Teoría de Códigos y Criptografía

Notas de Teoría

Laura Colmenarejo, Marithania Silvero,
M. Jesús Soto, José M. Tornero, José M. Ucha

Departamento de Álgebra, Universidad de Sevilla

Curso 2013/14

Índice

Tema 1. Fundamentos

Grupos.

Definición.– Un grupo es un par $(G, \star)$ formado por un conjunto $G \neq \emptyset$ y una operación binaria interna

$$\begin{array}{rcl} \star : G \times G & \longrightarrow & G \\ (x, y) & \longmapsto & \star(x, y) := x \star y \end{array}$$

tal que:

1. Verifica la propiedad asociativa.
2. Posee un elemento neutro. Esto es, existe $e \in G$ tal que para todo $x \in G$ se tiene que
$$e \star x = x \star e = x.$$
3. Todo elemento $x \in G$ posee un simétrico. Esto es, existe $y \in G$ tal que $x \star y$ e $y \star x$ son un elemento neutro.

Si el grupo G posee además la propiedad conmutativa, se dirá un grupo conmutativo o abeliano.

Observación.– Es inmediato comprobar que, en estas condiciones, el elemento neutro es único y lo notaremos $e \in G$. De la misma forma, dado $x \in G$, el opuesto de x también es único y se notará $x' \in G$.

Observación.– Habitualmente, cuando la operación se sobreentienda, diremos simplemente *"el grupo G"*.

Observación.– La notación escogida es intencionadamente neutra, en el sentido de que no tiene connotaciones, pero la mayoría de los grupos con los que vamos a trabajar vienen equipados con una operación denominada *suma* (notada $+$) o *producto* (notada $\cdot$ o por simple yuxtaposición de elementos). En estos casos, se aplican notaciones diferentes:

Operación	E. neutro	E. opuesto
Suma $(+)$	0	$-x$
Producto $(\cdot)$	1	$1/x$ o bien x^{-1}

Ejemplos.– Los grupos más habituales con los vamos a trabajar son los siguientes:

- El grupo de las permutaciones. Dado un conjunto X de n elementos, las biyecciones de X en sí mismo son un grupo con la composición de aplicaciones, denotado S_X. La representación habitual de este grupo suele hacerse tomando $X = \{1, 2, ..., n\}$ y denotando el grupo por S_n. Este grupo posee ciertas propiedades importantes, estudiadas en Álgebra Básica como son la factorización única en producto de ciclos disjuntos o factorización en producto de trasposiciones. No es abeliano para $n \geq 3$.

- Los grupos cíclicos finitos. Un grupo cíclico finito de n elementos, en notación multiplicativa, está formado por las potencias de un elemento x que verifica $x^n = 1$, esto es
$$\mathcal{C}_n := \{x^i \mid i = 0, ..., n-1\} = \{1 = x^0,\, x,\, x^2, ..., x^{n-1}\}.$$
En notación aditiva sólo hay que cambiar "potencia" por "múltiplo",
$$\mathcal{C}_n := \{ix \mid i = 0, ..., n-1\} = \{0 = 0x,\, x,\, 2x, ..., (n-1)x\}.$$
Hemos utilizado la misma notación porque, en un sentido que luego haremos preciso, ambos grupos son, en realidad, el mismo.

- El grupo cíclico infinito. El grupo cíclico infinito generado por n está formado por todas las potencias (positivas y negativas) de un elemento x, que son siempre distintas entre sí:
$$\mathcal{C}_\infty := \{x^i \mid i \in \mathbf{Z}\}.$$

Definición.– La subestructura natural de los grupos son los subgrupos. Dado un grupo $(G, \star)$, un subconjunto $H \subset G$ es un subgrupo cuando $\left(H, \star_{|H\times H}\right)$ es a su vez un grupo. Esto se denota abreviadamente por $H \leq G$.

Observación.– Una familia especialmente importante de subgrupos es la que permite inducir estructuras de grupos por relación de equivalencia: los denominados subgrupos normales. Para definirlos, fijamos un grupo $(G, \star)$ y un subgrupo $H \leq G$. Consideramos las órbitas de un elemento $x \in G$ cualquiera por H, esto es:
$$x \star H = \{x \star h \mid h \in H\}, \qquad H \star x = \{h \star x \mid h \in H\}$$

Definición.– Decimos que H es normal en G cuando, para todo $x \in G$, $x \star H = H \star x$. Esta condición se denota habitualmente $H \lhd G$.

Observación.– Algunas propiedades equivalentes a $H \lhd G$ son

1. Para todo $x \in G$, $x \star H \star x' := \{x \star h \star x' \mid h \in H\} = H$.
2. Para todo $x \in G$, $x \star H \star x' := \{x \star h \star x' \mid h \in H\} \subset H$.

Proposición.– Dado $H \lhd G$, la relación definida en G por
$$x \sim y \iff x \star y' \in H$$
es de equivalencia. La clase de equivalencia de x es, precisamente $x \star H$. El conjunto cociente, que se denota habitualmente G/H, tiene estructura de grupo con la operación (notada también $\star$, sólo por incordiar):
$$(x \star H) \star (y \star H) = (x \star y) \star H.$$

Observación.– Cuando G es abeliano (cosa que sucederá con frecuencia en los grupos que trataremos en este curso), es inmediato ver que todo subgrupo es normal y, por tanto, siempre podemos considerar la estructura cociente.

Definición.– La otra operación esencial entre grupos es el producto. Dados dos grupos $(G, \star)$ y $(L, \bullet)$ (eventualmente, se puede considerar una cantidad finita arbitraria de grupos), se define su producto (directo) como el grupo $G \times L$, con la operación $\diamond$ definida por

$$(x, a) \diamond (y, b) = (x \star y, a \bullet b).$$

Definición.– Un homomorfismo de grupos es una aplicación

$$f : (G, \star) \longrightarrow (L, \bullet)$$

que verifica que para cualquier par de elementos $x, y \in G$,

$$f(x \star y) = f(x) \bullet f(y),$$

esto es, un homomorfismo es una aplicación que respeta las operaciones de grupo.

Un homomorfismo biyectivo se denomina un isomorfismo. Cuando existe un isomorfismo entre G y L como arriba, esto se suele denotar $G \simeq L$. Ambos grupos se dicen isomorfos y, dado que desde el punto de vista de la Teoría de Grupos, ambos objetos son absolutamente equivalentes, se tratan como el mismo objeto.

Observación.– Algunos resultados importantes concernientes a isomorfismos son:

1. Si dos conjuntos X e Y tienen el mismo cardinal (finito), entonces $S_X \simeq S_Y$.
2. Dos grupos cíclicos finitos con el mismo números de elementos son siempre isomorfos.
3. $\mathcal{C}_p \times \mathcal{C}_q \simeq \mathcal{C}_{pq}$ si y sólo si[1] $\gcd(p, q) = 1$.
4. Todo grupo cíclico infinito es isomorfo a $(\mathbf{Z}, +)$.
5. Todo grupo abeliano finitamente generado es isomorfo a uno de la forma
$$\mathcal{C}_{d_1} \times ... \times \mathcal{C}_{d_n} \times \mathbf{Z}^r,$$
donde $d_1 \mid d_2 \mid ... \mid d_n$ y $r \geq 0$.
6. Todo grupo abeliano finitamente generado es isomorfo a uno de la forma
$$\mathcal{C}_{p_1^{\alpha_1}} \times ... \times \mathcal{C}_{p_n^{\alpha_n}} \times \mathbf{Z}^r,$$
donde p_i es primo para $i = 1, ..., n$, y $r \geq 0$.

Teorema.– (Primer Teorema de Isomorfía). Sea

$$f : (G, \star) \longrightarrow (L, \bullet)$$

y definimos

$$\ker(f) = \{g \in G \mid f(g) = e_L\}$$
$$\mathrm{Img}(f) = \{l \in L \mid \exists g \in G \text{ con } l = f(g)\}$$

Entonces $\ker(f) \lhd G$, $\mathrm{Img}(f) < L$ y $G/\ker(f) \simeq \mathrm{Img}(f)$.

[1] La definición de $\gcd(p, q)$, aunque es conocida, se recordará más adelante.

Anillos. Dominios euclídeos.

Definición.– Un *anillo* es una terna $(A, +, \cdot)$ formada por un conjunto A y dos operaciones internas y binarias $+, \cdot$ verificando:

1. El par $(A, +)$ es un grupo abeliano, cuyo elemento neutro llamaremos normalmente "cero (0)".
2. La operación binaria $\cdot$ es asociativa y tiene elemento neutro, que llamaremos normalmente "uno" (notado 1).
3. La operación $\cdot$ es *distributiva* a la derecha y a la izquierda respecto de la operación $+$, i.e. para todos $x, y, z \in A$, se tiene

$$(x + y) \cdot z = x \cdot z + y \cdot z, \quad x \cdot (y + z) = x \cdot y + x \cdot z.$$

Si además la operación $\cdot$ es conmutativa, diremos que el anillo es conmutativo. En esta asignatura no veremos anillos no conmutativos, aunque los hay y muy importantes (por ejemplo, los anillos de matrices cuadradas).

Observación.– Algunas notas a la definición.

1. En general se usará la expresión "sea A un anillo", sobreentendiendo las operaciones. La operación $\cdot$ se notará normalmente por simple yuxtaposición.
2. En un anillo A se tiene $0 \cdot x = x \cdot 0 = 0$ para todo $x \in A$.
3. Si en un anillo A se tiene $1 = 0$, entonces $A = \{0\}$.
4. Para todo $x, y \in A$, se tiene $x(-y) = (-x)y = -(xy)$.
5. Si $A_1, \ldots, A_n$ son anillos, el producto cartesiano $A_1 \times \cdots \times A_n$ posee una estructura natural de anillo, donde las operaciones están definidas componente a componente.

Definición.– Sea A un anillo. Una unidad es un elemento que posee un simétrico multiplicativo (a la izquierda y a la derecha), que llamaremos inverso. El conjunto de las unidades de A es un grupo para el producto y se notará A^*.

Un *cuerpo* es un anillo conmutativo tal que todo elemento distinto de cero es una unidad, i.e. $A^* = A \setminus \{0\}$.

Ejemplos.– Algunos casos sencillos de unidades.

1. Las unidades de $\mathbf{Z}$ son $1, -1$.
2. Los anillos $\mathbf{Q}, \mathbf{R}, \mathbf{C}$ son cuerpos.
3. El grupo de las unidades del anillo $\mathcal{M}(n, k)$ con $k = \mathbf{Q}, \mathbf{R}$ ó $\mathbf{C}$ es $\mathrm{GL}(n, k)$ (las matrices con determinante no nulo).

Definición.– Sea A un anillo. Un elemento $x \in A$ se llamará un *divisor de cero* si y sólo si es distinto de cero y existe $y \in A$, $y \neq 0$, tal que $xy = 0$. Un anillo sin divisores de cero se llama un *dominio de integridad* (o dominio, a secas).

Un elemento $x \in A$ se llamará *nilpotente* si es distinto de cero y existe un entero $n > 0$ tal que $x^n = 0$.

Observación.– En un dominio de integridad se da la propiedad cancelativa por el producto de elementos no nulos:

$$a \neq 0,\ ab = ac \Rightarrow b = c.$$

lo cual hace que estos anillos sean particularmente cómodos a la hora de hacer cálculos.

Observación.– El concepto equivalente a subgrupo para anillos no es, como podría esperarse, el de subanillo. En efecto, si $R \subset A$ son dos anillos (con las mismas operaciones) no se puede definir, en general, una estructura coherente de anillo en el grupo cociente A/R.

A mediados del siglo XIX, Dedekind buscaba una generalización de la factorización única en primos que se tiene en $\mathbf{Z}$ para unos anillos denominados *anillos de enteros algebraicos*. Estos anillos, que contienen a $\mathbf{Z}$ y están contenidos en $\mathbf{C}$, no verifican en general que exista una factorización única en elementos irreducibles. Por ello, Dedekind pensó sustituir la factorización tradicional, en producto de elementos del anillo, por un concepto que denominó *elementos ideales*. Conviene destacar que la idea original ya la tuvo Kummer algunos años antes, buscando la demostración del Teorema de Fermat.

Sin embargo, para mantener las propiedades de la divisibilidad tradicional (que después recordaremos), Dedekind pensó que sus elementos ideales debían verificar las propiedades análogas a la divisibilidad. Esto es:

1. Si $a|b$ y $a|c$, entonces $a|(b+c)$.
2. Si $a|b$, entonces $a|bc$, para todo c.

El resultado de esta magnífica idea fue el concepto que hoy denominamos *ideal de un anillo*.

Definición.– Sea A un anillo. Un *ideal* de A es un subconjunto I de A que verifica:

1. I es un subgrupo del grupo aditivo de A.
2. Para cualesquiera $a \in I\ ,\ x \in A$ se tiene $xa \in I$.

Observación.– Sea $I \subset A$ un ideal de A. Se tiene:

1. Si I contiene una unidad, entonces $I = A$.
2. A es un cuerpo si y sólo si sus únicos ideales son $\{0\}$ y A.

Observación.– En realidad no hace falta comprobar que I es un subgrupo aditivo. Para ver si I es ideal es suficiente probar (comparar con las propiedades que buscaba Dedekind):

1. Si $a, b \in I$, entonces $a + b \in I$.
2. Si $a \in I$ y $c \in A$, entonces $ac \in I$.

Asombrosamente, el concepto de Dedekind es precisamente el que permite dotar de estructura de anillo a los cocientes.

Proposición.– El grupo cociente A/I admite una estructura canónica de anillo, con el producto definido por

$$(a+I)(b+I) = ab + I.$$

Definición.– Sean A, B anillos, $f : A \longrightarrow B$ una aplicación. Se dirá que f es un *homomorfismo* de anillos si verifica:

1. Para cualesquiera $x, y \in A$, es $f(x+y) = f(x) + f(y)$[2].
2. Para cualesquiera $x, y \in A$, es $f(xy) = f(x)f(y)$[3].
3. $f(1) = 1$[4].

Un homomorfismo biyectivo se llama un *isomorfismo* y los anillos entre los cuales se puede establecer un isomorfismo se llaman anillos isomorfos. Análogamente al caso de grupos, trataremos los anillos isomorfos como el mismo anillo, a todos los efectos.

Se verifica también el Primer Teorema de Isomorfía, con el enunciado análogo (cuidado: ahora $\ker(f)$ son los elementos de A que van a 0_B).

De entre todos los dominios, nos interesan particularmente una familia que incluye a los enteros y a los polinomios: los dominios euclídeos, o dominios de división.

Definición.– Un anillo R se dice un dominio euclídeo si existe una aplicación

$$\mu : R \setminus \{0\} \longrightarrow \mathbf{Z}_{\geq 0} := \mathbf{N} \cup \{0\}$$

verificando que, dados $a, b \in R \setminus \{0\}$, $\mu(ab) \leq \mu(a)$ y, si $b \neq 0$, existen elementos únicos $q, r \in R$, denominados respectivamente cociente y resto, tales que:

$$a = b \cdot q + r, \text{ con } r = 0 \text{ o bien } \mu(r) < \mu(b).$$

Cuando $r = 0$, se dice que b divide a a, o que b es un divisor de a, o que a es un múltiplo de b o que a es divisible por b. Se suele denotar $b|a$.

Observación.– Las propiedades básicas de la divisibilidad son:

1. Si $b|a$ y $b|c$, entonces $b|(a \pm c)$.
2. Si $b|a$, entonces $b|(ac)$ para todo $c \in R$.

Todo elemento $a \in R$ es trivialmente divisible por las unidades de R y por a, así como por los productos de ambos. Cuando ésos son precisamente los únicos divisores de a, entonces a se dice que es un elemento irreducible.

Los resultados más importantes sobre elementos irreducibles son los siguientes:

Teorema de Euclides.– Sean $a, b, p \in R$ tales que p es irreducible y $p|(ab)$. Entonces $p|a$ o $p|b$.

[2] Ojo, no son la misma operación.
[3] Ojo, igual que el ojo anterior.
[4] Ojo, y van tres (¿?): no necesariamente son el mismo elemento.

Teorema.– Todo ideal I de un dominio euclídeo R es principal. Esto es, existe un $x \in I$ tal que I es, precisamente, el conjunto de múltiplos de x (notación habitual: $I = \langle x \rangle$).

Teorema de la factorización única.– Todo $a \in R$ distinto de 0 y no unidad es producto, salvo unidades y reordenación, de manera única de irreducibles. Esto también se expresa diciendo que R es un dominio factorial o de factorización única.

Observación.– Podemos usar "primo" como alternativa a "irreducible", pero hay que tener cuidado: esta identificación no es cierta en todos los anillos. Formalmente, un elemento irreducible es aquél que no tiene divisores, salvo los triviales, y un elemento primo es aquél que, cuando divide a un producto, necesariamente divide a alguno de los factores. En dominios euclídeos ambos conceptos son equivalentes, pero en general no es así.

Definición.– Dados $a, b \in R$ existe un único (salvo producto por unidades) $d \in R$ que verifica las dos propiedades siguientes:

- $d|a$ y $d|b$.
- Para todo $e \in R$ tal que $e|a$ y $e|b$, se tiene que $e|d$.

Ese entero d se denomina *máximo común divisor* de a y b y se denota $\gcd(a, b)$.

Un resultado esencial a cuenta del $\gcd$ es el conocido como Identidad de Bézout:

Identidad de Bézout.– Dados $a, b \in R$, existen $\alpha, \beta \in R$ tales que

$$\gcd(a, b) = \alpha \cdot a + \beta \cdot b.$$

De hecho, $\gcd(a, b)$ se puede tomar como un elemento de R que se puede expresar de esta forma y que es minimal para μ.

Observación.– La Identidad de Bézout se puede ampliar para demostrar que todo ideal finitamente generado es princial, esto es, tiene un sistema de generadores unitario.

Y una propiedad, imprescindible en muchos argumentos, es el Teorema Chino del Resto:

Teorema Chino del Resto.– Dados $m_1, ..., m_n \in R$ tales que $\gcd(m_i, m_j) = 1$ cuando $i \neq j$, se tiene un isomorfismo de anillos

$$R/\langle m_1 ... m_n \rangle \simeq R/\langle m_1 \rangle \times ... \times R/\langle m_n \rangle.$$

Observación.– Si disponemos de un algoritmo para hallar la identidad de Bézout, el Teorema Chino del Resto admite una demostración constructiva, esto es, da un procedimiento para hallar el isomorfismo. Sin entrar en los detalles:

$\longrightarrow$ Dado $x + \langle m_1 ... m_n \rangle$, tomamos como imagen

$$\left(x + \langle m_1 \rangle, ..., x + \langle m_n \rangle\right).$$

$\longleftarrow$ En el sentido contrario, procedemos como sigue, dado $(a_1 + \langle m_1 \rangle, ..., a_n + \langle m_n \rangle)$:

- Hallamos $M = m_1 ... m_r$ y también $M_i = M/m_i$ para $i = 1, ..., r$.

- Como $\gcd(m_i, M_i) = 1$, podemos hallar $\alpha_i, \beta_i \in R$ tales que

$$\alpha_i m_i + \beta_i M_i = 1,$$

por la Identidad de Bézout.

- Tomamos entonces el elemento

$$x = a_1\beta_1 M_1 + ... + a_r\beta_r M_r,$$

que verifica que

$$(x + \langle m_1\rangle, ..., x + \langle m_n\rangle) = (a_1 + \langle m_1\rangle, ..., a_n + \langle m_n\rangle)\,.$$

El estudio de los ideales de estos dominios arroja unos últimos resultados fundamentales (bastantes sencillos):

Teorema.– Sea A un dominio euclídeo, $I = \langle x\rangle \subset A$ un ideal. Entonces

$$A/I \text{ es dominio} \iff A/I \text{ es cuerpo} \iff x \text{ es irreducible.}$$

Proposición.– Sea A un dominio de ideales principales, $I = \langle x\rangle \subset A$ un ideal. Entonces para todo ideal $J \subset A/I$ existe $z \in A$ tal que:

- $z|x$.
- $J = \langle z + I\rangle$.

Congruencias.

Los enteros $(\mathbf{Z}, +, \cdot)$ son un dominio euclídeo, gracias al Teorema de la División Entera.

Teorema de la División Entera.– Dados $a, b \in \mathbf{Z}$, con $b \neq 0$, existen únicos $q, r \in \mathbf{Z}$ tales que

$$a = qb + r, \text{ con } 0 \leq r < b.$$

Como sabemos, las unidades de $\mathbf{Z}$ son ± 1 y los irreducibles son, precisamente, los números primos.

Todos los subgrupos de $(\mathbf{Z}, +)$ son de la forma

$$\mathbf{Z}n = \{\alpha \cdot n \mid \alpha \in \mathbf{Z}\},$$

y es inmediato comprobar que, efectivamente, esos conjuntos son ideales en el anillos $(\mathbf{Z}, +, \cdot)$. Las congruencias son simplemente los anillos cociente de $\mathbf{Z}$, esto es, los anillos de la forma $\mathbf{Z}/\mathbf{Z}n$.

Una forma alternativa, más elemental de verlo, es la siguiente. Dado $n \in \mathbf{Z}$, decimos que dos enteros $a, b \in \mathbf{Z}$ son *congruentes módulo* n, cuando a y b tienen el mismo resto al dividirlos por n. Esto se denota

$$a \equiv b \mod n,$$

y es equivalente a que $n|(a-b)$ (o sea, a $a+\mathbf{Z}n = b+\mathbf{Z}n$ en el argot de anillos cociente). Notaremos

$$x + \mathbf{Z}n = \{a \in \mathbf{Z} \mid a \equiv x \mod n\},$$

denominada la *clase de x módulo n*.

A partir de la definición anterior es obvio que todo entero es congruente con uno (y sólo uno) de los elementos de

$$\{0,\, 1, ...,\, n-1\},$$

esto es, el conjunto de clases módulo n, es, exactamente,

$$\mathbf{Z}/\mathbf{Z}n = \{0 + \mathbf{Z}n,\, 1 + \mathbf{Z}n, ...,\, (n-1) + \mathbf{Z}n\}.$$

El conjunto $\mathbf{Z}/\mathbf{Z}n$ tiene, como sabemos, estructura de anillo. El problema[5] de cuándo existe un inverso aditivo (en lenguaje algebraico, cuando $a + \mathbf{Z}n$ es una *unidad* en $\mathbf{Z}/\mathbf{Z}n$) se resuelve de la siguiente forma:

Proposición.– Las siguientes condiciones son equivalentes:

- Existe un inverso multiplicativo de $a + \mathbf{Z}n$, esto es, existe $b \in \mathbf{Z}$ tal que

$$(a + \mathbf{Z}n)(b + \mathbf{Z}n) = 1 + \mathbf{Z}n.$$

- $\gcd(a, n) = 1$.

De aquí deducimos inmediatamente algo que ya sabíamos:

Corolario.– El anillo $\mathbf{Z}/\mathbf{Z}n$ es un cuerpo si y sólo si n es primo.

Dado un primo $p \in \mathbf{Z}$, el cuerpo $\mathbf{Z}/\mathbf{Z}p$, notado habitualmente $\mathbf{F}_p$ se denomina el *cuerpo primo de p elementos*. Es importante acostumbrarse a *pensar* en términos de aritmética modular (esto es, a operar en $\mathbf{Z}/\mathbf{Z}n$), porque es el contexto natural donde se modelizan la gran mayoría de los esquemas de encriptación modernos, como iremos viendo.

