La Teoría de Códigos: una introducción a las Matemáticas de la transmisión de información

ADOLFO QUIRÓS GRACIÁN

Departamento de Matemáticas
Universidad Autónoma de Madrid, 28049 Madrid
adolfo.quiros@uam.es

Resumen. Presentamos, con distinto grado de detalle, tres maneras en las que las Matemáticas ayudan a la transmisión segura y fiable de la información. Los códigos detectores y correctores de errores se tratan con cierta amplitud, incluyendo varios ejemplos. En cuanto a la criptografía, se explican las principales diferencias entre la clásica y la de clave pública. Se mencionan brevemente los códigos para la compresión de datos. Terminamos el artículo con una bibliografía comentada.

1. ¿Por qué y para qué se codifica la información?

El problema que nos interesa es el de cómo poder transmitir información de manera *segura y fiable* a través de un canal que sea *poco seguro y poco fiable* (con ruido). Como ejemplos que todos conocemos podemos pensar en:

Ejemplo 1: Conversaciones telefónicas convencionales, donde el canal es el cable.

Ejemplo 2: Transmisiones desde un satélite o telefonía móvil. Aquí el canal es el espacio.

Ejemplo 3: Grabación y recuperación de datos en el disco duro de un ordenador o en un disco compacto (de música por ejemplo). El canal a través del que transmitimos los datos es en este caso el propio disco.

¿Qué quiere decir que un canal es *poco seguro*? Quiere decir que pueden enterarse de lo que dice un mensaje, o alterarlo, terceras personas distintas del emisor o de aquella a quien el mensaje estaba dirigido.

¿Qué quiere decir que el canal es *poco fiable*? Quiere decir que en el canal hay "ruido", de manera que hay una cierta probabilidad de que el mensaje llegue alterado. Por *ruido* entendemos no sólo lo que nos impide oír con claridad cuando se establece una conexión telefónica defectuosa, sino cualquier otra cosa que dificulte la correcta recepción del mensaje, como puede ser la existencia de huellas de dedos en un disco compacto.

Los códigos que atacan los problemas de seguridad se llaman *Códigos Secretos o Criptográficos* y los que intentan resolver las dificultades planteadas por la falta de fiabilidad del canal son los *Códigos Detectores y Correctores de Errores*. Ambos tipos de códigos se pueden combinar, y en la práctica se combinan. Sin embargo nosotros los trataremos de manera independiente.

Históricamente la utilidad principal de los códigos criptográficos se encontraba en las actividades militares y de espionaje, pero hoy en día tienen otras muchas aplicaciones entre las que podemos citar:

- Transmisión de datos confidenciales a través de líneas telefónicas, digitales o de satélites. Por ejemplo, Internet o las transferencias electrónicas de fondos que realizan los bancos.

- Almacenamiento de dichos datos en grandes sistemas de información. Piénsese en los historiales médicos, que puede ser muy útil tener almacenados en ordenadores o en tarjetas inteligentes personales, pero que no es deseable que estén al alcance de cualquiera.

- Autentificación del usuario autorizado a recibir las emisiones de un canal de TV de pago. Esto es lo que hacen los descodificadores de las televisiones digitales.

- Todos los problemas de "identificación electrónica": "password" para acceder a ordenadores o a cajeros automáticos; firmas digitales para garantizar los pagos electrónicos; autentificación de mensajes enviados por correo electrónico.

- Autentificación no sólo de personas, sino de datos, lo que por ejemplo dificultaría la introducción de virus en los ordenadores.

En lo que se refiere a los códigos correctores, tienen entre otras las siguientes aplicaciones:

- Recuperación correcta de los datos grabados en un disco duro o en un disco compacto. Esto es lo que hace que un CD suene bien si esta sucio o incluso arañado.

- Recepción correcta de mensajes digitales enviados a través de medios muy ruidosos. Un ejemplo impresionante es la recepción de las fotos de Júpiter o Saturno enviadas por las sondas espaciales.

- Detección y/o corrección de los errores que se producen al transmitir o almacenar datos de identificación. Esto es lo que hacen el N.I.F. (y es su ventaja con respecto al D.N.I.), el I.S.B.N. o los números de las tarjetas de crédito.

