(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 563 290**

(51) Int. CI.:

H04B 14/04 (2006.01)
H03M 13/37 (2006.01)
H04L 1/00 (2006.01)
H03M 13/29 (2006.01)
H03M 13/11 (2006.01)
H03M 13/19 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **16.02.2007 E 07757111 (5)**

(97) Fecha y número de publicación de la concesión europea: **09.12.2015 EP 1980041**

(54) Título: **Generador de códigos y descodificador basados en campos múltiples para sistemas de comunicaciones**

(30) Prioridad:

21.02.2006 US 775528 P
13.02.2007 US 674655

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
14.03.2016

(73) Titular/es:

DIGITAL FOUNTAIN, INC. (100.0%)
5775 Morehouse Drive
San Diego, CA 92121, US

(72) Inventor/es:

SHOKROLLAHI, M. AMIN;
LUBY, MICHAEL G.;
WATSON, MARK y
MINDER, LORENZ

(74) Agente/Representante:

FORTEA LAGUNA, Juan José

DESCRIPCIÓN

Generador de códigos y descodificador basados en campos múltiples para sistemas de comunicaciones

CAMPO DE LA INVENCIÓN

La presente invención se refiere a la codificación y descodificación de datos en sistemas de comunicaciones y, más específicamente, a sistemas de comunicaciones que codifican y descodifican datos para tener en cuenta errores y brechas en los datos comunicados. La comunicación se usa en un sentido amplio, e incluye, pero no se limita a, la transmisión de datos digitales en cualquier forma a través del espacio y / o el tiempo.

ANTECEDENTES DE LA INVENCIÓN

La transmisión de ficheros y flujos entre un remitente y un destinatario, por un canal de comunicaciones, ha sido el objeto de mucha bibliografía. Preferiblemente, un destinatario desea recibir una copia exacta de datos transmitidos por un canal por un remitente, con algún nivel de certidumbre. Allí donde el canal no tiene fidelidad perfecta (lo que abarca prácticamente todos los sistemas realizables), una preocupación es cómo tratar los datos perdidos o distorsionados en la transmisión. Los datos perdidos (borraduras) son a menudo más fáciles de tratar que los datos corrompidos (errores), porque el destinatario no siempre puede decir cuándo los datos corrompidos son datos recibidos con errores. Muchos códigos correctores de errores han sido desarrollados para corregir borraduras y / o errores. Habitualmente, el código específico se escoge en base a alguna información acerca de las infidelidades del canal a través del cual los datos están siendo transmitidos y a la naturaleza de los datos que están siendo transmitidos. Por ejemplo, allí donde se sabe que el canal tiene largos periodos de infidelidad, un código de error por ráfagas podría ser el mejor adaptado para esa aplicación. Allí donde solamente se esperan errores breves e infrecuentes, un sencillo código de paridad podría ser lo óptimo.

La transmisión de datos es directa cuando un transmisor y un receptor tienen toda la potencia de cálculo y la energía eléctrica necesarias para las comunicaciones y el canal entre el transmisor y el receptor está suficientemente limpio para permitir comunicaciones relativamente libres de errores. El problema de la transmisión de datos se torna más difícil cuando el canal está en un entorno adverso o el transmisor y / o el receptor tienen capacidad limitada.

Una solución es el uso de técnicas de corrección anticipada de errores (FEC), en las que los datos son codificados en el transmisor de modo que un receptor pueda recuperarse de borraduras y errores de transmisión. Allí donde sea factible, un canal inverso desde el receptor al transmisor permite que el receptor comunique errores al transmisor, que puede luego ajustar su proceso de transmisión en consecuencia. Sin embargo, a menudo un canal inverso no está disponible o no es factible, o está disponible solamente con capacidad limitada. Por ejemplo, allí donde el transmisor está transmitiendo a un gran número de receptores, el transmisor podría no ser capaz de gestionar canales inversos desde todos esos receptores. Como otro ejemplo, el canal de comunicación puede ser un medio de almacenamiento y por tanto la transmisión de los datos es directa a lo largo del tiempo y, a menos que alguien invente una máquina de viaje por el tiempo que pueda retroceder en el tiempo, un canal inverso para este canal no es factible. Como resultado, los protocolos de comunicación a menudo necesitan ser diseñados sin un canal inverso, o con un canal inverso de capacidad limitada y, de tal modo, el transmisor puede tener que tratar con condiciones de canal de amplia variación, sin una visión completa de esas condiciones de canal.