Aparte de los resultados que hemos visto en la sección anterior, y que se aplican directamente a $\mathbf{Z}$ por ser anillo euclídeo, hay algunas cuestiones propias de $\mathbf{Z}$ que nos interesa recordar ahora.

Definición.– La *función φ de Euler* se define como

$$\begin{array}{rcl} \varphi : \mathbf{N} & \longrightarrow & \mathbf{Z} \\ n & \longmapsto & \varphi(n) = \sharp\{0 < m < n \mid \gcd(n, m) = 1\} \end{array}$$

o dicho de otro modo, $\varphi(n)$ cuenta el número de unidades en $\mathbf{Z}/\mathbf{Z}n$.

Teorema.– La función φ verifica las siguientes propiedades:

- Si $\gcd(a, b) = 1$, $\varphi(ab) = \varphi(a)\varphi(b)$.
- Si $n = p_1^{r_1}...p_t^{r_t}$ es la descomposición de n en factores primos,

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)...\left(1 - \frac{1}{p_t}\right).$$

[5] Problema ¿para quién?

- (Teorema de Euler) Para todo $a \in \mathbf{Z}$ tal que $\gcd(a, n) = 1$ se tiene

$$a^{\varphi(n)} \equiv 1 \mod n.$$

Corolario.– (Pequeño Teorema de Fermat) Para todo p primo, se tiene que $\varphi(p) = p - 1$ y, por tanto,

$$a^{p-1} \equiv 1 \mod p$$

para todo a tal que $\gcd(a, p) = 1$. Por tanto,

$$a^p \equiv a \mod p, \ \forall a \in \mathbf{Z}.$$

Polinomios.

Fijemos un cuerpo k, que puede ser cualquiera. Los polinomios con coeficientes en k se definen, de la manera usual, como sumas formales finitas:

$$k[X] = \left\{ \sum_{i \in I} a_i X^i \mid a_i \in k, \ I \in \mathbf{N} \cup \{0\}, \ I \text{ finito} \right\}.$$

Como sabemos $k[X]$ es un anillo con la suma y el producto habituales, que de hecho contiene al cuerpo k y, además, es un dominio euclídeo gracias al Teorema de la División. Dado

$$f(X) = \sum_{i=0}^{\infty} a_i X^i,$$

definimos

$$\deg(f) = \max\{i \mid a_i \neq 0\}$$

cuando $f \neq 0$. De esta forma se verifica la fórmula del producto:

$$\deg(fg) = \deg(f) + \deg(g), \text{ para } f, g \neq 0.$$

Teorema de la División Polinómica.– Dados $f, g \in k[X]$, con $g \neq 0$, existen únicos $q, r \in k[X]$ tales que

$$f = gq + r, \text{ y } r = 0, \text{ o bien } \deg(r) < \deg(g).$$

Definición.– Dado $f \in k[X]$, una raíz de f es un elemento $\alpha \in k$ tal que $f(\alpha) = 0$.

Siendo estrictos, se puede hablar de raíces de f en cualquier extensión de f. En estas condiciones, si $\alpha \in K$, con $k \subset K$, y $f(\alpha) = 0$, diremos que α es una K–raíz de f. En cualquier caso, siempre se tiene que

$$f(\alpha) = 0 \iff (X - \alpha) \mid f(X).$$

Del mismo modo, independientemente de cuál sea el cuerpo K, con $k \subset K$, un polinomio de grado n no puede tener más de n K–raíces (eventualmente repetidas).

Como dominio euclídeo, se verifican el Teorema de Euclides, la Identidad de Bézout, el Teorema Chino del Resto y la factorización única en polinomios irreducibles. Pero ahora no está tan claro qué polinomios son irreducibles. De hecho, depende fuertemente del cuerpo en el que tomemos coeficientes. Lo que no depende, en modo alguno, es el conjunto de unidades, que es siempre k^*.

Proposición.– Sea $f(X) \in k[X]$.

1. Si $k = \mathbf{C}$, f es irreducible si y sólo si $\deg(f) = 1$ (Teorema Fundamental del Álgebra).
2. Si $k = \mathbf{R}$, f es irreducible si y sólo si $\deg(f) = 1$ o $\deg(f) = 2$ y las raíces de f son dos complejos no reales (conjugados, para más señas).

La irreducibilidad en $\mathbf{Q}[X]$ es un problema fascinante y complejo, y no lo es menos el caso de $\mathbf{F}_p[X]$, relacionado íntimamente, como veremos, con la clasificación de los cuerpos finitos.

Espacios vectoriales.

La estructura de espacio vectorial sobre un cuerpo k es de las más utilizadas y prácticas en las aplicaciones del álgebra. Por ello se estudia desde el principio de los programas de álgebra, a pesar de su complejidad intrínseca (posee un operación interna y otra externa, relacionadas por propiedades nada elementales). Supondremos que el alumno sabe, o al menos recuerda, o al menos recuerda que una vez supo, los siguientes conceptos básicos:

1. Espacio vectorial sobre un cuerpo k: (in)dependencia lineal, sistemas generadores, bases, dimensión.
2. Subespacio vectorial: generadores, sistemas de ecuaciones implícitas, operaciones básicas (suma, intersección, suma directa), espacios vectoriales cociente.
3. Homomorfismos: Matrices, isomorfismos, núcleo, imagen, Primer Teorema de Isomorfía, autovalores, autovectores.
4. Aplicaciones bilineales: Ortogonalidad, productos escalares.

Aparte de estos resultados muy generales, conviene tener presente un ejemplo que será recurrente en ciertas partes de la asignatura: dados dos cuerpos $k \subset K$, K es siempre un k–espacio vectorial (notemos que, al escribir $k \subset K$ estamos suponiendo implícitamente que las operaciones en k como cuerpo son las mismas que como subconjunto de K).

Cuerpos finitos.

Esta sección se puede resumir en el Teorema de Clasificación de los Cuerpos Finitos.

Teorema.– (T.C.C.F. I) Sea F un cuerpo finito. Entonces existe un primo $p \in \mathbf{Z}$ tal que $|F| = p^r$. Todo cuerpo con p^r elementos es isomorfo a F (como grupo abeliano). Cualquier elemento de esta familia de cuerpos isomorfos se denota $\mathbf{F}_{p^r}$.

Antes de probar el Teorema, debemos determinar y definir un concepto importante.

Definición.– Sea k un cuerpo. Definimos el homomorfismo de anillos

$$\begin{array}{rcl} ch : \mathbf{Z} & \longrightarrow & k \\ 1 & \longmapsto & 1_k \end{array}$$

Por el Primer Teorema de Isomorfía, existe un $n \in \mathbf{Z}$ tal que $\ker(ch) = \mathbf{Z}n$, y $\mathbf{Z}/\mathbf{Z}n \simeq \mathrm{Img}(ch) \subset k$. Este n puede ser 0 o no.

Si $n = 0$, k contiene una copia de $\mathbf{Z}$ (y por tanto de $\mathbf{Q}$, ya que es un cuerpo). Si no, como $\mathbf{Z}/\mathbf{Z}n$ debe ser un dominio, ha de ser n primo (llamemos $n = p$), luego k contiene una copia de algún $\mathbf{F}_p$. En el primer caso se dice que k es *de característica* 0, y en el segundo, *de característica* p.

Demostración.– Al ser F un cuerpo finito, ha de tener característica p, por lo que contiene a $\mathbf{F}_p$. Por tanto, es un $\mathbf{F}_p$–espacio vectorial, necesariamente de dimensión finita, luego es isomorfo a $\mathbf{F}_p^r$, para cierto $r \in \mathbf{N}$ y $|F| = p^r$.

Si tenemos un cuerpo con p^r elementos, es claro que ha de ser de característica p. Eso implica que ha de ser un $\mathbf{F}_p$–espacio vectorial de dimensión r, luego es isomorfo a F (como espacio vectorial y por tanto como grupo abeliano). □

Mucho más compleja es la segunda parte del Teorema

Teorema.– (T.C.C.F. II) Dado un primo p y un entero $r > 0$ existe siempre un cuerpo F tal que $|F| = p^r$. Dos de tales cuerpos son siempre isomorfos (como cuerpos)

Demostración.– Notemos, por abreviar $q = p^r$, y consideremos el polinomio $X^q - X \in \mathbf{F}_p[X]$. Claramente 0 es raíz del polinomio, y si tomamos $a \in \mathbf{F}_p^*$, tenemos

$$a^{q-1} = a^{p^r-1} = a^{\left[(p-1)\left(p^{r-1}+p^{r-2}+\ldots+1\right)\right]} = 1,$$

dado que $a^{p-1} = 1$, por el Pequeño Teorema de Fermat. Por tanto, si consideramos F, el conjunto de raíces de $X^q - X$ en una extensión de $\mathbf{F}_p$, tendremos que $\mathbf{F}_p \subset F$. Además, dadas dos raíces $\alpha, \beta \in F$ es inmediato (aunque aburrido) comprobar que $\alpha+\beta, \alpha\beta \in F$, utilizando intensivamente que, en característica p,

$$(\alpha + \beta)^{p^s} = \alpha^{p^s} + \beta^{p^s}.$$

Por tanto el conjunto F es un cuerpo que, además, contiene exactamente q elementos como buscábamos.

Además, se tiene que dos cuerpos de q elementos son cuerpos de descomposición de un mismo polinomio de $FF_p[X]$, con lo cual son isomorfos por un resultado general de Teoría de Galois[6]. □

Observación.– Un problema importante cuando trabajamos con un cuerpo finito $\mathbf{F}_q$ es cómo representamos los elementos de dicho cuerpo. Hasta ahora tenemos una opción

[6]No parece tan complicado, ¿verdad? ¿Podrías localizar dónde hemos usado un argumento no justificado?

evidente: dado que $\mathbf{F}_q \simeq \mathbf{F}_p^r$ podemos representar todo elemento de $\mathbf{F}_q$ como una r–upla de elementos de $\mathbf{F}_p$. Eso funciona, y además permite sumar elementos con facilidad, pero no nos permite multiplicar ya que la estructura de espacio vectorial no contempla el producto interno.

Teorema.– Dado un cuerpo finito $\mathbf{F}_q$, el grupo de las unidades $(\mathbf{F}_q^*, \cdot)$ es un grupo cíclico.

Demostración.– Dado un elemento $x \in \mathbf{F}_q$, denotamos

$$o(x) = \min\{n \mid x^n = 1\},$$

y observemos que $o(x)$ siempre es un número entre 0 y $q-1$. Consideremos entonces[7]

$$n = \text{lcm}\{o(x) \mid x \in \mathbf{F}_q^*\} = p_1^{\alpha_1} ... p_r^{\alpha_r}.$$

La factorización nos indica que existen elementos $x_1, ..., x_r$ tales que

$$o(x_i) = \beta_i p_i^{\alpha_i},$$

lo cual implica que

$$o\left(x_i^{\beta_i}\right) = p_i^{\alpha_i},$$

y, por tanto

$$o\left(x_1^{\beta_1} ... x_r^{\beta_r}\right) = p_1^{\alpha_1} ... p_r^{\alpha_r} = n,$$

por lo que basta demostrar que $n = q-1$ para probar el resultado. Por un lado, es claro que $n \leq q-1$, ya que hay un elemento de orden n. Pero además, por definición de n se tiene que

$$x^n = 1, \text{ para todo } x \in \mathbf{F}_q,$$

con lo que el polinomio $X^n - 1 \in \mathbf{F}_q[X]$ tiene, al menos $q-1$ raíces y por tanto $n \geq q-1$. □

Definición.– Un elemento $x \in \mathbf{F}_q^*$ que genere $\mathbf{F}_q^*$ como grupo cíclico multiplicativo se denomina un elemento primitivo de $\mathbf{F}_q$.

Para aquéllos que recuerden otro concepto de elemento primitivo (proveniente de la Teoría de Galois), les traemos un resultado tranquilizador.

Proposición.– Sea $\alpha \in \mathbf{F}_q^*$ un elemento primitivo. Entonces $\mathbf{F}_q = \mathbf{F}_p[\alpha]$.

Demostración.– Directa: si α generase un subcuerpo de la extensión $\mathbf{F}_p \subset \mathbf{F}_q$, no podría ser $o(\alpha) = q-1$.

Los elementos primitivos nos permiten ampliar nuestro espectro de representaciones para $\mathbf{F}_q$. Una, directa, es la siguiente: supongamos que $\alpha \in \mathbf{F}_q$ es un elemento primitivo. Entonces todo $x \in \mathbf{F}_q$ es necesariamente de la forma $x = \alpha^s$ para cierto $s \in \{0, ..., q-1\}$. Esta representación funciona muy bien con el producto, como es obvio. No es tan sencillo, sin embargo, obtener fórmulas simples para la suma.

Así pues, si queremos operaciones sencillas debemos optar por representaciones algo más complejas. La primera, denominada *representación polinomial*, proviene directamente de la Teoría de Galois. En efecto, consideramos la extensión $\mathbf{F}_p \subset \mathbf{F}_q$, de grado r,

[7] No hemos definido el mínimo común múltiplo (lcm), pero entedemos que el alumno está familiarizado con el concepto y sus propiedades básicas.

y un elemento primitivo α como antes. Hallamos su polinomio mínimo sobre $\mathbf{F}_p$, que ha de tener necesariamente grado r:

$$f(X) = X^r + a_{r-1}X^{r-1} + ... + a_1X + a_0, \quad a_i \in \mathbf{F}_p.$$

Entonces, por la teoría general de extensiones algebraicas, tenemos un isomorfismo de cuerpos

$$\mathbf{F}_q \simeq \mathbf{F}_p[X]/\langle f \rangle.$$

Con esto, todo elemento de $\mathbf{F}_q$ se puede ver como un polinomio de $\mathbf{F}_p[X]$ de grado menor que r, y las operaciones de suma y producto son las usuales de polinomios (eventualmente, tomando módulo $f(X)$). O, equivalentemente, como un polinomio en α de grado menor o igual que r, teniendo en cuenta que $f(\alpha) = 0$ a la hora de operar.

Una opción más sofisticada es considerar la denominada *matriz compañera* de $f(X)$, que no es sino

$$M_f = \begin{pmatrix} 0 & 0 & ... & 0 & -a_0 \\ 1 & 0 & ... & 0 & -a_1 \\ 0 & 1 & ... & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & ... & 1 & -a_{r-1} \end{pmatrix}.$$

Ejercicio.– Esta matriz se corresponde con el endomorfismo de $\mathbf{F}_p$–espacios vectoriales

$$\begin{array}{rccc} \mu_\alpha : \mathbf{F}_q & \longrightarrow & \mathbf{F}_q \\ \beta & \longmapsto & \alpha\beta \end{array}$$

y su polinomio característico es $\pm f(X)$.

Por el Teorema de Hamilton–Cayley, tenemos que $f(M_f) = 0$ por lo cual, formalmente, también podemos asociar cada elemento de $\mathbf{F}_q$ con una expresión polinómica en M_f con coeficientes e $\mathbf{F}_p$ y grado menor que r. La ventaja ahora es que las reglas de la suma y el producto son las reglas usuales del álgebra matricial, y que todas las entradas de M_f son enteros módulo p.

Complejidad.

La idea fundamental de la teoría de la complejidad es poder dar una estimación del tiempo que va a tardar en completarse un algoritmo. Esta estimación se realiza, esencialmente, contando *operaciones elementales*. En su forma más estricta, una operación elemental es una suma de bits. Esto es

$$0 + 0 = 0, \quad 1 + 0 = 1, \quad 0 + 1 = 1, \quad 1 + 1 = 10$$

Por este motivo, es habitual trabajar en base 2 cuando tratamos con asuntos de complejidad. A la hora de manejar estas estimaciones temporales debemos tener en cuenta, obviamente, la magnitud de los datos iniciales del problema. No es lo mismo

$$1 + 1$$

que

$$10101010101010101010111111110100 + 10110011100000000011110101010101011010.$$

Por ello se suele adoptar, en el contexto de la computación con números enteros, una medida del tamaño de los datos que no es sino el número de cifras. O, lo que es lo mismo, dado $x \in \mathbf{Z}$,

$$[\log_2(x)] + 1,$$

aunque en la práctica se suele tomar simplemente $[\log_2(x)]$.

Ejemplo.– Calcular el número de operaciones que implica sumar y multiplicar dos enteros de k cifras (binarias).

A la hora de contar operaciones, sin embargo, no podemos garantizar (salvo en casos muy especiales) que nuestro algoritmo sea *el mejor* posible para realizar el cálculo deseado. En realidad, al dar un algoritmo estamos dando una cota superior para el número de operaciones que se necesitan. En estas condiciones es habitual utilizar la siguiente notación:

$$f = \mathcal{O}(g).$$

Esta notación es una forma abreviada de decir

$$\exists C \in \mathbf{R}_+ \mid \forall n \in \mathbf{N},\ f(n) < C \cdot g(n).$$

Dicho de otro modo, el límite de f/g cuando tendemos a infinito es un número real (que puede ser eventualmente 0). Así, cuando $f(n)$ está contando el número de operaciones necesarias para efectuar un cálculo sobre enteros de *tamaño* n lo que estamos diciendo es que el algoritmo en cuestión necesita (en el peor de los casos) un múltiplo de $g(n)$ operaciones.

Otras notaciones relacionadas con ésta y utilizadas, sobre todo en textos no demasiado modernos, son

$$f = o(g) \iff \lim_{x\to\infty} \frac{f}{g} = 0$$

$$f \sim g \iff \lim_{x\to\infty} \frac{f}{g} = 1$$

Proposición.– Se verifican las siguientes propiedades:

1. Si $f_1 = \mathcal{O}(g_1)$ y $f_2 = \mathcal{O}(g_2)$, entonces $f_1 f_2 \in \mathcal{O}(g_1 g_2)$.
2. $f \cdot \mathcal{O}(g) \subset \mathcal{O}(fg)$.
3. Si $f \in \mathcal{O}(g)$, para todo $k \in \mathbf{R}_+$, $kf \in \mathcal{O}(g)$.
4. Si $f \in \mathcal{O}(g)$, para todo $k \in \mathbf{R}_+$, $f \in \mathcal{O}(kg)$.
5. Si $f_1, f_2 \in \mathcal{O}(g)$, entonces $f_1 + f_2 \in \mathcal{O}(g)$.

Tradicionalmente los algoritmos se dividen, en función de su eficiencia en:

1. Polinomiales: aquéllos que requieren un número de operaciones de $\mathcal{O}(\log^m(n))$, para cierto $m \in \mathbf{N}$.

2. Exponenciales: aquéllos que requieren un número de operaciones de $\mathcal{O}(n^m)$, para cierto $m \in \mathbf{N}$.

3. Subexponenciales: aquéllos que requieren (tal vez) más operaciones que uno polinomial, y (con seguridad) menos que uno exponencial.

Veremos ejemplos de las tres familias a lo largo del curso. En realidad, la base operativa de la criptografía en clave privada es encontrar algoritmos de codificación y decodificación que sean polinomiales conocida la clave y exponenciales cuando no se conoce. En cambio, la base de la criptografía en clave pública es encontrar operaciones que verifiquen que se puedan realizar en tiempo polinomial pero que requieran tiempo exponencial para deshacerse (ya veremos qué quiere decir *deshacer*).

Observación.– Notemos que, por las propiedades del logaritmo, podemos ya utilizar cualquier base para hablar de tamaño y complejidad, ya que la cantidad de dígitos en dos bases distintas son siempre proporcionales.

Observación.– Evidentemente podemos tratar con algoritmos con entradas de datos más complejas como polinomios, o simplemente varios enteros. En el primer caso, hay varias medidas posibles para el tamaño de un polinomio, pero tradicionalmente se utiliza un par de enteros:

$$f = \sum_{i=0}^{d} a_i x^i \implies f \simeq \left(\max_i \{[\log(a_i)]\}, d \right).$$

En el segundo caso, si tenemos varios enteros (o, análogamente, varios polinomios), la opción más habitual suele ser considerar el tamaño máximo entre los datos de entrada.

Observación.– En la práctica, esta forma de contar operaciones puede no ser un reflejo fiel del tiempo real que tarda un algoritmo. Detalles de enorme importancia desde el punto de vista de la eficacia, como son la cantidad de memoria que se necesita, el tiempo que se pierde en operaciones de reescritura o la gestión de variables pueden hacer que un algoritmo teóricamente menos complejo que otro resulte mucho más lento cuando los ponemos a calcular de manera explícita.

Ejercicios Tema 1

Ejercicio 1.01.– Hallar todos los posibles grupos abelianos de 40 elementos.

Ejercicio 1.02.– Hallar todos los grupos abelianos de orden 60 que:

1. Contengan un grupo cíclico de orden 6.
2. No contengan un grupo cíclico de orden 4.

Ejercicio 1.03.– Sea A un dominio euclídeo, $I = \langle x \rangle \subset A$ un ideal. Entonces

$$A/I \text{ es dominio} \iff A/I \text{ es cuerpo} \iff x \text{ es irreducible.}$$

Ejercicio 1.04.– Hallar un polinomio $f(X) \in \mathbf{F}_5[X]$ que verifique:

- $\deg(f) = 6$.
- Su resto al dividir entre $X^2 + 2$ es $X + 1$.
- $f(2) = 4$.

Ejercicio 1.05.– Probar que las siguientes condiciones son equivalentes:

- Existe un inverso multiplicativo de $a + \mathbf{Z}n$, esto es, existe $b \in \mathbf{Z}$ tal que
$$(a + \mathbf{Z}n)(b + \mathbf{Z}n) = 1 + \mathbf{Z}n.$$
- $\gcd(a, n) = 1$.

Ejercicio 1.06.– Probar el Teorema de Wilson:

$$p \text{ es primo} \iff (p-1)! \equiv -1 \mod p$$

Ejercicio 1.07.– Sea $N \in \mathbf{Z}$ del cual conocemos $\varphi(N)$ y también que es producto de dos primos. Dar un procedimiento para hallar la factorización de N.

Ejercicio 1.08.– Sea $m \in \mathbf{Z}$. Probar que

$$\sharp\{n \in \mathbf{Z} \mid \varphi(n) = m\} < \infty.$$

Ejercicio 1.09.– Probar que, para todo $n \in \mathbf{Z}_{\geq 0}$, $30 | (n^5 - n)$.

Ejercicio 1.10.– Usando el Pequeño Teorema de Fermat, dar un polinomio de $\mathbf{F}_p[X]$ sin raíces en $\mathbf{F}_p$.

Ejercicio 1.11.– Construir explícitamente $\mathbf{F}_4$ y dar sus elementos en forma matricial.

Ejercicio 1.12.– Construir explícitamente $\mathbf{F}_9$ y dar sus elementos en forma polinómica.

Ejercicio 1.13.– Representando $\mathbf{F}_9$ como $\mathbf{F}_3[\alpha]$, donde α verifica $\alpha^2 + 1 = 0$, discutir, en función del parámetro a, la existencia de soluciones del siguiente sistema, y en caso afirmativo, hallarlas:

$$\left\{ \begin{array}{rcrcl} (\alpha+2)X & + & (2\alpha+1)Y & = & 2 \\ aX & + & (\alpha+1)Y & = & 0 \end{array} \right.$$

Ejercicio 1.14.– Responder las siguientes cuestiones:

1. En el $\mathbf{F}_5$–espacio vectorial $\mathbf{F}_5^3$ determinar si los siguientes elementos son linealmente independientes o no:
$$\langle (1,4,2),\ (3,3,1),\ (4,0,4)\rangle, \qquad \langle (1,4,1),\ (3,3,3),\ (4,0,4)\rangle$$
2. Se considera el cuerpo $\mathbf{F}_{49} = \mathbf{F}_7[x]/\,(x^2+2x+2)$, donde expresaremos los elementos de forma matricial. En el $\mathbf{F}_{49}$–espacio vectorial $\mathbf{F}_{49}^2$, estudiar si
$$\langle (2M_f, M_f+1)\rangle = \langle (2M_f+5, 5M_f+1)\rangle.$$

Ejercicio 1.15.– Sea $n \in \mathbf{Z}$. Probar que convertir n a notación binaria tiene una complejidad $\mathcal{O}(\log^2 n)$.