- Detección y/o corrección de los errores que se producen en la lectura automática de datos, como hace un scanner al leer el código de barras de un producto en la caja de un supermercado

Por supuesto en la vida real el uso de estos códigos requiere no sólo el estudio de sus aspectos matemáticos, sino también, y de manera esencial, la fabricación de instrumentos que puedan aplicarlos de manera rápida y eficaz (es decir, barata). No sólo en este aspecto más tecnológico de la teoría de códigos, sino también en la parte

puramente matemática, juegan un papel fundamental los problemas de rapidez y coste.

Vamos a intentar representar gráficamente ambos tipos de códigos. El canal de transmisión (ruidoso e inseguro) aparecerá como $\Rightarrow$ y denotaremos por m el mensaje que deseamos transmitir. En principio tenemos el siguiente esquema sencillo de transmisión:

$$\begin{array}{ccc} \text{EMISOR} & \Rightarrow & \text{RECEPTOR} \\ m & \rightarrow & m \end{array}$$

Las operaciones de cifrar/descifrar (para códigos criptográficos) y de codificar/descodificar (en el caso de códigos detectores y correctores de errores) se representarán como flechas. Esto no es casual: intenta recordarnos la notación para funciones. De hecho cifrar y codificar no son otra cosa que aplicar funciones (es decir, reglas de transformación) adecuadas a los mensajes.

Para hacer esto es conveniente que el emisor transcriba los mensajes del lenguaje natural en el que se comunique con el receptor (castellano, inglés, jerga matemática,...) a un "alfabeto" adecuado para la aplicación de funciones para la transmisión a través del canal del que se trate. En general se sabe como definir muchas funciones de manera sencilla sobre los objetos "matemáticos" (números, vectores, matrices, etc.), por lo que resulta útil escribir en estos términos los mensajes. Por supuesto, si la transmisión se realiza a través de un canal digital, uno tiene que acabar transmitiendo los mensajes como sucesiones de ceros y unos.

2. Los Códigos Detectores y Correctores de Errores

¿Cómo representar un código que sirva para detectar o corregir errores? Intentemosló:

$$\begin{array}{ccccccccc} & & & & & & \text{ruido } (=r) & & \\ & & & & & & \downarrow & & \\ \text{EMISOR} & \rightarrow & \text{CODIFICADOR} & \Rightarrow & \text{DESCODIFICADOR} & \rightarrow & \text{RECEPTOR} \\ m & \rightarrow & u & \rightarrow & v=u+r & \rightarrow & m \end{array}$$

La idea es que, antes de enviarlo, el emisor codifica su mensaje m como u. Esto debe hacerlo añadiendo a m información redundante, de manera que si en el canal de transmisión se produce ruido r y el receptor en vez de u recibe un mensaje alterado $v=u+r$ sea, a pesar de todo, capaz de recuperar el mensaje original m.

Empecemos por un par de ejemplos sencillos de códigos detectores/correctores de errores.

Supongamos que tenemos sólo dos mensajes posibles, *SI* y *NO*, que podríamos codificar y enviar como *SI*=1, *NO*=0. Suponemos que la probabilidad, p, de alteración

de un bit (dígito binario, un 0 o un 1) en el canal es pequeña. Pero siempre que $p>0$ puede ocurrir:

$$\begin{array}{ccc} & \textit{ruido} & \\ & \downarrow & \\ 1 & \Rightarrow & 0 \end{array}$$

de manera que el receptor entienda *NO* cuando le queríamos decir que *SI*. (Es fácil imaginar consecuencias desastrosas de esta confusión.)

Para evitarlo codificamos los mensajes doblándolos: *SI*=11, *NO*=00. De esta manera si, como consecuencia del ruido, se produce un error, digamos que enviamos 11 y se recibe 01, el receptor detecta inmediatamente el error porque el mensaje 01 no forma parte de nuestro código, y puede pedir que volvamos a enviar el mensaje. Obsérvese que este código no permite corregir el error, dado que al recibir 01 es igual de probable que el mensaje original sea 11 o que sea 00.

Sí que podemos corregir un error si el código triplica el mensaje: *SI*=111, *NO*=000, ya que entonces si recibimos, por ejemplo, 101, es más probable que el mensaje original fuese 111 que que fuese 000. Utilizando por tanto la *descodificación al vecino más próximo* (que es la de *máxima verosimilitud*) podemos suponer que el mensaje enviado era *SI*. Es interesante observar que, si renunciamos a corregir errores y nos limitamos a detectarlos, este código detecta no sólo uno, sino incluso dos errores que puedan producirse al transmitir tres bits (lo cual pasa sólo en canales bastante ruidosos).