El problema de la transmisión de datos entre transmisores y receptores se dificulta más cuando los receptores necesitan ser dispositivos pequeños, de baja potencia, que podrían ser portátiles o móviles, y necesitan recibir datos en altos anchos de banda. Por ejemplo, una red inalámbrica podría configurarse para entregar ficheros o flujos desde un transmisor estático a un número grande, o indeterminado, de receptores portátiles o móviles, ya sea como una difusión o multi-difusión, donde los receptores están restringidos en su potencia de cálculo, tamaño de memoria, potencia eléctrica disponible, tamaño de antena, tamaño de dispositivo y otras restricciones de diseño. Otro ejemplo está en aplicaciones de almacenamiento donde el receptor extrae datos desde un medio de almacenamiento que exhibe infidelidades en la reproducción de los datos originales. Tales receptores están a menudo incrustados, con el mismo medio de almacenamiento, en dispositivos, por ejemplo, controladores de disco, que están sumamente restringidos en términos de potencia de cálculo y energía eléctrica.

En un sistema de ese tipo, las consideraciones a abordar incluyen tener un canal inverso escaso o nulo, memoria limitada, ciclos de cálculo, energía, movilidad y temporización limitados. Preferiblemente, el diseño debería minimizar la cantidad de tiempo de transmisión necesaria para entregar datos a una población potencialmente grande de receptores, donde los receptores individuales podrían ser encendidos y apagados en momentos impredecibles, entrar y salir del alcance, incurrir en pérdidas debidas a errores de enlace, movilidad, congestión que fuerza a que paquetes de ficheros o flujos de baja prioridad sean temporalmente descartados, etc.

En el caso de un protocolo de paquetes usado para el transporte de datos por un canal que puede perder paquetes, un fichero, flujo u otro bloque de datos a transmitir por una red de paquetes se divide en símbolos de entrada de

igual tamaño, los símbolos de codificación del mismo tamaño que los símbolos de entrada son generados a partir de los símbolos de entrada usando un código de FEC, y los símbolos de codificación son colocados y enviados en paquetes. El "tamaño" de un símbolo puede ser medido en bits, ya sea que el símbolo esté o no efectivamente descompuesto en un flujo de bits, donde un símbolo tiene un tamaño de M bits cuando el símbolo es seleccionado entre un alfabeto de 2^M símbolos. En un sistema de comunicación basado en paquetes de ese tipo, podría ser adecuado un esquema de codificación de FEC de borrado, orientado a paquetes. Una transmisión de ficheros se llama fiable si permite que el pretendido destinatario recupere una copia exacta del fichero original, incluso ante borraduras en la red. Una transmisión de flujo se llama fiable si permite que el pretendido destinatario recupere una copia exacta de cada parte del flujo de manera oportuna, incluso ante borraduras en la red. Tanto la transmisión de ficheros como la transmisión de flujos pueden ser también algo fiables, en el sentido de que algunas partes del fichero o del flujo no sean recuperables, o transmisibles por flujo, si algunas partes del flujo no son recuperables oportunamente. La pérdida de paquetes ocurre a menudo porque la congestión esporádica provoca que el mecanismo de almacenamiento temporal en un encaminador colme su capacidad, forzándolo a descartar paquetes entrantes. La protección contra borraduras durante el transporte ha sido el objeto de mucho estudio.

En el caso de un protocolo usado para la transmisión de datos por un canal ruidoso que puede corromper bits, un bloque de datos a transmitir por un canal de transmisión de datos es dividido en símbolos de entrada de igual tamaño, los símbolos de codificación del mismo tamaño son generados a partir de los símbolos de entrada y los símbolos de codificación son enviados por el canal. Para un canal ruidoso de ese tipo, el tamaño de un símbolo es habitualmente de un bit o de unos pocos bits, ya sea que un símbolo sea o no efectivamente descompuesto en un flujo de bits. En un sistema de comunicación de ese tipo, podría ser adecuado un esquema de codificación de FEC de corrección de errores orientado a flujos de bits. Una transmisión de datos se llama fiable si permite que el pretendido destinatario recupere una copa exacta del bloque original incluso ante errores (corrupción de símbolos, ya sean detectados o no detectados en el canal). La transmisión también puede ser algo fiable, en el sentido de que algunas partes pueden permanecer corrompidas después de la recuperación. Los símbolos son corrompidos a menudo por ruido esporádico, ruido periódico, interferencia, señal débil, bloqueos en el canal, y una amplia variedad de otras causas. La protección contra la corrupción de datos durante el transporte ha sido objeto de mucho estudio.