Ejercicio 1.16.– Sean $a, n, m \in \mathbf{Z}_{\geq 0}$. Hallar la complejidad de calcular $a^2 \mod m$ y $a^{2^n} \mod m$.

Ejercicio 1.17.– Sean $a, n, m \in \mathbf{Z}_{\geq 0}$.

1. Hallar la complejidad de calcular $a^n \mod m$ por el método que os parezca más simple.
2. Hallar la complejidad de calcular $a^n \mod m$ por un método que comience calculando la expresión binaria de n.

Ejercicio 1.18.– Sean $a, b \in \mathbf{Z}$, con $a > b$.

1. Probar que la división euclídea de a entre b (esto es, hallar cociente y resto) tiene complejidad $\mathcal{O}(\log^2 a)$.
2. Probar que hallar $\gcd(a,b)$ usando el algoritmo de Euclides tiene complejidad $\mathcal{O}(\log^3 a)$.
3. Probar que hallar los coeficientes de la Identidad de Bezout tiene complejidad $\mathcal{O}(\log^3 a)$.

Ejercicio 1.19.– Sea $n \in \mathbf{Z}$. Probar que calcular $n!$ tiene complejidad $\mathcal{O}(n^2 \log^2 n)$.

Ejercicio 1.20.– Calcular la complejidad de resolver un sistema de congruencias por el Teorema Chino del Resto.

Tema 2. Códigos correctores de errores

Códigos correctores y códigos detectores.

Fijaremos un alfabeto $\mathcal{A}$, que no es más que un conjunto de símbolos sin significado individual. Un código en bloque de n caracteres no es más que un subconjunto de $\mathcal{A}^n$, esto es, un conjunto de palabras de exactamente n letras en el alfabeto $\mathcal{A}$.

Ejemplo.– El código ASCII original consistía en un código en bloque de 8 caracteres binarios. Esto es, un subconjunto de $\mathbf{F}_2^8$ (ojo, sin estructura de espacio vectorial, únicamente como conjunto). Cada caracter del código ASCII, algunos imprimibles como `a` o `@` y otros no, como *Fin de línea* o *Tabulación horizontal*, tenían asignado un número entre 0 y 127, que corresponde a las 7 primeras cifras binarias. La última cifra se obtiene sumando las 7 primeras (esto se denominaba clásicamente *bit de paridad*). Así, por ejemplo, la `a` correspondía al 97, o sea:

$$\texttt{a} = 1100001 \Longrightarrow \text{Código ASCII de } \texttt{a} = 11000011$$

Es sencillo ver que si en el código ASCII cometemos un error al copiar o transmitir un carácter (o sea, 8 bits), ese error se detectará por la presencia del bit de paridad. Es un ejemplo rudimentario de detección de errores, esto es, *cómo el añadir información redundante a las palabras de un código puede resultar útil ante la presencia de errores en la transmisión*.

Definición.– Dado un código $\mathcal{C} \subset \mathcal{A}^n$, llamamos:

- Longitud del código al número n.
- Tamaño del código al número $|\mathcal{C}|$.

Dentro de $\mathcal{A}^n$ podemos definir una métrica en el sentido usual del término, que nos permite dotar al conjunto de una topología que nos será de utilidad.

Definición.– Definimos la métrica o distancia de Hamming en $\mathcal{A}^n$ como

$$\begin{array}{rcl} d : \mathcal{A}^n \times \mathcal{A}^n & \longrightarrow & \mathbf{R}_+ \\ (\underline{x}, \underline{v}) & \longmapsto & \sharp\{i \mid x_i \neq v_i,\ i = 1, ..., n\} \end{array}$$

esto es, $d(\underline{x}, \underline{v})$ mide el número de coordenadas distintas que poseen $\underline{x}$ y $\underline{v}$.

Ejercicio.– Probar que la distancia de Hamming es una distancia, en el sentido topológico.

Observación.– Para el caso más frecuente, cuando $\mathcal{A} = \mathbf{F}_2$, podemos definir equivalentemente

$$d(\underline{x}, \underline{v}) = \sum_{i=1}^{n} |x_i - v_i|$$

Intuitivamente, estamos interesados en códigos donde las palabras estén *cuanto más lejos mejor*. Esto nos permitirá poder detectar (y eventualmente corregir) errores, ya que si las palabras del código están muy separadas las unas de las otras, si se cometen (no demasiados) errores al transmitir una palabra, la transmisión errónea quedará, o eso esperamos, cerca únicamente de la palabra correcta, de entre todas las del código. Esto se conoce como el principio de la distancia mínima (*maximum likelihood decoding rule* en inglés) y es la base de la decodificación en códigos correctores de errores.

Ejercicio.– Sea $\mathcal{C} \subset \mathcal{A}^n$ un código, $\underline{x} \in \mathcal{C}$, $\underline{v}, \underline{z} \notin \mathcal{C}$ verificando $d(\underline{x}, \underline{v}) < d(\underline{x}, \underline{z})$. Supongamos que la probabilidad de que se produzca un error en la transmisión de $\underline{x}$ es $0 < p < 1/2$. Probar que la probabilidad de transmitir $\underline{x}$ y recibir $\underline{v}$ es estrictamente menor que la probabilidad de transmitir $\underline{x}$ y recibir $\underline{z}$.

Ejemplo.– El código de repetición de longitud n sobre el alfabeto $\mathcal{A}$ consiste en

$$\mathcal{C} = \{(a, a, ..., a) \mid a \in \mathcal{A}\} \subset \mathcal{A}^n.$$

Ejemplo.– El código de la cinta de papel de n agujeros consiste en

$$\mathcal{C} = \left\{ (x_1, x_2, ..., x_n) \mid x_i \in \mathbf{F}_2, \ \sum_{i=1}^{n} x_i = 0 \right\} \subset \mathbf{F}_2^n,$$

y es una generalización del código ASCII (x_n es aquí el bit de paridad).

Si calculamos las distancias mínimas entre estos ejemplos, notaremos que hay grandes diferencias entre ellos. Y, si pensamos un poco, descubriremos que la distancia mínima parece crecer inversamente a la cantidad de información transmitida. Por ejemplo, en el código de repetición tenemos una distancia mínima bastante grande (tanta como queramos, si podemos hacer crecer n), pero a cambio la información transmitida en cada palabra del código es escasa. Sin embargo, en la cinta de papel, los $n - 1$ primeros bits son información pura (dado que pueden variar como queramos) y sólo el último no lo es. Sin embargo, la distancia mínima en este código es siempre 1.

Definición.– Supongamos que $|\mathcal{A}| = q$. Dado un código $\mathcal{C} \subset \mathcal{A}^n$, llamamos:

- Distancia mínima del código al número

$$d(\mathcal{C}) = \min\{d(\underline{x}, \underline{v}) \mid \underline{x}, \underline{v} \in \mathcal{C}\}.$$

- Distancia mínima relativa del código al número

$$\delta(\mathcal{C}) = d(\mathcal{C})/n.$$

- Tasa de transmisión de información del código al número

$$R(\mathcal{C}) = \log_q(|\mathcal{C}|)/n.$$

- Redundancia del código al número

$$n - \log_q(|\mathcal{C}|).$$

Normalmente diremos que $\mathcal{C}$ es un código tipo (n, m, d) para decir que es un código de longitud n, tamaño m y distancia mínima d. Las demás constantes relacionadas con el código (denominadas *parámetros del código*) se pueden calcular a partir de éstas (conocido $\mathcal{A}$, claro está).

Lema.– Sea $\mathcal{C}$ un código tipo (n, m, d). Entonces, para todo $\underline{x} \in \mathcal{C}$, $\underline{v} \in \mathcal{A}^n$, si $d(\underline{x}, \underline{v}) = h < d$, se tiene que $\underline{v} \notin \mathcal{C}$.

Diremos que un código $\mathcal{C}$ detecta $d - 1$ errores cuando se da esta situación.

Demostración.– Ejercicio. □

Lema.– Sea $\mathcal{C}$ un código tipo (n, m, d). Entonces, para cualesquiera $\underline{x}_1, \underline{x}_2 \in \mathcal{C}$, $\underline{v} \in \mathcal{A}^n$, no es posible que

$$d(\underline{x}_i, \underline{v}) \leq \left[\frac{d-1}{2}\right], \quad i = 1, 2.$$

Dicho de otro modo,

$$\overline{B}\left(\underline{x}_1, \left[\frac{d-1}{2}\right]\right) \cap \overline{B}\left(\underline{x}_2, \left[\frac{d-1}{2}\right]\right) = \emptyset.$$

Diremos que un código $\mathcal{C}$ corrige $[(d-1)/2]$ errores cuando se da esta situación.

Demostración.– Ejercicio. □

Ejemplo.– El código original de Hamming, el primer ejemplo de la historia de código corrector de errores pensado como tal, era en un código $(7, 2^4, 3)$ sobre $\mathcal{A} = \mathbf{F}_2$. El código $\mathcal{C}$ era el conjunto de soluciones en $\mathbf{F}_2^7$ del sistema

$$\begin{cases} x_1 & + & x_3 & + & x_5 & + & x_7 & = & 0 \\ x_2 & + & x_3 & + & x_6 & + & x_7 & = & 0 \\ x_4 & + & x_5 & + & x_6 & + & x_7 & = & 0 \end{cases}$$

Es sencillo ver que podemos tomar x_3, x_5, x_6, x_7 como parámetros y los valores de x_1, x_2, x_4 vienen determinados por nuestra elección. La distancia mínima relativa es $3/7$ y la redundancia de este código es 3, siendo su tasa de transmisión $4/7$. Es un código que detecta dos errores y corrige uno.

Visto lo anterior la teoría de códigos correctores presenta algunos problemas interesantes, desde el punto de vista matemático:

- Encontrar un compromiso adecuado entre los errores que se pretenden detectar o corregir y la tasa de información, que está directamente involucrada con el coste de transmisión (a mejor tasa, menos cuesta enviar el mismo mensaje).
- Estudiar errores de distinta naturaleza, para adecuar los códigos a los errores más comunes, dependiendo del tipo de transmisión (ejemplo, ISBN vs. CD).
- Dar, si es posible, un algoritmo eficiente para detectar y corregir los errores de transmisión.

Códigos lineales.

A partir de ahora trabajaremos con el espacio vectorial $\mathbf{F}_q^n$, donde notaremos sus vectores como *columnas*.

Definición.– Un código lineal de longitud n sobre el alfabeto $\mathcal{A} = \mathbf{F}_q$ es un subespacio vectorial de $\mathbf{F}_q^n$.

Observación.– Un código lineal $\mathcal{C} \subset \mathbf{F}_q^n$, como subespacio vectorial que es, tiene una dimensión, pongamos $\dim(\mathcal{C}) = k$. En estas condiciones es claro que $\mathcal{C}$ es un código tipo (n, q^k, d) (no sabemos quién es d, al menos por ahora), con $R(\mathcal{C}) = k/n$ y redundancia $n-k$. Esto es, de las n coordenadas que tiene cada palabra de $\mathcal{C}$, k contienen información, lo cual es perfectamente coherente con el hecho de que $\dim(\mathcal{C}) = k$.

Ejercicio.– Estudia cuáles de los ejemplos que hemos visto en la sección anterior son códigos lineales y determina su dimensión.

Para hallar la distancia mínima de un código lineal nos apoyamos en un concepto auxiliar. Dado $\underline{x} \in \mathbf{F}_q^n$ definimos su peso como

$$w(\underline{x}) := \sharp\{i \mid x_i \neq 0\} = d(\underline{x}, \underline{0}).$$

Proposición.– Si $\mathcal{C}$ es un código lineal tipo (n, q^k, d), entonces

$$d = \min_{\underline{0} \neq \underline{x} \in \mathcal{C}} w(\underline{x}).$$

Demostración.– Basta aplicar que, para todo $\underline{x}, \underline{v} \in \mathcal{C}$, $\underline{x} - \underline{v} \in \mathcal{C}$, y

$$d(\underline{x}, \underline{v}) = d(\underline{x} - \underline{v}, \underline{0}) = w(\underline{x} - \underline{v}).$$

□

Falta por dar una manera explícita (a ser posible sencilla) de calcular la distancia mínima de un código lineal. Para ello recurriremos a la artillería del álgebra lineal y, en particular, a las dos maneras más habituales de expresar un subespacio vectorial: mediante una base y mediante un sistema de ecuaciones implícitas.

Definición.– Sea $\mathcal{C} \subset \mathbf{F}_q^n$ un código lineal. Diremos que M es una matriz generatriz de $\mathcal{C}$ cuando las columnas de M formen una base de $\mathcal{C}$ como subespacio vectorial.

Definición.– Sea $\mathcal{C} \subset \mathbf{F}_q^n$ un código lineal. Si $\mathcal{C}$ viene expresado por un sistema de ecuaciones implícitas independientes $AX = 0 \in \mathcal{M}((n-k) \times 1, \mathbf{F}_q)$, diremos que A es una matriz de control de $\mathcal{C}$.

Proposición.– Sea $\mathcal{C} \subset \mathbf{F}_q^n$ un código lineal tipo (n, q^k, d) con matriz generatriz M y matriz de control A. Entonces:

- $M \in \mathcal{M}(n \times k, \mathbf{F}_q)$, $A \in \mathcal{M}((n-k) \times n, \mathbf{F}_q)$
- $\text{rg}(M) = k$, $\text{rg}(A) = n - k$.
- $A \cdot M = 0 \in \mathcal{M}((n-k) \times k, \mathbf{F}_q)$.
- Dado un vector $\underline{x} \in \mathbf{F}_q^n$,

$$\underline{x} \in \mathcal{C} \iff A\underline{x} = 0 \in \mathcal{M}((n-k) \times 1, \mathbf{F}_q) \iff \text{rg}(M) = \text{rg}(M|\underline{x}).$$

Demostración.– Todas las afirmaciones son inmediatas. □

Proposición.– La distancia mínima de $\mathcal{C}$ es la menor cantidad de columnas de A que necesitamos para formar un conjunto linealmente dependiente.

Demostración.– Si podemos encontrar r columnas de A linealmente dependientes, tendremos una combinación de las columnas con r coeficientes distintos de cero (o tal vez menos, pero alguno). Pongamos

$$\alpha_1 C(A)_1 + ... + \alpha_n C(A)_n = \underline{0} \in \mathbf{F}_q^{n-k}.$$

Entonces tenemos que $(\alpha_1, ..., \alpha_n) \in \mathcal{C}$ y además esta palabra de $\mathcal{C}$ posee como mucho r coordenadas distintas de cero, con lo cual

$$d \leq w\,(\,(\alpha_1, ..., \alpha_n)\,) \leq r.$$

Análogamente se puede probar que d es precisamente el cardinal del *menor* conjunto de columnas de A que formen un conjunto linealmente dependiente. □

Ejercicio.– Calcular las matrices asociadas a los códigos de repetición, de la cinta de papel y de Hamming y estudiar la distancia mínima en cada caso.

La estructura extra que nos proporcionan los códigos lineales nos permite una decodificación sistemática, siguiendo el principio de distancia mínima. Supongamos que tenemos un código lineal $\mathcal{C} \subset \mathbf{F}_q^n$, con matriz de control A, y que hemos emitido una palabra $\underline{x} \in \mathcal{C}$, recibiendo en cambio

$$\underline{v} = \underline{x} + \underline{e},$$

donde $\underline{e}$ es el error de la transmisión.

Definición.– Llamamos síndrome de $\underline{v}$ al vector

$$s(\underline{v}) = A\underline{v} \in \mathbf{F}_q^{n-k}.$$

Observación.– Algunas propiedades elementales del síndrome son:

- $s(\underline{z}) = \underline{0} \iff \underline{z} \in \mathcal{C}$.
- Si $\underline{v} = \underline{x} + \underline{e}$, $s(\underline{v}) = s(\underline{e})$.
- El síndrome de un vector es una combinación lineal de las columnas de A correspondientes a los lugares en los que se ha cometido un error de transmisión.

En realidad, pues, el síndrome no es sino la aplicación lineal $s : \mathbf{F}_q^n \longrightarrow \mathbf{F}_q^{n-k}$, definida por la matriz A, que verifica $\ker(s) = \mathcal{C}$.

Ejemplo.– Consideramos el código original de Hamming, que tiene por matriz de control

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Supongamos que hemos emitido una palabra del código y hemos recibido 1010111 (que no está en el código). Entonces, haciendo

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

que es precisamente la sexta columna de A. Por tanto (suponiendo que sólo hayamos cometido un error), el error está en la sexta coordenada, luego la palabra original era

$$1010111 + 0000010 = 1010101.$$

Es sencillo interpretar el síndrome en términos de espacios cociente.

Observación.– Dado un código lineal $\mathcal{C} \subset \mathbf{F}_q^n$, de tamaño q^m, consideramos el espacio $\mathbf{F}_q^n/\mathcal{C}$, que es un espacio vectorial de dimensión $n - m$. Sus elementos son las clases

$$\underline{x} + \mathcal{C} = \{\underline{x} + \underline{u} \mid \underline{u} \in \mathcal{C}\},$$

cada una de ellas formada por q^m elementos. Dos vectores $\underline{u}, \underline{v}$ están en la misma clase si y sólo si

$$\underline{u} - \underline{v} \in \mathcal{C} \iff s(\underline{u}) = s(\underline{v}).$$

Por tanto en los elementos de una clase aparecen los vectores que provienen de cometer el mismo error de transmisión.

Definición.– Diremos que un elemento $\underline{u}$ es líder de una clase $\mathbf{F}_q^n/\mathcal{C}$ cuando sea *el único elemento de peso mínimo*.

Obviamente, no tiene por qué haber un único elemento de peso mínimo, pero, en la práctica, en las clases que nos interesan, sí que hay uno y sólo uno.

Proposición.– Sea $\mathcal{C} \subset \mathbf{F}_q^n$ un código lineal tipo (n, q^m, d). Sea $\underline{v} \in \mathbf{F}_q^n$ tal que $w(\underline{v}) \leq [(d-1)/2]$. Entonces, para todo $\underline{u} \in \underline{v} + \mathcal{C}$ se tiene que $w(\underline{u}) > [(d-1)/2]$.

Demostración.– Supongamos que existen dos vectores $\underline{u}, \underline{v}$, en la misma clase y con peso $t \leq [(d-1)/2]$. Entonces $\underline{u} - \underline{v} \in \mathcal{C}$ y

$$d(\underline{u}, \underline{v}) = w(\underline{u} - \underline{v}) \leq w(\underline{u}) + w(\underline{v}) \leq 2t < d(\mathcal{C}),$$

por lo que $d(\underline{u}, \underline{v}) = 0$ y $\underline{u} = \underline{v}$. □

Esta situación propicia el denominado *algoritmo del líder* para decodificar:

Paso 1. (Este paso sólo hay que realizarlo una vez). Se construye un tabla de doble entrada donde aparecen todos los vectores de peso menor o igual que $[(d-1)/2]$ y, para cada uno de ellos, su sídrome, que es en realidad el de la clase que determinan y de la cual el vector que hemos escogido es necesariamente el líder. Notemos que para vectores con pesos mayores la clase que determinan puede o no tener un líder.

Paso 2. Para descodificar $\underline{z}$ hallamos $s(\underline{z}) = A\underline{z}$, localizamos la clase correspondiente y su entrada en la tabla, que será del tipo $(s(\underline{z}), \underline{a})$, para cierto $\underline{a} \in \mathbf{F}_q^n$, o bien $s(\underline{z})$ no estará en la tabla.

Paso 3. En el primer caso la decodificación es $\underline{z} \longmapsto \underline{z} - \underline{a}$. En el segundo, la decodificación falla.

Como es obvio, la necesidad de mantener una tabla como la creada en el paso 1 implica que este método no es práctico para códigos de gran tamaño o longitud, como los que se demandan en muchas aplicaciones actuales. Sin embargo, no es menos cierto que la gran mayoría de los códigos actuales no son sino ejemplos (familias) de códigos lineales para los cuales existe una variación creativa del algoritmo del líder que permite decodificar de manera eficaz.

Observación.– La proposición anterior nos muestra que para crear la tabla podemos tomar los elementos de $\overline{B}(\underline{0}, [(d-1)/2])$ y considerar los síndromes correspondientes, ya que forzosamente ellos serán líderes de las clases en las que, con seguridad, podremos decodificar. Sin embargo, puede haber otras clases con líder, siendo éste de peso mayor que $[(d-1)/2]$. Calcular estas clases y añadirlas a la tabla pueden suponer una capacidad extra de corrección, a costa de aumentar la memoria utilizada.

Códigos cíclicos.

Definición.– Un código cíclico es un código lineal $\mathcal{C} \subset \mathbf{F}_q^n$ que verifica la siguiente propiedad:

$$(x_1, ..., x_n) \in \mathcal{C} \iff (x_n, x_1, ..., x_{n-1}) \in \mathcal{C}.$$

Observación.– Por motivos que después resultarán obvios, la notación habitual cambiará y consideraremos los elementos de $\mathbf{F}_q^n$ escritos como

$$\underline{x} = (x_0, x_1, ..., x_{n-1}).$$

Observación.– En lo sucesivo supondremos $\gcd(n, q) = 1$ para poder aplicar algunas de las propiedades más interesantes de los códigos cíclicos.

La descripción de los códigos cíclicos se hace en dos contextos distintos, que normalmente se usan de manera indiferenciada:

- Interpretación en el espacio vectorial de los polinomios: Identificamos $\mathbf{F}_q^n$ con $\mathbf{F}_q[X]_{(n-1)}$, el conjunto de polinomios sobre $\mathbf{F}_q$ con grado menor o igual que $n-1$. Esta identificación es, explícitamente:

$$\underline{u} = (u_0, u_1, ..., u_{n-1}) \longleftrightarrow u_0 + u_1 X + u_2 X^2 + ... + u_{n-1} X^{n-1}.$$

- Interpretación en el anillo cociente de polinomios: Alternativamente, interpretamos $\mathbf{F}_q^n \simeq \mathbf{F}_q[X]_{(n-1)}$ con el anillo $\mathbf{F}_q[X]/\langle X^n - 1\rangle$, usando las congruencias habituales (que no son sino polinomios de grado menor o igual que $n-1$). En esta interpretación, los códigos cíclicos verifican una propiedad curiosa.

Proposición.– Sea $\mathcal{C} \subset \mathbf{F}_q^n$ un código lineal. Entonces es cíclico si y sólo si es un ideal, como subconjunto de $\mathbf{F}_q[X]/\langle X^n - 1\rangle$.

