricos en los que se usan diferentes claves para cifrar y para descifrar y en los que, además, conocer una de las claves no ofrece mucha información útil para conocer la otra).

Los sistemas simétricos resultan convenientes porque:

- Son versátiles. En general es posible implementarlos fácilmente tanto en software como en hardware. El término "fácilmente" se refiere a que el software o hardware son simples, no se requiere de una infraestructura muy especial. Las operaciones de

DES, por ejemplo, son sólo corrimientos de bits y XOR, fácilmente implementables en hardware.

- Poseen alto rendimiento. Son algoritmos rápidos, no muy complejos y si se implementan en hardware procesan del orden de cientos de megabytes[1] por segundo, en software su velocidad es dos ordenes de magnitud menor [Sch96], pero en general bastante rápidos de todos modos. Por ejemplo el chip DES desarrollado en los *Sandia National Laboratories*[2] opera a una tasa de 6.7 Gigabits por segundo, es decir puede cifrar aproximadamente unos 100 millones de bloques de DES (64 bits) por segundo.
- Las claves son relativamente (respecto a la longitud del texto cifrado) cortas. DES, por ejemplo usa claves de 64 (56 realmente) bits, AES usa claves de 128, 192 o 256 bits.
- Se pueden componer para lograr cifrados más fuertes.
- En general se conocen muy bien sus fortalezas y debilidades.

Pero por supuesto también hay puntos en contra:

- La clave debe permanecer secreta. La misma clave se usa para cifrar y descifrar, así que debe haber una "doble garantía" de que la clave está segura: tanto el emisor como el receptor deben protegerla.
- Es difícil pensar en un sistema con una red de usuarios grande en el que todos puedan hablarse en secreto con todos. Si hay n usuarios debe haber $\left(\begin{array}{c} n \\ 2 \end{array} \right) = O(n^2)$ claves diferentes, cada vez que ingresa un nuevo usuario se deben acordar n claves nuevas. Si se decide que haya una entidad única que distribuya las claves entonces esta debe ser inviolable e incorruptible, 100% confiable (lo que suele decirse en inglés *Unconditionally Trusted Third Party*) [MvOV96] ya que sabe **todo** de **todos** los usuarios.
- Hay que cambiar las llaves con periodicidad para no favorecer el criptoanálisis por volumen de transmisión cifrada. Esto significa que el problema mencionado de establecer las llaves es todavía más complicado.

Por su parte la criptografía de llave pública también tiene sus fortalezas:

- Sólo la llave privada debe mantenerse en secreto y sólo es conocida por una persona.

[1] 1 megabyte (MB) = 8,388,608 bits.
[2] Más información en: `http://www.sandia.gov/media/NewsRel/NR1999/encrypt.htm`.

- Es posible pensar en una red de usuarios con un directorio público de llaves de cifrado.
- De haber un administrador central de llaves, éste no necesita ser tan confiable como en el caso de la criptografía simétrica. Aal tipo de autoridad de administración de llaves necesario se le conoce en inglés como *Functionally Trusted Third Party*, una confiabilidad más débil [MvOV96] ya que no conoce las llaves privadas de los usuarios.
- Las parejas (llave pública, llave privada) pueden ser usadas durante periodos de tiempo más prolongados que en la criptografía simétrica.
- Son fáciles de usar para implementar sistemas de firmas digitales.
- En una red con muchos usuarios se requieren pocas llaves, comparativamente a los mencionado arriba para los sistemas simétricos.

Las desventajas de la criptografía de llave pública son:

- Para implementarse requieren de mucha mayor infraestructura que los sistemas simétricos. En software se requiere de bibliotecas de manejo de grandes números y con funciones de aritmética modular y pruebas de primalidad. En hardware las operaciones son mucho más complicadas. Basta recordar que para elevar un número a una potencia por el método de elevar al cuadrado y multiplicar hay que implementar un algoritmo de orden $O(\log^3 n)$, donde n es el módulo.
- Son mecanismos mucho más lentos que los de criptografía simétrica usuales. Por ejemplo el chip[3] IBM 4758 tarda en promedio 3,568 ms (poco más de 3 segundos) en establecer la llave de RSA; es capaz de cifrar 505 mensajes por segundo y de descifrar 174.8 en el mismo tiempo, compárense estas cifras con las del chip DES mencionado arriba.
- El tamaño de las llaves es mucho más grande (al menos un orden de magnitud mayor) al requerido por los sistemas simétricos. Esto es porque los mecanismos de criptoanálisis de los sistemas simétricos terminan en una búsqueda exhaustiva, en los sistemas de llave pública, en cambio, hay muchos recursos teóricos que sustentan atajos posibles; así que para obtener en un sistema de llave pública una seguridad comparable a la de un sistema simétrico se requiere una longitud de llave mayor.
- A pesar de lo revisado en los capítulos anteriores, nuestro conocimiento de estos sistemas es más reducido que el que se posee de los simétricos. Están basados (en el mejor de los casos) en el supuesto de que hay algunos problemas difíciles y en el

[3]Mayor información en `http://www-3.ibm.com/security/cryptocards/html/perfcpq.shtml`.

supuesto de que no hay otro modo de hacer ciertas cosas más que siguiendo el camino difícil.

- Como la llave de cifrado es pública, si hay relativamente pocos mensajes posibles, es posible montar un ataque de texto claro elegido, algo que puede ocurrir cuando hay mensajes cortos con relleno mal elegido.

Hay que hacer notar que las ventajas de los sistemas simétricos y de los de llave pública son complementarias. Los de llave privada (simétricos) son rápidos y tienen una excelente relación costo-beneficio y los de llave pública son buenos para resolver el problema que les dió origen: la distribución de las llaves y autentificación. Así que ¿por qué no pensar en combinar ambos en un sistema híbrido?

11.2 Sistemas mixtos

Imaginemos que tenemos un sistema de llave pública ya establecido. Tenemos un conjunto de usuarios, cada uno tiene su pareja de llaves pública y privada y las llaves públicas, que sirven para cifrar, están disponibles en un directorio de usuarios que cualquiera puede consultar para poder enviar mensajes cifrados a otro usuario. Sean A y B dos usuarios del sistema[4] y que A quiere comunicarse en secreto con B. Entonces A puede proceder como sigue:

1. A obtiene la llave pública de B.
2. A usa la llave pública de B para cifrar un mensaje que contiene una llave de un sistema simétrico, que previamente han acordado A y B. Ésta es llamada la *llave de sesión*. A envía a B este mensaje.
3. B descifra el mensaje usando su llave privada y recupera la llave de sesión.
4. A y B se comunican, en adelante, usando el sistema simétrico acordado y cifrando los mensajes con la llave de sesión.

Suena bien, pero el paso 1 puede traernos problemas. ¿Cómo consigue A la llave pública de B? Consultando el directorio de llaves públicas que está guardado en algún lugar, típicamente en la base de datos de una computadora que reparte las llaves a quién lo solicita, una autoridad de distribución de llaves. En este escenario puede ocurrir algo como lo siguiente: imaginemos que un tercero C, quiere saber lo que A y B se dirán en secreto.

[4]Podemos pensar indistintamente que estos "usuarios" son personas o computadoras de una red.

1. A solicita, a quien cree es la autoridad de distribución de llaves, la llave pública de B.
2. C se hace pasar por la autoridad de distribución de llaves y le contesta a A con su propia llave pública.
3. C solicita a la verdadera autoridad la llave pública de B.
4. A genera la llave de sesión, la cifra con la llave pública de C (pensando que es la de B) y la envía.
5. C descifra la llave de sesión que envió A, la vuelve a cifrar usando la pública de B y la envía.
6. B recibe la llave de sesión cifrada y la descifra usando su llave privada.
7. A, B y C tienen ahora la misma llave de sesión y C puede leer todo lo que se digan A y B.

El problema se presenta también si A pregunta a B su llave pública. En ese caso C no se hace pasar por una autoridad de distribución de llaves, pero simplemente pretende que es B mismo. A este tipo de ataque se le llama *de intermediario* o en inglés *man-in-the-middle attack* (literalmente "el ataque del hombre en medio").

Para evitar el ataque de intermediario, Ron Rivest y Adi Shamir [RS84] propusieron el protocolo de interbloqueo (*interlock protocol*) que opera como se describe a continuación y en el que se intercambian, de ser necesario, dos llaves de sesión diferentes:

1. A envía a B su llave pública.
2. B envía a A su llave pública.
3. A toma su llave de sesión, la parte en dos, y cifra la primer mitad utilizando la llave de B. Evía únicamente esa mitad.
4. B toma su propia llave de sesión, la parte a la mitad, y cifra la primera mitad con la llave pública de A, enviando esta mitad a A.
5. Luego de recibir la primer mitad de la llave de sesión de B, A envía la otra mitad de su llave de sesión, también cifrada.
6. B pega las dos mitades que recibió, descifra, y obtiene así la llave de sesión generada por A; entonces envía en respuesta la otra mitad de su llave a A.
7. A pega las dos mitades que recibió, descifra, y recupera la llave de sesión generada por B.

Ahora A y B están de acuerdo en un par de llaves de sesión y pueden convenir usar una de ellas para sus comunicaciones futuras con un algoritmo simétrico.

El protocolo de interbloqueo está basado en el hecho de que no es posible descifrar los fragmentos de mensaje. Si C se hace pasar por A ante B y por B ante A, no puede descifrar ninguno de los mensajes parciales por lo que no puede enviar nada a B a menos que invente su propio mensaje.

El problema principal, como hemos visto, es certificar que la llave pública que se utiliza es realmente la llave de aquel a quien van dirigidos los mensajes. Si logramos eso tenemos gran parte de la batalla ganada, pero eso requiere de mecanismos de autentificación, los que abordaremos un poco más adelante. Si suponemos que ya tenemos un mecanismo de certificación de llaves podemos pensar en otras posibilidades para un sistema mixto. Por ejemplo, el protocolo de Beller-Yacobi [MvOV96], que permite el envío simultáneo de una llave de sesión (para sistema simétrico) y un mensaje cifrado:

1. A genera una llave de sesión K y cifra, con un método simétrico que ha acordado con B, el mensaje M. Esto lo denotaremos[5] con $\mathrm{E}_K(M)$.
2. A obtiene la llave pública certificada de B.
3. A cifra K con la llave pública de B, $\mathrm{E}_B(K)$, y envía a B la pareja $(E_K(M), E_B(K))$.
4. B descifra la parte del mensaje que contiene K ($\mathrm{E}_B(K)$) usando su llave privada y así obtiene K.
5. B usa la K obtenida para descifrar la otra parte del mensaje que contiene M ($\mathrm{E}_K(M)$).

11.3 Autentificación e intercambio de llave

Ya mencionamos que uno de los problemas serios para implementar un sistema mixto es la autentificación: estar seguros siempre de que hablamos con quien creemos estar hablando. Si en un sistema de llave pública todos los usuarios conocen la llave pública del administrador de llaves central y éste firma digitalmente sus mensajes entonces podemos garantizar que las comunicaciones se llevan a cabo con quien realmente deben ser y no con intermediarios. Si un usuario A pide al distribuidor la llave pública de B, con quien desea hablar en secreto, y el distribuidor entrega la llave firmada con su propia llave privada, entonces A puede obtener la llave pública de B y certificar que es el administrador quien se la dió, porque

[5] En general usaremos E para denotar cifrado, ya sea con la llave pública de alguien en un sistema de llave pública o con la llave privada de un sistema simétrico.

nadie podría haber firmado el mensaje de tal forma que la llave pública del administrador verifique la firma. Si un usuario malicioso C se pone en medio haciéndose pasar por el administrador de llaves no tendrá los medios para firmar el mensaje de tal forma que A crea que fue el administrador quien lo firmo. Aún si C solicita al distribuidor la verdadera llave pública de B y pretende pegar la firma de este mensaje a su propio mensaje A se percatará de ello: recordemos que la firma es una función tanto de la llave privada como del mensaje mismo, firma y mensaje están estrechamente ligados, así que no hay tal cosa como una firma "general" añadible a cualquier mensaje.

También podemos pensar en un protocolo de autentificación con llave simétrica. Supongamos que A y B se han puesto de acuerdo en un sistema simétrico y en una llave K para usar con este sistema. A y B pueden autentificarse mutuamente con un protocolo llamado *desafío-respuesta* [Sch96].

1. A genera un número aleatorio r_A y lo envía a B. Esto constituye el reto que A lanza a B.
2. B recibe el mensaje y genera su propio número aleatorio r_B; lo cifra con el algoritmo convenido usando K y envía a A: $E_K(r_A, r_B)$.
3. A recupera r_A y r_B descifrando el mensaje y envía de regreso a B r_B.

A sabe que habla con B porque nadie más podría conocer K; y B sabe que habla con A porque nadie que no sepa K podría haber descifrado el mensaje para conocer r_B. Un atacante que se haga pasar por A o B no puede descifrar los mensajes. Pero hay otro problema, existe un ataque similar al del intermediario llamado *de reflexión* que funcionaría como sigue:

1. A envía a C (el intermediario) r_A creyendo que es B.
2. C envía r_A a A de regreso (reflexión), esto inicia un nuevo protocolo con A.
3. A, en respuesta a este último mensaje (reto) envía un nuevo número aleatorio r'_A cifrado $E_K(r_A, r'_A)$, el r_A que envía es, de hecho el que recibió de C.
4. C recibe este mensaje y lo envía de regreso a A.
5. A supone que esta es la respuesta a su primer protocolo. Descifra y responde con r'_A. Esto termina el segundo protocolo.
6. C responde otra vez con lo mismo que recibe (r_A) con lo que termina el primer protocolo.

Ahora C tiene dos sesiones abiertas con A. Para evitar este ataque debemos forzar a que la contraparte cifre o descifre algo. Para esto necesitamos incluir en el mensaje algo que deba ser conocido y luego cambiado en función de su valor, por ejemplo un número aleatorio que se debe incrementar, un número de secuencia un identificador del remitente o del destinatario, o un sello de tiempo (*time-stamp*). Por ejemplo si en el segundo paso B tiene que enviar $E_K(r_A, r_B, A)$ entonces C ya no puede simplemente reflejar este mensaje porque tendría que reemplazar A por otro identificador, lo que significa que debe descifrar el mensaje.

Otro protocolo de autentificación con establecimiento de llave de sesión es el de Borrows-Abadi-Needham [Sch96, MvOV96], también conocido como el "protocolo de la rana hocicona" (habladora) o en inglés *wide-mouth frog*.

1. A genera K y envía a la autoridad de distribución de llaves (T) un mensaje cifrado con una llave que exclusivamente comparten A y T ($k_{A,T}$). El mensaje es: $A, E_{k_{A,T}}(B, K)$.
2. T descifra el mensaje y le avisa a B que A desea platicar con él con el mensaje: $E_{k_{B,T}}(A, K)$ donde $k_{B,T}$ es una llave que comparten exclusivamente B y T.
3. B descifra el mensaje y a partir de ese momento puede usar un algoritmo simétrico para comunicarse secretamente con A usando K.

La desventaja de este protocolo es que los mensajes viejos resultan útiles para un intruso C. Esto se debe a que C puede retomar algún mensaje previo que vio pasar, por ejemplo $(A, E_{k_{A,T}}(B, K))$, y enviar miles de estos mensajes a T. Por cada uno de estos mensajes T envía uno a B, así que C puede echar a perder el sistema saturando a T e indirectamente a B. Para resolver este problema se coloca un sello de tiempo en cada mensaje, los mensajes caducos se ignoran.