Los códigos de reacción en cadena son códigos de FEC que permiten la generación de un número arbitrario de símbolos de salida a partir de los símbolos de entrada fijos de un fichero o flujo. A veces, se mencionan como códigos de FEC de fuente, o sin tasa, dado que el código no tiene una tasa de transmisión fija a priori. Los códigos de reacción en cadena tienen muchos usos, incluyendo la generación de un número arbitrario de símbolos de salida de una manera aditiva para la información, a diferencia de una manera duplicativa para la información, en donde esta última se da allí donde los símbolos de salida recibidos por un receptor antes de poder recuperar los símbolos de entrada duplican la información ya recibida y por tanto no proporcionan información útil para recuperar los símbolos de entrada. Se muestran técnicas novedosas para generar, usar y operar códigos de reacción en cadena, por ejemplo, en Luby I, Luby II, Shokrollahi I y Shokrollahi II.

Una propiedad de los símbolos de salida producidos por un codificador de reacción en cadena es que un receptor es capaz de recuperar el fichero o bloque original del flujo original en cuanto hayan sido recibidos suficientes símbolos de salida. Específicamente, para recuperar los K símbolos originales de entrada con alta probabilidad, el receptor necesita aproximadamente K+A símbolos de salida. La razón A / K se llama el "sobregasto de recepción relativa". El sobregasto de recepción relativa depende del número K de símbolos de entrada, y de la fiabilidad del descodificador.

También se sabe cómo usar códigos de reacción en cadena de múltiples etapas ("MSCR"), tales como los descritos en Shokrollahi I y / o II, y los desarrollados por Digital Fountain, Inc., con el nombre comercial de códigos "Raptor". Los códigos de reacción en cadena de múltiples etapas se usan, por ejemplo, en un codificador que recibe símbolos de entrada desde un fichero de origen o un flujo de origen, genera símbolos intermedios a partir de los símbolos de entrada y codifica los símbolos intermedios usando códigos de reacción en cadena. Más específicamente, se genera una pluralidad de símbolos redundantes a partir de un conjunto ordenado de símbolos de entrada. Se genera una pluralidad de símbolos de salida a partir de un conjunto combinado de símbolos que incluyen los símbolos de entrada y los símbolos redundantes, en donde el número de posibles símbolos de salida es mucho mayor que el número de símbolos en el conjunto combinado de símbolos, en donde al menos un símbolo de salida se genera a partir de más de un símbolo en el conjunto combinado de símbolos y a partir de menos que todos los símbolos en el conjunto combinado de símbolos, y de modo que el conjunto ordenado de símbolos de entrada pueda ser regenerado, con un grado deseado de precisión, a partir de cualquier número predeterminado, *N*, de los símbolos de salida. También se conoce cómo usar las técnicas descritas anteriormente para codificar y descodificar códigos sistemáticos, en los cuales los símbolos de entrada están incluidos entre los posibles símbolos de salida del código. Esto puede lograrse según lo descrito en Shokrollahi II, aplicando primero una transformación a los símbolos de entrada, seguida por las etapas descritas anteriormente, dando como resultado dicho proceso mejorado que los primeros símbolos de salida generados por el código sean iguales a los símbolos de entrada. Como estará claro para los expertos en la técnica de codificación de errores y borraduras, las técnicas de Shokrollahi II pueden ser aplicadas directamente a los códigos descritos o sugeridos en la presente memoria.

Para algunas aplicaciones, otras variaciones de códigos podrían ser más adecuadas, o preferidas de otro modo.

Los códigos de MSCR y los códigos de reacción en cadena descritos anteriormente son extremadamente eficaces en términos de su complejidad de codificación y descodificación. Uno de los motivos para su eficacia es que las operaciones que se realizan son operaciones lineales sobre el campo GF (2), es decir, el campo sencillo sobre un bit donde la operación de sumar dos elementos de campo es sencillamente la operación lógica XOR, y la operación de multiplicar dos elementos de campo es sencillamente la operación lógica AND. En general, estas operaciones son realizadas sobre múltiples bits simultáneamente, p. ej., 32 bits a la vez o 4 octetos a la vez, y tales operaciones disponen de soporte nativo en todos los modernos procesadores de CPU. Por otra parte, cuando se usan como códigos de FEC de borraduras, debido a que las operaciones son sobre GF(2), resulta que la probabilidad de que el receptor pueda descodificar todos los símbolos de entrada desciende, a lo sumo, en aproximadamente la mitad para cada símbolo adicional recibido más allá de los primeros K, donde K es el número de símbolos de entrada originales. Por ejemplo, si se reciben K+A símbolos de codificación, entonces la probabilidad de que el proceso de recuperación no logre recuperar los K símbolos de entrada originales es al menos de 2^{-A}. Lo que sería un comportamiento más deseable es que la probabilidad del fallo de descodificación disminuyera más rápidamente como función de A.