Los dos códigos que hemos empleado se llaman *códigos de repetición*, y está claro que repitiendo más y más veces el mensaje podemos detectar y corregir cada vez más errores. Pero eso es extremadamente costoso. El objetivo fundamental de la Teoría de Códigos Detectores y Correctores de Errores es conseguir el mismo efecto con códigos más cortos o, dicho de otra manera, más baratos.

3. Algunos Códigos Detectores

Un ejemplo interesante son los números que forman los códigos de barras. Su versión más común en Europa, el EAN (*Europen Article Number*), es un número de trece cifras,

$$a_0a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}$$

de las que las siete primeras, a_0 ... a_6, constituyen el *código del fabricante* (e incluyen alguna información sobre el lugar de origen del código, que no siempre coincide con el del producto), las cinco siguientes, a_7 ... a_{11}, son el *código del producto*, y la última, a_{12}, es un *dígito de control* que se añade de manera que el número

$$3\,(a_1+a_3+a_5+a_7+a_9+a_{11}) + (a_0+a_2+a_4+a_6+a_8+a_{10}+a_{12})$$

sea divisible entre 10 o, en el lenguaje de las *congruencias módulo* 10,

$$3\,(a_1+a_3+a_5+a_7+a_9+a_{11}) + (a_0+a_2+a_4+a_6+a_8+a_{10}+a_{12}) \equiv 0 \quad mod\ 10.$$

Se puede comprobar que la fórmula funciona en el EAN de una lata de una conocida marca de cerveza, 8-411327-122016, donde el 6 es el dígito de control.

Aunque pueda parecer sorprendente, añadir un sólo dígito de control a los doce que dan la información sobre el producto permite detectar todos los errores en una cifra o recuperar una cifra que se haya borrado (pero de la que sepamos el lugar que ocupaba): "equivóquese" el lector al escribir uno de los números del EAN de la lata de cerveza y verá que no se satisface la fórmula que caracteriza a un EAN correcto; "borre" una de las cifras y compruebe que puede recuperarla si exige que la fórmula funcione (aunque debe prestar atención a si la cifra que se ha borrado ocupa un lugar par o impar, porque la manera de recuperarlas es distinta). También permite corregir casi todos los errores de un tipo muy frecuente: que se intercambie el orden de dos cifras consecutivas. Si en el EAN de nuestra lata se intercambian la segunda y tercera cifras, de manera que se obtiene 8-141327-122016, ya no se satisface la fórmula que caracteriza a un EAN correcto.

Pero hemos dicho "casi todos" los errores de este tipo. El EAN no detecta el intercambio de dos cifras consecutivas cuya diferencia sea 5: si en la lata de cerveza se intercambian la sexta y la séptima cifras, que son un 2 y un 7, el número resultante, 8-411372-122016, parece un EAN legítimo.

El motivo fundamental por el que se produce esta excepción es que el cálculo del EAN se realiza operando módulo 10, y 10 no es un número primo. Es más fácil detectar y corregir errores trabajando módulo un primo[1], y esto es lo que hace por ejemplo el ISBN (*International Standard Book Number*) que identifica a la mayoría de los libros modernos. Se trata de un número de diez cifras,

$$a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}$$

de las que las nueve primeras dan información sobre el libro (editorial, título, edición,...) y la última, a_{10}, es de nuevo un dígito de control, que se calcula exigiendo que

$$a_1+2a_2+3a_3+4a_4+5a_5+6a_6+7a_7+8a_8+9a_9+10a_{10}$$

sea divisible entre 11. En términos de congruencias, a_{10} se obtiene mediante la fórmula

$$a_{10} \equiv a_1+2a_2+3a_3+4a_4+5a_5+6a_6+7a_7+8a_8+9a_9 \quad mod\ 11.$$

[1] La explicación se puede resumir en que, si trabajamos módulo un primo, estamos en un cuerpo, y es más sencillo hacer Algebra Lineal sobre un cuerpo que sobre un anillo. La situación es todavía más delicado si el anillo tiene divisores de cero, como sucede al trabajar con un módulo no primo.

Con un poco de práctica en la aritmética modular es fácil comprobar que el ISBN detecta todos los errores en una cifra, puede recuperar un número borrado y detecta todos los intercambios de dos cifras consecutivas.