Demostración.– La suma es cerrada, por ser un código lineal (y por tanto un subespacio vectorial). Para ver que el producto de un polinomio $f(X) \in \mathcal{C}$ por uno arbitrario $g(X) \in \mathbf{F}_q[X]/\langle X^n - 1\rangle$ está siempre en $\mathcal{C}$ podemos suponer[8] que $g(X) = X$. Pero

$$f(X) = a_0 + a_1X + ... + a_{n-1}X^{n-1} \implies Xf(X) = a_{n-1} + a_0X + a_1X^2 + ... + a_{n-2}X^{n-1},$$

y por tanto $Xf(X) \in \mathcal{C}$ es, precisamente, la condición de que $\mathcal{C}$ sea cíclico. □

Observación.– Dado que en $\mathbf{F}_q[X]/\langle X^n - 1\rangle$ todos los ideales son principales, y están generados por un divisor de $X^n - 1$, podemos identificar cada código lineal $\mathcal{C}$ con un polinomio $g(X)$ tal que $g(X) \mid (X^n - 1)$ y, en esta identificación, $\mathcal{C} = \langle g(X)\rangle$.

Proposición.– Sea $\mathcal{C} = \langle g(X)\rangle \subset \mathbf{F}_q[X]/\langle X^n - 1\rangle$ un código cíclico, con $\deg(g) = n - k$. Entonces

$$\mathcal{B} = \left\{g(X),\, Xg(X),\, ...,\, X^{k-1}g(X)\right\}$$

es una base de $\mathcal{C}$ como espacio vectorial.

Demostración.– Vamos a llamar $h(X)$ al polinomio que verifica $gh = X^n - 1$, del que sabemos que $\deg(h) = k$.

Supongamos que tomamos un elemento genérico $f(X)g(X) \in \mathcal{C}$, para $f(X) \in \mathbf{F}_q[X]$ y sea $f_0(X)$ el resto de $f(X)$ módulo $h(X)$. Entonces

$$f(X)g(X) - f_0(X)g(X) = (f - f_0) \cdot g = q(X)h(X)g(X) \in \langle X^n - 1\rangle,$$

con lo cual

$$f(X)g(X) = f_0(X)g(X) \mod X^n - 1.$$

Así pues, en lugar de tomar fg tomaremos f_0g. Dado que $\deg(f_0) < k$, es elemental ver que f_0g se escribe como combinación lineal de los polinomios de $\mathcal{B}$. □

El hecho de que $\mathcal{B}$ es linealmente independiente viene motivado porque una combinación lineal nula debe verificar:

$$0 = \sum_{i=0}^{k-1} \left(a_iX^ig(X)\right) = g(X) \sum_{i=0}^{k-1} \left(a_iX^i\right) = g(X)f(X),$$

pero como $\deg(g) = n - k$, $\deg(f) \leq k - 1$, se tiene que $\deg(gf) \leq n - 1 < n$, por lo que sólo puede ser 0 en $\mathbf{F}_q[X]/\langle X^n - 1\rangle$ cuando $a_i = 0$ para $i = 0, ..., k - 1$. □

Corolario.– En las condiciones anteriores, $\dim(\mathcal{C}) = k$ y, si $g(X) = g_0 + g_1X + ... + g_{n-k}X^{n-k}$, una matriz generatriz de $\mathcal{C}$ es, precisamente:

$$M = \begin{pmatrix} g_0 & 0 & 0 & \dots & 0 \\ g_1 & g_0 & 0 & \dots & 0 \\ g_2 & g_1 & g_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & & g_0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & g_{n-k} \end{pmatrix} \in \mathcal{M}(n \times k; \mathbf{F}_q).$$

El cálculo de la matriz de control también puede realizarse en términos polinomiales. Dado $\mathcal{C} = \langle g(X)\rangle \subset \mathbf{F}_q[X]/\langle X^n - 1\rangle$, consideremos el polinomio

$$h(X) = \frac{X^n - 1}{g(X)} = h_0 + h_1X + ... + h_kX^k.$$

[8] ¿Por qué?

Corolario.– En las condiciones anteriores, una matriz de control de $\mathcal{C}$ es

$$A = \begin{pmatrix} 0 & \dots & & \dots & h_2 & h_1 & h_0 \\ 0 & \dots & & \dots & h_1 & h_0 & 0 \\ 0 & \dots & & \dots & h_0 & 0 & 0 \\ \vdots & & & & \vdots & \vdots & \vdots \\ h_k & \dots & h_0 & \dots & 0 & 0 & 0 \end{pmatrix} \in \mathcal{M}((n-k) \times n, \mathbf{F}_q).$$

Demostración.– Únicamente (!) hay que darse cuenta de que, si hacemos AM, el elemento (i, j) del producto es precisamente el coeficiente de grado $n - i - j + 1$ del polinomio $gh = X^n - 1$. □

El algoritmo del líder, aplicado a los códigos cíclicos, resulta particularmente simple y ésta es una de las razones por las cuales resulta interesante esta subfamilia de códigos. Veamos cómo funciona.

Paso 1. En lugar de construir la tabla completa de síndromes y líderes, construiremos una tabla reducida, donde únicamente aparecerán las parejas en las cuales el líder tenga su última coordenada *no nula*.

Paso 2. Para decodificar $\underline{z} = (z_0, ..., z_{n-1})$ calculamos como de costumbre su síndrome $s(\underline{z})$, y nos fijamos en la tabla. Si aparece, corregimos con el líder, si no, seguimos adelante, considerando ahora $(z_{n-1}, z_0, ..., z_{n-2})$.

Paso 3. Una vez que hemos realizado el paso 2 n veces, hemos terminado de corregir $\underline{z}$.

Observación.– Para hacernos una idea del avance que esto supone, si tenemos un código lineal binario de tipo $(23, 2^{12}, 7)$, (por ejemplo, el denominado G_{23}, de la familia de códigos de Golay) tendremos que calcular síndromes y líderes, al menos para

$$1 + 23 + \begin{pmatrix} 23 \\ 2 \end{pmatrix} + \begin{pmatrix} 23 \\ 3 \end{pmatrix} = 2025$$

casos, los que tenemos seguridad que podemos corregir. En cambio, si únicamente nos fijamos en los que tienen la última coordenada no nula, necesitamos

$$1 + 22 + \begin{pmatrix} 22 \\ 2 \end{pmatrix} = 133$$

entradas.

Una de las familias más utilizadas de códigos correctores en la actualidad son los códigos BCH (Bose, Chaudhuri, Hocquenghem). Es el tipo de código que utilizan, por ejemplo, los lectores de CD. Son códigos cíclicos y se basan en la estructura de las raíces n–ésimas de la unidad sobre $\mathbf{F}_q$.

Cotas asociadas a un código.

Hasta ahora hemos hablado de códigos, pasando superficialmente por cuestiones de optimización. Aunque sí ha debido quedar claro a estas alturas que un código no es, en principio, bueno o malo, salvo a la luz de las circunstancias en las que se utilice. El

código de repetición, por ejemplo, es óptimo para transmisiones donde el coste es escaso, pero la necesidad de que el mensaje llegue íntegro es enorme. La cinta de n agujeros, en cambio, es ideal para situaciones donde la probabilidad de error es muy escasa, la transmisión es muy costosa y los posibles errores no corregidos carecen de importancia (si son pocos). Ahora vamos a intentar dar formalismo a estas consideraciones.

Fijamos entonces un alfabeto $\mathcal{A}$ tal que $|\mathcal{A}| = q$, y un código $\mathcal{C}$ sobre $\mathcal{A}$ de tipo (n, m, d). Al hablar de decodificación, es claro que hablamos de bolas cerradas, luego una primera idea a considerar es cuántas palabras de $\mathcal{A}^n$ entran en la bola centrada en una palabra de $\mathcal{C}$. Claramente, para todo $\underline{x} \in \mathcal{A}^n$,

$$|\overline{B}(\underline{x}, r)| = |\overline{B}(\underline{0}, r)|,$$

por lo que designaremos este número entero por $V(r)$. Si hubiéramos de ser precisos, la notación sería $V_q(n, r)$ (dado que q y n son importantes, aunque para nosotros estén fijados de antemano).

Proposición.– En las condiciones anteriores

$$V(r) = \sum_{j=0}^{r} \binom{n}{j} (q-1)^j.$$

Demostración.– Directa. □

Cota de Hamming.– En las condiciones anteriores,

$$mV\left(\left[\frac{d-1}{2}\right]\right) \leq q^n.$$

Observación.– Esta cota, que involucra los parámetros principales de $\mathcal{C}$, ilustra la defínción de *código perfecto*, que es aquél que verifica

$$mV\left(\left[\frac{d-1}{2}\right]\right) = q^n,$$

esto es, el espacio $\mathcal{A}^n$ está recubierto (de forma disjunta) por las bolas centradas en las palabras de $\mathcal{C}$. Lamentablemente, no existen muchos códigos perfectos, y están bien estudiados. Forman dos familias importantes, los *códigos de Hamming* (ver ejercicios) y los *códigos de Golay*.

Para códigos lineales podemos refinar mucho más. Supongamos a partir de ahora, pues, que $\mathcal{C}$ es lineal, de tipo (n, q^k, d).

Cota de Singleton.– En las condiciones anteriores, $k + d \leq n + 1$.

Demostración.– Sea A una matriz de control de $\mathcal{C}$. Como la distancia mínima es el menor número de columnas de A que necesitamos para formar un conjunto linealmente dependiente, y $\mathrm{rg}(A) = n - k$, $d \leq n - k + 1$. □

Observación.– La cota de Singleton pone de manifiesto de forma precisa que la distancia mínima y el tamaño del código no pueden crecer *a la vez*, algo que ya habíamos intuido.

Definición.– Los códigos que verifican $k + d = n + 1$ se llaman códigos de máxima distancia de separación, o códigos MDS.

Cota de Plotkin.– En las condiciones anteriores

$$d \leq \frac{nq^{k-1}(q-1)}{q^k-1}.$$

Demostración.– Aplicamos la fórmula de la dimensión de los espacios vectoriales a $\mathcal{C}$ y a $H : x_i = 0$,

$$\dim(\mathcal{C}) + \dim(H) = \dim(\mathcal{C}+H) + \dim(\mathcal{C}\cap H).$$

Tenemos dos casos posibles. El primero, que todas las palabras de $\mathcal{C}$ tengan coordenada i–ésima nula. Entonces $\mathcal{C} \subset H$, y hay exactamente q^k tales palabras.

El otro caso es que no sea así. Entonces $\mathcal{C} + H = \mathbf{F}_q^n$, luego $\dim(\mathcal{C}\cap H) = k-1$, y por tanto las palabras de $\mathcal{C}$ con coordenada i–ésima nula son q^{k-1}. Por tanto

$$\sum_{\underline{x}\in\mathcal{C}} w(\underline{x}) \leq n\left(q^k - q^{k-1}\right) = nq^{k-1}(q-1).$$

Por otra parte, todas las palabras (salvo $\underline{0}$) tienen al menos d coordenadas no nulas, luego

$$\sum_{\underline{x}\in\mathcal{C}} w(\underline{x}) \geq d\left(q^k-1\right),$$

y esto finaliza la demostración. □

La última cota que daremos será distinta: hasta ahora hemos demostrado lo que *no* puede hacer un código. Gilbert, Shannon y Varshamov demostraron lo que puede hacer un código.

Cota de Gilbert–Shannon–Varshamov.– Si se verifica

$$q^{n-k+1} > V(r),$$

entonces existe un código lineal de tipo (n, q^k, d) con $d > r$.

Demostración.– La construcción del código es inductiva. Comenzamos fijando un vector cualquier $\underline{x}_1$ tal que $w(\underline{x}_1) > r$, y hacemos $\mathcal{C}_1 = \langle \underline{x}_1 \rangle$.

En el paso general, supongamos construido $\mathcal{C}_{j-1} = \langle \underline{x}_1, ..., \underline{x}_{j-1} \rangle$ de tipo (n, q^{j-1}, d), con $d > r$ y $j-1 < k$. Entonces, por hipótesis,

$$q^{j-1}V(r) < q^n,$$

luego existe algún vector $\underline{x}_j$ cuya distancia con las palabras de $\mathcal{C}_{j-1}$ es mayor que r. Hacemos entonces $\mathcal{C}_j = \langle \underline{x}_1, ..., \underline{x}_j \rangle$ y continuamos. □

A nivel de aplicaciones interesantes, lo que buscamos no son códigos individuales, sino *familias de códigos*. Eso hace que el resultado anterior carezca de interés práctico *per se*, ya que los códigos que nos promete deben crearse de manera artesanal. Y, de hecho, nos interesa que podamos variar la longitud arbitrariamente, con lo cual estamos en el terreno de las denominadas *familias asintóticas de códigos*. Aquí pierde interés ya hablar de n y de d, y hemos de sustituir estos parámetros por otros menos dependientes del *contexto*, como la tasa de información y la distancia relativa.

Definición.– Sean $d, n \in \mathbf{Z}_+$. Definimos

$$a_q(n,d) = \max\{\ k \in \mathbf{N} \mid \exists \mathcal{C} \text{ lineal sobre } \mathbf{F}_q \text{ de tipo } (n, q^k, d)\}.$$

Así mismo, recordemos que $\delta = d/n$ y definamos

$$\alpha_q(\delta) = \limsup_{n\to\infty} \frac{a_q(n, n\delta)}{n}.$$

Esto es, $\alpha_q(\delta)$ nos mide cómo de grande puede ser, idealmente, la tasa de información de un código de distancia relativa δ. Las cotas que hemos estudiado anteriormente tienen todas una versión asintótica (no siempre inmediata a partir de las versiones absolutas, pero no daremos las demostraciones).

Cota asintótica de Hamming.– En las condiciones anteriores,

$$\alpha_q(\delta) \leq 1 - \lim_{n\to\infty} \frac{\log_q(V([\delta/2]-1))}{n}$$

Cota asintótica de Singleton.– En las condiciones anteriores, $\alpha_q(\delta) \leq 1 - \delta$.

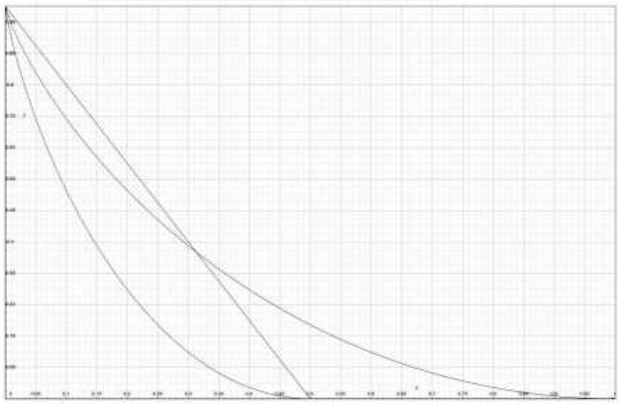
Cota asintótica de Plotkin.– En las condiciones anteriores, si $\theta = (q-1)/q$,

$$\alpha_q(\delta) \begin{cases} \leq 1 - \delta/\theta & \text{si } \delta \leq \theta \\ = 0 & \text{si } \delta > \theta \end{cases}$$

Cota asintótica de Gilbert–Shannon–Varshamov.– En las condiciones anteriores, si $\delta \leq \theta$,

$$\alpha_q(\delta) \geq 1 - \lim_{n\to\infty} \frac{\log_q(V(\delta))}{n}.$$

La *zona interesante*, el espacio que nos queda entre la curva de Plotkin y la de Gilbert–Shannon–Varshamov, es el objetivo de los constructores de buenos códigos. Este espacio ha estado durante muchos años desprovisto de familias, hasta que Tsfasman, Vladut y Zink[9], en 1982 crearon la primera familia de códigos que mejoraba la cota GSV. Su trabajo combinaba la teoría clásica de códigos de Goppa con los espacios vectoriales de funciones asociados a curvas algebraicas proyectivas planas. Lejos de terminar el problema, este trabajo abrió una conexión inesperada y muy fértil entre la teoría de códigos, la teoría de números y la geometría algebraica.



[9] M.A. Tsfasman, S.G. Vladut, Th. Zink: *On Goppa codes which are better than the Varshamov–Gilbert bound*. Mathematische Nachrichten **109** (1982) 21–28.

Ejercicios Tema 2

Ejercicio 2.01.– El código NIF funciona de la siguiente manera: las palabras son pares (n, l), donde $n \in \mathbf{Z}$ y $l \in \{0, 1, ..., 22\}$, siendo $n \equiv l \mod 23$. Demostrar que el código NIF detecta un error en la transmisión de n, así como dos errores, si corresponden a la permutación de dos cifras decimales consecutivas. ¿Puede detectar dos errores cualesquiera? ¿Puede corregir un error?

Ejercicio 2.02.– El código ISBN (International Standard Book Number), utilizado para catalogar libros, utiliza el alfabeto $\mathcal{A} = \{0, 1, ..., 9, X\}$, donde la X debe entenderse como 10. El código ISBN es, entonces

$$\mathcal{C} = \left\{ (a_1, ..., a_{10}) \mid a_i \in \mathcal{A}, \sum_{i=1}^{10} (11 - i)a_i = 0 \mod 11 \right\} \subset \mathcal{A}^{10}.$$

Estudiar el tamaño del código y si es lineal o no. Demostrar que el código ISBN detecta un error, aunque no lo corrige y que no detecta dos errores, aunque sí lo hace cuando corresponden a la permutación de dos cifras consecutivas.

Ejercicio 2.03.– Sea $\mathcal{C} \subset \mathbf{F}_2^n$ un código lineal de dimensión k. Probar que, bien todas, bien exactamente la mitad de las palabras de $\mathcal{C}$ tienen peso par.

Ejercicio 2.04.– Supongamos que enviamos una cadena de, pongamos 100.000 caracteres (por ejemplo, un programa de ordenador o un conjunto de datos) y que tenemos una probabilidad de 10^{-5} de que se cometa un error en la transmisión de cada carcter. Hallar la probabilidad de que la transmisión resulte correcta. Si decidimos utilizar el código de Hamming, hallar la probabilidad de que la transmisión resulte correcta (eventualmente, tras corregir errores).

Ejercicio 2.05.– Sea $\mathcal{C}$ un código lineal tipo (n, q^k, d), con matriz generatriz M y matriz de control A. La matriz A^t puede entonces interpretarse como la matriz generatriz de un código lineal $\mathcal{C}^\perp$, denominado código dual de $\mathcal{C}$. Probar que $\mathcal{C}^\perp$ coincide con el subespacio ortogonal de $\mathcal{C}$ respecto del producto escalar usual y, en particular:

- $\mathcal{C}^\perp$ es un código de tipo (n, q^{n-k}, d') (para cierta $d' \in \mathbf{N}$).
- $(\mathcal{C}^\perp)^\perp = \mathcal{C}$.

Ejercicio 2.06.– Un código lineal tipo (n, q^k, d) se dice que está en forma sistemática cuando viene dado por una matriz generatriz de la forma

$$M = \left(\frac{I_k}{M_k} \right),$$

donde I_k es la matriz unidad $k \times k$ y $M_k \in \mathcal{M}((n - k) \times k, \mathbf{F}_q)$. Probar que todo código lineal se puede escribir de forma sistemática, si admitimos permutaciones entre las coordenadas. Hallar la matriz de control de un código en forma sistemática.

Ejercicio 2.07.– Dado un código lineal $\mathcal{C} \subset \mathbf{F}_2^n$, definimos el código truncado de $\mathcal{C}$ como

$$\mathcal{C}^- = \{(x_1, ..., x_{n-1}) \mid \exists x_n \in \mathbf{F}_2 \text{ con } (x_1, ..., x_n) \in \mathcal{C}\}.$$

Estudiar si $\mathcal{C}^-$ es lineal, así como sus parámetros (en función de los de $\mathcal{C}$).

Ejercicio 2.08.– Dado un código lineal $\mathcal{C} \subset \mathbf{F}_2^n$, definimos el código con extensión de paridad de $\mathcal{C}$ como

$$\mathcal{C}^+ = \left\{ (x_1, ..., x_{n+1}) \mid (x_1, ..., x_n) \in \mathcal{C} \text{ y } \sum_{i=1}^{n+1} x_i = 0 \right\}.$$

Estudiar si $\mathcal{C}^+$ es lineal, así como sus parámetros.

Ejercicio 2.09.– Dados dos códigos lineales $\mathcal{C}_1 \subset \mathbf{F}_2^n$ y $\mathcal{C}_2 \subset \mathbf{F}_2^m$, se define su suma directa como

$$\mathcal{C}_1 \oplus \mathcal{C}_2 = \{(\underline{x}|\underline{y}) \mid \underline{x} \in \mathcal{C}_1, \underline{y} \in \mathcal{C}_2\}.$$

Probar que $\mathcal{C}_1 \oplus \mathcal{C}_2$ es un código lineal que verifica

$$d(\mathcal{C}_1 \oplus \mathcal{C}_2) = \min\{d(\mathcal{C}_1),\, d(\mathcal{C}_2)\}.$$

Ejercicio 2.10.– Dados dos códigos lineales $\mathcal{C}_2 \subset \mathcal{C}_1 \subset \mathbf{F}_2^n$, se define su producto como

$$\mathcal{C}_1 \mid \mathcal{C}_2 = \{(\underline{x}|\underline{x} + \underline{y}) \mid \underline{x} \in \mathcal{C}_1, \underline{y} \in \mathcal{C}_2\}.$$

Probar que $\mathcal{C}_1 \mid \mathcal{C}_2$ es un código lineal que verifica

$$d(\mathcal{C}_1 \mid \mathcal{C}_2) \geq \min\{2d(\mathcal{C}_1),\, d(\mathcal{C}_2)\}.$$

Ejercicio 2.11.– El código de Hamming de redundancia r, notado $\mathcal{H}(r)$ es el código lineal definido por una matriz de control en la que aparecen todos los vectores no nulos de $\mathbf{F}_2^r$.

- Probar que $\mathcal{H}(r)$ tiene longitud $2^r - 1$, dimensión $2^r - r - 1$ y distancia 3. Por tanto, corrige exactamente un error.
- Demostrar que el cálculo del síndrome es suficiente para decodificar en $\mathcal{H}(r)$ (supuesto que se ha cometido a lo más un error).
- Probar que toda palabra está a distancia menor o igual que uno de un elemento de $\mathcal{H}(r)$. Dicho de otro modo, las bolas cerradas centradas en los elementos de $\mathcal{H}(r)$ y radio 1 recubren todo el espacio de forma disjunta. Esto es, $\mathcal{H}(r)$ es un código perfecto.

Ejercicio 2.12.– Construir explícitamente $\mathcal{H}(2)$ y $\mathcal{H}(3)$.

Ejercicio 2.13.– Consideramos el código lineal $\mathcal{C}$ sobre $\mathbf{F}_2$, definido por la matriz generatriz

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Hallar los parámetros del código y una matriz de control. Calcular todos los síndromes con líder de peso menor o igual que 2. Si recibimos la palabra 11011011 y sabemos que se han cometido, a lo sumo, dos errores, ¿cuál es la palabra enviada y cuál el error cometido?

Ejercicio 2.14.– Imaginemos que tenemos un código lineal $\mathcal{C}$ sobre $\mathbf{F}_q$. Probar que existen exactamente q^r síndromes, donde r es la redundancia de $\mathcal{C}$.

Ejercicio 2.15.– Imaginemos que tenemos un código lineal binario $\mathcal{C}$, tipo $(63, 2^{51}, 5)$ (un tal código existe). Hallar cuántos síndromes posee $\mathcal{C}$ y, de entre estos, cuántos tendrán un líder de peso 1 y cuántos un líder de peso 2. ¿Por qué no tiene sentido considerar pesos mayores?

Ejercicio 2.16.– Sabemos que, en un código perfecto de tipo (n, m, d),

$$m = \frac{2^n}{V([(d-1)/2])}.$$

Sin embargo, eso no quiere decir que si $2^n/V([(d-1)/2])$ es un entero m, exista un código perfecto tipo (n, m, d).