Hay otros códigos de FEC de borradura y corrección de errores que funcionan sobre campos más grandes, por ejemplo, los códigos de Reed-Solomon, que funcionan sobre GF(4), o sobre GF(8), o sobre GF(256) o, más generalmente, sobre GF(2^L) para cualquier L > 1, y también los códigos LDPC que funcionan sobre campos mayores. La ventaja de tales códigos de FEC es que, por ejemplo, en el caso de códigos de FEC de borradura, la probabilidad del fallo de descodificación disminuye mucho más rápidamente como función de A que los códigos de FEC sobre GF(2). Por otra parte, estos códigos de FEC son habitualmente mucho menos eficaces en términos de complejidad de codificación y descodificación, y uno de los motivos principales para eso es porque las operaciones sobre campos más grandes son mucho más complejas y / o no disponen de soporte nativo en las CPU modernas, y la complejidad crece habitualmente según crece el tamaño del campo. Por tanto, los códigos de FEC que funcionan sobre campos finitos más grandes son a menudo mucho más lentos o imprácticos en comparación con los códigos de FEC que funcionan sobre GF(2).

Por tanto, lo que se necesita son códigos de FEC de borradura y corrección de errores que sean extremadamente eficaces en términos de su complejidad de codificación y descodificación, con la propiedad de que la probabilidad de fallo de descodificación disminuya muy rápidamente como función del número de símbolos recibidos, más allá del número mínimo que necesita un código de FEC ideal para recuperar los símbolos de entrada originales.

Se reclama atención nuevamente al documento US 2005 / 219070 A1 (citado en la página 1) que describe un codificador que usa sub-símbolos de símbolos de entrada para efectuar o controlar un equilibrio entre el esfuerzo de cálculo y la eficacia de sobregasto, por ejemplo, para reducir en gran medida el esfuerzo de cálculo para el coste de una pequeña magnitud de eficacia de sobregasto. Un codificador lee una pluralidad ordenada de símbolos de entrada, que comprenden un fichero de entrada o un flujo de entrada, y produce sub-símbolos de salida. La pluralidad ordenada de símbolos de entrada son seleccionados, en cada caso, a partir de un alfabeto de entrada, y los sub-símbolos de salida generados comprenden selecciones entre un alfabeto de sub-símbolos de salida. Un sub-símbolo de salida se genera usando un evaluador de funciones aplicado a sub-símbolos de los símbolos de entrada. El codificador puede ser invocado una o más veces, produciendo cada vez un sub-símbolo de salida. Los sub-símbolos de salida pueden luego ser ensamblados en símbolos de salida y transmitidos a su destino. Las funciones usadas para generar los sub-símbolos de salida a partir de los sub-símbolos de entrada pueden ser las operaciones lógicas XOR de algunos de los sub-símbolos de entrada, y estas funciones se obtienen a partir de un código lineal definido por un campo de extensión de GF(2), transformando cada entrada, en un generador o matriz de control de paridad de este código, en una matriz binaria adecuada, usando una representación regular del campo de extensión sobre GF(2). En un descodificador, los sub-símbolos de salida recibidos por el destinatario son obtenidos a partir de símbolos de salida transmitidos desde un remitente que generó esos símbolos de salida en base a una codificación de una secuencia de entrada (fichero, flujo, etc.).

BREVE SUMARIO DE LA INVENCIÓN

De acuerdo a la presente invención, se proporcionan un procedimiento de codificación y un procedimiento de descodificación, según lo enunciado, respectivamente, en las reivindicaciones independientes. Las realizaciones preferidas de la invención están descritas en las reivindicaciones dependientes.