El ISBN tiene sin embargo una pequeña dificultad técnica: uno de los posibles restos al dividir entre 11 es 10, y por tanto 10 es uno de los valores que puede tomar a_{10}, que sin embargo debe ser una sola cifra. Esto se resuelve admitiendo que a_{10} pueda valer X (10 en números romanos), como sucede en el ISBN 84-239-5921-X, correspondiente a un tomo de una popular enciclopedia (dado que el ISBN debe identificar totalmente al libro, cada uno de los tomos tiene un ISBN distinto). Pero si se insiste en utilizar sólo las 10 cifras de nuestro tradicional sistema de numeración no se podrá trabajar con un código basado en la aritmética módulo 11.

Utilizar letras no pareció importar a quienes diseñaron el NIF (*Número de Identificación Fiscal*), y la letra que se añade al número del DNI para formar el NIF no es otra cosa que el resto al dividir el número del DNI entre 23. Este resto está codificado como una letra de una manera más o menos aleatoria (0=T, 1=R, 2=W, 3=A, ... , 22=E), que el lector no tendrá dificultad en descubrir si tiene acceso a suficientes NIFs (se puede hacer en una clase con ayuda de los alumnos). El NIF es un código con buenas propiedades de detección de errores, que no contiene ninguna información adicional a la del DNI.

4. Algunos Códigos Correctores

Hasta ahora sólo hemos dado ejemplos de códigos detectores de errores. En realidad, la forma en que el EAN se escribe empleando un código binario (1 = barras negras, 0 = espacios en blanco) para formar el código de barras que lee la caja del supermercado le da también la capacidad de corregir errores. Pero en la práctica esta capacidad de corrección no se aprovecha, puesto que es preferible utilizar toda la potencia del código en detectar el mayor número posible de errores, aunque no se puedan corregir. Esto no es un problema dado que, una vez que un pitido nos avisa de un error, no es demasiado costoso volver a pasar el producto por el lector.

Bien distinta es la situación cuando se trata de escuchar la música grabada en un disco compacto: ¡sería inaceptable que cada vez que el lector detectase un error en el CD pitase! En este caso es esencial poder corregir los errores en una sucesión de ceros y unos que, al fin y al cabo, es como está escrita la música en el CD. No explicaremos como lo hace exactamente un lector de discos compactos, sino que daremos un ejemplo más sencillo (y admitimos que algo ficticio) que esperamos que sirva como ilustración.

Volvamos a una situación tratada anteriormente: debemos transmitir las contestaciones a una serie de preguntas para las que hay dos posibles respuestas, *SI*=1 y *NO*=0. En lugar de transmitir las respuestas de una en una las agrupamos de cuatro en cuatro, de manera que los mensajes a transmitir son bloques de cuatro ceros y unos. Por ejemplo, 1101 representa el grupo de cuatro respuestas *SI, SI, NO, SI*.

Antes de transmitir el mensaje completamos estos cuatro bits, $x_1x_2x_3x_4$, con tres bits de control, $x_5x_6x_7$, que elegimos de manera que los tres números $x_1+x_2+x_3+x_5$, $x_1+x_2+x_4+x_6$ y $x_1+x_3+x_4+x_7$ sean pares. Podemos escribir esto en forma modular como

$$\begin{array}{rrrrrrrrrr} x_1 & +x_2 & +x_3 & & +x_5 & & & \equiv 0 & mod\ 2 \\ x_1 & +x_2 & & +x_4 & & +x_6 & & \equiv 0 & mod\ 2 \\ x_1 & & +x_3 & +x_4 & & & +x_7 & \equiv 0 & mod\ 2 \end{array}$$

o dicho de manera equivalente

$$\begin{array}{rcrrrrr} x_5 & \equiv & x_1 & +x_2 & +x_3 & & mod\ 2 \\ x_6 & \equiv & x_1 & +x_2 & & +x_4 & mod\ 2 \\ x_7 & \equiv & x_1 & & +x_3 & +x_4 & mod\ 2 \end{array}$$

de modo que nuestro mensaje 1101 se transmitirá como 1101010.

Este código es capaz no sólo de detectar, sino también de corregir un error en cada mensaje, es decir, en cada grupo de 7 bits. Lo ideó R. W. Hamming en 1948, con él se inició la Teoría de los Códigos Detectores y Correctores de Errores, y es sólo uno de entre la familia de *Códigos de Hamming*.