Hemos mencionado ya dos veces los sellos de tiempo como una alternativa para prevenir ataques. Se oye fácil pero no lo es. De hecho es uno de los problemas más importantes de la teoría de la computación distribuida. Incluir sellos de tiempo en los mensajes supone que todas las máquinas tienen la misma hora todo el tiempo, de otra manera ¿cómo garantizar que no se ignoran mensajes válidos? Así que los relojes de todas las máquinas deben estar sincronizados, ese es el problema.

Otro problema que hemos dejado de lado es la generación de números aleatorios, algo que usamos para obtener las llaves de sistema simétrico. Estos números aleatorios deben poseer "alta calidad", algo que no es trivial dado que los generadores de números pseudoaleatorios convencionales son bastante predecibles. Se requiere de recursos especiales para generar números aleatorio con calidad criptográfica, así que sería bueno que no todos los posibles interlocutores tuvieran esa capacidad.

Un protocolo parecido al de la rana hocicona es el de Needham-Schroeder [Sch96] de 1987. En el de la rana A le dice a la autoridad que quiere hablar con B y la autoridad avisa a B de ello; en el protocolo de Needham-Schroeder, A le dice a la autoridad que desea hablar con B y la autoridad le da a A un "pase" para poder hablar con B.

1. A envía a T (la autoridad de distribución) el mensaje: (A, B, r_A).
2. T envía a A: $E_{k_{A,T}}(r_A, B, K, E_{k_{B,T}}(K, A))$.
3. A envía a B: $E_{k_{B,T}}(K, A)$.
4. B envía a A: $E_K(r_B)$.
5. Finalmente A responde a B con $E_K(r_B - 1)$.

Los dos últimos pasos garantizan que no se puede atacar por reflexión: hay un número aleatorio que debe poder ser conocido e incrementado antes de enviar respuesta. A sabe, al final, que está hablando con B porque nadie más podría conocer la llave que B comparte con T, con la que se cifró el mensaje que A envió a B. Por su parte, B sabe que habla con A porque recibió el número aleatorio que envió incrementado. Este protocolo tiene la ventaja de que sólo T debe tener la capacidad de generar buenas llaves de sesión. Lo malo de este protocolo es que si C conoce una llave vieja K, puede usarla para hacerse pasar por A:

1. C envía a B un mensaje que ya ha visto pasar antes y que fue cifrado usando la llave K que conoce: $E_{k_{B,T}}(K, A)$, no conoce $k_{B,T}$ pero no importa porque este mensaje es idéntico a uno que en algún momento envió A a B.
2. B responde a C con: $E_K(r_B)$.
3. Como C conoce K descifra el mensaje y cifra de nuevo para enviar $E_K(r_B - 1)$.

Al final, B termina convencido de que habla con A.

Ante esto Needham y Schroeder modificaron su protocolo y su idea fue la misma que se les ocurrió a Otway y Rees. El protocolo modificado, conocido como de Otway-Rees es el siguiente:

1. A genera un mensaje en el que pone un número de índice (I) y un número aleatorio r_A: $(I, A, B, E_{k_{A,T}}(r_A, I, A, B))$.
2. B genera un nuevo mensaje: $(I, A, B, E_{k_{A,T}}(r_A, I, A, B), E_{k_{B,T}}(r_B, I, A, B))$ que envía a T.

3. El distribuidor T genera la llave de sesión y envía a B: $(I, E_{k_{A,T}}(r_A, K), E_{k_{B,T}}(r_B, K))$.

4. B envía a A el mensaje que recibió de T pero destinado a A: $(I, E_{k_{A,T}}(r_A, K))$.

5. A descifra y puede platicar en secreto con B.

Otra variante del protocolo de Needham-Schoeder es la utilizada por el sistema Kerberos, sólo que en ella se requiere de sellos de tiempo y por tanto de una sincronización aproximada entre las máquinas del sistema.

11.4 SSL

El protocolo SSL (*Secure Sockets Layer*) usado en Internet es de hecho la base de un sistema mixto. Las comunicaciones cifradas entre las máquinas interlocutoras (comúnmente llamadas anfitriones o *hosts* en la terminología de Internet) se llevan a cabo usando un sistema simétrico y para esto es necesario que primero se pongan de acuerdo en una llave.

SSL ha tenido varias versiones. La original fue diseñada por Taher ElGamal en Netscape. En 1997 SSL cambió su nombre por el de TLS (*Transport Layer Security*) y se ha convertido en un borrador de protocolo estándar para Internet (lo que se conoce como *Internet Draft*). El borrador más reciente es de marzo de 2002 y expira en septiembre del mismo año [DR02].

El objetivo de TLS es dar privacía e integridad de datos a dos programas de aplicación que se intercomunican. Se asume una arquitectura de red de computadoras subyacente basada en los protocolos de Internet TCP/IP. De hecho TLS se monta sobre los servicios provistos por el último de estos protocolos (TCP) que hace que la comunicación sea confiable (garantiza la entrega ordenada de mensajes). TLS está constituido de dos protocolos más pequeños: el protocolo de registro (*Record Protocol*) y el protocolo de saludo (*Handshake Protocol*).

El protocolo de registro es el encargado de la transmisión y recepción de mensajes cifrados con métodos simétricos. Este protocolo puede utilizar (o en terminología usual de computación "soporta") varios algoritmos de cifrado, a saber: RC4 (40 y 128 bits de llave), RC2 (40 bits de llave), DES (40 y 56 bits de llave), 3DES (triple DES con 168 bits de llave) e IDEA (128 bits de llave). También se encarga de garantizar la integridad de los datos para lo que usa funciones llamadas "de hash" como MD5 (*Message Digest* que genera 128 bits a partir de un tamaño arbitrario de datos) y SHA (*Secure Hash Algorithm* que genera 160 bits). El protocolo de registro toma el mensaje a enviar, lo fragmenta en bloques, lo comprime (opcionalmente), aplica una de las funciones hash mencionadas, lo cifra con alguno de los métodos simétricos mencionados y lo transmite.

El protocolo de saludo o, literalmente, de "apretón de manos" es el encargado de establecer los parámetros de sesión, entre otras cosas debe establecer la llave del sistema simétrico a usar. El cliente y el servidor se intercambian mensajes de saludo y parámetros criptográficos, que les permiten ponerse de acuerdo en lo que se denomina en el borrador un "secreto pre-maestro" (*pre-master secret*). Luego a partir de esto se genera el secreto maestro y se establece todo lo necesario para que opere el protocolo de registro.

El protocolo de saludo procede como sigue:

1. El cliente (por ejemplo un "browser" de HTML) envía un mensaje de saludo que contiene:
   - La hora.
   - Un número de 28 bytes (224 bits) aleatorio al que llamaremos r_c.
   - La versión del protocolo que corre en la máquina cliente.
   - Un identificador de sesión.
   - Una lista de opciones (ordenada por preferencia) acerca de:
     - Algoritmos de criptografía simétrica soportados (mencionados arriba).
     - Métodos de intercambio de llave. Hay dos posibilidades: RSA y Diffie-Hellman.
     - Métodos de verificación de integridad (funciones hash mencionadas arriba).
2. El servidor responde con un mensaje de saludo que contiene la misma información (con las preferencias del servidor, por supuesto). Al número aleatorio contenido en este mensaje le llamaremos r_s.
3. El servidor envía también un mensaje de certificado que contiene:
   - Un número k que en el futuro será utilizado, según el caso, como la llave pública del servidor (si se decide usar RSA como método de intercambio de llave) o como los valores del módulo p, el generador g, y $Y = g^{\beta} \pmod{p}$ (si se decide usar Diffie-Hellman).
   - El identificador del certificado.
   - Periodo de validez del certificado.
   - Nombre del servidor.
   - Nombre del servicio (e.g. httpd para intercambió de páginas web).
   - Firma del certificado.

4. Ocasionalmente el servidor puede enviar un mensaje más si no le alcanza el anterior para enviar todo lo que se requiere.

5. El cliente envía un mensaje de intercambio de llave que sirve para establecer el secreto pre-maestro. Si se eligió RSA el cliente envía 48 bytes (384 bits) aleatorios cifrados con k que se interpreta como la llave pública del servidor. Si se eligió Diffie-Hellman entonces se envía $X = g^{\alpha} \pmod{p}$, k se interpreta como Y, como se mencionó antes.

6. Tanto el cliente como el servidor envían un mensaje llamado *finish* que consiste en la función hash elegida para verificar integridad, evaluada sobre el conjunto de todos los mensajes previos concatenados.

El secreto pre-maestro consiste en 48 bytes (384 bits). Si como método de intercambio de llave se eligió RSA entonces el secreto pre-maestro son los 48 bytes enviados por el cliente en el paso 5 del protocolo. Si se eligió, en cambio, Diffie-Hellman entonces el secreto pre-maestro es la llave que calculan ambas partes $K = g^{\alpha\,\beta} \pmod{p}$.

Para especificar cómo se calcula el secreto maestro debemos especificar un par de cosas. Sabemos que hay dos funciones hash de verificación de integridad, ambas reciben dos argumentos: un texto al que se le aplica la función y un número usado como semilla de un generador de números pseudoaleatorios, podemos especificar genéricamente esto como:

$$\begin{aligned} &\text{MD5} \quad (\mathit{texto}, \mathit{semilla}) \\ &\text{SHA} \quad (\mathit{texto}, \mathit{semilla}) \end{aligned}$$

con base en estas funciones se define otra, en el borrador de IETF se denomina *función pseudoaleatoria* (en siglas PRF):

$$\begin{aligned} \text{PRF}(\mathit{textolargo}, \mathit{etiqueta}, \mathit{semilla}) \quad = \quad & \text{MD5}(\mathit{textolargo}_{izq}, \mathit{etiqueta} \sqcup \mathit{semilla}) \\ & \oplus \text{SHA}(\mathit{textolargo}_{der}, \mathit{etiqueta} \sqcup \mathit{semilla}) \end{aligned}$$

donde $\sqcup$ denota la yuxtaposición o concatenación de las cadenas de bits que constituyen sus operandos, $\oplus$ denota el XOR (disyunción exclusiva o suma módulo 2), *textolargo* es una cadena binaria del doble de longitud del *texto* aceptado como argumento de las funciones hash, $\mathit{textolargo}_{izq}$ y $\mathit{textolargo}_{der}$ denota las mitades derecha e izquierda de esa cadena de bits.

Con base en esto, el secreto maestro es:

$$\mathit{mastersecret} = \text{PRF}(\mathit{premastersecret}, \text{“}\mathit{master\ secret}\text{”}, r_c \sqcup r_s)$$

el segundo argumento es, de hecho, la cadena de texto "master secret" (incluyendo el espacio entre las palabras). Los valores r_c y r_s son los números aleatorios enviados en los mensajes de saludo por el cliente y el servidor respectivamente.

El secreto maestro puede ser usado ahora como la llave de alguno de los sistemas simétricos, aquel que hayan elegido de acuerdo a sus preferencias el cliente y el servidor.

TLS puede utilizar el protocolo de Diffie-Hellman para el intercambio de llaves. Otro protocolo de uso común hoy en día entre la comunidad de usuarios de computadoras en red es SSH (*Secure shell*), que permite la realización de sesiones de trabajo en computadoras remotas en las que el usuario está registrado. La versión más reciente de este protocolo es la 2 y también usa el protocolo de Diffie-Hellman para el intercambio de la llave, de hecho, a diferencia de TLS en el que puede usar otro mecanismo de llave pública, en SSH el único mecanismo especificado como obligatorio por el estándar es Diffie-Hellman.

En SSH también hay varios sub-protocolos:

- El protocolo de transporte, encargado de autentificación del servidor, confidencialidad y, opcionalmente, compresión de datos.
- El protocolo de autentificación de usuario, se encarga de validar al usuario que pretende entrar en sesión en un sistema remoto.
- El protocolo de conexión, que es el encargado de la transmisión de datos cifrados en algún mecanismo criptográfico simétrico.

El protocolo de transporte es el equivalente al protocolo de saludo de TLS. En el estándar está especificado el número que se utiliza como módulo al ejecutar el protocolo de Diffie-Hellman [YKS+02]:

$$p = 2^{1024} - 2^{960} - 1 + 2^{64}\lfloor 2^{894}\,\pi + 129093 \rfloor.$$

Éste es un número primo de 1024 bits, 309 dígitos decimales. El generador que se utiliza es 2.

El catálogo de sistemas simétricos que especifica el estándar de SSH es [YKS+02]:

- Obligatoriamente 3DES.
- Recomendados: blowfish, twofish 128, AES 128.
- Opcionales: serpent, IDEA, algunos de los anteriores con otros tamaños de llave.

Al igual que TLS, SSH también verifica la integridad de los datos. Para ello usa SHA. Además provee de mecanismos de firmas digitales, para lo que se especifica como obligatorio DSS y como recomendado RSA.

Cuando un usuario trata de hacer una conexión mediante SSH desde su máquina local a alguna otra máquina en la que tiene cuenta, el protocolo de transporte se encarga de validar al servidor. La primera vez que entran en contacto la máquina A y la máquina B ejecutan Diffie-Hellman y se ponen de acuerdo en una llave, esta es almacenada para futuras conexiones de tal forma que la próxima vez que A trate de hablar con B tratará de hacerlo usando la llave en la que ya se habían puesto de acuerdo antes. Si esta llave fue borrada A y B deben volver a ejecutar Diffie-Hellman para establecer una nueva. Una vez establecida la llave de sesión el protocolo de autentificación de usuario entra en acción, se encarga de verificar que el usuario de A es también un usuario legal de B. Una vez hecho esto es el protocolo de conexión el que se encarga del resto de la comunicación cifrada entre A y B.

SSH es susceptible al ataque del intermediario. Si A quiere hablar con B y otro anfitrión C se interpone haciéndose pasar por B, basta con que finja que ha perdido la llave de sesión en la que se habían puesto de acuerdo en el pasado A y B, entonces se procederá a determinar una nueva llave y ya. Cuando un anfitrión se pretende conectar con otro con el que se ha conectado previamente y este dice desconocer la llave de sesión que habían acordado, la versión 2 de SSH informa de esto al usuario que pretende conectarse y le pregunta si realmente desea continuar con la conexión, es decir proceder a la ejecución de Diffie-Hellman para determinar la nueva llave. En las versiones previas del protocolo el usuario no era informado, simplemente se volvía a determinar la llave.

También TLS es susceptible al ataque del intermediario, a menos que tanto el cliente como el servidor sean mutuamente autentificados. Por cierto esta es una opción de TLS, de hecho se puede autentificar solo al servidor o al servidor y al cliente. en este caso debe haber una autoridad de autentificación.

APÉNDICE

A.1 Datos obtenidos por los autores

Los datos estadísticos elaborados por los autores fueron obtenidos con una muestra de 319,958 letras de texto literario y periodístico contemporáneo, en su mayoría escrito con el uso del idioma en México. La tabla siguiente tiene la frecuencia absoluta y porcentual de cada letra, incluyendo las vocales acentuadas y la Ñ.

Letra	Frec.	%	Letra	Frec.	%	Letra	Frec.	%
E	40511	12.67	U	12835	4.01	F	1862	0.58
A	38327	11.98	M	9319	2.91	J	1726	0.54
O	29106	9.10	P	7410	2.32	Z	1446	0.45
S	23297	7.28	B	5536	1.73	Á	1429	0.45
N	21904	6.85	H	3572	1.12	É	1275	0.40
R	20677	6.46	Y	3507	1.10	Ñ	557	0.17
L	18927	5.92	V	3411	1.07	X	324	0.10
I	17836	5.58	Q	3398	1.06	Ú	292	0.09
D	15303	4.78	G	3392	1.06	K	123	0.04
T	13751	4.30	Í	2848	0.89	W	62	0.02
C	13125	4.10	Ó	2761	0.86	Ü	9	0.00

Índice de coincidencias (IC): 0.0694 (contando los 33 símbolos como parte del alfabeto).