De acuerdo a una realización de la invención, se proporciona un procedimiento de codificación de datos para transmisiones desde un origen a un destino, por un canal de comunicaciones. El procedimiento funciona sobre un conjunto ordenado de símbolos de entrada y puede generar cero o más símbolos redundantes a partir de los símbolos de entrada, siendo cada símbolo redundante igual a una combinación lineal de un cierto número de los símbolos de entrada, con coeficientes tomados de uno o más campos finitos, en donde el campo finito usado puede diferir, entre distintos símbolos de entrada y entre distintos símbolos redundantes. El procedimiento incluye la generación de una pluralidad de símbolos de salida a partir del conjunto combinado de símbolos que incluye los

símbolos de entrada y los símbolos redundantes, si hay algún símbolo redundante, en donde cada símbolo de salida puede ser generado a partir de uno o más de los símbolos combinados de entrada y redundantes, en donde cada símbolo de salida es generado como una combinación lineal de un cierto número de los símbolos de entrada y redundantes, con coeficientes tomados de uno o más campos finitos, en donde el campo finito usado puede diferir, entre distintos símbolos de entrada y redundantes, entre distintos símbolos de salida y entre los símbolos de salida y los símbolos redundantes, y de modo que el conjunto ordenado de símbolos de entrada pueda ser regenerado, en un grado deseado de precisión, a partir de cualquier número predeterminado de los símbolos de salida.

Los procedimientos también pueden ser usados para generar símbolos de salida, en donde el número de posibles símbolos de salida que pueden ser generados a partir de un conjunto fijo de símbolos de entrada puede ser mucho mayor que el número de símbolos de entrada.

De acuerdo a un ejemplo, el procedimiento incluye recibir en un destino al menos algunos de los símbolos de salida enviados desde un origen por un canal de comunicaciones, donde la transmisión por el canal puede dar como resultado la pérdida o corrupción de algunos de los símbolos enviados, y donde puede saberse que algunos de los símbolos recibidos sean correctamente recibidos, y también puede proporcionarse información acerca del grado de corrupción de los símbolos. El procedimiento incluye regenerar en el destino el conjunto ordenado de símbolos de entrada, con un grado deseado de precisión, que depende de cuántos símbolos son recibidos y del conocimiento de la corrupción de los símbolos recibidos.

Este ejemplo también puede incluir recibir en un destino al menos algunos de los símbolos de salida, en donde el número de posibles símbolos de salida que pueden ser recibidos puede ser mucho mayor que el número de símbolos de entrada.

De acuerdo a otro ejemplo, se proporciona un procedimiento de codificación de datos para su transmisión desde un origen a un destino, por un canal de comunicaciones. El procedimiento funciona sobre un conjunto ordenado de símbolos de entrada e incluye la generación de una pluralidad de símbolos redundantes a partir de los símbolos de entrada. El procedimiento también incluye generar una pluralidad de símbolos de salida a partir de un conjunto combinado de símbolos que incluyen los símbolos de entrada y los símbolos redundantes, en donde la operación aplicada en la generación de símbolos de salida es sobre un pequeño campo finito (por ejemplo, GF(2)), y tal que el conjunto ordenado de símbolos de entrada pueda ser regenerado, con un grado deseado de precisión, a partir de cualquier número predeterminado de los símbolos de salida. La pluralidad de símbolos redundantes se genera a partir del conjunto ordenado de símbolos de entrada, en donde las operaciones para generar los símbolos redundantes son sobre un campo finito que no es GF(2) (por ejemplo, GF(256)), o es sobre una mezcla de más de un campo finito (por ejemplo, algunas operaciones sobre GF(2), algunas operaciones sobre GF(256)).

De acuerdo a otro ejemplo más, se proporciona un sistema para recibir datos transmitidos desde un origen por un canal de comunicaciones, usando técnicas similares. El sistema comprende un módulo de recepción acoplado con un canal de comunicaciones, para recibir símbolos de salida transmitidos por el canal de comunicaciones, en donde cada símbolo de salida es generado a partir de al menos un símbolo en el conjunto combinado de símbolos que incluyen los símbolos de entrada y los símbolos redundantes, en donde la operación aplicada en la generación de símbolos de salida es sobre un pequeño campo finito (por ejemplo, GF(2)), y tal que el conjunto ordenado de símbolos de entrada pueda ser regenerado, con un grado deseado de precisión, a partir de cualquier número predeterminado de los símbolos de salida, en donde los símbolos de entrada provienen de un conjunto ordenado de símbolos de entrada, en donde los símbolos redundantes son generados a partir de entrada y en donde la pluralidad de símbolos redundantes es generada a partir del conjunto ordenado de símbolos de entrada, en donde las operaciones para generar los símbolos redundantes son sobre un campo finito que no es GF(2) (por ejemplo, GF(256)), o es sobre una mezcla de más de un campo finito (por ejemplo, algunas operaciones sobre GF(2), algunas operaciones sobre GF(256)).