La gran ventaja del código de Hamming sobre los de repetición es su economía. Para conseguir corregir errores mediante repetición hay que triplicar cada bit, y por tanto sólo 1/3 de los bits enviados son información, mientras que 2/3 son bits de control. Sin embargo el código de Hamming permite corregir errores enviando 4/7 de bits de información y sólo 3/7 de control.

En realidad, para ser totalmente honrados hay que tomar en consideración que el código de Hamming envía los bits de información de cuatro en cuatro. Supongamos a título de ejemplo que estamos transmitiendo por un canal bastante ruidoso, en el que hay una probabilidad del 5% de que un bit se altere. Si no codificamos en absoluto, la probabilidad de leer sin errores un bloque de 4 bits de información es de aproximadamente el 81,5%. Triplicando cada bit dicha probabilidad sube enormemente, a poco más del 97%. Por su parte el código de Hamming permite leer correctamente un 95,5% de los bloques de 4 bits de información recibidos. Como se ve, el código de Hamming es ligeramente menos fiable que triplicar cada bit, ¡pero a cambio permite transmitir casi al doble de velocidad! (7 bits en lugar de 12 para enviar 4 bits de información).

Terminaremos esta sección señalando que el código de Hamming se puede describir matricialmente como el conjunto de los vectores $(x_1,x_2,x_3,x_4,x_5,x_6,x_7)$ que satisfacen la ecuación

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} mod 2.$$

La matriz que aparece en esta ecuación se llama *matriz comprobadora de paridad*. Los códigos que se pueden expresar mediante ecuaciones matriciales de este estilo se llaman *códigos lineales*. Otro ejemplo son los llamados *Códigos de Reed-Solomon*, de los que dos de ellos, definidos sobre un cuerpo finito con 256 elementos, y entrelazados, son los que permiten que un CD suene bien. Podemos decir que en un disco compacto un 75% es música y un 25% son Matemáticas[2].

5. Códigos Criptográficos

Obsérvese que, dado que el interés de los códigos correctores de errores está en poder corregir errores provocados por el medio y no por algún enemigo mal intencionado, los métodos que utilicemos para codificar y descodificar pueden ser públicos y no hay necesidad de cambiarlos si los que estamos utilizando funcionan adecuadamente.

Muy distinta es la situación cuando pensamos que alguien que no nos quiere bien puede haber interceptado el canal con el objetivo de enterarse de un mensaje sin autorización, o de alterar el mensaje con intenciones perversas. Veamos el esquema:

[2]Y eso tomando en consideración sólo los bits grabados en el CD. En otros aspectos de la tecnología del CD las Matemáticas desempeñan también un papel esencial, aunque no sería justo despreciar, por ejemplo, el de la electrónica.

$$
\begin{array}{ccccccc}
 & & & & \text{INTERCEPTOR} & & \\
 & & & & \downarrow & & \\
\text{EMISOR} & \rightarrow & \text{CIFRADOR} & \Rightarrow & \text{DESCIFRADOR} & \rightarrow & \text{RECEPTOR} \\
m & \rightarrow & E_k(m) & \rightarrow & D_k(E_k(m)) & \rightarrow & m
\end{array}
$$

Observemos que aquí lo que envía el Cifrador y lo que recibe el Descifrador es lo mismo, el mensaje no se altera al pasar por el Canal. Sin embargo lo que enviamos no es m, sino una versión cifrada[3] $E_k(m)$. La idea es que la única forma de recuperar m a partir del conocimiento de $E_k(m)$ sea aplicando la correspondiente función para descifrar, D_k, y que esto no sepa como hacerlo el Interceptor. El subíndice k representa una clave que en principio sólo pueden conocer el Emisor y el Receptor, y que es la que imposibilita el desciframiento no autorizado.

Un ejemplo muy antiguo de código criptográfico es el que utilizaba Julio Cesar[4]. Consistía en sustituir cada letra por la que iba tres lugares detrás de ella en el alfabeto: la A se sustituía por la D, la B por la E, etc. Si suponemos para simplificar que su alfabeto coincidiese con el nuestro (la CH y la LL las vemos cada una como dos letras) y Cesar quisiese firmar con su primer nombre, CAYO, escribiría FDBR. Sin querer menospreciar a Julio Cesar, este es un ejemplo de método criptográfico muy poco seguro.