- Probar que
$$\frac{2^{90}}{V_2(90, 2)} = 2^{78}.$$

- Sea $\mathcal{C}$ un código binario perfecto tipo $(90, 2^{78}, 5)$ sobre $\mathbf{F}_2$ y supongamos[10] que $\underline{0} \in \mathcal{C}$. Consideremos el conjunto
$$X = \left\{ \underline{x} \in \mathbf{F}_2^{90} \mid x_1 = x_2 = 1,\ d(\underline{x}, \underline{0}) = 3 \right\}.$$
Hallar el número de elementos de X y probar que, para cada $\underline{x} \in X$ existe una única $\underline{z} \in \mathcal{C}$ tal que $d(\underline{x}, \underline{z}) = 2$.

- Para $\underline{x}$ y $\underline{z}$ como antes, probar que $d(\underline{z}, \underline{0}) = 5$, que $z_i = 1$ si $x_i = 1$.

- Dados ahora dos $\underline{x}, \underline{x}' \in X$, construir los $\underline{z}, \underline{z}' \in \mathcal{C}$ correspondientes y demostrar que $d(\underline{z}, \underline{z}') = 6$. Deducir que cada $\underline{z} \in \mathcal{C}$ está asociada a exactamente tres elementos de X y llegar a una contradicción que demuestra que no existe un código perfecto $(90, 2^{78}, 2)$.

[10] ¿Por qué suponer esto no hace que perdamos generalidad?

Ejercicio 2.17.– Sean $n \in \mathbf{Z}$, $\gcd(n, q) = 1$. Supongamos que el orden de q en el grupo multiplicativo $(\mathbf{Z}/\mathbf{Z}n^*, \cdot)$ es d. Entonces el polinomio $X^n - 1 \in \mathbf{F}_q[X]$ factoriza en $\varphi(n)/d$ polinomios mónicos irreducibles. Para demostrar esto, probar:

- Hay exactamente $\varphi(n)$ raíces n–ésimas primitivas de la unidad en $\mathbf{F}_q$ (o sea, no son raíces m–ésimas de la unidad, con $m < n$).
- Sea β una raíz n–ésima primitiva de la unidad en $\mathbf{F}_q$. El entero d se puede caracterizar como el menor entero positivo tal que $\beta^{q^d-1} = 1$.
- Deducir que $[\mathbf{F}_q[\beta] : \mathbf{F}_q] = d$ y el resultado.

Ejercicio 2.18.– Decimos que un vector $\underline{x} \in \mathbf{F}_q^n$ es una ráfaga si todas sus coordenadas no nulas son consecutivas. Las ráfagas son particularmente interesantes cuando representan errores en códigos binarios. Probar que un código cíclico $\mathcal{C} \subset \mathbf{F}_2^n$ de dimensión k no contiene ninguna ráfaga de peso $l \leq n - k$. Probar que, por este motivo, detecta cualquier error ráfaga de peso menor o igual a $n - k$.

Ejercicio 2.19.– Aplicar el procedimiento Gilbert–Shannon–Varshamov para crear un código sobre $\mathbf{F}_3$ de tipo $(6, 3^4, 2)$.

Ejercicio 2.20.– Probar que el procedimiento inductivo de Gilbert–Shannon–Varshamov crea, efectivamente, un código con las condiciones enunciadas.

Tema 3. Criptografía de clave privada

Generalidades.

El problema de la criptografía de clave secreta es básicamente el siguiente: tenemos dos individuos, clásicamente notados A (Alice) y B (Bob) que quieren establecer un canal seguro de comunicación, y un tercer individuo E (Eve) (o, en términos más genéricos *el adversario*), que tiene la capacidad de interceptar y leer los mensajes que se intercambian por este canal.

Para simplificar los problemas de logística, supondremos que los mensajes que Alice y Bob se quieren intercambiar son números enteros, en una base previamente acordada (típicamente 2) y conocida por todos los actores, o bien una congruencia módulo un cierto entero N, también conocido por todos (incluida Eve). Hay muchas formas de hacer esto, la más simple es asignar a cada símbolo de texto un número (como por ejemplo en el código ASCII) y luego yuxtaponer los números para conseguir un mensaje numérico. Si, por el motivo que fuera, el mensaje fuese demasiado largo (por ejemplo, si estamos trabajando en un anillo de congruencias) se puede dividir el mensaje cuantas veces sea necesario.

Para lograr que su correspondencia sea confidencial, Alice y Bob crean un *protocolo criptográfico* que no es, fundamentalmente, más que una aplicación biyectiva. Alice halla la imagen por esta aplicación de su mensaje para Bob, y esto es lo que se envía por el canal de comunicación. Posteriormente Bob recibe la imagen y calcula el original, hallando así el mensaje de Alice. Veamos la formulación precisa.

Para facilitar la comprensión del protocolo, notaremos con letras minúsculas los objetos (enteros, aplicaciones,...) que sólo conocen Alice y Bob, y con mayúsculas aquellos objetos a los que Eve tiene acceso[11]. En estas condiciones, en un protocolo de clave privada, los objetos involucrados son:

1) El mensaje que Alice quiere hacer llegar a Bob: m.

2) Una clave para encriptar y desencriptar, que han acordado Alice y Bob: k.

3) Una aplicación de encriptación conocida, que toma dos valores (clave y mensaje): $F(\cdot,\cdot)$.

4) El mensaje encriptado $F(k,m) = M$.

5) Una aplicación de desencriptación $G(\cdot,\cdot)$, que es la inversa de la anterior en el sentido de que verifica: $G(k,M) = m$.

[11] No necesariamente de forma legal.

Idealmente, se deben dar las siguientes condiciones:

1) Los cálculos $F(k, m)$ y $G(k, M)$ deben ser rápidos (esto es, algoritmos polinomiales).

2) No es posible (en la práctica) para Eve averiguar m a partir de M *sin conocer* k (esto es, el cálculo de m requiere un algoritmo exponencial, como mínimo).

Se podría razonar que supone una pérdida innecesaria de seguridad el que Eve conozca F y G. Esto es, a priori resulta mucho más seguro que Eve no sepa cómo se cifra y se descifra. Pero en la práctica esto no es así. Muchos protocolos criptográficos han basado su seguridad en la opacidad del cifrado y, cuando éste ha sido desvelado (y, tarde o temprano, sucede), toda el sistema se ha visto comprometido. Una postura de cifrado abierto, donde todo el mundo es *retado* a derrotar al protocolo, garantiza mucha más fiabilidad.

Formalmente, éstas son sólo parte de las exigencias que debe cumplir un protocolo. Tradicionalmente, se asumen los denominados *Principios de Kerckhoffs*:

1) Si el sistema no es teóricamente irrompible, debe serlo en la práctica.

2) La efectividad del sistema no debe depender de que su diseño permanezca en secreto.

3) La clave debe ser fácilmente memorizable de manera que no haya que recurrir a notas escritas.

4) Los mensajes encriptados deberán dar resultados alfanuméricos.

5) El sistema debe ser operable por una única persona.

6) El sistema debe ser fácil de utilizar.

Evidentemente, algunos de estos principios no son de aplicación ahora (los principios de Kerckhoffs datan de finales del siglo XIX).

Galia est omnis divisa in partes tres.

En el siglo I A.C., aparece una técnica de cifrado conocida con el nombre genérico de cifrado de César en honor a Julio César, que dejó constancia escrita de cómo utilizó este código durante la guerra de las Galias. Consiste en la transformación del mensaje mendiante un desplazamiento constante de tres caracteres. Esto es, asociamos a cada letra un entero módulo 27:

A=00	B=01	C=02	D=03	E=04	F=05	G=06	H=07	I=08
J=09	K=10	L=11	M=12	N=13	˜=14	O=15	P=16	Q=17
R=18	S=19	T=20	U=21	V=22	W=23	X=24	Y=25	Z=26

Observación.– El sistema LaTeX no permite usar la letra Ñ en modo verbatim (o sea, en tipografía de *máquina de escribir*) que es, por ciertos motivos, aconsejable en este contexto. Por ello, sustituiremos la letra por la tilde, `~`, que sí es un símbolo admitido.

Entonces el cifrado que debe llevar a cabo Alice consiste en hacer $F(m) = m + 3 \mod 27$, letra a letra. O, siendo más exhaustivo:

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

El descifrado que realiza Bob es, obviamente, $G(M) = M - 3 \mod 27$.

Ejercicio.– Cifrar los siguiente mensajes:

Mensaje I: `VENI, VIDI, VICI.`
Mensaje II: `JULIO CESAR HA SIDO ASESINADO.`

Observación.– El sistema de César se puede complicar, pero no mucho, de al menos dos formas:

- Cambiando el usar tres desplazamientos por un número arbitrario $k \in \mathbf{Z}/\mathbf{Z}27$, que actúa de clave para cifrar y descifrar.
- Complicar el cifrado usando una *función afín*. Para ello seleccionamos una clave $k = (a, b) \in \mathbf{Z}/\mathbf{Z}27^* \times \mathbf{Z}/\mathbf{Z}27$ y fijamos el cifrado $F(k, m) = am + b \mod 27$.

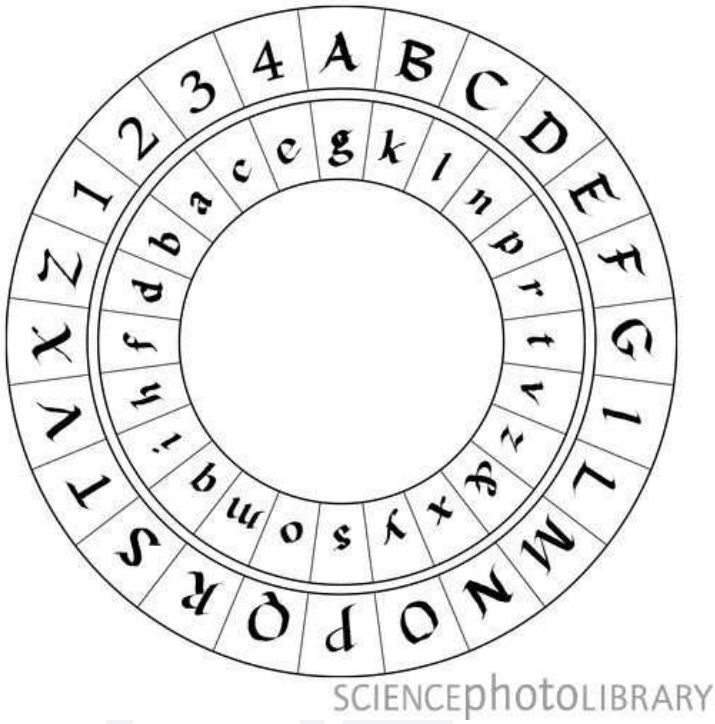
Ejercicio.– Halla el procedimiento de descifrado del protocolo de función afín.

Por supuesto, ninguno de estos sistemas tiene vigencia actualmente: un ataque básico de Eve, pero efectivo, puede consistir en probar todas las posibles claves de descifrado (27 en el primer caso, 486 en el segundo). Ambos son ejemplos de cifrados por sustitución: aquéllos en los que cada letra (a veces incluyendo en *letra* los espacios o símbolos de puntuación) se le asociaba un símbolo concreto (otra letra o tal vez no), siempre el mismo para todo el mensaje. Algunos ejemplos notables de esta técnica fueron:

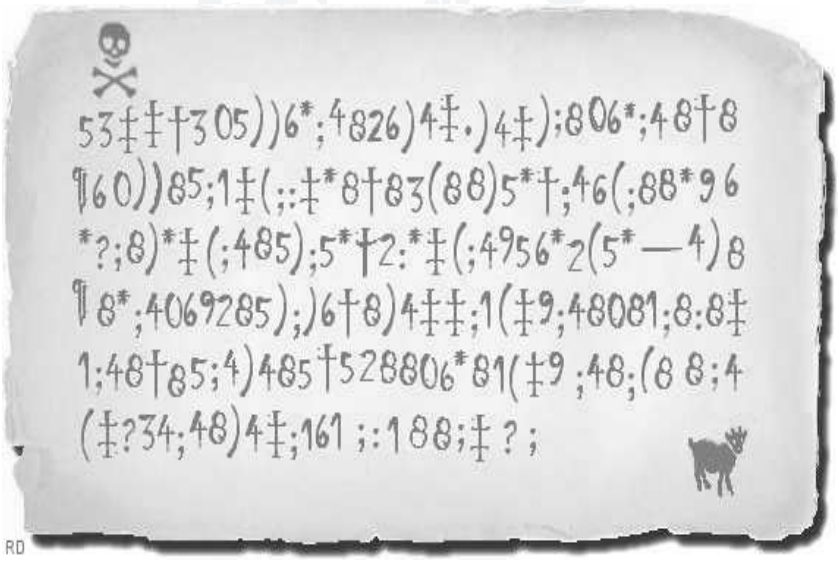
- El disco cifrador de Alberti (Leon Battista Alberti fue un estudioso genovés del siglo XV) consistía, materialmente, en dos discos concéntricos.

  En el exterior aparecían los 20 caracteres del latín, esto es, los mismos del alfabeto castellano excepto las letras `H`, `J`, `~`, `K`, `U`, `W` e `Y`, y se incluyen los números `1`, `2`, `3` y `4` para códigos especiales. En el disco interior aparecen todos los caracteres del latín además del signo `&` y las letras `H`, `K` e `Y`. Para cifrar un mensaje, Alice escoge una posición inicial del disco (por ejemplo, en el dibujo: disco interior `p`, disco exterior `E`) y cambiaba cada letra de m (en el disco exterior) por su correspondiente (en el disco interior).

  Además este sistema permitía cambiar de posición en medio del texto, bien tras cifrar una longitud fija de texto, bien cuando aparece un símbolo acordado. Decodificar el mensaje para Bob es igualmente simple (en este caso hay que ir del disco interno al externo). Conviene fijarse en que si hay que enviar el procedimiento de cifrado, esto puede representar una debilidad potencial del protocolo.



- *El escarabajo de oro*, uno de los cuentos más famosos de Edgar Allan Poe, donde el protagonista se enfrenta a la tarea de descifrar este mensaje:



- *La aventura de los bailarines*, una de los relatos de *El retorno de Sherlock Holmes*. En este relato en concreto aparece un enfoque muy constructivo del ataque por análisis de frecuencias que veremos más adelante. Entre otras cosas, Holmes (que juega aquí el papel del adversario) hace hincapié en un factor muy importante a tener en cuenta: la seguridad del sistema disminuye conforme aumenta la cantidad de texto cifrado con el mismo procedimiento.

La criptografía como arma infalible para ligar.

El siguiente código que estudiaremos es el código de Vigenère y durante varios siglos (desde el XVII hasta finales del XIX) se consideró seguro e infranqueable (aunque la verdad es que no era para tanto, como veremos). El cifrado toma su nombre en honor al criptólogo francés Blaise de Vigenère (1523-1596), y utiliza, realmente, el mismo método que el cifrador del César, una sustitución por desplazamiento de k caracteres en el texto, donde k es un número módulo 27. Pero la diferencia (y lo que lo hace mucho más robusto) es que dicho desplazamiento viene indicado por el valor numérico asociado a uno de los caracteres de una clave que se escribe cíclicamente bajo el mensaje.

Por ejemplo, si tenemos la clave `CERVANTES` para cifrar un texto (que a lo mejor conocéis), se realiza la cifra siguiente:

Mensaje (m): `ENUNLUGARDELAMANCHADECUYONOMBRE ...`
Clave (k): `CERVANTESCERVANTESCERVANTESCERV ...`

O sea, a nivel numérico tendríamos:

Mensaje (m): `04 13 21 13 11 21 06 00 18 03 04 11 00 ...`
Clave (k): `02 04 18 22 00 13 20 04 19 02 04 18 22 ...`
Cifrado (M): `06 17 12 08 11 07 26 04 10 05 08 12 22 ...`

El mensaje cifrado completo queda `GQMIL HZEKF ICVMN GGZCH VXULI....`

Como podéis ver, una misma letra se puede encriptar de maneras diferentes, dependiendo de qué letra de la clave le toque para encriptarse en cada momento. La posibilidad de elegir la propia clave (caso pionero de *customización*[12]) fue clave para el éxito de este cifrado.

Observación.– Para cifrar (y descifrar) se usaban normalmente tablas como la siguiente (donde aparecen todas las letras del alfabeto inglés *sumadas*.

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

[12]Neologismo aberrante por personalización.

Ejercicio.– Cifrar el siguiente mensaje según el método de Vigenère:

Mensaje (m): `THERE ARE MORE THINGS IN HEAVEN AND EARTH`
`HORATIO THAN ARE DREAMT OF IN YOUR PHILOSOPHY`
Clave (k): `HAMLET`

Observación.– Giacomo Casanova fue muchas cosas (no todas ellas decentes). Entre otras fue diplomático y espía (la distinción entre ambos trabajos, todavía a día de hoy, es a menudo difusa) y como tal fue un gran experto en cifrados y códigos. En sus memorias cuenta la siguiente anécdota acerca de cómo utilizó sus conocimientos en este campo para propósitos algo alejados de su profesión.

Cinco o seis semanas después, ella (Madame d'Urfé) me preguntó si había descifrado el manuscrito donde se encontraba el proceso de transmutación. Le dije que así era.

– Sin la clave, señor, perdóneme pero creo que es imposible.

– ¿Quiere que le diga cuál es la clave que ha utilizado?

– Si no le importa.

Entonces le dije la palabra clave, que no pertenecía a ningún idioma, y pude ver su sorpresa. Me dijo que era imposible, pues ella era la única poseedora de la palabra clave y la guardaba en su memoria, sin haberla puesto nunca por escrito.

Le podía haber dicho la verdad (que el mismo cálculo que me permitió descifrar el texto me había permitido obtener la clave) pero por capricho le dije que un genio me la había revelado. Esta falsa revelación hizo que Madame d'Urfé se rindiera a mí. Ese día me convertí en el poseedor de su alma, y abusé de dicho poder. Cada vez que pienso en ello me avergüenzo de lo que hice, y ahora hago penitencia obligándome a contar la verdad al escribir estas memorias.

¿Cómo se las apañó Casanova para romper el código de Madame d'Urfé? El hecho de que existiese una *palabra clave* nos da a entender que, en efecto, se trataba de un código de Vigenère. A pesar de ser considerado irrompible (por Madame d'Urfé entre otros, como hemos visto), a un criptoanalista sagaz no se le escapa el hecho de que, en el fondo, el código de Vigenère y el código de César no son tan distintos.

Observación.– Veamos ahora el método de ataque más habitual y exitoso contra un cifrado por sustitución (y contra un código Vigenère). Consiste en la siguiente idea: las letras, en un texto en castellano (o en inglés o en cualquier idioma) no aparecen de manera aleatoria, ni se distribuyen equiprobablemente. Algunas aparecen más frecuentemente, otras menos y algunas de hecho son muy raras. Estudiando entonces cantidades enormes de texto podemos crearnos una tabla donde aparezca el porcentaje (aproximado) de veces que aparece determinada letra en un texto. En castellano, por ejemplo, este estudio (denominado *análisis de frecuencias*, que suena bastante sofisticado) es, aproximadamente, el siguiente:

A	B	C	D	E	F	G	H	I
10.60	1.16	4.85	5.87	13.11	1.13	1.40	0.60	7.16

J	K	L	M	N	Ñ	O	P	Q
0.25	0.11	4.42	3.11	7.14	0.10	8.23	2.71	0.74

R	S	T	U	V	W	X	Y	Z
6.95	8.47	5.40	4.34	0.82	0.12	0.15	0.79	0.26

Dependiendo del tipo de texto analizado, aparecerán ligeras diferencias, si bien podemos concluir que los valores se mantienen en un rango manejable y no muy amplio. Esto quiere decir que es posible considerar, por ejemplo, la letra `L` con más peso que la `D`, considerar con más frecuencia la letra `C` en vez de la letra `T`, etc.

Por supuesto, estamos asumiendo que los únicos símbolos que aparecen son letras (de un tipo, mayúsculas o minúsculas). Si introducimos más simbología (puntuación, distintas grafías, separación de palabras,...) estas cifras pueden cambiar. Por ejemplo, en Enigma, los alemanes usaban la letra `X` para separar palabras. Así pasaba de ser un símbolo muy poco frecuente a ser una de las que más aparecía en el texto descifrado (y eso que el idioma alemán no es precisamente famoso por usar palabras breves).

Por supuesto, a idiomas distintos, análisis de frecuencias distintos. Aunque no es menos cierto que, por ejemplo, todas las lenguas romances tienen análisis similares. El del inglés, por ejemplo, varía más:

A	B	C	D	E	F	G	H	I	J	K	L	M
8.2	1.5	2.8	4.2	12.7	2.2	2.0	6.1	7.0	0.1	0.8	4.0	2.4

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
6.7	7.5	1.9	0.1	6.0	6.3	9.0	2.8	1.0	2.4	0.1	2.0	0.1

Dado que en los cifrados de sustitución el mismo símbolo siempre corresponde a la misma letra, estos cifrados suelen ser bastante débiles ante un ataque por análisis de frecuencias. Es el caso que nos encontramos en *El escarabajo de oro* y *La aventura de los bailarines*.

Para atacar el código de Vigenère, tenemos que tener en cuenta que el análisis de frecuencias no se puede aplicar tal cual, pues una misma letra se puede codificar de diferentes maneras, según su posición en el texto. Sin embargo, sí es cierto que, si nuestra palabra clave es, pongamos, `CRYPTO`, de 6 caracteres, todas las letras cuya posición sea congruente con 1 módulo 6 se codificarán de la misma forma (sumando `C`, concretamente). Análogamente con las letras en posición 2 módulo 6, 3 módulo 6,... De esta forma, conocida (o supuesta) la longitud de la palabra clave, digamos l, realmente nos encontramos con l cifrados distintos tipo César *sofisticado* y, por tanto, vulnerable al ataque por análisis de frecuencias.

De esta forma, la única dificultad para descifrar un mensaje estriba en averiguar la longitud de la clave. Aquí entra en juego el método Kasiski, que esencialmente consiste en la siguiente idea: en los lenguajes humanos, los digrafos (conjuntos de dos letras) y trigrafos se distribuyen de manera aún más brusca que las letras aisladas. Así, determinados digrafos y/o trigrafos son *mucho* más frecuentes que otros. Por tanto, en un texto largo, tenemos una probabilidad alta de que dos (o más) de esos digrafos frecuentes hayan sido cifrados de la misma forma.

Por tanto, el método de Kasiski sacaba ventaja de esta debilidad potencial buscando digrafos (o trigrafos) idénticos *en el texto cifrado*. Suponiendo que correspondan con digrafos idénticos en el texto original, habrán sido codificados de la misma forma, y por

tanto, la distancia en letras entre uno y otro habrá de ser un múltiplo de la longitud de la clave. Si tenemos suerte y encontramos varios casos similares, el $\gcd$ de las distancias puede ser una buena alternativa para la longitud de la clave.