De acuerdo a otro ejemplo más, se proporciona una señal de datos de ordenador realizada en una onda portadora.

Se logran numerosos beneficios por medio de la presente invención. Por ejemplo, en una realización específica, el gasto de cálculo para codificar datos para su transmisión por un canal está reducido. En otra realización específica, el gasto de cálculo para descodificar tales datos está reducido. En otra realización específica más, la probabilidad de error del descodificador está reducida, manteniendo a la vez bajo el gasto de cálculo de la codificación y la descodificación. Según la realización, pueden lograrse uno o más de estos beneficios. Estos y otros beneficios se proporcionan en más detalle en toda la extensión de la presente especificación, y más específicamente a continuación.

Una comprensión adicional de la naturaleza y las ventajas de las invenciones divulgadas en la presente memoria puede ser realizada con referencia a las partes restantes de la especificación y a los dibujos adjuntos.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La Fig. 1 es un diagrama de bloques de un sistema de comunicación de acuerdo a una realización de la presente invención.

La Fig. 2 es un diagrama de bloques de un codificador de acuerdo a una realización de la presente invención.

La Fig. 3 es un diagrama de bloques simplificado de un procedimiento de generación de símbolos redundantes, de acuerdo a una realización de la presente invención.

La Fig. 4 es un diagrama de bloques simplificado del funcionamiento básico de un codificador estático, de acuerdo a una realización de la presente invención.

La Fig. 5 es un diagrama de bloques simplificado de un codificador dinámico, de acuerdo a una realización de la presente invención.

La Fig. 6 es un diagrama de bloques simplificado de una operación básica de un codificador dinámico, de acuerdo a una realización de la presente invención.

La Fig. 7 es un diagrama de bloques simplificado de un codificador estático, de acuerdo a una realización de la presente invención.

La Fig. 8 es un diagrama de bloques simplificado del funcionamiento básico de un codificador estático, de acuerdo a una realización de la presente invención.

La Fig. 9 es un diagrama simplificado de un procedimiento para calcular parámetros de codificación, de acuerdo a una realización específica de un codificador estático.

La Fig. 10 es un diagrama de flujo simplificado de un codificador estático, de acuerdo a otra realización de la presente invención.

La Fig. 11 es un diagrama de bloques simplificado de un descodificador, de acuerdo a una realización de la presente invención.

La Fig. 12 es un diagrama de flujo simplificado de una operación de un descodificador, de acuerdo a una realización de la presente invención.

La Fig. 13 es un diagrama de flujo simplificado de una operación de un descodificador, de acuerdo a otra realización de la presente invención.

La Fig. 14 es un diagrama de flujo simplificado de una operación de un descodificador, de acuerdo a otra realización más de la presente invención.

La Fig. 15 es un diagrama de bloques simplificado de un descodificador dinámico, de acuerdo a una realización de la presente invención.

La Fig. 16 es un diagrama de bloques simplificado de un descodificador estático, de acuerdo a una realización de la presente invención.

La Fig. 17 ilustra símbolos de origen a partir de correlaciones de sub-símbolos.

La Fig. 18 ilustra posibles configuraciones de parámetros de descarga de ficheros, para diversos tamaños de ficheros.

La Fig. 19 ilustra posibles configuraciones de parámetros de transmisión por flujo, para diversos tamaños de bloques de origen.

La Fig. 20 ilustra una forma de una matriz que representa una relación entre símbolos de origen e intermedios.

La Fig. 21 ilustra una distribución de grados para el generador de grados.

La Fig. 22 ilustra una forma de la matriz **A** que puede ser usada para la descodificación.

La Fig. 23 ilustra una descomposición en bloques de la matriz **A** que puede ser usada para la descodificación.

La Fig. 24a ilustra una descomposición en bloques de la matriz **X** que puede ser usada para la descodificación

La Fig. 24b ilustra una descomposición en bloques de la matriz **X** después de varias etapas de la primera fase del proceso de descodificación.

La Fig. 25 ilustra una descomposición en bloques de la matriz **X** después de algunas etapas de eliminación.

La Fig. 26 ilustra una descomposición en bloques de una sub-matriz de **X** después de etapas adicionales de eliminación.

La Fig. 27 ilustra una descomposición en bloques de la matriz **A** después de etapas de eliminación y borrado.

La Fig. 28 ilustra una descomposición en bloques de la matriz **A** después de etapas adicionales de eliminación y borrado.