Usando un alfabeto de 26 letras (eliminando acentos, diéresis y considerando las Ñ como igual a N) se obtiene:

Letra	Frecuencia	%	Letras	Frecuencia	%
E	41786	13.06	P	7410	2.32
A	39756	12.43	B	5536	1.73
O	31867	9.96	H	3572	1.12
S	23297	7.28	Y	3507	1.10
N	22461	7.02	V	3411	1.07
I	20684	6.47	Q	3398	1.06
R	20677	6.46	G	3392	1.06
L	18927	5.92	F	1862	0.58
D	15303	4.78	J	1726	0.54
T	13751	4.30	Z	1446	0.45
U	13136	4.11	X	324	0.10
C	13125	4.10	K	123	0.04
M	9319	2.91	W	62	0.02

Frecuencia relativa de uso de acentos y diéresis: 0.03%.

Índice de coincidencias (IC): 0.0744 (alfabeto de 26 letras), 0.0741 (alfabeto de 27 letras, añadiendo la ñ)

Si se utiliza un alfabeto de 27 letras entonces la distribución uniforma tiene un índice de coincidencias de $1/27 = 0.037$, en vez de el $1/26 = 0.038$ usual.

Índice de coincidencias (IC) para textos cifrados polialfabéticamente en español (alfabeto de 26 letras)

Alfabetos	IC
1	0.0744
2	0.0546
3	0.0495
4	0.0473
5	0.0439
6	0.0449
7	0.0423
8	0.0419
9	0.0415
10	0.0414
15	0.0395
20	0.0388

A.2 Datos de otras fuentes

En [Lib] se muestran los siguientes datos:

Letra	%	Letra	%	Letra	%
E	13.676	C	4.679	Q	0.875
A	12.529	T	4.629	H	0.704
O	8.684	U	3.934	F	0.694
S	7.980	M	3.150	Z	0.523
R	6.873	P	2.505	J	0.443
N	6.712	B	1.420	X	0.221
I	6.249	G	1.006	W	0.023
D	5.856	Y	0.895	K	0.004
L	4.971	V	0.895		

que se categorizan como sigue:

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

También se dice que los textos de menos de 500 palabras son inadecuados para usar análisis de frecuencias. No indica el tamaño de la muestra de la que se obtuvieron los datos mostrados. El índice de coindicencias con estos datos sería 0.0755 (alfabeto de 26 letras) En [Nic] se usan 60,115 de texto como muestra representativa y se obtienen gran cantidad de datos que se muestran a continuación.

Índice de coincidencias[1] de textos en español: 0.0747.

Frecuencias relativas.

Letra	**%**	**Letra**	**%**	**Letra**	**%**
E	13.0	D	4.5	Y	0.7
A	11.1	L	3.6	H	0.6
O	9.7	U	3.6	Q	0.6
I	8.2	P	3.0	J	0.3
N	8.0	M	2.9	Z	0.3
R	7.7	G	1.4	X	0.2
S	6.9	B	1.3	W	0.1
T	5.3	V	1.0	K	0.0
C	5.2	F	0.8		

[1]Cabe señalar que en la referencia citada el índice de coincidencias tiene un significado ligéramente distinto, es el valor mostrado (0.0747) multiplicado por el numero de letras en el alfabeto. A lo que aquí llamamos índice de coincidencias, en [Nic] se le llama Kappa. Para nosotros en el texto ambos conceptos son el mismo.

Vocales: A, E, I, O, U, Y = 46.3%, 45.6% (sin la Y).

Consonantes de alta frecuencia: N, R, S = 22.6%

Consonantes de frecuencia media: C, D, L, M, P, T = 24.5%

Consonantes de baja frecuencia: B,F,G,H,J,K,Q,V,W,X,Z = 6.6 % 7

Letras más frecuentes: (E, A, O, I, N, R, S) = 64.6%

Letras iniciales de palabra (no se consideraron las palabras de una sola letra). Se consideró un texto de 10,129 letras.

P	1,128	L	435	Q	286	V	183	Y	27
C	1,081	R	425	I	281	F	177	W	19
D	1,012	M	403	H	230	O	169	Z	2
E	989	N	346	U	219	B	124	K	1
S	789	T	298	G	206	J	47	X	0
A	761								

Digramas (sobre una muestra de 60,115 letras) el renglón corresponde a la letra que se encuentra a la izquierda en el digrama, la columna a la de la derecha.

	A	B	C	D	E	F	G	H	I	J	K	L	M
A	12	14	54	64	15	5	8	4	10	8		41	30
B	11				5				14	1		12	
C	39		5		17			8	80			3	
D	32		1	2	84			1	30				
E	20	5	47	26	17	8	21	6	9	3		44	26
F	2				9				12			1	
G	12				12				5			1	
H	15				3				5				
I	43	8	42	29	40	5	8			1		14	16
J	4				5								
K					1								
L	44		5	5	35	1	3		28			9	5
M	32	10			42				30				
N	41	2	33	37	41	10	6	2	28	1		5	4
O	19	17	28	26	16	6	5	5	4	1		22	33
P	30		1		16				5			8	
Q													
R	74	1	12	10	94	1	12		45	1	1	6	15
S	32	2	18	15	57	3	2	4	41	1		5	7
T	60		1		67				35				
U	13	6	11	5	52	1	3		9			9	6
V	12			1	15				15				
W	1				1								
X			1		4								
Y	5	1	3	2	5	1	1					1	1
Z	6		1	1									

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	64	4	24	5	81	62	18	9	9			11	4
B		5			12	2	1	3					
C		69			6		13	18					
D	1	59	2	1	3	1		6				1	
E	126	5	23	4	94	119	17	5	10	1	8	2	3
F		7			4			5					
G	2	15			11		1	11					
H		6						1					
I	50	67	4	1	16	27	24	1	8				5
J		3						3					
K													
L	1	17	5	1	2	4	5	5	3			1	
M		15	10					6					
N	3	43	10	2	4	21	91	12	6			1	1
O	104	4	29	7	58	73	12	3	5		2	9	1
P		31			34	1	3	19					
Q								29					
R	11	43	7	3	10	10	15	9	6			1	1
S	5	22	26	4	6	10	57	23	2			4	
T		56			34			11					
U	34	1	3		9	10	4		1			2	
V		7											
W												1	
X			3				2						
Y	1	5	2	1	1	3	1	1					
Z		3						2					

Índice de coincidencias de digramas = 0.0091,

Digramas que constituyen el 75% de una muestra de 5000 digramas.

Los 15 más frecuentes (constituyen el 25% del total de la muestra).

EN	126	ER	94	DE	84	RA	74	IO	67
ES	119	RE	94	AR	81	OS	73	TE	67
ON	104	NT	91	CI	80	CO	69	AN	64

Las 25 siguientes (constituyen el 25% del total de la muestra).

AD	64	SE	57	IN	50	RO	43	AL	41
AS	62	ST	57	EC	47	NO	43	SI	41
TA	60	TO	56	RI	45	IA	43	NE	41
DO	59	AC	54	EL	44	IC	42	NA	41
OR	58	UE	52	LA	44	ME	42	IE	40

Las restantes dentro del 75%.

CA	39	OM	33	PA	30	NI	28	AP	24	NS	21	EE	17
ND	37	NC	33	AD	30	OC	28	IT	24	EA	20	OB	17
TI	35	DA	32	DI	30	IS	27	EP	23	OA	19	CE	17
LE	35	MA	32	ID	29	EM	26	SU	23	PU	19	ET	17
TR	34	SA	32	QU	29	SP	26	SO	22	SC	18	LO	17
UN	34	PO	31	OP	29	ED	26	OL	22	AT	18		
PR	34	MI	30	LI	28	OD	26	EG	22	CU	18		

Digramas con reverso frecuente.

EN	126	NE	41	CI	80	IC	42	AC	54	CA	39
ES	119	SE	57	AN	64	NA	41	LA	44	AL	41
ON	104	NO	43	AD	64	DA	32	EL	44	LE	35
ER	94	RE	94	AS	62	SA	32	MA	32	AM	30
AR	81	RA	74	OR	58	RO	43				

Digramas con reverso infrecuente.

NT	91	TN	0	ND	37	DN	1
IO	67	OI	4	NC	33	CN	0
ST	57	TS	0				

Letras dobles.

EE	17	RR	10	LL	9	OO	4	DD	2
AA	12	SS	10	CC	5	NN	3		

Digramas iniciales de palabra (sobre una muestra de 10,129 palabras).

CO	684	PR	307	PA	263	SE	189	CA	151	PE	111	MA	101
RE	335	ES	286	PO	247	DI	175	SI	137	UN	109	CU	100
DE	323	QU	286	IN	235	PU	157	MI	117	HA	108	SO	100

Trigramas frecuentes (sobre una muestra de 60,115 letras).

ENT	596	ARA	229	POR	176	OSE	147	ERO	131	NDE	121	PER	111
ION	564	ONE	227	TER	174	ONS	144	ONT	131	RAN	121	ASE	109
CIO	502	ESE	202	ODE	168	REC	144	ANA	130	STE	119	CAN	109
NTE	429	ADE	293	ERE	166	ORE	143	ARE	129	REN	118	UNI	108
CON	415	PAR	190	ERA	165	OCO	142	UNT	127	ARI	117	OSI	107
EST	355	CIA	190	TRA	165	EDE	141	ANO	127	TEN	116	GEN	105
RES	335	ENC	188	AME	165	ICI	140	TAR	126	OND	115	NCO	105
ADO	307	NCI	184	ERI	163	END	139	ANT	126	RIA	115	RIO	105
QUE	294	PRE	183	MER	162	SEN	139	ESA	126	ECI	114	ERN	104
ACI	277	DEL	183	ELA	159	TAD	138	IER	125	IST	113	OMI	104
NTO	270	NDO	183	PRO	158	ECO	135	ADA	125	ONA	113	SCO	104
IEM	267	NES	183	ACO	155	STR	134	DEN	124	DAD	112	TES	103
COM	246	DOS	182	ENE	153	TOS	133	AND	123	INT	112	BIE	101
ICA	242	MEN	181	UES	151	IDA	132	DES	121	NTR	112	NTI	100
STA	240	NTA	176	ESP	149	SDE	132	IDO	121	ESI	111	TOR	100

Tetragramas frecuentes (sobre la misma muestra).

CION	444	CONS	104	ERNO	79	AMER	72	FORM	62	EEST	55
ACIO	252	CONT	99	IERN	78	IEND	72	SENT	62	SCON	55
ENTE	233	PUNT	95	OQUE	78	IDAD	71	ICIO	61	SIDE	55
ESTA	174	ANDO	91	IONA	77	ENDO	70	ONTR	60	CIEN	54
IONE	159	TADO	91	UEST	77	ERIC	70	SION	60	NFOR	54
MENT	150	ACON	90	BIER	76	NTOS	70	CCIO	59	OPOR	54
ONES	146	ANTE	89	ICAN	76	MIEN	69	GENT	58	RESP	54
IENT	141	NTER	85	RESE	76	IOND	67	COMA	57	ARIO	53
ENTO	137	INTE	84	GOBI	75	MERI	67	ESDE	57	ESTR	53
ENCI	128	NTES	82	OBIE	75	NTRA	67	ORES	57	ARGE	51
PARA	117	ADOS	81	ECON	74	DELA	65	RECI	57	ECTO	51
ENTA	115	AMEN	81	RGEN	73	ENTI	64	AQUE	56	PART	51
NCIA	115	OCON	81	RICA	73	NTIN	64	IONP	56	POSI	51
PRES	111	ESEN	80	STAD	73	COMI	63	QUES	56	EPRE	50
UNTO	111	ONDE	80								

Trigramas frecuentes como iniciales de palabra.

CON	298	PAR	154	PUN	93	INT	72	UNI	55	CUA	52
COM	218	PRO	139	PER	80	RES	72	DES	53	TRA	52
EST	194	PRE	114	GOB	66	NUE	66	INF	53	REP	51

Longitud promedio de palabra en español: 5.9 letras.

Palabras de una sola letra: Y(63%), A(32%), O(4%), N(1%), E

Palabras de dos letras: DE, LA, EL, EN, ES, UN, NO, SE, SU, LO, LA, HA, MI, ME, AL, YO.

Palabras de tres letras: QUE, LOS, UNA, POR, DEL, CON, LAS, MAS, SON, SER, UNO, SIN, HAY, MIS, SUS, ESE.

Letras frecuentemente encontradas como iniciales de palabra: C, P, A, S, M, E, D, T, H, V, R, U, N, I, L, B, O, F, Q, G.

Letras frecuentemente encontradas como final de palabra: O, A, S, E, N, R, B, D, L, I, Z.

Peculiaridad: la Q siempre es seguida de UE o UI.

APÉNDICE

En este apéndice vamos a hablar de curvas elípticas, que en años recientes se han vuelto una herramienta importante en criptología; por un lado tenemos la idea de utilizar curvas elípticas como base de criptosistemas, propuesta independientemente por Koblitz y Miller; y por otro lado tenemos el uso de curvas elípticas para resolver problemas asociados a criptosistemas de llave pública, como el algoritmo de factorización de Lenstra, y algoritmos para encontrar logaritmos discretos.

Vamos a empezar con una breve introducción con definiciones y propiedades generales. Después vamos a hablar de algunos criptosistemas que se basan en curvas elípticas. Y finalmente vamos a describir el algoritmo de Lenstra para factorizar usando curvas elípticas.

B.1 Definiciones y propiedades básicas

El tema de curvas elípticas se cubre extensivamente en la literatura. Como dijo Serge Lang, es posible escribir indefinidamente sobre curvas elípticas. Nuestra presentación, por supuesto, será muy breve.

Lo primero que uno nota al iniciar un estudio de curvas elípticas es que no son elipses, y por ello una breve explicación del nombre es apropiada.

Cuando tratamos de calcular la longitud de arco de segmentos de un círculo, obtenemos las funciones trigonométricas sen, cos, y tan. Si hacemos un estudio similar de longitud de arco de segmentos de elipses, entonces terminamos considerando las llamadas *integrales elípticas*, que son integrales de la forma:

$$\int \frac{dx}{\sqrt{4x^3 - g_2 x - g_2}}.$$

Estas integrales toman varios valores sobre los números complejos, y sólo están bien definidas módulo una retícula periódica. Uno puede entonces considerar que los valores que toma una integral elíptica se encuentran en un toro. La función 'inversa' de una integral elíptica es una función doblemente periódica llamada una *función elíptica*. De hecho, cualquier función meromórfica doblemente periódica sobre los números complejos es la inversa de una integral elíptica.

Resulta que toda función doblemente periódica $\mathfrak{P}$ cuyos periodos son independientes sobre $\mathbb{R}$ satisface una ecuación de la forma

$$\mathfrak{P}' = 4\mathfrak{P}^3 - g_2\mathfrak{P} - g_3,$$

para ciertas constantes g_2 y g_3. Una función $\mathfrak{P}$ que satisface estas condiciones se llama una *función $\mathfrak{P}$ de Weiestrass.* Si consideramos la pareja $(\mathfrak{P}, \mathfrak{P}')$ como un punto en el espacio, la ecuación nos da una función del toro a la curva

$$Y^2 = 4X^3 - g_2 X - g_3.$$

Este es un ejemplo de una curva elíptica. El coeficiente de 4 de la X^3 es tradicional entre analistas, pero se puede escalar y eliminar, que es lo que se suele hacer entre algebristas.