Desde el siglo I a.C. la criptografía avanzó muchísimo, y alcanzó una de sus cimas con las máquinas Enigma que utilizaban los ejércitos alemanes durante la II Guerra Mundial. Pero tanto Enigma como el criptosistema de Julio Cesar comparten con todos los demás métodos criptográficos utilizados por la humanidad hasta los años 70 del siglo XX dos características:

1) ¿Cómo pueden ponerse de acuerdo el emisor y el receptor en una clave secreta? Evidentemente no pueden hacerlo a través del canal de comunicación, ¡porque entonces también la conocería el Interceptor! Es necesario que tengan un segundo canal, este seguro, a través del que transmitir la clave. El canal seguro tradicional era el contacto personal con anterioridad a la transmisión de mensajes, pero esto no es muy útil para una red grande de comunicaciones.

2) Si se conoce la clave utilizada para cifrar es fácil encontrar la clave para descifrar. Esto fuerza a mantener ambas claves secretas y plantea al menos dos problemas:

- cada pareja de corresponsales debe tener su propia clave para cifrar, lo que supone que una red de N personas necesite $N(N-1)/2$ claves distintas;

- si un nuevo miembro desea incorporarse a la red debe previamente acordar claves con cada uno de los N miembros anteriores.

[3] Es cada vez más frecuente el uso del anglicismo *encriptar*. Pero nosotros preferimos el clásico y muy matemático *cifrar*, que tiene su origen en la palabra árabe para el cero.
[4] c. f. Suetonio, *La vida de los Césares. Cayo Julio Cesar, LVI.*

Ninguno de los dos problemas es serio si N es pequeño, pero de nuevo suponen una dificultad para una red grande de comunicaciones.

La *Criptografía Clásica* era por tanto poco adecuada para la transmisión segura de mensajes entre, pongamos, todas las sucursales de todos los bancos del mundo.

En 1976 Diffie, Hellman y Merkle propusieron un sistema de *Criptografía de Clave Pública* que resolvía todos estos inconvenientes. Esta basado en la existencia de ciertas funciones, llamadas *funciones de un solo sentido*, con la propiedad de que, incluso conociendo la función f, es *imposible en la práctica* encontrar la correspondiente función inversa f^{-1}.

Se pueden utilizar estas funciones para proteger las comunicaciones de una red como sigue. La usuaria A calcula una función de un solo sentido f_A, que se utilizará para cifrar, con su correspondiente inversa f_A^{-1}, que servirá para descifrar, y lo mismo hacen todos los demás. Las funciones para cifrar f_A, f_B, f_C, etc. se publican (de ahí el nombre clave pública) en una guía, similar a una guía de teléfonos, pero cada usuario mantiene secreta su función para descifrar. Si la usuaria B quiere enviar un *mensaje* a A no tiene más que mirar en la guía la clave pública de A y enviar $f_A(mensaje)$. Ahora A, ¡y sólo ella!, conoce la clave para descifrar f_A^{-1} y puede recuperar el *mensaje*. Hay que insistir en que C no puede hacer esto porque no se puede encontrar f_A^{-1} a partir del conocimiento de f_A.

Obsérvese que ya no hay que intercambiar claves, puesto que éstas se publican en una guía; además cada usuario necesita una sola clave, que utilizarán todos los demás usuarios para comunicarse con él; y para unirse a la red basta con entrar en contacto con el administrador que edita la guía. El método permite incluso diseñar un sistema de *firmas digitales* para autentificar los mensajes.

La primera propuesta operativa, y la más empleada, de un criptosistema de clave pública la hicieron en 1978 Rivest, Shamir y Adleman, y en su honor se llama *Criptosistema RSA*. Está basado en que es muy fácil encontrar primos grandes, y sin embargo es muy difícil factorizar un número del que se sepa que es compuesto. En el momento actual, es esencialmente imposible factorizar un número de 400 cifras del que se sabe que es producto de dos primos de unas 200 cifras cada uno. Por el contrario, el mayor primo encontrado hasta ahora (el $2^{13.466.917}$-1, encontrado el 14 de noviembre de 2001) tiene 4.053.946 cifras, se conocen varios miles de *primos gigantes* (con más de 10.000 cifras), y resulta rutinario encontrar *primos titánicos* (con más de 1.000 cifras). Esta diferencia en la dificultad es lo que utiliza el criptosistema RSA para construir funciones de un solo sentido.