Ejercicio.– Intenta descifrar el siguiente texto. Está escrito en inglés.

```
ZMXZBVIDYAWAEALGTPTEAMZRVNMOCZWBAHNDSIWGZGAINYIUBOWKMIFWIHWF
AWZGBXBXMPJMQDVFOXSSLQFGFWMPGJZQXSVOIEGYWMBWAYIZRGZMHWFAWZHU
SBIOFHTMBGWLUBZQJDOVFAFWYDZQANAVEKVLPUBGZMECHFLATFATQBPWQZFR
KBXSFKLDSNEAUKNDSQRNDWZSASZDCJKBDSRLAATPGJNZRKBABRFMMHULPQVN
DWATNKBDSRLTMACABGFAWLYMPGTXOELWFVRUWXRNFLPOZHETSAEGQMRKEQFR
KBMPOWLNMGZMRZNKPATNFMABYAOTHGZIFGCDQFHUWVUUULIZRGGCOVRVBTSF
GCZRBXAUZRFKQOAVQZHUWVMYRVTUUULQEOJLMZHUGCEOAVXQCCDMYOLTMYCE
WXQCCDMFOYCQZUJABTCHLABSNCQZUCWWBZRZMMFVFOIWGZWGHYAAFSAAVSDR
GXXSJJQFWAYAABTKBTOGNWUQRKVQJRJATOEWIZRAGWZSQSZQRQAAFIETBTSF
GCZRBXAUZRFKQTBGTEGNALUMBMLABBLSZCJKQXSAUMXWXWIOOAUMDUEGEEVR
SZYMJGZPGGZIFWZAOTHGWIOVLGCFOXWUKOEEAFVNLQYWTZBDSNUPKCHTCFAL
OWDRFDQWSFATQBGJIUBQJWBGSWTXOAVMOVBWLUBGZMISYDAATFATQBPWIZRG
ZMBSBHTQPBOMPOAVXDOLWLFCGZMZSBFOARGZMKANVMMBQLPQGVYVRZNKPQRB
MBUHFOIDBVFOUBGZMICEVAFVNLQFKNKNAFZAVSOAVBTSFAOZGNALFVROWDRF
GNFVRHZADUWBEOEWEDWGLMZCALPQGHTEMMJSTXGNFLFSAWUQBGZIXZFSVPKU
AABSEWLUBGZMECHFLECSKQXSAUM
```

Permutando, que es udogreni.

El cifrado por permutación, como su propio nombre indica, consiste en aplicar una permutación al texto original m. En este caso, tanto la clave como el proceso se identifican con la permutación. Normalmente se utilizaban permutaciones de pocos elementos, por ejemplo 7, y entonces se permutaban las primeras 7 letras, luego las 7 siguientes, y así sucesivamente.

Ejemplo.– Tenemos el texto

```
NOW IS THE WINTER OF OUR DISCONTENT
MADE GLORIOUS SUMMER BY THIS SUN OF YORK,
```

al cual queremos aplicar la permutación

$$k = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 5 & 8 & 7 & 1 & 3 & 6 & 2 \end{pmatrix}$$

Para ello escogemos entonces los primeros 8 caracteres y los permutamos siguiendo a k. Obtenemos (poniendo puntos en los espacios, para verlo mejor)

$$\texttt{NOW.IS.T} \Longrightarrow \texttt{ITSNO..W}$$

Ejercicio.– Termina de cifrar el mensaje, y luego aplica la permutación

$$k = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 5 & 2 & 1 \end{pmatrix}$$

al siguiente mensaje

```
MUCHOS A~OS DESPUES FRENTE AL PELOTON DE FUSILAMIENTO
EL CORONEL AURELIANO BUENDIA HABIA DE RECORDAR AQUELLA
TARDE REMOTA EN QUE SU PADRE LO LLEVO A CONOCER EL HIELO
```

Descifrar el mensaje no es difícil para Bob; simplemente hay que reordenar utilizando la *permutación inversa*. Por ejemplo, en el caso

$$k = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 5 & 8 & 7 & 1 & 3 & 6 & 2 \end{pmatrix}$$

tendríamos que la inversa sería

$$k' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 5 & 8 & 6 & 1 & 2 & 7 & 4 & 3 \end{pmatrix}$$

Observación.– El análisis de frecuencias no sirve con el cifrado por permutaciones. De hecho, en el texto cifrado, las frecuencias son, lógicamente, las mismas que en el original. ¿Cómo actuar entonces? Supongamos que nos encontramos el siguiente mensaje

```
H.OANAM.LYEUQ...ROBP.NENIEV.NOAG
```

Para empezar, no sabemos la longitud de la permutación utilizada. Eso es un problema, obviamente. Pero sí sabemos que, si la longitud es, pongamos n, entonces las n primeras letras, permutadas, tienen que tener algún sentido. Por ejemplo, si n fuera 3, el comienzo del mensaje serían las tres primeras letras, permutadas de alguna forma. Pero todas las permutaciones posibles son:

```
H.O     O.H     .HO     .OH     HO.     OH.
```

Tal vez la última nos sirva, pero sólo en un contexto poético (o extremadamente cursi)[13]. Así que mejor podemos probar con $n = 4$.

Ejercicio.– Escribe todos los posibles inicios del mensaje original con $n = 4$ y, si no encuentras ninguna opción viable, con $n = 5$.

Ejercicio.– Separa el texto en bloques de un tamaño fijo, e intenta descartar como viables algunos de esos tamaños por la composición de alguno de los bloques.

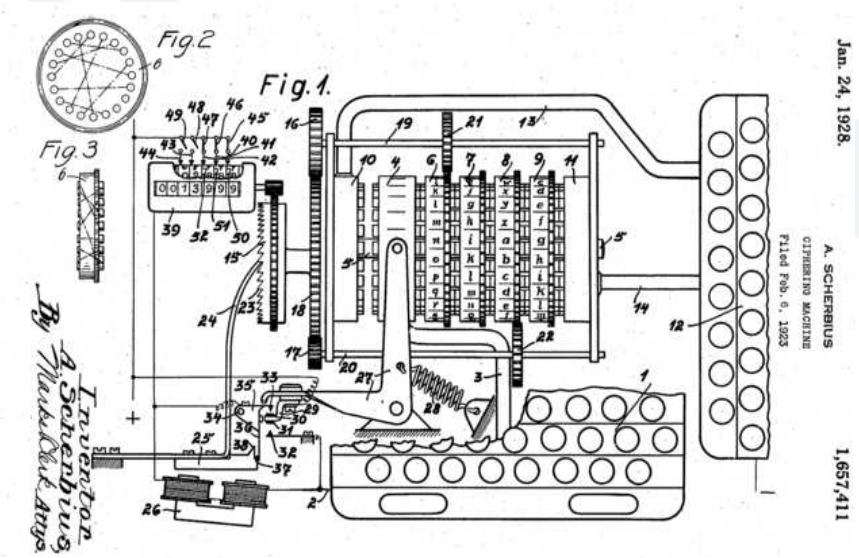
Una vez fijada una posible longitud de permutación conviene acortar la cantidad de permutaciones posibles. Hay que tener en cuenta que, si $n = 5$, tenemos $5! = 120$ opciones, y si $n = 6$, tendríamos $6! = 720$. Utilizando un ordenador es manejable, pero no tanto a mano. Para ello, podemos analizar el texto y fijarnos en que tenemos la suerte de tener una letra peculiar: la `Q`. Peculiar porque *debe* ir escrita seguida de la `U`. Así que, como en el texto tenemos `...UQ...`, sabemos a ciencia cierta que, en nuestra permutación, hay dos números de la forma $\{i, i+1\}$, que van en números de la forma $\{j-1, j\}$.

Ejercicio.– Escribe las permutaciones, para $n = 5$ que verifican esta propiedad. ¿Cuántas hay? Descifra el mensaje utilizando estas ideas.

[13] Moraleja: NO utilizar caracteres especiales para marcar el espacio entre palabras.

Enigma.

La máquina Enigma, creada por Arthur Scherbius en 1918 y patentada en 1928, supuso un hito definitivo en la encriptación. En el fondo era una evolución de otros modelos electromecánicos que intentaban hacer más sencilla y automatizada la tediosa tarea de encriptar y desencriptar mensajes (sobre todo de contenido militar). Sin embargo, la sutileza de su diseño y la versatilidad del sistema la convirtieron, con justicia, en la joya de la corona de la criptografía de clave privada de la era pre–digital.



Enigma consistía, en su modelo más sencillo (el modelo comercial, no el que se diseñó para uso militar) en tres rotores, que no eran sino permutaciones de las letras que actuaban consecutivamente. Pongamos por ejemplo[14] la siguiente situación, donde los rotores son las filas de abajo de los tres primeros bloques (las de arriba están puestas como cortesía):

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
EKMFLGDQVZNTOWYHXUSPAIBRCJ

ABCDEFGHIJKLMNOPQRSTUVWXYZ
AJDKSIRUXBLHWTMCQGZNPYFVOE

ABCDEFGHIJKLMNOPQRSTUVWXYZ
BDFHJLCPRTXVZNYEIWGAKMUSQO

ABCDEFGHIJKLMNOPQRSTUVWXYZ
YRUHQSLDPXNGOKMIEBFZCWVJAT
```

Si pulsábamos, por ejemplo `Y`, el camino que seguía la letra era

$$\texttt{Y} \longmapsto \texttt{C} \longmapsto \texttt{D} \longmapsto \texttt{H}$$

[14] Ésta era una configuración auténtica de un modelo de Enigma.

La letra H llegaba entonces a la última parte de la máquina, denominada *reflector* que, como se puede ver, no es sino un producto de trasposiciones disjuntas. Esto es, empareja las letras dos a dos y cambia cada letra por su pareja. En concreto, cambia H por D, y luego D realiza el camino inverso por los tres rotores:

$$\mathtt{H} \longmapsto \mathtt{D} \longmapsto \mathtt{B} \longmapsto \mathtt{J} \longmapsto \mathtt{Z}$$

Así, Y se cifraba como Z. Para evitar el posible ataque por análisis de frecuencias, una vez codificada una letra, el primer rotor se movía un lugar a la izquierda, con lo que la nueva configuración resultaba:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
KMFLGDQVZNTOWYHXUSPAIBRCJE

ABCDEFGHIJKLMNOPQRSTUVWXYZ
AJDKSIRUXBLHWTMCQGZNPYFVOE

ABCDEFGHIJKLMNOPQRSTUVWXYZ
BDFHJLCPRTXVZNYEIWGAKMUSQO

ABCDEFGHIJKLMNOPQRSTUVWXYZ
YRUHQSLDPXNGOKMIEBFZCWVJAT
```

Una vez que el primer rotor daba una vuelta completa, el segundo se movía un lugar, y cuando el segundo había dado la vuelta completa, se movía un lugar el tercero. El reflector, obviamente, era invariante.

Ejercicio.– ¿Cuántas letras había que cifrar para que la configuración de los rotores volviese a la posición original?

Una de las grandes ideas que subyacen en este aparato, y que lo convierten en una maravilla del diseño industrial, es precisamente la presencia del reflector. Gracias a él, el proceso de cifrado y de descifrado es *exactamente el mismo*. Dicho de otro modo, si Alice decide enviar Y a Bob (un mensaje bastante escueto, pero mensaje al fin y al cabo), lo cifra como Z y lo envía. Bob entonces, *cifra* Z y, como se puede comprobar inmediatamente, gracias al diseño del cifrado, obtiene Y, el mensaje original.

En este modelo simple, la clave de cifrado para Alice (y la de descifrado para Bob) consistía únicamente en tres letras: la posición inicial de los tres rotores (en el ejemplo anterior sería EAB), que se podían escoger libremente, dando lugar a un total de 26^3 claves, un número más que respetable para una seguridad robusta (en los años 20). Los modelos militares, sin embargo, eran bastante más complejos. Uno de los más básicos[15] consistía típicamente en:

- Un conjunto de cinco rotores. De ellos, se elegían tres, que se podían hacer funcionar en el orden que se quisiera.
- Un controlador para el movimiento del segundo y el tercer rotor. Esto es, en lugar de que cada rotor girase cuando el anterior hubiera dado una vuelta completa, se podía

[15] Los más complejos, que eran los que utlizaba la Kriegsmarine, la Marina alemana, no fueron completamente desencriptados hasta la década de los 70.

hacer que el segundo rotor girara, por ejemplo, cuando `N` pasara por la primera posición en el primer rotor (y análogamente, escoger una posición concreta para el giro del tercer rotor).

- Un tablero de *stecker pairs*. Esto era un tablero de clavijas, cada una asociada a una letra. Si se conectaban dos de estas clavijas mediante un cable, las dos letras conectadas eran intercambiadas en el proceso de cifrado por la máquina. Un sistema de encriptación estándar incluía seis stecker pairs.

Ejercicio.– Estudiar qué datos debía contener la clave de encriptación del modelo militar de Enigma que hemos descrito. ¿Cuántas eran las posibles claves para ese modelo?

La proeza (no puede denominarse de otra manera) del equipo de matemáticos, lingüistas, criptográfos y personal de apoyo del Servicio de Inteligencia Británico, enclaustrados en la mansión de Bletchley Park (los equipos denominados *Ultra* y *Estación X*) representa un hito espectacular en la coordinación de investigación pura, desarrollo tecnológico y recursos humanos[16]. Constancia de ello deja, por ejemplo, la fabricación (casi artesanal) de *Colossus*, uno de los primeros ordenadores de la historia, pensado y diseñado por Alan Turing con el único objetivo de filtrar las posibles claves alemanas mediante el ataque que él mismo concibió, el denominado ataque de texto escogido o *chosen plaintext attack*.

Cifrado de un solo uso.

Llegados a este punto podríamos preguntarnos si existe un cifrado que resista, al menos por ahora, cualquier ataque. La respuesta es sí y el protocolo se denomina cifrado de un solo uso, o *one–time pad*. La idea de este protocolo perfecto es tan sencilla como un código de Vigenère, pero con una clave tan larga como el propio mensaje. De esta forma cada letra se encripta de una manera completamente diferente (si no, lo es por coincidencia, no por método).

¿Es inatacable este sistema? La respuesta, sorprendetemente (o quizás no tanto porque lo hemos mencionado arriba) es sí. Los problemas de este protocolo no tienen que ver con la seguridad sino con la gestión y la logística. Sin embargo, para comunicaciones donde la seguridad era prioritaria sobre cualquier otra cuestión, es el sistema que se ha utilizado hasta tiempos relativamente modernos (concretamente hasta los años 80–90 en los últimos coletazos de la Guerra Fría)[17].

En la práctica, un one–time pad *de andar por casa* se puede tomar como un código de Vigenère, tomando como clave un texto previamente acordado (típicamente se trataba de un pasaje de un libro). A día de hoy, un cifrado de este tipo estaría a salvo de los mejores equipos de las agencias de inteligencia de cualquier país.

[16]Por no mencionar que representa la mayor inversión jamás realizada por un gobierno en una investigación matemática.

[17]Una referencia excelente, sobre todo como novela en sí misma, es *El espejo de los espías*, de John LeCarré.

Códigos actuales.

En la codificación en clave privada, tal y como la entendemos hoy, hay a grandes rasgos dos familias importantes que operan de forma cualitativamente distinta:

1. Códigos de flujo[18]. Estos códigos están asociados a una función (conocida), denominada *generador de claves*, que denotarmos $K(\cdot)$. Esta función genera una clave para encriptar a partir de nuestra clave privada k, y esta clave $K(k)$ es la que realmente se usa en la encriptación. Si la función está *bien escogida* la seguridad es próxima a la que proporciona un one–time pad. Con "bien escogida" queremos decir:

   - Que la longitud de la clave $K(k)$ debe ser grande. Esto es necesario porque una clave k genera siempre la misma clave de encriptación, de manera que este hecho puede usarse para atacar el encriptado. Cuanto más largo sea el periodo de repetición (y menos conocida su longitud), menos opciones habrá de que un ataque de estas características tenga éxito.
   - Que la clave $K(k)$ tiene que ser *esencialmente* aleatoria. Esto quiere decir que debe pasar unos test, denominados *test aleatorios*, que determinan que, en la práctica, el resultado obtenido es indistinguible de una cadena aleatoria de números.
   - Que la complejidad del algoritmo para hallar $K(k)$ sea polinomial.

2. Códigos de bloque. Estos códigos funcionan rompiendo el mensaje en bloques de un tamaño prefijado anteriormente y, por supuesto, conocido. En los casos que veremos con detalle son de 64 bits (DES) y 128 bits (AES). La clave para encriptar se aplica entonces a cada bloque por separado. Es un sistema más versátil que el anterior. En particular, los códigos que se utlizan en la actualidad con más frecuencia son códigos de bloque iterados. En estos códigos, la misma operación de encriptación se realiza varias veces (16 en el caso de DES, una cantidad variable en el caso de AES).

Ejemplo.– Veremos un código en bloque *no* iterado: el cifrado de Feistel que, como veremos más adelante, es la base del sistema DES y que tiene ciertas similitudes con los cifrados de flujo. En el cifrado de Feistel fijamos (públicamente) una función $F(\cdot,\cdot)$ que toma como entrada la clave de encriptación k (o algo obtenido a partir de ella) y un bloque de texto que se *la mitad* del texto a encriptar. Operamos como sigue:

a) Dividimos nuestro mensaje en dos mitades: $m = [l|r]$.

b) Hallamos $F(k, r)$.

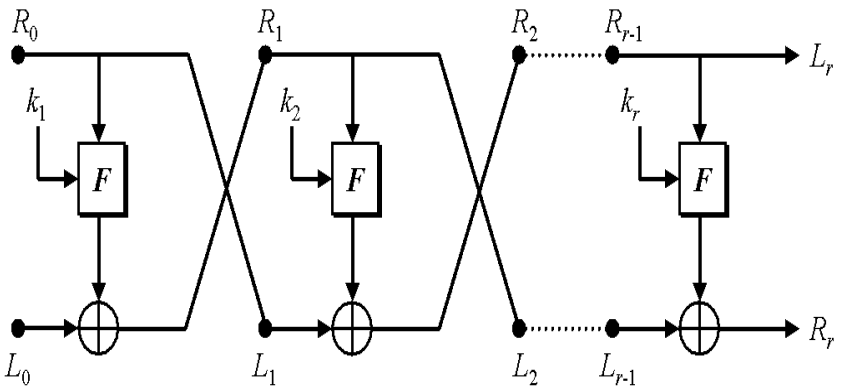
c) Calculamos[19] $M = [r|l \oplus F(k, r)]$

[18] Algunos ejemplos son el LFSR y el RC4.

[19] La operación $\oplus$, en el contexto de encriptación binaria, es la operación definida por $1+1 = 0+0 = 0$, $1+0 = 0+1 = 1$, o sea, es la *suma sin llevada*, también llamada XOR.

A primera vista no parece gran cosa. De hecho, la primera mitad del texto encriptado es la segunda mitad del texto original. Sin embargo, si realizamos una nueva codificación, ya todo resto del texto original desaparece. El protocolo fue originalmente diseñado (por Feistel y otros investigadores de IBM en los años 70) pensando desde un principio en aplicarlo de forma iterada.

Gracias a su peculiar y simple estructura, la decodificación es muy simple. Bob, una vez recibido $M = [r|l \oplus F(k,r)] = [L|R]$ y calculada $F(k,r)$ (Bob conoce r ya que $r = L$), puede recuperar $m = [R \oplus F(k,L)|L]$. Notemos que la sencillez de ambos procesos *no dependen* de la función $F(\cdot,\cdot)$, lo cual confiere a este protocolo una enorme capacidad de adaptación a las necesidades concretas de Alice y Bob.



DES.

Data Encryption Standard (DES) ha sido el protocolo de encriptación en clave privada más usado desde mediados de los 70 (¿cómo eran los ordenadores entonces?) hasta el año 2000 con la llegada de AES–Rijndael. El asombroso progreso en el campo de la computación del que esos años fueron testigo, y el hecho de que DES se mostrara resistente a los sucesivos ataques dan una idea de su enorme versatilidad y de lo bien diseñado que estaba el protocolo. La idea básica está en utilizar de forma iterada las rondas de Feistel que acabamos de ver. Concretamente, los datos técnicos del sistema son los siguientes:

- Es un código en bloque, que usa bloques de 64 bits (si pensamos, por ejemplo, en ASCII, tendríamos bloques de 8 caracteres).
- La clave secreta que comparten Alice y Bob, k, posee 56 bits.
- La operación básica es un cifrado de Feistel, que se repite 16 veces (más detalles a continuación).

Recordemos que el cifrado de Feistel operaba esencialmente como sigue:

$$m = [l|r] \longmapsto M = [r|l \oplus F(k,r)]$$

Aparte de realizar esta operación 16 veces, DES comienza aplicando una permutación y termina aplicando la inversa de esta permutación. El motivo de la existencia de esta permutación, habitualmente llamada IP es desconocido, o cuando menos sospechoso. Pero el caso es que el esquema completo sería:

$$m = [l|r] \longmapsto IP(m) \longmapsto \text{(aplicar 16 veces el cifrado de Feistel)} \longmapsto$$
$$\longmapsto N \longmapsto M = IP^{-1}(N).$$

El primer y el último paso son sencillos, simplemente aplicamos una permutación. En las tablas al final de la sección podéis ver la permutación IP y su inversa (también llamada FP). Están escritas de manera que las imágenes se leen siguiendo las filas (hay 8 filas y 8 columnas). Esto es, el bit 37, tras la permutacin IP, por ejemplo, ocupa el puesto 52, ya que en la table 37 es el cuarto elemento de la sexta fila, mientras que $IP^{-1}(57)$ es el penúltimo elemento de la última fila, o sea, que el bit 57 pasa a ocupar la posición 63. Así el bit 52 de $IP(m)$ será el bit 37 de m y el bit 63 de M es el 57 de N.

Lo realmente interesante de DES es, por tanto, el funcionamiento de la función básica del cifrado Feistel. La función F no es compleja, aunque sí bastante intrincada. Primero veamos los pasos:

1. La mitad derecha de nuestro mensaje original, que denotamos r, posee 32 bits. Mediante una permutación, denominada permutación de expansión, o simplemente E, esos 32 bits se convierten en 48.

2. Esos 48 bits se suman sin llevada (operación XOR o $\oplus$) con 48 bits que provienen de la clave k.

3. Los 48 bits resultantes de esta suma se distribuyen en grupos de 6 y se introducen en 8 *S-cajas*, de las cuales salen 4 bits.

4. Los 32 bits que salen de las 8 S-cajas se permutan utilizando una permutación de S_{32}, denotada habitualmente P.

Veamos con más detenimiento cada paso.

En el primer paso, aplicamos E que no es en sentido estricto una permutación, ya que toma 32 bits y nos devuelve 48, a costa de repetir 16 de estos bits (y reordenarlos de forma no trivial). El motivo de esta expansión, o más precisamente, cuáles son los bits que se repiten y dónde se colocan se hará explícito luego.