Por el resto del apéndice, K es un campo. Siempre vamos a pensar en que K es $\mathbb{R}$, los números reales; $\mathbb{Q}$, los números racionales; $\mathbb{C}$, los números complejos; o bien $\mathbb{F}_q$, el campo finito de q elementos, con $q = p^r$, p un primo, $r > 0$. Pero las definiciones se pueden hacer en general.

Definición B.1 *Sea K un campo de característica distinta de 2 y 3, y sea $x^3 + ax + b$, con $a, b \in K$, un polinomio cúbico sin raíces múltiples. Una* curva elíptica sobre K *es el conjunto de puntos (x, y) con $x, y \in K$ que satisfacen la ecuación*

$$y^2 = x^3 + ax + b, \tag{B.1.1}$$

junto con un elemento, denotado O y llamado el "punto al infinito."

Definición B.2 *Si K es un campo de característica 2, entonces una* curva elíptica sobre K *es el conjunto de puntos qu satisfacen una ecuación de tipo*

$$y^2 + cy = x^3 + ax + b \tag{B.1.2}$$

o bien una ecuación de la forma

$$y^2 + xy = x^3 + ax^2 + b, \tag{B.1.3}$$

y donde no nos importa si el polinomio de grado tres tiene raíces múltiples o no; mas un "punto al infinito" O.

Definición B.3 *Si K es un campo de característica 3, entonces una* curva elíptica sobre K *es el conjunto de puntos que satisfacen la ecuación*

$$y^2 = x^3 + ax^2 + bx + c \tag{B.1.4}$$

mas un "punto al infinito" O, y donde no nos importa si el polinomio de grado tres tiene raíces múltiples o no.

B.1.1 Curvas elípticas sobre los reales

Un hecho centralmente importante acerca el conjunto de puntos de una curva elíptica es que forman un grupo abeliano. Para poder visualizar cómo funciona ésto, por el momento vamos a sumir que $K = \mathbb{R}$, es decir, la curva elíptica es una curva en el plano real usual (más el otro punto O "al infinito"; pensamos en O como un punto infinitamente distante en la dirección vertical).

Definición B.4 *Sea E una curva elíptica sobre los reales, y sean P y Q does puntos de E. Definimos el negativo de P y la suma $P + Q$ de acuerdo a las siguientes reglas:*

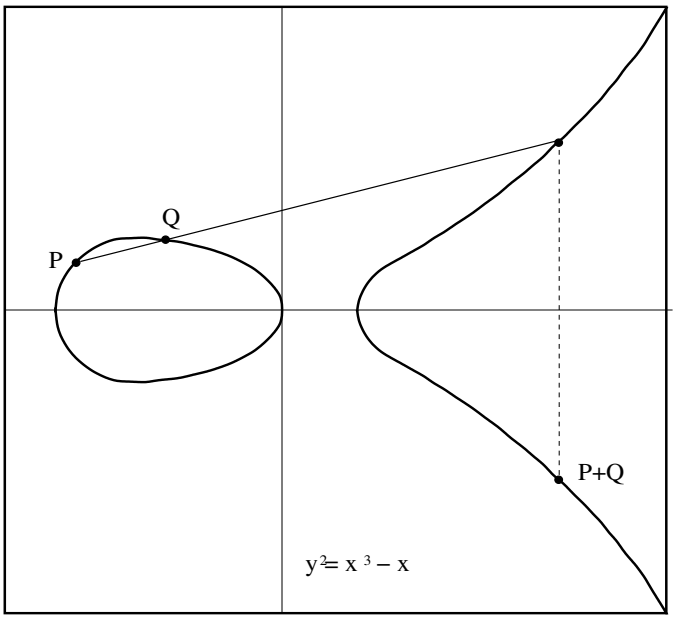

Figura B.1: La suma de los puntos P y Q en la curva elíptica $y^2 = x^3 - x$.

1. *Si P es el punto al infinito O, entonces definimos $-P = O$ y $P + Q = Q$; es decir, O será el cero del grupo. Suponemos ahora que ni P ni Q son iguales al punto al infinito.*

2. *El negativo $-P$ es el punto que tiene la misma coordenada x que P, y que tiene por coordenada y el negativo de la coordenada y de P. Es decir, $-(x, y) = (x, -y)$. De la ecuación B.1.1, es claro que si (x, y) está en E, $(x, -y)$ también lo está.*

3. *Si P y Q tienen distintas coordenadas x, entonces no es dificil verificar que la recta $\ell = \overline{PQ}$ intersecta la curva en exactamente un tercer punto R (a menos que ℓ sea tangente a la curva en P, en cuyo caso tomamos $R = P$; o que ℓ sea tangente a la curva en Q, en cuyo caso tomamos $R = Q$). Entonces definimos $P + Q$ como $-R$.*

4. *Si $Q = -P$ (es decir, Q y P tienen la misma coordenada x, y la coordenada y de Q es menos la de P) entonces definimos $P + Q = O$, el punto al infinito.*

5. *Si $P = Q$, entonces sea ℓ la recta tangente a la curva en el punto P, y sea R el tercer punto de intersección de ℓ con la curva. Definimos $P + Q = -R$.*

Ejemplo B.1 La curva elíptica $y^2 = x^3 - x$ está en la Figura B.1.

La figura presenta el caso típico de una suma de dos puntos P y Q. Para encontrar $P+Q$, trazamos la cuerda que pasa por P y Q, y $P+Q$ es el reflejo a lo largo del eje X del tercer punto de intersección. Si P y Q son el mismo punto, trazamos la tangente en P, y $2P$ es el reflejo a lo largo del eje X del tercer punto de intersección de la tangente con la curva. $\lhd$

Se pueden dar fórmulas para calcular las coordenadas de $P+Q$ en términos de las coordenadas de P y de Q. Sean (x_1, y_1), (x_2, y_2), (x_3, y_3) las coordenadas de P, Q, y $P+Q$, respectivamente. Queremos expresar x_3 y y_3 en términos de x_1, x_2, y_1, y y_2.

Si las coordenadas x de P y Q son distintas, entonces la fórmula, que se puede verificar con simple geometría analítica, es:

$$\begin{aligned} x_3 &= \left(\frac{y_2-y_1}{x_2-x_1}\right)^2 - x_1 - x_2; \\ y_3 &= -y_1 + \left(\frac{y_2-y_1}{x_2-x_1}\right)(x_1 - x_3). \end{aligned} \tag{B.1.5}$$

En el caso en que $P = Q$, usando derivadas implícitas podemos obtener las siguientes fórmulas:

$$\begin{aligned} x_3 &= \left(\frac{3x_1^2+a}{2y_1}\right)^2 - 2x_1; \\ y_3 &= -y_1 + \left(\frac{3x_1^2+a}{2y_1}\right)(x_1 - x_3). \end{aligned} \tag{B.1.6}$$

Con estas fórmulas, no es dificil probar que la suma es asociativa y los puntos de la curva se convierten en un grupo abeliano; se puede probar con geometría proyectiva, o un argumento de análisis complejo, o un argumento de geometría algebráica.

En el caso de geometría proyectiva, podemos considerar el punto al infinito O como el punto en la recta al infinito en la dirección vertical, y todo funciona perfectamente.

B.1.2 Curvas elípticas sobre los complejos

Las fórmulas (B.1.5) y (B.1.6) tienen sentido en cualquier campo (si el campo tiene característica 2 ó 3, uno puede derivar ecuaciones similares a partir de la ecuación general (B.1.2), (B.1.3), o (B.1.4)). Se puede demostrar que las fórmulas dan un grupo abeliano para los puntos de una curva elíptica sobre cualquier campo.

En particular, sea E una curva elíptica definida sobre el campo $\mathbb{C}$ de los números complejos. Entonces E es el conjunto de parejas (x, y) de números complejos que satisfacen una ecuación

$$y^2 = x^3 + ax + b$$

con $a, b \in \mathbb{C}$, más el punto al infinito O. Aunque E es una "curva", si pensamos en términos de nuestra geometría usual, en realidad es de dimensión dos: es una superficie en el espacio real de dimensión cuatro cuyas coordenadas son las partes reales e imaginarias de x y y.

Para ver cómo visualizar la curva como una superficie, sea L un retícula en el plano complejo; es decir, L es el grupo abeliano de todas las combinaciones lineales con coeficientes enteros de dos números complejos ω_1 y ω_2 que no son colineales en el plano complejo. Es decir, $L = \mathbb{Z}\omega_1 + \mathbb{Z}\omega_2$. Por ejemplo, si $\omega_1 = 1$, $\omega_2 = i$, entonces L son los enteros Gaussianos.

Dada una curva elíptica E sobre los números complejos, resulta que existe una retícula L y una función compleja, la función $\mathfrak{P}$ de Weiestrass, denotada $\mathfrak{P}_L(z)$ o $\mathfrak{P}(z)$ si no hay peligro de confusión, que es analítica excepto por un polo doble en cada punto de L, y que satisface una ecuación diferencial de la forma

$$\mathfrak{P}'^2 = \mathfrak{P}^3 + a\mathfrak{P} + b;$$

$\mathfrak{P}(z_1) = \mathfrak{P}(z_2)$ si y sólo si $z_1 - z_2 \in L$ (doblemente periódica); $\mathfrak{P}$ asocia cada $z \notin L$ con el punto $(\mathfrak{P}(z), \mathfrak{P}'(z))$ de E, dando una correspondencia uno a uno de E con $\mathbb{C}/L$; y esta correspondencia es un isomorfismo de grupos abelianos, donde $\mathbb{C}/L$ tiene la estructura de grupo abeliano heredada de la estructura aditiva de $\mathbb{C}$.

Es decir, podemos visaulizar a E como un grupo cuyos elementos son las clases laterales $z + L$; es decir, los complejos de la forma $a\omega_1 + b\omega_2$, con $0 \leq a, b < 1$. Este se llama el "paralelograma fundamental." Luego tenemos que identificar los lados paralelos de la frontera del paralelograma, lo cual no da un toro.

Como grupo, el toro es el producto de dos copias del círculo; es decir, cada punto se puede parametrizar con pares ordenados de ángulos (α, β). Es decir, podemos pensar que una curva elíptica sobre los complejos es una generalización a dos dimensiones del círculo en el plano real.

B.1.3 Curvas elípticas sobre los racionales

Si tomamos una curva elíptica E dada por la ecuación

$$y^2 = x^3 + ax + b$$

con $a, b \in \mathbb{Q}$, es natural buscar soluciones (x, y) con $x, y \in \mathbb{Q}$. Hay una teoría muy grande sobre curvas elípticas sobre los racionales; por ejemplo, notemos que las fórmulas para sumar puntos garantizan que la suma de dos puntos con coordenadas racionales vuelve a ser un racional (o el punto al finfinito).

Se sabe que el grupo abeliano de puntos de una curva elíptica sobre $\mathbb{Q}$ es finitamente generado (el Teorema de Mordell). Esto quiere decir que consiste de un subgrupo finito, llamado el "grupo de torsión" (los elementos de orden finito), y un subgrupo generado por un número finito de puntos de orden infinito; es decir, copias del grupo cíclico infinito. El número de generadores que se necesitan para la parte infinita se llama el *rango* r de la curva; vale cero si y sólo si el grupo completo es finito. El estudio del rango r y otras propiedades del grupo de una curva elíptica sobre $\mathbb{Q}$ están relacionados con muchas preguntas de teoría de números y geomtría algebráica. Por ejemplo, una pregunta de los antiguos griegos:

> "Dado un entero positivo n, ¿existe un triángulo rectángulo tal que la longitud de sus lados son números racionales, y cuya area es n?"

resulta ser equivalente a la pregunta:

> "Dado un entero positivo n, ¿es mayor que cero el rango de la curva elíptica $y^2 = x^3 - n^2x$ sobre $\mathbb{Q}$?"

B.1.4 Puntos de orden finito

El *orden* N de un punto P en una curva elíptica es el menor entero positivo tal que $NP = O$; tal número N puede no existir. En muchos casos, particularmente en el caso de curvas elípticas sobre $\mathbb{Q}$, hay puntos que son de *orden infinito*, para los cuales N no existe.

Ejemplo B.2 Encuentre el orden de $P = (2, 3)$ en la curva $y^2 = x^3 + 1$.

Usando la ecuación (B.1.6), tenemos que $2P = (0, 1)$, y que $4P = 2(2P) = 2(0, 1) = (0, -1)$. Por lo tanto, $4P = -2P$, o $6P = O$. Por lo tanto, el orden de P es 2, 3, ó 6. Ya vimos que $2P = (0, 1) \neq O$, y si P tuviera órden 3, entonces $4P = P$, pero eso no es cierto. De manera que P tiene orden 6. $\lhd$

B.1.5 Curvas elípticas sobre campos finitos

Por el resto de la sección, vamos a suponer que K es el campo finito $\mathbb{F}_q$ de $q = p^r$ elementos. Sea E una curva elíptica definida sobre $\mathbb{F}_q$. Si $p = 2$ ó 3, E está dada por una

ecuación de la forma (B.1.2) ó (B.1.3) en el primer caso, (B.1.4) en el segundo.

Es fácil ver que una curva elíptica tiene a lo mucho $2q + 1$ puntos en $\mathbb{F}_q$, es decir, el punto al infinito y $2q$ parejas (x, y) con $x, y \in \mathbb{F}_q$ que satisfacen la ecuación correspondiente. A saber, para cada uno de los q valores posibles de x, tenemos a lo mucho dos valores de y.

Pero como sólo la mitad de los elementos de F_q^* tienen raíz cuadrada, uno esperaría (si $x^3 + ax + b$ es un elemento aleatorio del campo) que sólo haya alrededor de la mitad de esos puntos en $\mathbb{F}_q$. En efecto, salvo por un factor de posible error, tenemos el siguiente resultado:

Teorema B.1 (Teorema de Hasse) *Sea N el número de puntos definidos sobre $\mathbb{F}_q$ de una curva elíptica. Entonces*

$$|N - (q + 1)| \leq 2\sqrt{q}.$$

Además de saber cuántos puntos N tiene la curva, también nos gustaría saber la estructura del grupo abeliano. El grupo abeliano no tiene que ser cíclico, pero se puede demostrar que siempre es el producto de a lo mucho dos grupos cíclico. Es decir, es isomorfo al producto de grupos p-primarios, cada uno de la forma forma

$$\frac{\mathbb{Z}}{p^\alpha \mathbb{Z}} \times \frac{\mathbb{Z}}{p^\beta \mathbb{Z}}$$

con $\alpha \geq 1$, $\beta \geq 0$. El *tipo* del grupo de puntos de E sobre $\mathbb{F}_q$ es una lista $(\ldots, p^\alpha, p^\beta, \ldots)_{p|N}$ de los ordenes de los factores cíclicos p-primarios (omitimos p^β cuando $\beta = 0$), tomada sobre todos los primos p que dividen N.

B.1.6 Extensiones de campos finitos y las conjeturas de Weil

Si tenemos una curva elíptica E definida sobre $\mathbb{F}_q$ (es decir, los coeficientes de la ecuación están en $\mathbb{F}_q$), entonces también está definida sobre $\mathbb{F}_{q^r}$ para $r = 1, 2, \ldots$, y por lo tanto tiene sentido hablar de los puntos con coordenadas en $\mathbb{F}_{q^r}$, es decir, soluciones a la ecuación es éstas extensiones del campo. Si empezamos con $\mathbb{F}_q$ como nuestro campo sobre el cual E está definida, sea N_r el número de puntos con coordenadas en F_{q^r} de E (de manera que $N_1 = N$).