6. Cómo hacer más cortos los mensajes

Además de la transmisión segura y fiable de la información la Teoría de Códigos tiene un tercer aspecto, la llamada *Compresión de Datos*. Esta estudia como codificar los mensajes de la manera más corta posible, eliminando información que sea redundante, de manera que transmitirlos sea poco costoso (en tiempo o dinero, que

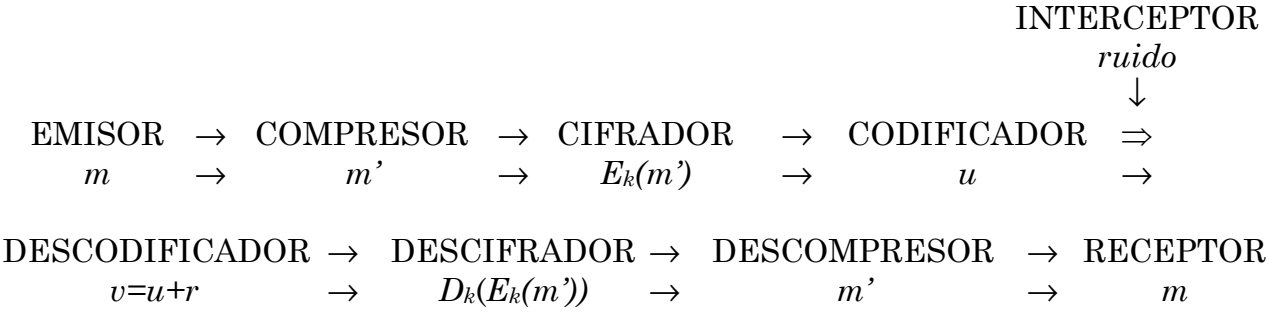
aquí acaba siendo más o menos lo mismo). No vamos a tratar este tema con amplitud, y nos limitaremos a dar un ejemplo.

Con frecuencia se codifican las 27 letras del alfabeto castellano A,...,Z (con Ñ y W, pero la CH=C+H y la LL=L+L) como los números del 0 al 26. Este método no tiene en cuenta que unas letras se utilizan más que otras en castellano (y en cualquier idioma, aunque no sean siempre las mismas las más utilizadas). El alfabeto Morse, que es una manera de codificar mediante puntos y rayas, sí tiene esto en cuenta, y reserva los códigos cortos para las letras más utilizadas (por ejemplo, la E=• y la T=-) mientras que emplea códigos más largos para las letras no muy comunes (Q=--•-). De esta manera los mensajes transmitidos resultan más cortos.

La manera óptima de comprimir datos es utilizar un *Código de Huffman*. Pero hay otros códigos compresores que, sin ser óptimos, son también muy eficaces, y presentan la ventaja añadida de ser más sencillos de poner en práctica. Un ejemplo son los códigos *ZIP* que se emplean para comprimir documentos electrónicos.

7. ¿Se puede hacer todo a la vez?

Podemos unir las tres componentes de la Teoría de Códigos, compresión de datos, protección de la información y corrección de errores producidos por ruido en el canal, en un sólo diagrama que represente el proceso completo:



Conviene aclarar que, aunque parezca absurdo primero eliminar redundancias al comprimir los datos y luego añadirla al utilizar un código corrector, esto no supone una contradicción: la redundancia que quitamos al comprimir corresponde a información inútil, mientras la que añade el código corrector está pensada para ser útil.

Bibliografía Comentada

Las referencias que damos son sólo una selección de entre los numerosos libros y artículos dedicados a diversos aspectos de la Teoría de Códigos, y en particular no reflejan el notable incremento del número de publicaciones sobre criptografía (por desgracia no tanto sobre códigos detectores y correctores de errores) aparecidas en España. Confiamos en que, por ser estos títulos sin duda más accesibles, el lector no tenga dificultades para localizarlos por otros medios.

Dos auténticas joyas de la divulgación matemática

[1] COMAP, Las Matemáticas en la vida cotidiana. Addison-Wesley/UAM (1999). Los capítulos 9 y 10 de este fascinante libro tratan distintos tipos de códigos. Los restantes 20 capítulos no desmerecen.

[2] I. Stewart, De aquí al infinito. Crítica (1998). Un capítulo trata de primos y criptografía, y otro de "fácil y difícil". Todo el libro es una delicia.

Divulgación en castellano

[3] M. C. Espinel - P. Caballero, *La matemática que protege de errores a los números de identificación*. Suma 20 (1995) 77-84. Explica varios ejemplos de códigos detectores de errores y concluye con algunas reflexiones para el aula.