La Fig. 29 ilustra una descomposición en bloques de la matriz **A** después de etapas adicionales de eliminación.

La Fig. 30 ilustra una descomposición en bloques de la matriz **A** después de más etapas adicionales de eliminación.

La Fig. 31 muestra una tabla de probabilidades de fallo de código para un código (120, 100) construido de acuerdo a una realización preferida de la invención.

La Fig. 32 muestra una tabla de probabilidades de fallo de código para un código (1.110, 100) construido de acuerdo a una realización preferida de la invención.

La descripción detallada es seguida por tres apéndices: el Apéndice A contiene valores ejemplares para los índices sistemáticos *J(K)*; el Apéndice B.1 contiene valores ejemplares para la tabla V_0; y el Apéndice B.2 contiene valores ejemplares para la tabla V_1.

DESCRIPCIÓN DETALLADA DE REALIZACIONES ESPECÍFICAS

Las invenciones descritas en la presente memoria hacen uso de operaciones matemáticas para la codificación y la descodificación, en base a operaciones en uno o más campos finitos. Los campos finitos son estructuras algebraicas finitas para las cuales están definidas las cuatro operaciones aritméticas, y que forman un campo con respecto a estas operaciones. Su teoría y su construcción son bien entendidas por los expertos en la técnica.

En la descripción que sigue requeriremos que esté definido un proceso de multiplicación entre los elementos de un campo finito y los símbolos que representan, o se obtienen de, los datos a codificar o descodificar. Se consideran tres tipos distintos de símbolos en esta descripción: los símbolos de entrada comprenden información conocida para el remitente, que ha de ser comunicada al receptor, los símbolos redundantes comprenden símbolos que se obtienen de los símbolos de entrada y los símbolos de salida comprenden símbolos que son transmitidos por el remitente al receptor. De las muchas posibilidades para definir un proceso de multiplicación de ese tipo, nos concentramos en dos específicos: transformaciones simples y transformaciones entrelazadas.

Transformaciones simples

En este caso, el proceso de multiplicación está definido entre un elemento *a* de un campo finito GF(2^M) y un símbolo *S* que tiene *M* bits de longitud. Según se usa en la presente memoria, “símbolo” se refiere a un trozo de datos que es habitualmente más pequeño que el bloque de origen. El tamaño de un símbolo puede medirse a menudo en bits, donde un símbolo tiene el tamaño de *M* bits y el símbolo se selecciona entre un alfabeto de 2^M símbolos. En aplicaciones de transmisión fiable de información por redes de paquetes, por ejemplo, el tamaño de un símbolo podría ser igual al tamaño del paquete, o podría ser más pequeño, de modo que cada paquete contenga uno o más símbolos.

En el caso de una transformación simple, el símbolo *S* es interpretado como un elemento de GF(2^M), y la multiplicación *a*S* se define como la multiplicación normal en el campo GF(2^M). La operación realizada sobre el símbolo se llama una “transformación simple” del símbolo. Como ejemplo ilustrativo, considérese el campo GF(4). Los elementos de GF(4) pueden ser representados con 2 bits, por ejemplo, de acuerdo a su expansión binaria. El campo GF(4) tiene cuatro elementos de campo 00, 01, 10, 11, donde la suma es el ‘o’ lógico exclusivo normal de cadenas de bits, y la multiplicación se define mediante la tabla:

	00	01	10	11
00	00	00	00	00
01	00	10	10	11
10	00	10	11	10
11	00	11	10	10

De acuerdo a la tabla de multiplicación anterior, el resultado de 10*01 sería 10, dado que 01 es el elemento multiplicativo neutro (a veces llamado el elemento de identidad) del campo.

Transformaciones entrelazadas

Para ilustrar las transformaciones entrelazadas, haremos uso del concepto matemático de un anillo. Como es bien sabido por los medianamente expertos en la técnica, un anillo es un conjunto en el cual están definidas dos operaciones, la suma y la multiplicación, de modo que estas operaciones satisfagan las leyes distributivas. Además, el conjunto considerado con la suma sola forma un grupo abeliano, es decir, el resultado de una suma es independiente del ordenamiento de los sumandos, hay un elemento neutro 0 para la suma y para cada elemento hay otro elemento tal que la suma de estos elementos es 0. El otro requisito es que la multiplicación tenga un elemento neutro 1, de modo que la multiplicación de cualquier elemento por 1 no cambie el valor de ese elemento. Para un anillo general, no requerimos que ningún elemento no nulo tenga una inversa multiplicativa, ni requerimos que la multiplicación sea conmutativa. Cuando ambas condiciones se satisfacen, sin embargo, llamamos al anillo un "campo". Esta notación es estándar en el área del álgebra.