Usando los números N_r, uno puede definir una "serie generadora" $Z(T;\ E/\mathbb{F}_q)$, que es una serie formal en $\mathbb{Q}[[T]]$, dada por

$$Z(T;\ E/\mathbb{F}_q) = e^{\sum N_r T^r / r}, \tag{B.1.7}$$

en la cual T es la variable, y la notación $E/\mathbb{F}_q$ designa la curva elíptica y el campo que estamos tomando como campo inicial, y la suma esta tomada sobre toda $r = 1, 2, \ldots$ Se puede probar que la sere tiene coeficientes que son enteros positivos, y es llamada la *función zeta de la curva elíptica E (sobre $\mathbb{F}_q$).* Es un objeto muy importante asociado a E.

Las "Conjeturas de Weil" (que ya son un Teorema de P. Deligne) dicen en un contexto más general que la función zeta tiene una forma muy especial. En el caso de una curva elíptica $E/\mathbb{F}_q$, Weil probó lo siguiente:

Teorema B.2 (Conjetura (Teorema) de Weil para curvas elípticas) *La función Z es una función racional de T de la forma*

$$Z(T;\ E/\mathbb{F}_q) = \frac{1 - aT + qT^2}{(1-T)(1-qT)}, \tag{B.1.8}$$

donde sólo el entero a depende de la curva elíptica particular E. El valor de a está relacionado a $N = N_1$ de la siguiente forma: $N = q + 1 - a$. Además, el discriminante del polinomio cuadrático en el numerador es negativo (i.e., $a^2 < 4q$, que es el Teorema de Hasse), y por lo tanto el polinomio cuadrático tiene dos raíces conjugadas complejas α y β, ambos de valor absoluto $\sqrt{q}$; más precisamente, $1/\alpha$ y $1/\beta$ son las raíces, y α, β son las "raíces recíprocas."

Hay muchas analogías entre el grupo de puntos de una curva elíptica definidos sobre $\mathbb{F}_q$ y el grupo multiplicativeo $(\mathbb{F}_q)^*$. Por ejemplo, tienen aproximadamente el mismo número de elementos, por el Teorema de Hasse. Pero el grupo de puntos de una curva elíptica tiene una ventaja que explica su utilidad para criptografía: para una misma q (relativamente grande), hay muchas curvas elípticas de donde escoger, y muchas N correspondientes. Las curvas elípticas dan entonces una gran cantidad de grupos abelianos finitos "naturales."

B.2 Criptosistemas de curvas elípticas

Ya vimos cómo usar el grupo abeliano finito $\mathbb{F}_q^*$ para crear criptosistemas de llave pública. Para ser más precisos, vimos que resolver el problema del logaritmo discreto en campos finitos es dificil, y eso nos llevó a considerar ciertos criptosistemas. Vamos ahora a construir criptosistemas de llave pública análogos, usando ahora el grupo finito abeliano de puntos de una curva elíptica E definida sobre $\mathbb{F}_q$.

B.2.1 Múltiplos de puntos

El análogo a multipicar dos elementos de $\mathbb{F}_q^*$ para una curva elíptica E es *sumar* dos puntos de E. Entonces, el análogo a tomar $\alpha \in \mathbb{F}_q^*$ y tomar α^k sería tomar un punto $P \in E$ y tomar kP, P sumado consigo mismo k veces. Para elevar un elemento a la k-ésima potencia en un campo finito podemos usar el método de elevar al cuadrado y multiplicar, lo cual nos da un algoritmo de $O(\log(k)\log^3(q))$ operaciones de bits. De manera similar, calcular $kP \in E$ se puede hacer en $O(\log(k)\log^3(q))$ con un método semejante, de "duplica y suma."

Por ejemplo, para calcular $100P$, podemos calcular

$$100P = 2(2(P + 2(2(2(P + 2P))))).$$

B.2.2 Textos como puntos

Para poder usar curvas elípticas como criptosistemas, tenemos que poder codificar nuestros textos como si fueran puntos en una curva elíptica dada E sobre un campo finito $\mathbb{F}_q$. Queremos hacerlo de alguna manera sistemática, para que dado un texto m (que podemos pensar se trata de un entero en un rango conocido, como lo hacíamos con los criptosistemas de llave pública), sea posible determinar cuáles son las coordenadas del punto P_m correspondiente. Nótese que ésta codificación *no es* un cifrado; después vamos a hablar de maneras mediante las cuales podemos cifrar el punto P_m. Pero necesitamos alguna manera en la cual un usuario autorizado pueda recuperar m una vez de descifra el texto y obtiene el punto P_m.

Hay dos cosas que hay que decir. En primer lugar, no se conoce ningún algoritmo de tiempo polinomial (en $\log(q)$) que sea determinístico para encontrar un número grande de puntos en una curva elíptica arbitraria E sobre $\mathbb{F}_q$. Sin embargo, hay algoritmos probabilísticos para los cuáles la probabilidad de fracaso es muy pequeña. En segundo lugar, no basta con generar muchos puntos aleatorios de E: para poder codificar un gran número de posibles mensajes m, necesitamos una manera sistemática de generar puntos que esté relacionada de alguna manera a m, por ejemplo, tales que la coordenada x esté relacionada de alguna manera sencilla con el entero m.

Damos un ejemplo de un algoritmo probabilístico para codificar textos como puntos en una curva elíptica E definida sobre $\mathbb{F}_q$. Sea κ un entero suficientemente grande para que estemos satisfechos si la probabilidad de fracaso del algoritmo es 1 en 2^κ; normalmente $\kappa = 30$ basta, $\kappa = 50$ en casos extremos. Supongamos que nuestras unidades de mensaje m son enteros $0 \leq m < M$. Y suponemos que nuestro campo finito $\mathbb{F}_q$ se elige de tal manera

que $q > M\kappa$. Escribimos los enteros de 1 a $M\kappa$ en la forma $m\kappa + j$, con $1 \leq j \leq \kappa$, y elegimos una biyección entre estos números y un conjunto de elementos de $\mathbb{F}_q$. Por ejemplo, escribimos el entero como un entero de r dígitos en base p, y tomamos los r dígitos, considerados como elementos de $\mathbb{Z}_p$, como los coeficientes de un polinomio de grado $r-1$ que corresponde a elementos de $\mathbb{F}_q$ (recuerde que $\mathbb{F}_q$ se puede ver como el cociente de $\mathbb{Z}_p[x]$ por un polinomio irreducible de grado r).

Entonces, dado un entero m, para cada $j = 1, 2, \ldots, \kappa$ obtenemos un elemento $x \in \mathbb{F}_q$ que corresponde a $m\kappa + j$. Dada x, calculamos el lado derecho de la ecuación

$$y^2 = f(x) = x^3 + ax + b$$

y buscamos una raíz cuadrada de $f(x)$. Si encontramos tal y, con $y^2 = f(x)$, tomamos $P_m = (x, y)$. Si $f(x)$ no tiene raíz cuadrada, entonces sustituímos j por $j+1$ y volvemos a tratar de encontrar la x y y correspondiente. Siempre que podamos encontrar una x tal que $f(x)$ es un cuadrado antes de que j crezca más allá de κ, podemos recuperar m del punto (x, y) mediante la fórmula $m = \lfloor (\tilde{x} - 1)/\kappa \rfloor$, donde $\tilde{x}$ es el entero que corresponde a x bajo la correspondencia con elementos de $\mathbb{F}_q$. Como $f(x)$ es un cuadrado para aproximadamente la mitad de los posibles valores de x, la probabilidad de que el método falle es $2^{-\kappa}$.

B.2.3 Logaritmo discreto en E

Definición B.5 *Si E es una curva elíptica sobre $\mathbb{F}_q$, y $\mathcal{B}$ es un punto de E, entonces el Problema del Logaritmo Discreto en E (respecto a la base $\mathcal{B}$) es el problema de, dado un punto $P \in E$, encontrar un entero $x \in \mathbb{Z}$ tal que $x\mathcal{B} = P$ si tal entero x existe.*

Es probable que el Problema del Logaritmo Discreto en curvas elípticas sea más dificil que el Problema del Logaritmo Discreto en campos finitos. Las técnicas más poderosas para campos finitos, el cálculo de índices, no parece tener un buen análogo para curvas elípticas; en particular, la relativa facilidad del Problema del Logaritmo Discreto para campos de característica 2 no parece aplicarse para curvas elípticas definidas sobre $\mathbb{F}_{2^r}$. Es decir, si bien para tener una garantía de seguridad para logaritmo dsicreto en $\mathbb{F}_{2^r}$ se requiere de una r relativamente grande, parece ser que el problema análogo para curvas elípticas requiere de una r sustancialmente más pequeña para alcanzar el mismo grado de seguridad.

Hasta 1990, los únicos algoritmos que se conocían para calcular logaritmo discreto en una curva elíptica eran aquellos que funcionan en cualquier grupo abeliano. Estos algoritmos son de tiempo exponencial cuando el orden del grupo es divisible por algún primo grando. Pero en 1990 Menezes, Okamoto, y Vanstone dieron un nuevo ataque al problema para

curvas elípticas definidas sobre $\mathbb{F}_q$. Usaron el aparamiento de Weil[1] para inyectar el grupo E al grupo multiplicative de una extensión $\mathbb{F}_{q^k}$ de $\mathbb{F}_q$. Esta inyección reduce el Problema de Logaritmo Discreto de E al Problema del Logaritmo Discreto para $\mathbb{F}^*_{q^k}$. Nótese que la complejidad aumenta al pasar de $\mathbb{F}_q$ a $\mathbb{F}_{q^k}$.

Sin embargo, para que la reducción mediante el apareamiento de Weil ayude, es necesario que el grado de la extensión, es decir k, sea pequeño. Pero las únicas curvas elípticas para las cuales k es pequeño son las curvas llamadas *"supersingulares."* Los ejemplos más familiares de ellas son curvas de la forma $y^2 = x^3 + ax$ cuando la característica p de $\mathbb{F}_q$ satisface $p \equiv -1 \pmod 4$; y las curvas de la forma $xy^2 = x^3 + b$ con $p \equiv -1 \pmod 3$. La gran mayoría de las curvas elípticas, sin embargo, **no** son supersingulares, y para ellas la reducción de Menezes, Okamoto, y Vanstone, casi nunca llevan a un algoritmo subexponencial.

Entonces, si evitamos las curvas supersingulares (lo cuál es fácil de hacer), no se conoce ningún algoritmo subexponencial que resuelva el Problema de Logaritmo Discreto para curvas elípticas.

B.2.4 Diffie-Hellman en curvas elípticas

Si A y B quieren intercambiar llaves para usarlas en un criptosistema simétrico, primero eligen públicamente un campo finito $\mathbb{F}_q$, y una curva elíptica E definida sobre $\mathbb{F}_q$. Su llave va a ser construída a partir de un punto aleatorio P de la curva elíptica. Por ejemplo, si tienen un punto aleatorio $P \in E$, entonces pueden tomar la coordenada x de P para obtener un elemento aleatorio de $\mathbb{F}_q$, que luego se convierte en un entero de r dígitos base p (donde $q = p^r$) que sirve como llave. El objetivo de A y B es elegir un punto P de tal manera que su comunicación para elegirlo es pública, pero sólo ellos dos saben cuál es el punto P.

Primero escogen públicamente un punto $\mathcal{B} \in E$ que sirva de "base." El punto $\mathcal{B}$ juega el mismo papel que el generador g en el intercambio de llaves de Diffie-Hellman sobre un campo finito. Sin embargo, no vamos a insistir que $\mathcal{B}$ sea un generador del grupo de puntos de E; después de todo, es posible que el grupo de puntos de E no sea cíclico, y por ende no tenga generador. Pero queremos que el subgrupo que genera $\mathcal{B}$ sea relativamente grande, de preferencia del mismo órden de magnitud que el grupo de puntos de E. Este problema lo vamos a discutir luego; por lo pronto, supongamos que ya fijamos de manera pública el punto $\mathcal{B}$, cuyo orden es relativamente grande (ya sea N mismo, o un divisor grande de N).

Para generar la llave, primero A elige un entero arbitrario a del mismo orden de magnitud

[1] El aparamiento de Weil es una función bilinear, alternante, no degenerada que va de parejas de puntos de la curva E cuyo orden divide un entero m a las raíces m-ésimas de la unidad en $\overline{\mathbb{F}_q}$, la cerradura algebraica de $\mathbb{F}_q$. Véase el Capítulo III, Sección 5 de [BSS99].

que q (que es aproximadamente el mismo que N, según el Teorema de Hasse), y lo mantiene secreto. Luego calcula $a\mathcal{B} \in E$, y lo hace público. La otra persona, B, hace lo mismo: elige un entero b aleatorio, que mantiene secreto, y calcula $b\mathcal{B} \in E$ y lo hace público. La llave secreta que van a usar es $P = ab\mathcal{B} \in E$. Ambos la pueden calcular: A conoce $b\mathcal{B}$ (que es público), y a (que es su secreto), y entonces puede calcular $a(b\mathcal{B}) = ab\mathcal{B} \in E$. De manera simétrica, B conoce $a\mathcal{B}$ y b, y puede calcular $b(a\mathcal{B}) = ab\mathcal{B}$. Pero un criptoanalista conoce $\mathcal{B}$, $a\mathcal{B}$, y $b\mathcal{B}$. Hasta donde sabemos, sin resolver el problema del logaritmo discreto no tiene manera de calcular $ab\mathcal{B}$.

B.2.5 Massey-Omura en curvas elípticas

Como en el caso de campos finitos, este criptosistema se usa para transmitir un mensaje m, que asumimos que ya tenemos codificados como un punto P_m en alguna curva elíptica E que está fija, y que se conoce públicamente. También asumimos que N, el número de puntos de E, se ha calculado y se conoce públicamente.

Cada usuario del sistema elige secretamente un entero e entre 1 y N tal que $(e, N) = 1$, y usa el algoritmo de Euclides para calcular su inverso, $d \equiv e^{-1} \pmod{N}$, es decir, el entero d, $1 \leq d < N$ tal que $de \equiv 1 \pmod{N}$. Si A le quiere enviar el mensaje P_m a B, primero le envía el punto $e_A P_m$ (donde el subíndice A denota que se trata de la llave de A). Esto no le dice nada a B, que no conoce d_A, e_A, ni P_m. Sin tratar de entender el mensaje, B multiplica por su llave e_B y le manda $e_B e_A P_m$ de regreso a A. Entonces A multiplica por d_A y envía $d_A e_B e_A P_m$ de regreso a B. Como $NP_m = O$, y $d_A e_A \equiv 1 \pmod{N}$, el resultado de esta operación es envíar $e_B P_m$ a B. Entonces B multiplica por d_B y obtiene $d_B e_B P_m = P_m$, y el mensaje que se buscaba.

El criptoanalista conoce entonces $e_A P_m$, $e_B e_A P_m$, y $e_B P_m$. Si pudiera resolver el problema del logaritmo discreto, podría recuperar el valor de e_B de los primeros dos puntos, y usarlo para calcular d_B; luego le aplica d_B al último punto para obtener P_m. Pero el Problema del Logaritmo Discreto es intratable en esta situación.

B.2.6 ElGamal en curvas elípticas

Aquí también estamos tratando de transmitir un mensaje ya codificado como un punto P_m de una curva elíptica E definida sobre un campo $\mathbb{F}_q$. En este caso, tanto $\mathbb{F}_q$ como E son públicos, lo mismo que el punto base $\mathcal{B} \in E$ (para ElGamal no necesitamos saber el número N de puntos de la curva). Cada usuario elige un entero aleatorio a_i, que mantiene secreto, y calcula y publica el valor de $a_i\mathcal{B}$, su llave pública.