[4] M. Gardner, *Juegos Matemáticos: Claves de nuevo tipo cuyo desciframiento ocuparía unos cuantos millones de años*. Investigación y Ciencia (octubre 1977) 96-101. La versión española del artículo donde se presentó en sociedad RSA.

[5] G. Lachaud - S. Vladut, *Los códigos correctores de errores*. Mundo Científico 161 (octubre 1995) 864-868. El título se explica sólo.

[6] S. Singh, Los Códigos Secretos. Debate (2000). Muy amena historia de la criptografía, desde los egipcios hasta nuestros días, con breves apuntes matemáticos.

[7] I. Stewart. *Juegos Matemáticos: Recibo de compra en Internet*. Investigación y Ciencia (abril 1996) 87-89. Una interesante aplicación de la criptografía de clave pública.

[8] I. Stewart. *Juegos Matemáticos: Caza mayor en territorio primo*. Investigación y Ciencia (julio 1997) 87-89. ¿Qué hay que hacer para encontrar primos grandes?

[9] I. Stewart. *Juegos Matemáticos: Cribas en la tierra de los factores*. Investigación y Ciencia (agosto 1997) 88-90. Presenta dos de los más modernos métodos de factorización: la criba cuadrática y la criba en cuerpos de números.

Divulgación en inglés

[10] M. E. Hellman, *The Mathematics of public-key Cryptography*. Scientific American (August 1979) 130-139. La criatura presentada por uno de sus progenitores.

[11] J. Gallian - S. Winters, *Modular Arithmetics in the Market Place*. The American Mathematical Monthly 95 (1988) 548-551. Explica por qué la aritmética modular es Matemática Aplicada.

[12] J. H. van Lint, *Matematics and the Compact Disc*. Nieuw Archief voor Wiskunde 16, no. 3 (november 1998) 183-190. Da el nivel justo de detalle para entender como funciona un CD.

Algunos libros de un nivel razonablemente asequible

[13] R. Hill, A First Course in Coding Theory, Oxford University Press (1986). Explica las bases de la teoría de códigos detectores y correctores sin suponer más conocimientos que el álgebra lineal y las congruencias.

[14] N. Koblitz, A course in Number Theory and Criptography, 2nd ed., Springer-Verlag (1994). Claro y completo, pero requiere una cierta madurez.

[15] J. Pastor - M. A. Sarasa, Criptografía Digital. Prensas Universitarias de Zaragoza (1998). Dirigido más bien a ingenieros. Pero es muy completo (y está en castellano).

[16] P. Ribenboim, The Little Book of Big Primes, Springer-Verlag (1991). Es la versión para no especialistas del "Book of Prime Number Records", en el que Ribenboim explica "Todo lo que usted siempre quiso saber....". No es estrictamente un libro sobre criptografía, pero vale la pena.

[17] S. Roman, Coding and Information Theory, Springer-Verlag (1992). La parte de "Teoría de la Información" no aparece en otros libros.

[18] J.H. van Lint, Introduction to Coding Theory, Springer-Verlag (1982). Más completo que el de Hill, pero también más difícil.

[19] J.H. van Lint - G. van der Geer, Introduction to Coding Theory and Algebraic Geometry, Birkhäuser (1988). Este libro requiere bastante más preparación que los otros, dado que la Geometría Algebraica es más difícil que el Algebra Lineal. Seguramente está fuera de lugar en esta bibliografía, pero no nos hemos podido resistir a incluir al menos una referencia en la que se traten los códigos geométricos.

En Internet

[20] http://pass.maths.org.uk/issue3/codes. *Coding Theory: the first 50 years*. Un artículo de R. Pinch en *Plus Magazine* que explica, incluyendo fotografías, como se usan los códigos correctores en los viajes espaciales.

[21] http://www.spatula.net/proc/barcode/index.src. Información sobre distintos tipos de códigos de barras (EAN, Postnet, Código 3 de 9,...). Ofrece la posibilidad de generar las correspondientes imágenes.

[22] http://www.kriptopolis.com. Revista de criptografía en castellano. Tiene acceso a libros y documentación gratuita.

[23] http://www.utm.edu/research/primes. *The prime number page*. Aquí se puede encontrar información actualizada casi a diario, y conexiones a otras páginas interesantes.

[24] http://entropia.com/ips. *Internet Prime Net Server*. Para quienes quieran colaborar en la búsqueda de primos enormes.

[25] http://www.rsa.com. La compañía de los inventores de RSA. Información sobre sus productos y sobre el "RSA Challenge".