En el segundo paso realizamos un XOR con una clave de 48 bits obtenida de nuestra clave k (que posee 56 bits, como dijimos). Esta clave de 48 bits varía dependiendo de cuál de las 16 vueltas al cifrado de Feistel sea la que estamos efectuando. Veamos cómo se obtienen estas *claves de vuelta*. Primero añadimos, por cada 7 bits, un bit de paridad (o sea, la suma de los 7 anteriores módulo 2), y así obtenemos 64 bits. Ahora aplicamos una permutación, denominada comúnmente $PC-1$, que toma estos 64 bits y nos devuelve 56. Si nos fijamos, esta permutación *sí* es una permutación de verdad, porque en realidad descarta los bits de paridad (que se introducen para controlar errores en la clave). Esta clave $PC-1(k)$ se divide en dos mitades

$$PC-1(k) = [c_0|d_0]\,,$$

y ahora se aplica la siguiente operación de forma iterada:

- Si estamos en la ronda 1, 2, 9 o 16; *giramos* 1 bit a la izquierda (o sea, movemos todos los bits a la izquierda, y el primero lo colocamos al final).
- Si estamos en otra ronda, giramos 2 bits a la izquierda

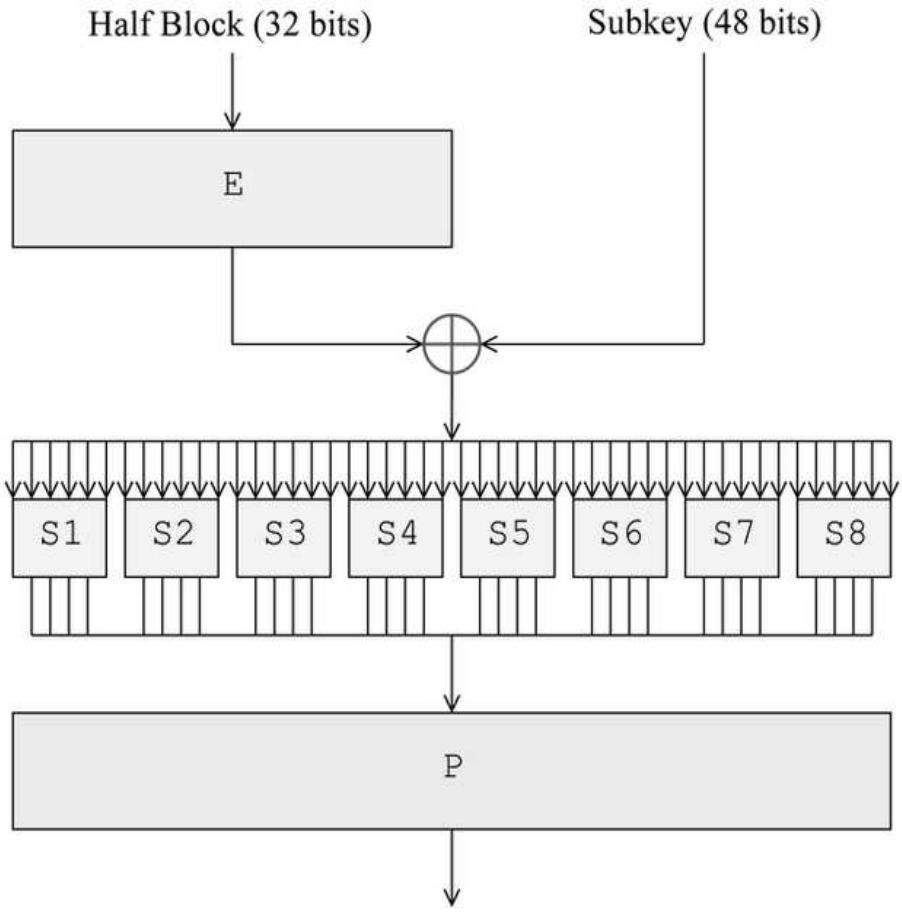
Cuando finalizamos hemos obtenido dos mitades c_i y d_i, que se vuelven a unir en $[c_i|d_i]$. Estos 56 bits se someten ahora a la acción de $PC-2$ (otra falsa permutación), que los reordena (y descarta los bits 9, 18, 22, 25, 35, 38, 43, 54. Así se obtiene la clave k_i, correspondiente a la ronda i, de 48 bits.

En el tercer paso pasamos cada grupo de 6 bits por la S-caja correspondiente. Notemos que los bits de una S-caja son todos distintos (esto es, nunca provienen dos del mismo bit original de m). De hecho, el diseño es tal que, en cada caja, de los 6 bits que entran, 4 son siempre bits que están representados en otra caja. Esto es esencial para que el proceso sea reversible, ya que al pasar de 6 bits a 4 no *se pierde* información. La forma de determinar la salida de cada caja es la siguiente:

- Los bits 1 y 6, entendidos como un número en base 2, nos dan un entero entre 0 y 3, al cual le sumamos 1.
- Los bits 2, 3, 4 y 5, entendidos como un número en base 2, nos dan un entero entre 0 y 15, al cual le sumamos 1.
- Usando ambos números, miramos en la caja, que no es más que una matriz 4×16, y obtenemos un elemento de esta matriz, que es un entero entre 0 y 15. Las cifras en base 2 de este entero es la salida de la S-caja.

El cuarto paso no es más que aplicar la permutación P. Que ya está bien, digo yo.

Permutación E	**Permutación** P
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	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



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

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

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

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

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

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

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

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

Permutación IP							
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

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

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

Permutación $PC-2$					
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

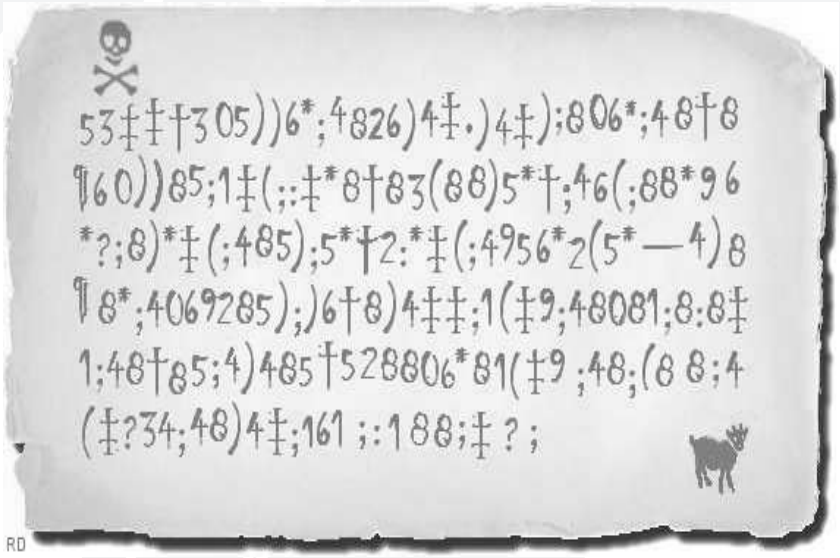
Ejercicios Tema 3

Ejercicio 3.01.– Al enfrentarnos a un texto cifrado por sustitución tipo César, un método alternativo a la fuerza bruta consiste en hacer un diagrama de frecuencias con los símbolos del texto. Explica cómo el análisis de este diagrama puede conducirnos a averiguar la clave de encriptación.

Ejercicio 3.02.– Descifra el siguiente mensaje en inglés, cifrado por sustitución

PXPXKXENVDRUXVTNLXHYMXGMAXYKXJNXGVRFXMAHWGXXWLEHGZXKVBIAXKMXQM

Ejercicio 3.03.– Descifra el texto de *El escarabajo de oro*. Está escrito en inglés.



Ejercicio 3.04.– Una versión algo más compleja del cifrado por funciones afines consiste en usar funciones afines de varias variables. Esto es, por ejemplo, funciones del tipo

$$\begin{array}{rcl} f : (\mathbf{Z}/\mathbf{Z}27)^2 & \longrightarrow & (\mathbf{Z}/\mathbf{Z}27)^2 \\ \begin{pmatrix} x \\ y \end{pmatrix} & \longmapsto & A\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \end{array}$$

donde $A \in \mathcal{M}(2; \mathbf{Z}/\mathbf{Z}27)$ y $a, b \in \mathbf{Z}/\mathbf{Z}27$. Estudiar qué condiciones se deben verificar para que f sea biyectiva (y, por tanto, pueda usarse para encriptar)[20].

Ejercicio 3.05.– Estudia de cuántas claves distintas se puede disponer usando el cifrado de funciones afines bidimensionales.

[20]El 27 se ha usado por tomar el número de letras del alfabeto castellano, pero se puede (y se debe) cambiar por el número de símbolos del alfabeto que estemos considerando.

Ejercicio 3.06.– Razona por qué el análisis de frecuencias no es útil para el cifrado por funciones afines bidimensionales. ¿Es posible que alguna variante de este ataque sí sea útil?

Ejercicio 3.07.– Cifra el texto siguiente:

SWEARNOTBYTHEMOONTHEINCONSTANTMOON

mediante una función afín bidimensional (ojo, consideramos 26 símbolos) dada por

$$A = \begin{pmatrix} 2 & 3 \\ 7 & 8 \end{pmatrix}, \quad \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 11 \\ 23 \end{pmatrix}.$$

Ejercicio 3.08.– Se ha interceptado un mensaje. Sabemos que el emisor del mensaje utiliza una encriptación por funciones afines bidimensionales, y el siguiente alfabeto:

A=0,..., Z=25, _=26, .=27, ,=28, ?=29

El texo del mensaje es S_GNLIKD?KOZQLLIOMKUL.VY y sospechamos que el emisor ha finalizado el mensaje con su firma: KARLA.. Descifra el mensaje.

Ejercicio 3.09.– Cifrar el siguiente mensaje según el método de Vigenère:

Mensaje (m): THERE ARE MORE THINGS IN HEAVEN AND EARTH
HORATIO THAN ARE DREAMT OF IN YOUR PHILOSOPHY
Clave (k): HAMLET

Ejercicio 3.10.– Supongamos que tenemos que descifrar un texto del cual sabemos que ha sido cifrado por el método de Vigènere. Explica cómo es posible confirmar o descartar una hipótesis sobre la longitud de la clave (formulada, por ejemplo, usando el método Kasiski), calculando *un único* diagrama de frecuencias.

Ejercicio 3.11.– Intenta descifrar el siguiente texto. Está escrito en inglés y cifrado por el método de Vigènere.

ZMXZBVIDYAWAEALGTPTEAMZRVNMOCZWBAHNDSIWGZGAINYIUBOWKMIFWIHWF
AWZGBXBXMPJMQDVFOXSSLQFGFWMPGJZQXSVOIEGYWMBWAYIZRGZMHWFAWZHU
SBIOFHTMBGWLUBZQJDOVFAFWYDZQANAVEKVLPUBGZMECHFLATFATQBPWQZFR
KBXSFKLDSNEAUKNDSQRNDWZSASZDCJKBDSRLAATPGJNZRKBABRFMMHULPQVN
DWATNKBDSRLTMACABGFAWLYMPGTXOELWFVRUWXRNFLPOZHETSAEGQMRKEQFR
KBMPOWLNMGZMRZNKPATNFMABYAOTHGZIFGCDQFHUWVUUULIZRGGCOVRVBTSF
GCZRBXAUZRFKQOAVQZHUWVMYRVTUUULQEOJLMZHUGCEOAVXQCCDMYOLTMYCE
WXQCCDMFOYCQZUJABTCHLABSNCQZUCWWBZRZMMFVFOIWGZWGHYAAFSAAVSDR
GXXSJJQFWAYAABTKBTOGNWUQRKVQJRJATOEWIZRAGWZSQSZQRQAAFIETBTSF
GCZRBXAUZRFKQTBGTEGNALUMBMLABBLSZCJKQXSAUMXWXWIOOAUMDUEGEEVR
SZYMJGZPGGZIFWZAOTHGWIOVLGCFOXWUKOEEAFVNLQYWTZBDSNUPKCHTCFAL
OWDRFDQWSFATQBGJIUBQJWBGSWTXOAVMOVBWLUBGZMISYDAATFATQBPWIZRG
ZMBSBHTQPBOMPOAVXDOLWLFCGZMZSBFOARGZMKANVMMBQLPQGVYVRZNKPQRB
MBUHFOIDBVFOUBGZMICEVAFVNLQFKNKNAFZAVSOAVBTSFAOZGNALFVROWDRF
GNFVRHZADUWBEOEWEDWGLMZCALPQGHTEMMJSTXGNFLFSAWUQBGZIXZFSVPKU
AABSEWLUBGZMECHFLECSKQXSAUM

Ejercicio 3.12.– Aplica la permutación

$$k = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 5 & 2 & 1 \end{pmatrix}$$

al siguiente mensaje

```
MUCHOS A~OS DESPUES FRENTE AL PELOTON DE FUSILAMIENTO
EL CORONEL AURELIANO BUENDIA HABIA DE RECORDAR AQUELLA
TARDE REMOTA EN QUE SU PADRE LO LLEVO A CONOCER EL HIELO
```

Ejercicio 3.13.– Descifra el siguiente texto, cifrado por una permutación de longitud indeterminada.

```
LOOECRENCNLESTSEIOTNAEETCNOYIC~SOAOLSTNSEEACNTYIO~SCAO
EUIDSVAIUDMNOMNTAITPRUOALEAALGASIREESATNTNSSNEEIIPRTOU
   EPIOXLIONCTIECBVNIENLLEEOETMMNDRSOEEODRPNEIRAMED
```

Ejercicio 3.14.– Estudiar qué datos debía contenter la clave de encriptación del modelo militar de Enigma que descrito en los apuntes de teoría. ¿Cuántas eran las posibles claves para ese modelo?

Ejercicio 3.15.– Descifrar el siguiente mensaje codificado en Enigma:

```
RFXLH PYIYW XGHHD CIJVR WVBXW RHMGB JIROE DLHWD JCLPS LIF
```

Las siguientes claves diarias utilizadas son:

- Rotores: `3-1-2`
- Posición de giro del rotor siguiente: `1-1-1`
- Stecker pairs: `AF WZ PR HL`
- Posición inicial de los rotores: `F P W`

Se recomienda usar el simulador de Enigma disponible en la página web de Bletchley Park: `http://www.bletchleypark.org.uk/content/enigmasim.rhtm#`.

Ejercicio 3.16.– El ataque por índice de coincidencia fue un ataque a Enigma que no se pudo llevar a cabo en Bletchley Park (aunque se diseñó teóricamente) por falta de recursos computacionales. Supongamos que tenemos un texto escrito en un alfabeto de m símbolos, y que consta de n caracteres. Llamamos $f_1, ..., f_m$ a las frecuencias de cada símbolo en el texto y definimos el índice de coincidencia del texto como

$$IC = \frac{1}{n(n-1)} \sum_{i=0}^{m} f_i(f_i - 1).$$

En un texto aleatorio se tiene $IC \sim 0.038$. Escoge un texto cualquier en castellano y en inglés (de, al menos, 300 caracteres sin contar espacios) y calcula su índice de coincidencia.

Ejercicio 3.17.– Para reforzar la seguridad del sistema Enigma, los alemanes decidieron que no se cifraría todo el texto con la clave del día. En lugar de eso, sólo se cifrarían las tres primeras letras de cada mensaje, que corresponderían con una nueva posición inicial de los rotores, con la que se cifraba *realmente* el mensaje. Explica cómo esto aumentaba la seguridad del sistema.

Paradójicamente, esto supuso una debilidad por el uso habitual de ciertas claves de mensaje (generalmente vulgarismos o nombres personales). Explica cómo una elección inadecuada de claves de mensje hacen que este procedimiento se convierta en un gran fallo de seguridad que compromete todo el sistema a partir del error de una única estación.

Ejercicio 3.18.– Un ataque de texto escogido o *chosen plaintext attack* se basa en la hipótesis de que en un determinado texto cifrado aparece una palabra para intentar deducir la clave, como en el ejercicio 3.08. Razona en qué tipo de cifrado (de los estudiados en este tema) es más eficaz este ataque. En Bletchley Park se utilizó fundamentalmente sobre mensajes cifrados que incluían partes meteorológicos y sobre las últimas palabras de los mensajes.

Ejercicio 3.19.– Suponemos que ciframos en bloques de 64 bits y con claves de 32 bits. Describe explícitamente un cifrado iterado de Feistel con tres aplicaciones, donde la función F sea simplemente

$$F(k, r) = \sigma(k \oplus r),$$

donde σ es la operación que permuta los 16 primeros bits con los 16 últimos.

Ejercicio 3.20.– Como DES cifra en bloques de 64 bits, en la práctica un bloque cifra un texto de 8 caracteres ASCII (imprimibles o no). Cifra el mensaje `RIJNDAEL` tomando como clave (de 56 bits, o sea, 7 caracteres ASCII) `FEISTEL`. Se recomienda disponer de bastante tiempo por delante.

Tema 4. Criptografía de clave pública

Lobos y corderos. Y lechuga. Y regalos.

Planteamos un antiguo problema, bastante simple:

Ejercicio.– Un barquero tiene que transportar al otro lado de un río tres cosas: un lobo, un cordero y una lechuga. El problema es que sólo puede transportar una cosa cada vez, y si deja solos al cordero y a la lechuga, cuando vuelva sólo quedará uno de ellos (y *no* será la lechuga). Del mismo modo, si deja solos al lobo y al cordero, cuando vuelva se lo va a encontrar todo perdido de sangre. ¿Qué puede hacer el barquero?

Aunque resulte algo extraño, este problema no está muy alejado del que motivó, a principios de los años 70, el nacimiento de un concepto que revolucionó completamente la criptografía moderna: la clave pública. Los primeros[21] en plantearse un concepto diferente de criptografía fueron Diffie y Hellman a mediados de los 70.

En aquel entonces, el enorme desarrollo de las comunicaciones y la tecnología de la información que se experimentó a partir de los años 60 comenzó a hacer posibles aplicaciones criptográficas a entornos distintos de los tradicionales (el militar, básicamente). En concreto, el mundo comercial comenzó a interesarse por las posibilidades que ofrecía la comunicación segura, al principio entre corporaciones (evitando los problemas derivados del espionaje industrial), hasta llegar al día de hoy en el cual también las comunicaciones entre clientes y empresas están sujetas a encriptaciones que velan por la privacidad de estos mensajes.

En términos criptográficos, el problema al que se enfrentaron Diffie y Hellman era complejo. Las comunicaciones se estaban desarrollando a una velocidad vertiginosa, y por primera vez en la historia había grupos no militares interesados en tener canales seguros: las nuevas empresas demandaban comunicaciones rápidas (eso se lo dejamos a los ingenieros) y fiables. De hecho, a diferencia de la situación tipo en el ámbito militar, donde el protocolo era uno a uno, las empresas normalmente querían poder comunicarse *con cualquier otra empresa*.

Ellos pensaban en realidad en una dificultad derivada de la criptografía de clave privada, que no era otra que el intercambio de claves. Con este aumento exponenecial de mensajes, los usuarios de protocolos criptográficos se enfrentaban al siguiente problema: las claves bien usadas tienen una vida útil y, si se quiere que el sistema sea robusto, es procedente tener un sistema flexible y sencillo de intercambio de claves que se pueda usar

[21] En realidad no, pero la confidencialidad de los archivos del GCHQ, el servicio de comunicaciones del espionaje británico, hicieron que ésta fuera la historia *oficial*.

con frecuencia. El problema que se planteaba para la gestión de claves, en el contexto de la criptografía de clave privada, era por tanto muy difícil con las herramientas disponibles.

El problema que ilustra mejor la inspiración de Whitfield Diffie y Richard Hellman es el siguiente:

Ejercicio.– Alice quiere enviar un regalo de cumpleaños a Bob que está pasando unas vacaciones en el extranjero. Sin embargo, Bob le dice que allí (en el extranjero, así, en general) el correo no es nada seguro y que cualquier paquete que no vaya en una caja con candado es sistemáticamente abierto y robado. Alice puede comprar candados y Bob también, pero obviamente ninguno de los dos tiene una llave que le valga a los candados que compre el otro. ¿Cómo puede hacer llegar Alice su regalo?

La idea de Diffie y Hellman cristalizó en el denominado DHEP (Diffie–Hellman Exchange Protocol) y se centraba en el siguiente hecho: escojamos un primo p y un g, un natural menor que p. Ahora escogemos un entero $a \in \mathbf{Z}$ cualquiera. Entonces:

- Calcular $g^a \mod p$ es un problema sencillo (complejidad logarítmica, concretamente $\mathcal{O}(\log a \log^2 p)$, ver ejercicio 1.17.).
- Sin embargo, el problema inverso (esto es, dado h del cual sabemos que es de la forma $g^a \mod p$, hallar a) es un problema extremadamente complejo.

Este problema,

$$\text{Conocidos } \{h = g^a \mod p,\ p,\ g\}, \text{hallar } a$$

conocido por el sugerente nombre de *problema del logaritmo discreto* (DLP) es un primer ejemplo de lo que se ha dado en llamar *funciones trampilla*, funciones en las que es muy sencillo calcular la imagen de un elemento dado, mientras que hallar la inversa de un elemento concreto es mucho más complejo aún en este caso, en el que existen infinitas soluciones para a. Hay varios algoritmos que resuelven este problema, la mayoría de los cuales están diseñados para ser eficaces sobre familias particulares de módulos.

Diffie y Hellman utilizaron astutamente este hecho para resolver el problema del intercambio de claves de la siguiente manera: Alice y Bob quieren compartir una clave (que supondremos que es un entero mod p) y acuerdan, *por un canal de comunicación no necesariamente seguro* que van a usar para generarla un primo P y un $G \mod P$. Por otra parte, Alice escoge un entero $a \in \mathbf{Z}$ y Bob escoge un entero $b \in \mathbf{Z}$. Entonces Alice calcula $H_1 = G^a$ y lo envía a Bob, que hace lo propio con $H_2 = G^b$. Finalmente Alice puede hallar

$$h = H_2^a = G^{ab}$$

y Bob puede hallar

$$h = H_1^b = G^{ab},$$

que será la clave que usen ambos en lo sucesivo.

Si nos situamos en la perspectiva de Eve, que quiere hallar la clave acordada, podemos comprobar que la información a la que tiene acceso, porque ha sido transmitida por un canal no seguro, sería:

$$P,\ G,\ H_1 = G^a,\ H_2 = G^b.$$

Si Eve quiere hallar h necesita entonces a o b, con lo cual debe resolver el problema del logaritmo discreto, y esto (si están bien escogidos los actores matemáticos de este drama) no es factible computacionalmente. A fecha de hoy, el DLP no es resoluble si uno fija $P \simeq 2^{1024}$.

For your eyes only.

Tras la idea de Diffie–Hellman, establezcamos un protocolo formal de criptografía de clave pública. La idea fundamental es abandonar los conceptos tradicionales:

- Dejamos de pensar en dos usuarios: pensamos en una comunidad de usuarios que incluye a Alice, Bob y, por qué no, a Eve.
- Dejamos de pensar que la clave de cifrado y la clave de descifrado son esencialmente lo mismo (o sea, a partir de una se calcula la otra de forma trivial). En lugar de eso tenemos una función trampilla: conociendo la clave de cifrado es sencillo averiguar la de descifrado, pero el camino inverso *no es factible (computacionalmente)*.
- Dejamos de pensar en encriptar para que sólo remitente y destinatario puedan desencriptar: en lugar de eso encriptaremos de forma que *sólo el destinatario pueda desencriptar*.

Así, tenemos que Alice tiene una clave pública K_A que todo el mundo conoce. En la vida real, todas las claves aparecen y se pueden mirar en una especie de *listín telefónico*, y así mismo Bob tiene la suya K_B (incluso Eve, como usuaria del sistema). Lo importante es que K_A viene acompañada, cuando se crea, de una clave privada k_A que es la que se usa para desencriptar. Teóricamente k_A se puede hallar a partir de K_A, pero en la práctica no es posible, y sólo Alice conoce k_A (y análogamente sólo Bob conoce k_B,...).

Tenemos una función de encriptación pública que depende de dos parámetros: la clave de encriptación *pública* y el mensaje a encriptar. Tenemos también una función de desencriptación que verifica dos condiciones:

- El cálculo de la desencriptación conociendo la clave de desencriptado es rápido.
- El cálculo de la desencriptación *sin conocer la clave de desencriptado* es demasiado lento.