Para mandar el mensaje P_m a B, el usuario A elige un entero aleatorio k y envía el par de puntos $(k\mathcal{B}, P_m + k(a_B\mathcal{B}))$. Para leer el mensaje, B multiplica el primer punto de la pareja por su llave secreta, a_B, y obtiene $a_B k\mathcal{B}$. Luego le resta ese valor al segundo punto de la pareja, y obtiene

$$P_m + ka_B\mathcal{B} - ka\mathcal{B}K = P_m,$$

el mensaje. El criptoanalista que escucha la transmisión necesita resolver el problema del logaritmo discreto para obtener el valor de k, y con ello encontrar P_m.

B.2.7 La elección de la curva y del punto

Hay varias maneras de elegir la curva elíptica E que se va a usar, y para el caso de Diffie-Hellman y ElGamal, el punto $\mathcal{B}$ que servirá de base.

Elección aleatoria

Una vez que hemos elegido el campo finito $\mathbb{F}_q$, con q grande, podemos elegir tanto E como $\mathcal{B} = (x, y) \in E$ al mismo tiempo de la siguiente manera[2]: Sean x, y, a tres elementos arbitrarios de $\mathbb{F} + q$. Luego definimos $b = y^2 - (x^3 + ax)$. Verificamos que la cúbica $x^3 + ax + b$ no tenga raíces múltiples, que es equivalente a verificar que el discriminante $4a^3 + 27b^2 \neq 0$ en $\mathbb{F}_q$; si el discriminante es cero, hacemos una nueva elección aleatoria de x, y, a. Si el discriminante es distinto de cero, tomamos $\mathcal{B} = (x, y)$ y E la curva dada por $y^2 = x^3 + ax + b$.

Si necesitamos saber el valor de N, el número de puntos de E, hay varias técnicas para calcularlo. El primer algoritmo de tiempo polinomial para calcular $\#E$ fue descubierto por René Shoof, y es determinístico. Consiste en contrar el valor de $\#E$ módulo ℓ para todo primo ℓ menor o igual a cierta cota. Esto se puede hacer examinando la acción del mapa de Frobenius en los puntos de orden ℓ de E; el mapa de Frobenus manda a cada elemento $x \in \mathbb{F}_q$ a x^p. El algoritmo es suficientemente rápido para que si q es un entero de entre 50 y 100 dígitos decimales, el algoritmo se puede aplicar.

Vale la pena mencionar que si bien no es necesario conocer el valor de N para el intercambio de Diffie-Hellman o para el sistema de ElGamal, muchas veces uno quiere tener una buena medida de la seguridad del sistema, y ésta depende de que N tenga un factor primo grande (para que los algoritmos generales de logaritmo discreto no sean aplicables en tiempo subexponencial).

[2] Asumimos que la característica del campo es mayor a 3; uno hace la obvia modificación al método si $q = 2^r$ ó $q = 3^r$, usando la ecuación correcta.

Reducción de un punto global a un punto módulo p

Una segunda manera de elegir la curva E y la base $\mathcal{B} \in E$ consiste en primero escoger una curva elíptica "global" y un punto de orden infinito en ella, y reducir. Es decir, primero escogemos una curva elíptica E definida sobre los racionales (o incluso, sobre un campo numérico), y escogemos a $\mathcal{B} \in E$ que sea un punto de orden ifinito.

Por ejemplo, el punto $\mathcal{B} = (0, 0)$ es un punto de orden infinito en la curva elíptica $y^2 + y = x^3 - x$, y genera el grupo de puntos racionales de la curva. También el punto $\mathcal{B} = (0, 0)$ es un punto de orden infinito en la curva elíptica $Y^2 + y = x^3 + x$, ambas definidas sobre $\mathbb{Q}$.

Después de elegir nuestra curva E y punto $\mathcal{B}$ sobre $\mathbb{Q}$, elegimos un primo p grande, y consideramos la *reducción* de E y $\mathcal{B}$ módulo p. Es decir, en general, los coeficientes de E no son divisibles entre p, así que podemos considerar la ecuación de E definida módulo p en vez de sobre $\mathbb{Q}$. Si hacemos luego un cambio de variables, podemos tomar la ecuación resultante sobre $\mathbb{F}_q$ y expresarla en la forma usual $y^2 = x^3 + ax + b$; excepto para primos pequeños, la cúbica del lado derecho no tiene raíces múltiples, de manera que obtenemos una curva elíptica sobre $\mathbb{F}_p$, a la que llamamos $E \bmod p$. Las coordenadas de $\mathcal{B}$ también se pueden reducir módulo p para obtener un punto, al que llamamos $\mathcal{B} \bmod p$, en la curva $E \bmod p$.

Para usar este método, primero elegimos E y $\mathcal{B}$, y las mantenemos fijas, y obtenemos varias distintas posiblidades de curvas y puntos $(E \bmod p, \mathcal{B} \bmod p)$ variando el primo p.

B.2.8 Orden del punto $\mathcal{B}$

¿Cuál es la probabilidad de que un punto "aleatorio" $\mathcal{B}$ de una curva elíptica "aleatoria" E sea un generador o tenga orden relativamente grande? O, en el caso del segundo método para elección descrito arriba, ¿cuál es la probabilidad de que, cuando p varía, el punto $\mathcal{B} \bmod p$ sea un generador o tenga orden relativamente grande en $E \bmod p$? Ésta última pregunta está muy relacionada con la siguiente pregunta sobre grupos multiplicativos de campos finitos: Dado un entero b, ¿cuál es la probabilidad de que, a medida que varía p, b sea un generador de $\mathbb{F}_p^*$?

Una manera de garantizar que $\mathcal{B}$ es adecuada para nuestros usos (y de hecho, que genera el grupo de puntos de la curva elíptica) es elegir la curva elíptica y el campo finito de tal manera que el número de puntos N de la curva sea un número primo. Si hacemos eso, entonces cualquier punto $\mathcal{B} \neq O$ es un generador. Si usamos el primer método descrito arriba para elegir $(E, \mathcal{B})$, buscamos hasta encontrar uno donde el número de puntos de E

sea un primo. Si usamos el segundo método, entonces dado un punto $\mathcal{B}$ fijo sobre una curva elíptica global fija E definida sobre $\mathbb{Q}$, escogemos primos p hasta encontrar uno donde el número de puntos de E mod p sea un primo. ¿Qué tanto vamos a tener que esperar hasta encontrar una pareja que satisfaga las condiciones?

La pregunta de cuánto tenemos que esperar está relacionada con la siguiente pregunta sobre los grupos $\mathbb{F}_p^*$: ¿Es $(p-1)/2$ un primo? Es decir, ¿es cierto que cualquier elemento de $\mathbb{F}_p^*$ que no sea 1 ni -1 es un generador, o el cuadrado de un generador, de $\mathbb{F}_p^*$? Ni la respuesta para el caso de curvas elípticas, ni las respuesta para el caso de campos finitos se conoce de manera definitiva. Pero se conjetura que en ambos casos la probabilidad de que un p elegido tenga la propiedad deseada es aproximadamente $O(1/\log(p))$.

Nótese que para que el orden de E mod p tenga la posiblidad de ser un número primo N para $p \gg 0$, debemos elegir E de tal manera que tenga torsión trivial; es decir, que el grupo de puntos no tenga ningún punto distinto de O de orden finito. De lo contrario, N siempre es divisible entre el orden del subgrupo de torsión de E.

B.3 Factorización de enteros por curvas elípticas

La principal razón por la que ha habido un aumento en el interés de criptografos y criptoanalistas sobre curvas elípticas es el ingenioso uso que les dió H.W. Lenstra, Jr. para encontrar un nuevo método de factorización que es, en mucho sentidos, mejor que los que se conocían antes, aunque la mejora en eficiencia no es lo suficientemente significativa para convertirse en un peligro para los criptosistemas cuya seguridad se basa en la dificultad de factorizar.

En esta sección vamos a describir el método de Lenstra. La base principal es el artículo original de Lenstra mismo [Len87] y la presentación de Koblitz en [Kob94].

B.3.1 Reducción módulo n de curvas elípticas

Por el resto de ésta sección, sea n un entero impar compuesto, y sea p un factor (desconocido) de n. Vamos a suponer que $p > 3$ (podemos verificar el caso de $p = 3$ directamente).

Para cualquier entero m, y cualesquiera dos racionales x_1 y x_2 cuyos denominadores son primos relativos a m, escribimos $x_1 \equiv x_2 \pmod{m}$ si y sólo si $x_1 - x_2$, escrito en mínimos términos, es un racional cuyo numerador es divisible entre m. Para cualquier racional x_1 cuyo denominador es primo relativo a m, existe un único entero x_2, llamado el *mínimo residuo no negativo*, $0 \leq x_2 \leq m-1$, tal que $x_1 \equiv x_2 \pmod{m}$. A veces escribimos

$x_1 \bmod m$ para representar este mínimo residuo no negativo.

Supongamos que tenemos una ecuación de la forma $y^2 = x^3 + ax + b$, con $a, c \in \mathbb{Z}$, y un punto $P = (x, y)$ que la satisface. En la práctica, la curva E y el punto P son generados de alguna manera "aleatoria", por ejemplo, eligiendo tres enteros aleatorios x, y, z y luego calculando $b = y^2 - x^3 - ax$. Suponemos que la cúbica tiene tres raíces distintas (i.e. $4a^3 + 27b^2 \neq 0$); esto es cierto casi siempre con coeficientes elegidos aleatoriamente. Por simplicidad, asumimos que el discriminantes $4a^3 + 27b^2$ es primo relativo con n (si no lo es, podemos obtener un factor propio de n usando el algoritmo de Euclides; a menos que $n|4a^3 + 27b^2$, en cuyo caso elegimos otra curva); esto garantiza que $x^3 + ax + b$ no tiene raíces múltiples módulo p para cualquier divisor p de n.

Consideramos el punto P y todos sus múltiplos módulo n. Esto quiere decir, tomamos $P \bmod n = (x \bmod n, y \bmod n)$ y, cada vez que calculamos kP, calculamos la reducción de las coordenadas módulo n. Para poder trabajar módulo n, hay una condición no trivial que debe ser cierta cuando tratamos de sumar dos puntos o de sumar un punto consigo mismo. A saber, todos los denominadores que aparecen tienen que ser primos relativos a n.

Proposición B.1 *Sea E una curva elíptica sobre $\mathbb{Q}$ con ecuación $y^2 = x^3 + ax + b$, donde $a, b \in \mathbb{Z}$ y $(4a^4 + 27b^2, n) = 1$. Sean P_1 y P_2 dos puntos de E cuyas coordenadas tienen denominadores que son primos relativos con n, y donde $P_1 \neq -P_2$. Entonces $P_1 + P_2 \in E$ tiene coordenadas cuyos denominadores son primos relativos con n si y sólo si no existe ningún primo p, $p|n$, con la siguiente propiedad: los puntos $P_1 \bmod p$ y $P_2 \bmod p$ en la curva elíptica $E \bmod p$ tienen como suma el punto $O \bmod p \in E \bmod p$.*

B.3.2 El método de Lenstra

Nos dan un entero impar compuesto n, y queremos encontrar un factor no trivial d, $1 < d < n$, $d|n$. Empezamos tomando una curva elíptica E: $y^3 = x^3 + ax + b$ con coeficientes enteros, y un punto $P = (x, y)$ en la curva. La pareja (E, P) es generada de alguna manera aleatoria, pero necesitamos un método que permita generar muchas parejas como ésta. Tratamos de usar E y P para factorizar n, de la manera que explicaremos en un momento; si nuestro intento falla, tomamos otro par (E, P) y continuamos hasta encontrar un factor $d|n$. Si la probabilidad de fracaso es $\rho < 1$, entonces la probabilidad de que h intentos sucesivos fallen es ρ^h, que es muy pequeño si h es muy grande. Entonces con una probabilidad muy alta vamos a lograr factorizar n en un número "razonable" de intentos.

Una vez que tenemos una pareja (E, P), elegimos un entero k que es B-homogéneo para alguna cota B, y tales que las potencias de primos que lo dividen son menores que alguna

segunda cota C. Es decir, tomamos

$$k = \prod_{\ell \leq B} \ell^{\alpha_\ell} \tag{B.3.9}$$

donde $\alpha_\ell = \lfloor \log(C)/\log(\ell) \rfloor$ es el máximo exponente tal que $\ell^{\alpha_\ell} \leq C$.

Después tratamos de calcular kP, trabajando todo el tiempo módulo n. El cálculo no sirve de gran cosa a menos que nos encontremos con una dificultad. Esto se debe a que según la Proposición B.1, esto occure cuando tenemos un múltiplo $k_1 P$ (una suma parcial que calculamos mientras estamos calculando kP) tal que para algún $p|n$, $k_1(P \bmod p) = O \bmod p$; es decir, el punto P mod p del grupo E mod p tiene orden que divide a k_1. Cuando tratamos de usar el algoritmo de Euclides para encontrar el inverso módulo n de un denominador divisible entre p, encontramos el máximo común divisor de n y el denominador. Este máximo común divisor es un divisor propio de n, a menos que n mismo sea el máximo común divisor; es decir, a menos que n divida el denominador. Pero eso quiere decir, según la Proposición B.1 que $k_1 P \bmod p = O \bmod p$ para *todos* los divisores primos p de n, algo que es muy poco probable si n tiene uno o dos divisores primos relativamente grandes. De manera que es virtualmente seguro que cuando tratamos de calcular $k_1 P$ mod n para alguna k_1 que es un múltiplo del orden de P mod p para alguna $p|n$, encontremos un divisor propio de n.

B.3.3 El algoritmo

Sea n un entero impar positivo, compusto. Supongamos que tenemos un método para generar pares (E, P) de curvas elípticas E: $y^2 = x^3 + ax + b$ con $a, b \in \mathbb{Z}$, y un punto $P = (x, y) \in E$. Dado un par como éste, hacemos el proceso que describimos a continuación. Si el proceso fracasa y no entrega un factor propio de n, generamos un nuevo par (E, P) y repetimos el proceso.

Antes de empezar a trabajar con nuestra E módulo n, verificamos que es una curva elíptica módulo cualquier $p|n$; esto sucede si y sólo si el discriminante $4a^3 + 27b^2$ es primo relativo con n. Si $(4a^3 + 27b^2, n) = 1$, procedemos. De lo contrario, si el máximo común divisor está entre 1 y n, hemos encontrado un factor propio de n y terminamos. Si el máximo común divisor es igual a n, elegimos una nueva curva y punto.

Después, suponemos que hemos elegido dos cotas, enteros positivos B y C. La cota B es la cota de homogeneidad para los divisores de k. Si B es grande, hay una gran probabilidad de que nuestra pareja (E, P) tenga la propiedad de que $kP \bmod p = O \bmod p$ para alguna $p|n$; por otro lado, entre más grande sea B más tiempo nos lleva calcular kP mod n. Así que B se debe elegir de alguna manera en que estimamos se minimice el tiempo que nos lleva

correr el algoritmo. La cota C es, en términos generales, una cota de los divisores primos $p|n$ para los cuales tenemos probabilidad de encontrar una relación $kP \bmod p = O \bmod p$. Después definimos k por la fórmula B.3.9; es decir, k es el producto de todas las potencias menores o iguales a C de todos los primos menores o iguales a B. El Teorema de Hasse nos dice que si p es un primo tal que $p+1+2\sqrt{p} < C$ y el orden de $E \bmod p$ no es divisible entre ningún primo mayor que B, entonces k es un múltiplo de éste orden, y $kP \bmod p = O \bmod p$.

Ejemplo B.3 Supongamos que elegimos $B = 20$, y queremos factorizar un entero n de diez dígitos decimales, que puede ser el produto de dos primos de cinco dígitos (es decir, ya sabemos que no es divisible por ningún primo de menos de cinco dígitos). Entonces tomamos $C = 100700$ y

$$k = 2^{16} \times 3^{10} \times 5^7 \times 7^5 \times 11^4 \times 13^4 \times 17^4 \times 19^3.$$

◁

Continuamos con la descripción del algoritmo. Trabajando módulo n, tratamos de calcular kP usando el método de duplicación y suma: calculamos $2P$, $2(2P)\ldots$, $2^{\alpha_2}P$, y luego calculamos $3(2^{\alpha_2})P$, $3(3 \cdot 2^{\alpha_2}P), \ldots$, $3^{\alpha_3}2^{\alpha_2}P$, etc., hasta tener $\prod_{\ell \leq B} \ell^{\alpha_\ell} P$. Al hacer estos cálculos, cada vez que tengamos que hacer una división módulo N, usamos el algoritmo de Euclides para encontrar el inverso multiplicativo módulo n. Si en cualquier paso el algoritmo de Euclides no nos proporciona un inverso, entonces hemos obtenido un divisor no trivial de n, o bien hemos obtenido a n mismo como máximo común divisor de n y el denominador en cuestión. En el primer caso, el algoritmo termina con éxito, reportando el factor propio. En el segundo caso, debemos regresar y elegir una nueva pareja (E, P). Si el algoritmo de Euclides siempre logra darnos un inverso, de manera que $kP \bmod n$ se puede calcular, entonces también tenemos que regresar y elegir una nueva pareja (E, P).

Ejemplo B.4 Vamos a usar la familiar de curvas elípticas $y^2 = x^3 + ax - a$, $a = 1, 2, \ldots$ cada una con el punto $P = (1, 1)$. Antes de usar una a para una n dada, verificamos que el discriminante $4a^3 + 27a^2$ es primo relativo con n. Vamos a tratar de factorizar $n = 5429$ con $B = 3$ y $C = 92$. La elección de C está motivada por el deseo de encontrar un factor primo p que es casi tan grande como $\sqrt{n} \approx 73$; para $p = 73$, la cota para el número de puntos de una curva elíptica definida sobre $\mathbb{F}_p$ es $72 + 2\sqrt{73} < 92$. Usando B.3.9 tomamos $k = 2^6 \cdot 3^4$. Para cada valor de a, multiplicamos P por 2 seis veces, y luego por 3 cuatro veces, trabajando módulo n en la curva elíptica correspondiente. Si $a = 1$, la operación se puede realizar sin problemas. Tratamos entonces $a - 2$, y pero al tratar de calcular $3^2 2^6 P$, tenemos un denominador cuyo máximo común divisor con n es el factor propio 61. Es decir, el punto $(1, 1)$ tiene como orden un divisor de $3^2 2^6$ en la curva $y^2 = x^3 + 2x - 1 \pmod 6$1. Si tratamos con $a = 3$, el método encuentra el otro factor primo de n, 89. ◁

En general, aunque no siempre, el método nos da el factor primo de n más pequeño.

B.3.4 Complejidad

El principal problema para calcular la complejidad del algoritmo es calcular, para una p fija y una elección dada de B, la probabilidad de que una curva elíptica elegida aleatoriamente, tenga orden N al reducir módulo p, y N no sea divisible por ningún primo mayor que B. Es decir, que N sea B-homogéneo.

Los ordenes N de las curvas elípticas módulo p están distribuidas de manera razonablemente uniforme en el intervalo

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}.$$

Entonces la probabilidad es más o menos igual a la probabilidad de que un entero aleatorio elegido en ese rango sea B-homogeneo. Esto nos lleva a una estimación de la forma

$$O\Big(\exp\Big(C\sqrt{\log_2(n)\log(\log_2 n)}\Big)\Big).$$

La derivación formal está en el artículo de Lenstra [Len87].

La complejidad en sí es comparable con la de otros métodos que ya se conocían, pero el método de Lenstra tiene ciertas ventajas sobre sus competidores:

1. Es el único método que es sustancialmente más rápido si n es divisible por un primo que es mucho más pequeño que $\sqrt{n}$.
2. Por ello, puede ser usado en combinación con otros métodos para factorizar.
3. Se puede paralelizar con facilidad.
4. Tiene muy poco requerimiento de memoria.

APÉNDICE

Dados dos polinomios

$$\begin{aligned} f(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \\ g(x) &= b_m x^m + b_{m-1}x^{m-1} + \cdots + b_0, \end{aligned}$$

con $a_n, b_m \neq 0$, el *resultante de* f *y* g, denotado por $R(f, g)$, se define como el determinante de la matriz de $(m + n) \times (m + n)$, dada por:

$$R(f,g) = \begin{vmatrix} a_n & a_{n-1} & \dots & a_0 & & & & & \\ & a_n & a_{n-1} & \dots & a_0 & & & & \\ & & & & \vdots & & & & \\ & & & & & a_n & a_{n-1} & \dots & a_0 \\ b_m & b_{m-1} & \dots & b_0 & & & & & \\ & b_m & b_{m-1} & \dots & b_0 & & & & \\ & & & & \vdots & & & & \\ & & & & & b_m & b_{m-1} & \dots & b_0 \end{vmatrix}.$$

En esta matriz, repetimos m veces los coeficientes de $f(x)$, y n veces los de $g(x)$. Los espacios en blanco en la representación de arriba llevan ceros.

Si las raíces de $f(x)$ son $t_1, \dots, t_n$, y las raíces de $g(x)$ son $u_1, \dots, u_m$, entonces se puede probar que

$$R(f,g) = a_n^m b_m^n \cdot \prod_{i=1}^{n} \prod_{j=1}^{m} (t_i - u_j).$$

Por lo tanto, $f(x)$ y $g(x)$ tienen al menos una raíz común si y sólo si $R(f,g) = 0$.

Si empezamos con dos polinomios en dos variables, $f(x,y)$, $g(x,y)$, entonces podemos considerarlos como polinomios en x y calcular el resultante, $R_x(f,g)$; en éste caso, $R_x(f,g)$ será un polinomio en la variable y; las raíces de éste polinomio indican los valores de y para los cuáles existe una x tal que (x,y) es raíz común para f y g.

APÉNDICE

En éste apéndice mencionamos algunas conjeturas de Teoría de Números y Geometría Algebráica Aritmética, que tienen implicaciones para la criptografía; en particular, para los métodos de cálculo de logaritmo discreto y los métodos de factorización.

D.1 La Hipótesis de Riemann

Sea ζ la función zeta de Riemann; ésta es la continuación analítica de la función:

$$\begin{aligned} \zeta(s) &= 1 + \frac{1}{2^s} + \frac{1}{3^s} + \cdots \\ &= \sum_{n=1}^{\infty} \frac{1}{n^s} \end{aligned}$$

$$= \prod_{p \text{ primo}} \frac{1}{1-p^s}.$$

Esta función está definida para toda $s \in \mathbb{C}$, con $\mathrm{Re}(s) > 1$; la continuación analítica está definida en el plano.

La función zeta de Riemman fue introducida por éste como una herramienta para probar el Teorema de los Números Primos, conjeturado por Gauss; a saber, que

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1,$$

donde $\pi(x)$ es el número de primos (positivos) menores o iguales a x. Su programa fue completado por Hadamard y Vallée-Poussin a finales del siglo XIX, culminando en una demostración analítica del Teorema de los Números Primos.

Riemann observó que la función ζ tiene ceros en $-2, -4, -6, \ldots$, llamados los "ceros triviales," y que todos los demás ceros de la función se encuentran en la franja $0 < \mathrm{Re}(\alpha) < 1$, distribuídos de manera simétrica alrededor de la recta $\mathrm{Re}(\alpha) = 1/2$. Estos se conocen como los "ceros no triviales" de ζ.

Conjetura D.1 (Hipótesis de Riemann) *Los ceros no triviales de la función ζ de Riemann ocurren en la recta* $\mathrm{Re}(s) = 1/2$.

La Hipótesis de Riemann se puede generalizar usando funciones semejantes a ζ en campos numéricos y en campos funcionales. La versión generalizada se llama la **Hipótesis Generalizada de Riemann**. Ésta ha sido demostrada en ciertos casos específicos.

Tanto la Hipótesis de Riemann como su versión generalizada están íntimamente ligadas con la distribución de los números primos, y tiene implicaciones para la distribución de números homogéneos. La complejidad de varios algoritmos que usan pruebas de primacidad o métodos probabilísticos dependen directamente de dichas distribuciones, y por ende su complejidad exacta depende de la Hipótesis de Riemann.

D.2 La Conjetura ABC

Sea n un entero positivo. Definimos el *radical de n,* $\mathrm{rad}(n)$ como la parte libre de cuadrados de n. Es decir, si la factorización prima de n está dada por:

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$$

donde $p_1 < p_2 < \cdots < p_r$, y $e_i > 0$ para cada i, entonces

$$\mathrm{rad}(n) = p_1 p_2 \cdots p_r.$$

Por ejemplo, $\mathrm{rad}(15) = 15$, $\mathrm{rad}(81) = 3$, y $\mathrm{rad}(1210) = 110$.

Sean A y B enteros positivos, primos relativos, y sea $C = A + B$. Por ejemplo, si $A = 4$, $B = 21$, entonces $C = 25$. Podemos ahora calcular $\mathrm{rad}(ABC)$, que es

$$\mathrm{rad}(ABC) = \mathrm{rad}(2^2 \times 3 \times 7 \times 5^2) = 2 \times 3 \times 5 \times 7 = 210.$$

Notemos que en este caso, tenemos que $\mathrm{rad}(ABC) > C$. De hecho, en casi todos los ejemplos que tomemos, tenemos que $\mathrm{rad}(ABC)/C > 1$. Pero de vez en cuando, sucede que ésto no es cierto. Por ejemplo, si $A = 1$, $B = 8$, y $C = 9$, entonces

$$\mathrm{rad}(ABC) = \mathrm{rad}(2^3 \times 3^2) = 6 < C;$$

si $A = 3$, $B = 125$, y $C = 128$, entonces

$$\mathrm{rad}(ABC) = \mathrm{rad}(3 \times 5^3 \times 2^7) = 2 \times 3 \times 5 = 30 < C.$$

De hecho, Masser probó que, eligiendo A y B con cuidado, podemos tener $\mathrm{rad}(ABC)/C$ tan pequeño como se quiera. Es decir:

Teorema D.1 (Masser)

$$\inf\left\{ \frac{\mathrm{rad}(ABC)}{C} \;\middle|\; A, B > 0,\ (A, B) = 1,\ A + B = C \right\} = 0.$$

La Conjetura ABC por otro lado dice que:

Conjetura D.2 (Conjetura ABC; Oesterlé y Masser) *Si $\epsilon > 0$, entonces los valores de*

$$\frac{(\mathrm{rad}(ABC))^{1+\epsilon}}{C}$$

tomados sobre todos los tripletes de enteros positivos (A, B, C) tales que A y B son primos relativos y $A + B = C$, tienen mínimo estrictamente positivo. Es decir: para toda $\varepsilon > 0$, existe una constante $c_\varepsilon > 0$ tal que para todos enteros positivos A, B, y C que satisfacen $A + B = C$ y $(A, B) = 1$,

$$C \leq c_\varepsilon \Big(\mathrm{rad}(ABC) \Big)^{1+\varepsilon}.$$

La Conjetura ABC es una de las preguntas abiertas más importantes en Teoría de Números hoy en día, y tiene demasiadas aplicaciones para mencionarlas todas. Por ejemplo, la Conjetura ABC implica el Último Teorema de Fermat y la Conjetura de Catalán de manera casi inmediata.

Las implicaciones más importantes para la criptografía están en la distribución de números homogéneos: la Conjetura ABC implica una distribución uniforme como se asume en la mayoría de los algoritmos probabilísticos que presentamos. La Conjetura ABC también tiene implicaciones para propiedades de curvas elípticas, que afectan los cálculos de la complejidad del algoritmo de factorización por curvas elípticas, así como los criptosistemas que se basan en curvas elípticas directamente.

D.3 Las Conjeturas de Vojta

En 1987, Paul Vojta propuso varias conjeturas de Geometría Algebráica Aritmética y Aproximación Diofantina, que indican una profunda conexión entre las ecuaciones diofantinas, y una rama del análisis complejo conocido como Teoría de Nevanlinna. Usando esta conexión, Vojta dio una segunda y totalmente nueva prueba de la Conjetura de Mordell, que había sido probada unos años antes por Gerd Faltings.

Tendríamos que salirnos mucho del tema para poder desarrollar el lenguaje necesario para expresar las Conjeturas de Vojta, de manera que no lo intentaremos. Las Conjeturas expresan de una manera cuantitativa cuántos puntos con coordenadas racionales (y qué tan "complicados" son los racionales involucrados) puede haber en una compleja superficie definida mediante cierto tipo de polinomios con coeficientes racionales.

En 1998, Vojta demostró que sus conjeturas implican la Conjetura ABC; Machiel van Frankenhuysen ha demostrado recientemente que la Conjetura ABC implica un caso especial de las Conjeturas de Vojta.

Además de las implicaciones relacionadas con la Conjetura ABC, las Conjeturas de Vojta tienen implicaciones importantes a las soluciones de cierto tipo de ecuaciones diofantinas.

APÉNDICE

El Teorema de Reciprocidad Cuadrática es el resultado de matemáticas que tiene el mayor número de pruebas esencialmente distintas publicadas. Gauss lo consideró su mejor resultado, y dió al menos cuatro pruebras distintas.

En este apéndice vamos a dar las definiciones y resultados importantes relacionados con Reciprocidad Cuadrática. También vamos a ver cómo usarla para decidir de manera eficiente si una congruencia de la forma $x^2 \equiv a \pmod{p}$ tiene solución, dónde p es un número primo. No vamos a proporcionar demostraciones. Éstas se pueden encontrar en casi cualquier libro de Teoría de Números, o en la Sección 2 del Capítulo II de [Kob94].

E.1 Residuos cuadráticos

Sea p un primo, $p > 2$. Queremos saber cuáles de los elementos distintos de cero, $\{1, 2, \ldots, p-1\}$ de $\mathbb{F}_p$ son cuadrados. Si alguna $a \in \mathbb{F}_p^*$ es un cuadrado, digamos $b^2 = a$, entonces a tiene exactamente dos raíces cuadradas: b y $-b$, pues la ecuación $x^2 - a = 0$ tiene a lo mucho dos soluciones en el campo. Entonces, los cuadrados de $\mathbb{F}_p^*$ se pueden encontrar calculando todos los valores $b^2 \bmod p$, para $b = 1, 2, \ldots, (p-1)/2$, y exactamente la mitad de los elementos de $\mathbb{F}_p^*$ son cuadrados.

Definición E.1 *Los cuadrados en $\mathbb{F}_p^*$ son llamados **residuos cuadráticos módulo** p. El resto de los elementos se llaman **residuos no cuadráticos módulo** p. Si no hay peligro de confusión, omitiremos el "módulo p."*

Por ejemplo, con $p = 11$, los cuadrados de $\mathbb{F}_{11}^*$ son $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 5$, y $5^2 = 3$. Es decir, los residuos cuadráticos módulo 11 son 1, 3, 4, 5, y 9; los residuos no cuadráticos son 2, 6, 7, 8, y 10. Hay $(p-1)/2$ residuos cuadráticos, y $(p-1)/2$ residuos no cuadráticos.