Si Alice quiere enviar un mensaje m a Bob, realiza el cálculo de encriptar, con la clave pública de Bob, y halla el mensaje encriptado M. Este mensaje *ni siquiera Alice podría descifrarlo* (aunque no lo necesita, porque sabe qué pone). Envía M y entonces Eve, si quiere recuperar m debe desencriptar M sin conocer k_B, mientras que Bob puede realizar este cálculo sin problema.

Por tanto, la idea es dejar de pensar en la encriptación como un protocolo entre dos usuarios idénticos para pasar a pensar en un protocolo donde ser emisor y ser receptor son conceptos distintos, porque poseen información distinta. En este contexto *se encripta*

pensando en que sólo el receptor pueda leer el mensaje. Ésta diferencia de acceso a la información hace que estos códigos se denominen en ocasiones *códigos asimétricos*, por contraposición a los códigos simétricos, que serían aquéllos en los que emisor y receptor (legítimo) comparten la misma información.

Es obvio que el diseño de un tal protocolo ha de ser muy cuidadoso para evitar un ataque alternativo al esperado que rompa la seguridad del sistema. Toda la seguridad del sistema recae en la función trampilla, ese sutil sacrificio de información que hace público un dato (la clave de encriptación) que, teóricamente, permite calcular otro dato (la clave de desencriptación) el cual, sin embargo, hay que mantener en secreto.

Las funciones trampilla no son sencillas de encontrar. De hecho, en los últimos años cada problema del que se sabe que es un buen candidato a función trampilla ha sido probado como base para un criptosistema de clave pública. En la actualidad, existen varias opciones interesantes. No por casualidad, casi todas las funciones que se usan tienen que ver con aritmética modular:

- La factorización. Si tomamos dos primos grandes (del orden de 2^{512}), pongamos p y q, cualquier ordenador multiplica enseguida $p \cdot q = N$. Sin embargo, dado N, hallar p y q no es tan simple. Este función trampilla se utiliza en la actualidad en el protocolo RSA, que veremos a continuación con detalle. Es el método más usado en la actualidad.

- El DLP. Dado un primo p y un natural $g < p$, hallar $h = g^a \mod p$ para un a cualquiera es muy sencillo. Pero dado h hallar el exponente a es, como hemos comentado, imposible (para p grande). Esto se usa en el DHEP y en un método de encriptación llamado ElGamal, que es la base de dos de los métodos más usados en internet hoy en día: PGP y GPG.

- El problema de las raíces cuadradas. Dado un entero NO primo n y $g \mod n$, es sencillo hallar $x = g^2 \mod n$. Pero dado a, hallar g (su *raíz cuadrada módulo* n) es impracticable. Este problema es la base del método de Rabin.

Cifrado RSA.

En el protocolo RSA no hay ningún tipo de dato compartido entre todos los usuarios. Cada uno de ellos, por ejemplo Alice, genera su clave de la siguiente forma:

- Escoge dos primos grandes, p_A, q_A y los multiplica $N_A = p_A \cdot q_A$.
- Calcula $\varphi(N_A) = (p_A - 1)(q_A - 1)$.
- Escoge un número E_A, primo con $\varphi(N_A)$.
- Calcula d_A, verificando $E_A \cdot d_A = 1 \mod \varphi(N_A)$.

La clave pública de Alice es entonces el par (N_A, E_A). El resto lo mantiene privado.

Supongamos que Bob quiere enviar un mensaje m a Alice. Supongamos, sin perder generalidad, que $m < N_A$. Entonces Bob lo encripta haciendo

$$M = m^{E_A} \mod N_A.$$

Una vez recibido M, Alice lo desencripta haciendo

$$M^{d_A} = \left(m^{E_A}\right)^{d_A} = m^{E_A d_A} = m^{1+r\varphi(N_A)} = m \mod N_A,$$

por el Teorema de Euler.

Observación.– Los parámetros de seguridad actuales son $N_A \sim 2^{1024}$. Además de eso, la elección de p_A y q_A debe ser cuidadosa, para evitar que la factorización de N_A sea factible por alguno de los métodos específicos existentes en la actualidad (esto es, métodos que no son buenos *en general*, pero que funcionan muy bien para factorizar determinados números).

Ejercicio.– Si Eve quiere descifrar el mensaje, una opción obvia es factorizar N_A y actuar como lo hace Alice. Sin embargo lo que Eve *necesita* es d_A. ¿Es necesario factorizar N_A para resolver este problema?

Cifrado de ElGamal.

En el cifrado de ElGamal, todos los usuarios del sistema comparten cierta información. Se escoge un cuerpo finito $\mathbf{F}_P$, que debe ser muy grande, y otro primo Q de tal forma que $Q|(P-1)$. Se escoge, así mismo, una base común, $G \in \mathbf{F}_P^*$, que tenga orden divisible por Q. En realidad pedimos que

$$G^{(P-1)/Q} \neq 1 \mod P.$$

Ahora Alice crea su clave pública escogiendo un entero n_A, con la única condición de que se cumpla $n_A < P-1$. La clave pública de Alice es entonces

$$E_A = G^{n_A} \mod P.$$

La encriptación se realiza como sigue: si Bob quiere encriptar m, primero genera una clave *efímera* h (un entero cualquiera), y calcula

$$M = (M_1, M_2) = \left(G^h, m \cdot E_A^h\right).$$

Alice, al recibir M, descifra como sigue

$$\frac{M_2}{M_1^{n_A}} = \frac{m \cdot E_A^h}{G^{h \cdot n_A}} = m \mod P.$$

Observación.– Los parámetros de seguridad actuales son $P \sim 2^{1024}$, $Q \sim 2^{160}$. Una de las grandes ventajas de este protocolo es que las elecciones de parámetros que deben ser cuidadosas las realiza *el sistema*, mientras que a los usuarios individuales sólo les compete elegir la clave privada n_A, que no tiene incidencia sobre la fortaleza del protocolo.

Ejercicio.– ¿Qué objeto puede tener el escoger G con orden divisible por Q?

Cifrado de Rabin.

El cifrado de Rabin se basa en el problema de las raíces cuadradas, que mencionamos anteriormente. Este problema es esencialmente equivalente en dificultad a factorizar (ver hoja de problemas para una de las implicaciones).

Para crear su clave, Alice escoge dos primos p_A y q_A, verificando

$$p_A = q_A = 3 \mod 4,$$

y un entero E_A tal que $0 < E_A < N_A = p_A q_A$. La clave pública de Alice es, entonces, (N_A, E_A).

Si Bob quiere enviar un mensaje m a Alice, calcula

$$M = m\,(m + E_A) \mod N_A.$$

Alice, al recibirlo, calcula

$$\sqrt{\frac{E_A^2}{4} + M} - \frac{E_A}{2} = \sqrt{\frac{(2m + E_A)^2}{4}} - \frac{E_A}{2} = \frac{2m + E_A}{2} - \frac{E_A}{2} \mod N_A,$$

con lo cual, en realidad, Alice calcula *varios* descifrados (porque hay varias raíces cuadradas, ver ejercicio 4.10.), pero obviamente sólo uno tendrá sentido.

Observación.– El hecho de escoger p_A y q_A congruentes con 3 en $\mathbf{Z}/\mathbf{Z}4$ se basa en que, en ese caso, el cálculo de raíces cuadradas módulo N_A es mucho más rápido usando el algoritmo de Shanks, que no estudiaremos en este curso.

Ejemplo.– ¿Por qué se encripta haciendo $m\,(m + E_A)$ en lugar de simplemente m^2?

Yo no he dicho eso. Bueno, sí.

La ley de acción y reacción o el principio del yin y el yang también tienen su versión criptográfica. En efecto, la criptografía de clave pública nos ha proporcionado, de repente, una manera mucho más eficaz de gestionar las claves en un entorno de múltiples usuarios: basta un listado público (y un protocolo bien diseñado, claro) y nos podemos olvidar de los problemas de almacenamiento e intercambio de claves de los viejos tiempos... ¿o no?

La verdad es que no. Para empezar, un aspecto que se ha perdido con el uso de las claves privadas es el de la *confianza*. Con la clave privada, cuando Bob recibía un mensaje encriptado de Alice, podía estar razonablemente seguro de que, en efecto, el mensaje había sido enviado por Alice. En cambio, en el nuevo contexto, cualquiera puede enviar a Bob un mensaje, firmarlo como Alice y ¿cómo puede saber Bob que en efecto es Alice quien lo manda?

Los problemas de suplantación de identidad cobraron un interés enorme en criptografía cuando por primera vez se diseñó un ataque a un protocolo (en este caso el DHEP que vimos antes) bajo la hipótesis de que el atacante (Eve, cómo no) era capaz de fingir identidades. Este tipo de ataques se conoce, habitualmente, con el nombre de *man in the middle*.

Ejemplo.– La cosa va como sigue: Supongamos que, en efecto, Eve es capaz de hacerse pasar por Bob cuando escribe a Alice, y viceversa (nada descabellado en un mundo donde las comunicaciones rara vez son personales).

Entonces en el DHEP, Eve intercepta los mensajes de Alice a Bob y, actuando como Bob, acuerda una clave privada con Alice, pongamos $K_{A,E}$. Actuando de forma análoga con Bob, acuerda una clave privada con él haciéndose pasar por Alice ($K_{B,E}$).

Ahora supongamos que Alice quiere mandar un mensaje a Bob. Lo codifica con $K_{A,E}$ y lo manda. A Eve le basta interceptar el mensaje, descifrarlo (posee la clave $K_{A,E}$), volver a cifrarlo con $K_{B,E}$ y enviarlo a Bob. Bob recibirá el mensaje convencido de que viene de Alice (y así es, en cierto sentido). De esta forma Eve puede monitorizar toda la correspondencia sin ser ni siquiera detectada.

La solución a los eventuales problemas de identidad falsa que se plantearon en los protocolos de clave pública se resolvieron con el uso de la denominada *firma digital*.

Ejercicio.– Intenta pensar cómo se podría garantizar la identidad del remitente en un protocolo de clave pública. La solución, después de la publicidad.

La idea es muy simple. Pongamos que estamos en una comunidad con un protocolo de cifrado público y que Alice quiere enviar un mensaje m a Bob, para lo cual lo cifra con la clave pública de Bob k_B, obteniendo M. A continuación, Alice genera una *firma* s (que puede ser simplemente su nombre y dirección, o una frase que le guste o lo que sea) y lo codifica *usando SU propia clave privada de descifrado* k_A, obteniendo S. Finalmente Alice envía $[M|S]$.

Bob al recibir $[M|S]$ decodifica la primera parte usando su clave privada, que sólo él conoce, k_B, y recuperando m. Después decodifica S usando K_A, *la clave pública de Alice*, y recuperando s.

Esto confirma que sólo Alice puede haber mandado el mensaje. En efecto, cualquiera puede decoficar la firma (sólo hace falta K_A, y es pública), pero en la firma no hay ninguna información relevante. La única información que contiene es que el remitente es, inequívocamente, Alice, puesto que *sólo ella puede haber codificado un mensaje de tal forma que* K_A *lo decodifique*.

Ejercicio.– Explica por qué no es una buena idea que Alice escoja un texto fijo como firma.

Ejemplo.– En el protocolo RSA, una opción para firmar el texto por parte de Alice es hacer

$$[m|s] \longmapsto \left[m^{E_B}|s^{d_A}\right]$$

Ejemplo.– El sistema DSS (Digital Signature Standard) está basado en el DLP y fue propuesto en 1991 como sistema de firma estandarizado por el NIST (National Institute of Standards and Technology), jugando un papel análogo al DES en encriptación. Es muy parecido a los sistemas de firma de Schnorr y de ElGamal (ver hoja de ejercicios).

Alice crea sus parámetros para firma digital como sigue: primero escoge un primo $Q_A \sim 2^{160}$, luego escoge otro $P_A \sim 2^{1024}$, verificando $P_A = 1 \mod Q_A$.

El grupo $\mathbf{F}_{P_A}^*$ tiene un único subgrupo cíclico de orden Q_A. Alice escoge un entero aleatorio g y calcula

$$g^{(P_A-1)/Q_A} \mod P_A.$$

Si este cálculo es disinto de 1, $G_A = g^{(P_A-1)/Q_A}$ es un generador del subgrupo cíclico de orden Q_A. Alice finalmente escoge un entero aleatorio $0 < x_A < Q_A$, como clave privada, y adopta como clave pública

$$(Q_A,\ P_A,\ G_A, Y_A = G_A^{x_A} \mod P_A)\,.$$

Para firmar un mensaje, Alice escoge una firma (o el encabezamiento del mensaje, o, más generalmente, la imagen del mensaje por una función hash[22]), llamémosla S, de forma que sea $0 < S < Q_A$. La forma de escoger S es pública, así que S lo es (no estamos suponiendo que el mensaje se vaya a encriptar, sólo que se va a firmar digitalmente).

A continuación, Alice escoge un entero aleatorio (clave efímera) $0 < k < Q_A$, y calcula

$$R = \left[\ G_A^k \mod P_A\ \right] \mod Q_A,$$

para después hallar T verificando

$$Tk = S + x_A R \mod Q_A.$$

La firma de Alice para este mensaje es entonces (R, T). Cuando Bob recibe (R, T), calcula

$$U_1 = T^{-1}S \mod Q_A, \qquad U_2 = T^{-1}R \mod Q_A,$$

y calcula

$$G_A^{U_1} Y_A^{U_2} \mod P_A.$$

Si la firma es correcta, entonces este cálculo módulo Q_A debe ser igual que R.

[22]Las funciones hash tienen la ventaja de que proporcionan, a un tiempo, firma digital y garantía de que el mensaje no ha sido manipulado. Son objeto de un trabajo personal, pero no las trataremos en el curso.

Ejercicios Tema 4

Ejercicio 4.01.– Alice quiere enviar un regalo de cumpleaños a Bob que está pasando unas vacaciones en el extranjero. Sin embargo, Bob le dice que allí (en el extranjero, así, en general) el correo no es nada seguro y que cualquier paquete que no vaya en una caja con candado es sistemáticamente abierto y robado. Alice puede comprar candados y Bob también, pero obviamente ninguno de los dos tiene una llave que le valga a los candados que compre el otro. ¿Cómo puede hacer llegar Alice su regalo?

Ejercicio 4.02.– Supongamos que queremos usar RSA en una comunidad de confianza y, para ahorrar costes, utilizamos todos el mismo $N = pq$, dado que no tenemos problemas en que otros usuarios de la comunidad lean los mensajes que nos intercambiamos. Supongamos que Eve (que no es miembro de la comunidad) intercepta un mensaje que Alice ha enviado a dos usuarios, Bob y Chuck. Las claves públicas de éstos son, respectivamente, (N, E_B) y (N, E_C).

Eve conoce por tanto $M_1 = m^{E_B}$ y $M_2 = m^{E_C}$. Calcula entonces los siguientes números

$$T_1 = E_B^{-1} \mod E_C, \qquad T_2 = \frac{1 - T_1 E_B}{E_C}.$$

Probar que con estos datos, Eve puede recuperar el mensaje original m.

Ejercicio 4.03.– Dado que en RSA hay libertad para escoger la clave de encriptación, los usuarios podrían estar tentados de escoger una clave sencilla, como por ejemplo 3. Supongamos que Alice, Bob y Chuck escogen $E_A = E_B = E_C = 3$ y que otro usuario les envía el mismo mensaje, m, interceptado por Eve en la forma

$$M_1 = m^3 \mod N_A, \qquad M_2 = m^3 \mod N_B, \qquad M_3 = m^3 \mod N_C.$$

Probar que Eve puede recuperar el mensaje m hallando la solución al siguiente sistema de congruencias (por el Teorema Chino del Resto, por ejemplo):

$$\begin{cases} x = M_1 & \mod N_A \\ x = M_2 & \mod N_B \\ x = M_3 & \mod N_C \end{cases}$$

Una elección habitual hoy para los sistemas RSA es escoger $E = 65537$.

Ejercicio 4.04.– Este ejercicio estudia, en un protocolo RSA y dado un usuario con clave pública (N, E), cómo factorizar N (de manera eficaz) si conocemos d. Por tanto, factorizar N y desencriptar RSA son problemas equivalentes. Supongamos pues que conocemos N, E y d tales que $d \cdot E = 1 \mod \varphi(N)$.

- Supongamos que tenemos $y \in \mathbf{Z}$ tal que $y^2 = 1 \mod N$. Probar que entonces podemos calcular en tiempo polinomial un divisor de N
- Probar que el divisor calculado anteriormente es no trivial si $y \neq \pm 1 \mod N$
- Probar que si escogemos un entero cualquiera x podemos hallar (en tiempo polinomial) $y_1 = x^{(d \cdot E - 1)/2}$, que es una raíz cuadrada de 1 en $\mathbf{Z}/\mathbf{Z}N$.

- Probar que, si en el paso anterior, $y_1 = 1 \mod N$, podemos calcular $y_2 = \sqrt{y_1} \mod N$ y es de nuevo raíz cuadrada de 1 en $\mathbf{Z}/\mathbf{Z}N$.

Este algoritmo es un algoritmo *Las Vegas*. Esta clase de algoritmos son probabilísticos, y bien pueden no terminar o no dar una respuesta. Sin embargo, si la dan, es siempre una respuesta correcta.

Ejercicio 4.05.– Intenta factorizar, usando el algoritmo anterior, $N = 1441499$, conocidos $E = 17$ y $d = 507905$.

Ejercicio 4.06.– Supongamos que estamos creando una clave para RSA y hemos escogido ya p, q y E. Probar que podemos escoger d como

$$d = E^{-1} \mod \text{lcm}(p-1, q-1).$$

Ejercicio 4.07.– Probar que si escogemos en RSA (por error, se entiende) N de manera que es primo, Eve puede descifrar fácilmente cualquier mensaje que nos envíen.

Ejercicio 4.08.– Una idea que liga factorización con diferencias de cuadrados, y que debemos a Fermat, funciona como sigue: imaginemos que $N = pq$. Entonces tenemos que

$$N = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2 = t^2 - s^2.$$

La idea de Fermat para factorizar consistía en buscar enteros $t > \sqrt{N}$ tales que $t^2 - N$ es un cuadrado perfecto. Explica por qué esto hace que escoger dos primos muy próximos p y q no es buena idea en RSA.

Ejercicio 4.09.– Factoriza 23360947609 usando la idea del ejercicio anterior.

Ejercicio 4.10.– Probar que un elemento no nulo de $\mathbf{F}_p$ tiene, exactamente, cero o dos raíces cuadradas. ¿Cuántas raíces cuadradas tiene un elemento no nulo de $\mathbf{Z}/\mathbf{Z}pq$?

Ejercicio 4.11.– Desarrolla un procedimiento para calcular todas las raíces cuadradas de un elemento de $\mathbf{Z}/\mathbf{Z}pq$, y halla las raíces de 15 módulo 1441499. El cálculo de raíces cuadradas en un cuerpo $\mathbf{F}_p$ se puede hacer en tiempo polinomial con el *algoritmo de Shanks*, implementado en la mayoría de los programas de cálculo simbólico.

Ejercicio 4.12.– Probar que, si podemos hallar en tiempo polinomial raíces cuadradas en $\mathbf{Z}/\mathbf{Z}N$, podemos factorizar $N = pq \in \mathbf{Z}$. Para ello, escojamos un $x \in (\mathbf{Z}/\mathbf{Z}N)^*$, calculemos $z = x^2$ y hallemos $\sqrt{z}$.

Probar que, de entre todos los valores posibles de $\sqrt{z}$, exactamente la mitad nos conducen (en tiempo polinomial) a una factorización de N. Por tanto, estamos ante un algoritmo probabilístico con probabilidad de éxito $1/2$.

Ejercicio 4.13.– Asumiendo que es posible calcular raíces cuadradas en $\mathbf{F}_p$ en tiempo polinomial (ver 4.11.), probar que, si conocemos la factorización de $N = pq$, es posible

hallar raíces cuadradas módulo N en tiempo polinomial (ver 4.11.). ¿Para qué enteros se puede aplicar exactamente el mismo procedimiento?

Ejercicio 4.14.– Una variante del DLP utilizada en criptografía es el protocolo Massey–Omura, que tiene la particularidad de no usar claves públicas. En este protocolo, todos los usuarios acuerdan trabajar en un cuerpo primo $\mathbf{F}_P$. Alice escoge entonces e_A y d_A tales que $e_A \cdot d_A = 1 \mod P-1$.

Si ahora Alice quiere enviar un mensaje m a Bob, calcula $M_1 = m^{e_A}$ y lo envía. Bob, por su parte, halla $M_2 = M_1^{e_B}$ y lo envía a Alice. Intenta imaginar cómo continúa el protocolo, sabiendo que para que Bob recupere m hace falta un envío más.

Ejercicio 4.15.– Estudia la fragilidad del sistema Massey–Omura a ataques *man in the middle*.

Ejercicio 4.16.– Un protocolo de intercambio de claves alternativo al DHEP es el denominado protocolo MQV (Menezes, Qu, Vanstone). Trabajamos en un grupo cíclico $\langle G\rangle$, y Alice y Bob han generado, respectivamente, pares de claves público / privadas:

$$(K_A = G^a, a) \qquad \left(K_B = G^b, b\right),$$

y ahora quieren acordar una clave privada común (por ejemplo, para usar DES). Primero genera, cada uno, un nuevo par de claves público / privadas efímeras:

$$(K_C = G^c, c) \text{ (Alice)} \qquad \left(K_D = G^d, d\right) \text{ (Bob)},$$

y se intercambian K_C y K_D. Éste es el único intercambio de mensajes del protocolo (comparar con DHEP).

Supongamos que $|G| = 2^t$, y tomamos $l = t/2$. En parámetros de seguridad actuales, se puede considerar $t = 160$ y, por tanto, $l = 80$.

Alice calcula entonces:

- Convierte K_C y K_D en enteros. Esto casi siempre es inmediato, pues se toma $G = \mathbf{F}_P^*$, pero asumiremos, en general, que se hace de una manera conocida y previamente fijada.
- Calcula
$$S_A = \left(K_C \mod 2^l\right) + 2^l.$$
- Calcula
$$T_A = \left(K_D \mod 2^l\right) + 2^l.$$
- Halla $h_A = c + S_A \cdot a$.
- Halla $k = \left(K_D K_B^{T_A}\right)^{h_A}$.

Razona qué calculos debe hacer Bob para llegar, con los datos de los que dispone, a k.

Ejercicio 4.17.– Prueba que tanto Alice como Bob pueden realizar los cálculos del MQV en tiempo polinomial, mientras que Eve necesita resolver el DLP.

Ejercicio 4.18.– Explica por qué no es una buena idea que Alice escoja un texto fijo como firma.

Ejercicio 4.19.– Demuestra que, en el sistema DSS, $\left[G_A^{U_1} Y_A^{U_2} \mod P_A \right] = R \mod Q_A$.

Ejercicio 4.20.– Un sistema de firma digital debido a ElGamal que usa también el DLP funciona como sigue: se fijan un primo P y un $G \in \mathbf{F}_P^*$. Alice escoge $0 < x_A < P-1$ y publica $Y_A = G^{x_A}$.

Para firmar un mensaje m (del que supondremos $0 < m < P-1$) Alice escoge un entero aleatorio k tal que $\gcd(k, P-1) = 1$, calcula

$$R = G^k \mod P, \qquad S = (m - x_A R)k^{-1} \mod P-1$$

y envía (R, S). Probar que Bob puede verificar la autenticidad del mensaje comprobando

$$G^m = Y_A^R R^S \mod P.$$