Si g es un generador de $\mathbb{F}_p^*$, cualquier elemento se puede escribir de la forma g^j. Los cuadrados son entonces elementos de la forma g^j con j par. Conversamente, cualquier elemento de la forma g^j con j par es un cuadrado, a saber, el cuadrado de $\pm g^{j/2}$.

E.2 El símbolo de Legendre

Definición E.2 *Sea a un entero, y $p > 2$ un primo. El **símbolo de Legendre** $\left(\frac{a}{p}\right)$ se define como 0, 1, ó -1, de acuerdo a la siguiente regla:*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{si } p|a. \\ 1 & \text{si } a \text{ es un residuo cuadrático módulo } p. \\ -1 & \text{si } a \text{ es un residuo no cuadrático módulo } p. \end{cases}$$

El símbolo de Legendre es entonces una manera de identificar si un entero es un residuo cuadrático módulo p o no.

La siguiente es una manera de encontrar el valor de $\left(\frac{a}{p}\right)$, pero es impráctica para valores grandes de a o de p:

Proposición E.1

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$

Las propiedades básicas del símbolo de Legendre son las siguientes:

Proposición E.2 *El símbolo de Legendre tiene las siguientes propiedades:*

1. $\left(\frac{a}{p}\right)$ *depende sólo del valor de* $a \bmod p$.
2. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.
3. *Para todo* b *primo relativo con* p, $\left(\frac{ab^2}{p}\right) = \left(\frac{a}{p}\right)$.
4. $\left(\frac{1}{p}\right) = 1$ *y* $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$.

Proposición E.3 (Caracter cuadrático de 2)

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8} = \begin{cases} 1 & \text{si } p \equiv \pm 1 \pmod{8}. \\ -1 & \text{si } p \equiv \pm 3 \pmod{3}. \end{cases}$$

Teorema E.1 (Reciprocidad Cuadrática**)** *Sean* p *y* q *dos primos impares. Entonces:*

$$\left(\frac{p}{q}\right) = (-1)^{(p-1)(q-1)/4}\left(\frac{q}{p}\right) = \begin{cases} -\left(\frac{q}{p}\right) & \text{si } p \equiv q \equiv 3 \pmod{4}. \\ \left(\frac{p}{q}\right) & \text{en cualquier otro caso.} \end{cases}$$

Es decir, si tanto p como q son congruentes con 3 módulo 4, entonces p es residuo cuadrático módulo q si y sólo si q no es residuo cuadrático módulo p. En cualquier otro caso, p es residuo cuadrático módulo q si y sólo si q es residuo cuadrático módulo p.

El carácter cuadrático de 2, de -1, y reciprocidad cuadrática nos permiten resolver fácilmente el problema de decidir si un entero es un cuadrado módulo un primo impar p o no.

Ejemplo E.1 Determine si 7411 es un cuadrado módulo el primo 9283.

Como ambos son primos, y los dos son congruentes con 3 módulo 4, tenemos que

$$\left(\frac{7411}{9283}\right) = -\left(\frac{9283}{7411}\right).$$

Podemos reemplazar 9283 por su residuo módulo 7411, que es 1872. De manera que

$$\left(\frac{7411}{9283}\right) = -\left(\frac{1872}{7411}\right).$$

Ahora factorizamos 1872. Afortunadamente es sencillo:

$$1872 = 2^4 \times 3^2 \times 13,$$

de manera que

$$\left(\frac{7411}{9283}\right) = -\left(\frac{1872}{7411}\right) = -\left(\frac{2^4}{7411}\right) \cdot \left(\frac{3^2}{7411}\right) \cdot \left(\frac{13}{7411}\right).$$

Los dos primeros símbolos de Legendre valen 1, pues 2^4 y 3^2 son cuadrados en los enteros. Por lo tanto,

$$\left(\frac{7411}{9283}\right) = -\left(\frac{13}{7411}\right).$$

Como $13 \equiv 1 \pmod 4$, tenemos que $\left(\frac{13}{7411}\right) = \left(\frac{7411}{13}\right)$; reemplazamos 7411 por su residuo módulo 13, que es 1, y tenemos que

$$\left(\frac{7411}{9283}\right) = -\left(\frac{1}{13}\right) = -1.$$

De manera que 7411 es un residuo no cuadrático módulo 9283. ◁

La única dificultad en el proceso anterior es que en cada paso tenemos que factorizar el "numerador" (aunque éste siempre se puede tomar más pequeño que el "denominador"). Afortunadamente, hay una generalización de reciprocidad cuadrática que permite realizar el cálculo sin factorizar (excepto por los factores de 2, que son muy fáciles de obtener).

E.3 El símbolo de Jacobi

Definición E.3 *Sea a un entero, y n un entero positivo impar. Escribimos $n = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$, su factorización en primos distintos, con $\alpha_i > 0$ para cada i. Definimos el* ***símbolo de***

Jacobi $\left(\frac{a}{n}\right)$ *como el producto de los símbolos de Legendre de los factores primos de* n*; es decir:*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \cdots \left(\frac{a}{p_r}\right)^{\alpha_r}.$$

No hay problema de confusión con el símbolo de Legendre, pues en el caso en que n es primo, los valores son idénticos.

NOTA: Si $\left(\frac{a}{n}\right) = -1$ con n un número compuesto, entonces a no es un cuadrado módulo n. Sin embargo, puede ser que $\left(\frac{a}{n}\right) = 1$, pero que a sea un residuo no cuadrático módulo n. Por ejemplo,

$$\left(\frac{2}{15}\right) = \left(\frac{2}{3}\right)\left(\frac{2}{5}\right) = (-1)(-1) = 1,$$

pero 2 no es un cuadrado módulo 15 (pues no es un cuadrado módulo 3).

Ahora generalizamos las propiedades del símbolo de Legendre al símbolo de Jacobi:

Proposición E.4 *Sean* m *y* n *dos enteros positivos impares,* a, b *dos enteros cualesquiera.*

1. *El valor de* $\left(\frac{a}{n}\right)$ *depende únicamente del valor de* a *módulo* n*.*
2. $\left(\frac{a}{n}\right) = 0$*,* 1*, ó* -1*, y vale* 0 *si y sólo si* $(a, n) \neq 1$*.*
3. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$*.*
4. $\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{a}{n}\right)$*.*
5. *Para todo entero* b*, primo relativo con* n*,* $\left(\frac{ab^2}{n}\right) = \left(\frac{a}{n}\right)$*.*

Proposición E.5 (Caracter cuadrático de -1**)** *Para todo entero positivo impar* n*,*

$$\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}.$$

Por lo tanto, $\left(\frac{-1}{n}\right) = 1$ *si* $n \equiv 1 \pmod 4$*, y* $\left(\frac{-1}{n}\right) = -1$ *si* $n \equiv 3 \pmod 4$*.*

Proposición E.6 (Caracter cuadrático de 2**)** *Para todo entero positivo impar* n*,*

$$\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}.$$

Por lo tanto, $\left(\frac{2}{n}\right) = 1$ *si* $n \equiv \pm 1 \pmod 8$*, y* $\left(\frac{2}{n}\right) = -1$ *si* $n \equiv \pm 3 \pmod 8$*.*

Teorema E.2 (Reciprocidad Cuadrática) *Para cualesquiera dos enteros positivos impares m y n,*

$$\left(\frac{m}{n}\right) = (-1)^{(m-1)(n-1)/4} \left(\frac{n}{m}\right).$$

En particular,

$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right), & \text{si } m \equiv n \equiv 3 \pmod 4, \\ \left(\frac{n}{m}\right) & \text{en otro caso.} \end{cases}$$

Ejemplo E.2 Ahora podemos hacer el cálculo anterior sin tener que factorizar ni verificar si los dos valores son primos. Sólo tenemos que "sacar" el factor de 2^4 de 1872:

$$\begin{aligned} \left(\frac{7411}{9283}\right) &= -\left(\frac{9283}{7411}\right) \\ &= -\left(\frac{1872}{7411}\right) \\ &= -\left(\frac{16}{7411}\right)\left(\frac{117}{7411}\right) \\ &= -\left(\frac{7411}{117}\right) \\ &= -\left(\frac{40}{117}\right) \\ &= -\left(\frac{8}{117}\right)\left(\frac{5}{117}\right) \\ &= -\left(\frac{2}{117}\right)\left(\frac{117}{5}\right) \\ &= \left(\frac{117}{5}\right) \\ &= \left(\frac{2}{5}\right) \\ &= -1. \end{aligned}$$

◁

Ejemplo E.3 Diga si 43691 es un residuo cuadrático módulo 65537.

$$\left(\frac{43691}{65537}\right) = \left(\frac{65537}{43691}\right)$$

$$\begin{aligned}
&= \left(\frac{21846}{43691}\right) \\
&= \left(\frac{2}{43691}\right)\left(\frac{10923}{43691}\right) \\
&= -\left(\frac{10923}{43691}\right) \\
&= \left(\frac{43691}{10923}\right) \\
&= \left(\frac{10922}{10923}\right) \\
&= \left(\frac{-1}{10923}\right) \\
&= -1.
\end{aligned}$$

Por lo tanto, 43691 es un residuo no cuadrático módulo 65537. ◁

E.4 Complejidad del algoritmo

Calcular el valor de $\left(\frac{a}{n}\right)$, donde n es un entero positivo impar (ya sea primo o no) lleva aproximadamente $O((\log(n)^2))$ operaciones de bits.

Bibliografía

[Bau00] Friedrich L. Bauer. *Decrypted Secrets. Methods and Maxims of Cryptology*. Springer-Verlag, 2nd revised and extended edition, 2000. MR **2001a**:94021.

[Beu94] Albrecht Beutelspacher. *Cryptology*. Spectrum. The Mathematical Association of America, 1994. MR **94m**:94015.

[Bih95] Eli Biham. On Matsui's linear cryptanalysis. In Alfredo de Santis, editor, *Advances in Cryptology–EUROCRYPT '94*, number 950 in Lecture Notes in Computer Science, pages 341–455. Springer-Verlag, 1995.

[Bon99] Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems (extended abstract). In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology–CRYPTO '90*, number 537 in Lecture Notes in Computer Science, pages 2–21. Springer-Verlag, 1991.

[BS93a] Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round DES. In Ernest F. Brickell, editor, *Advances in Cryptology–CRYPTO '92*, number 740 in Lecture Notes in Computer Science, pages 487–496. Springer-Verlag, 1993.

[BS93b] Eli Biham and Adi Shamir. *Differential Cryptoanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[BSS99] Ian Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic Curves in Cryptography*. Number 256 in LMS Lecture Note Series. Cambridge University Press, 1999. MR **2001i**:94048.

[BW00] David Bressoud and Stan Wagon. *A Course in Computational Number Theory*. Springer-Verlag, 2000.

[Cop97] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10:233–260, 1997.

[CW93] K.W. Campbell and M.J. Wiener. DES is not a group. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO '92 Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 280–291. Springer-Verlag, 1993. MR **95b**:94001.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, IT-22(6):644–654, 1976. MR **55** #10141.

[DR02] Tim Dierks and Eric Rescorla. *The TLS Protocol*. Internet Engineering Task Force (IETF), 2002. `http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc2246-bis-01.txt`.

[FdlGH$^+$01] A. Fúster, D. de la Guía, L. Hernández, F. Montoya, and J. Muñoz. *Técnicas Criptográficas de Protección de Datos, 2a. Ed.* Alfaomega, 2001.

[Fei73] Horst Feistel. Cryptography and computer privacy. *Scientific American*, pages 15–23, May 1973.

[Hey] Howard M. Heys. A tutorial on linear and differential cryptanalysis. `www.engr.mun.ca/ howard/PAPERS/ldc_tutorial.ps`.

[Hil29] Lester S. Hill. Cryptography in an algebraic alphabet. *American Mathematical Monthly*, 36:306–312, 1929.

[Hil31] Lester S. Hill. Concerning certain linear transformation apparatus of cryptography. *American Mathematical Monthly*, 38(3):135–154, 1931.

[Kah99] David Kahn. *The Codebreakers*. Scribner, 2nd edition, 1999.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, IX:5–38, 1883. `http://www.cl.cam.ac.uk/~fapp2/kerckhoffs/index.html`.

[Kob94] Neal Koblitz. *A Course in Number Theory and Cryptography*. Number 114 in GTM. Springer-Verlag, second edition, 1994. MR **95h**:94023.

[Len87] H.W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of Math.*, 126:649–673, 1987. MR **89g**:11125.

[Lew00] Robert Edward Lewand. *Cryptological Mathematics*. The Mathematical Association of America, 2000.

[LH94] S. Langford and M. Hellman. Differential-linear cryptanalysis. In Yvo G. Desment, editor, *Advances in Cryptology–CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 26–39. Springer-Verlag, 1994.

[Lib] Santa Cruz Public Libraries. Frecuency of occurrence of letters in spanish. `http://www.santacruzpl.org/readyref/files/g-l/ltfrqsp.shtml`. Tomado de: Fletcher Pratt, *Secret and Urgent: the Story of Codes and Ciphers*, Blue Ribbon Books, 1939, pp. 254-255.

[LL93] A.K. Lenstra and H.W. Lenstra, Jr., editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993. MR **96m**:11116.

[Lov86] L. Lovasz. *An Algorithmic Theory of Numbers, Graphs, and Convexity*. SIAM Publications, 1986. MR **87m**:68066.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology–EUROCRYPT '93*, number 765 in Lecture Notes in Computer Science, pages 386–397. Springer-Verlag, 1994.

[Mat95] Mitsuri Matsui. On correlation between the order of S-boxes and the strength of DES. In Alfredo de Santis, editor, *Advances in Cryptology–EUROCRYPT '94*, number 950 in Lecture Notes in Computer Science, pages 366–375. Springer-Verlag, 1995.

[Mil96] A. Ray Miller. *The Cryptographic Mathematics of Enigma*. National Security Agency, 1996.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. MR **99g**:94015.

[Nic] Randall K. Nichols. Classical cryptography course, lecture 2. `http://ads.fortunecity.com/RealMedia/ads/adstream_sx.ads/net/locale@x32`. Tomado de: Nichols Randall K., *Classical Cryptography Course*, Vol. I, Aegean Park Press, 1995.

[oST99] National Institute of Standards and Technology. *Data Encryption Standard (DES)*. Federal Information Processing Standards Publication 146-3, 25 de octubre de 1999. Disponible en `http://csrc.nist.gov/cryptval/` en línea en: `http://www.itl.nist.gov/fipspubs/fip46-2.htm`.

[oST01] National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication 197, 26 de noviembre de 2001. Disponible en `http://csrc.nist.gov/cryptval/`.

[otA90] Department of the Army, editor. *Basic Cryptoanalysis*. Number 34-40-2 in Field Manual. HQ Dept. of the Army, 1990.

[Pom96] Carl Pomerance. A tale of two sieves. *Notices of the AMS*, 43(12):1473–1485, 1996. MR **97f**:11100.

[RS84] R. Rivest and A. Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–395, 1984.

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, second edition, 1996.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(4):379–423, 623–656, 1948.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[Sin66] Abraham Sinkov. *Elementary Cryptanalysis: A Mathematical Approach*. Number 22 in New Mathematical Library. The Mathematical Association of America, 1966.

[Sin99] Simon Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*. Doubleday, 1999.

[Wil01] Jennifer E. Wilcox. *Solving the Enigma: History of the Cryptanalytic Bombe*. National Security Agency, 2001.

[YKS+02] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. *SSH Transport Layer Protocol*. Network working Group, Internet Engineering Task Force (IETF), 2002. `http://www.ietf.org/internet-drafts/`.