Una correlación (suma símbolo a símbolo) es una estructura lógica implementable en hardware, software, almacén de datos, etc., que correlaciona pares de símbolos del mismo tamaño con otro símbolo de ese tamaño. Indicamos esta correlación con $\oplus$, y la imagen de esta correlación del par (S, T) de símbolos con $S \oplus T$. Un ejemplo de una correlación de ese tipo es el 'o' lógico exclusivo bit a bit (XOR).

Otra estructura usada aquí es la de la "acción" de un tipo especial de conjuntos sobre símbolos. Supongamos que A es un conjunto equipado con una operación de suma conmutativa "+" que tiene un elemento neutro y que, para cada elemento, contiene su inversa aditiva. Un conjunto de ese tipo también se llama usualmente un grupo abeliano. Una "acción" de este grupo sobre el conjunto de símbolos es una correlación que correlaciona un par, que comprende un elemento r de grupo y un símbolo S, con otro símbolo. Indicamos la imagen con $r*S$, donde esta correlación respeta la suma en el grupo, es decir, para cada par de elementos a y b en el grupo A, $(a+b)*S = a*S \oplus b*S$. Si A es un anillo y la acción también respeta la multiplicación en A, donde el operador de multiplicación en A es *, es decir, $(a*b)*S = a*(b*S)$, entonces esta acción es el proceso de multiplicación deseado entre elementos de un campo finito y símbolos. En este entorno, decimos que el campo "opera" sobre el conjunto de símbolos. La operación realizada sobre símbolos de esta forma se llama una "transformación entrelazada".

Hay abundantes ejemplos de tales procesos de multiplicación. Se mencionan más adelante unos pocos ejemplos. Esta lista de ejemplos está concebida solamente con fines ilustrativos, y no debería ser considerada como una lista exhaustiva, ni debería ser interpretada para limitar el ámbito de esta invención.

El campo GF(2) con elementos de campo 0 y 1, siendo la suma el 'o' lógico exclusivo (XOR) y siendo la multiplicación la operación lógica AND, opera sobre el conjunto de símbolos definiendo $1 * S = S$ y $0 * S = 0$, en donde S indica un símbolo arbitrario y **0** indica el símbolo que tiene todos ceros.

El campo GF(4) puede operar sobre símbolos de tamaño par de la siguiente manera: para un símbolo S de ese tipo indicamos con $S[0]$ y $S[1]$, respectivamente, su mitad primera y segunda, de modo que S = ($S[0]$, $S[1]$) sea la concatenación de $S[0]$ y $S[1]$. Luego, definimos

$$00 * S = \mathbf{0}$$

$$01 * S = S$$

$$10 * S = (S[1], S[0] \oplus S[1])$$

$$11 * S = (S[0] \oplus S[1], S[0]).$$

Puede verificarse rápidamente que esta es en efecto una operación válida. Puede verse que la tabla de multiplicación del campo describe una operación que coincide con la operación definida anteriormente en el caso de símbolos de 2 bits.

Alternativamente, el campo GF(4) puede operar sobre símbolos de tamaño par de la siguiente manera: para un símbolo *S* de ese tipo indicamos con *S*[0] la concatenación de los bits en posiciones pares dentro de *S* y, de manera similar, indicamos con *S*[1] la concatenación de los bits en posiciones impares dentro de *S* (donde las posiciones se numeran secuencialmente partiendo de cero). Para dos cadenas de bits de igual longitud *A* y *B*, sea (*A* | *B*) definida como la cadena de bits *C* del doble de longitud, donde el bit en la posición 2**i* de *C* es el bit en la posición *i* de *A* y el bit en la posición 2**i*+1 de *C* es el bit en la posición *i*+1 de *B*. Luego, definimos

$$00 * S = \mathbf{0}$$

$$01 * S = S$$

$$10 * S = (S[1] \mid S[0] \oplus S[1])$$

$$11 * S = (S[0] \oplus S[1] \mid S[0]).$$

Puede verificarse rápidamente que esta es en efecto una operación válida. Puede verse que todas las operaciones definidas anteriormente son las mismas en el caso de símbolos de 2 bits.

Las transformaciones entrelazadas descritas anteriormente pueden ser vistas como un caso particular de una transformación entrelazada en la cual la longitud binaria de un elemento del campo coincide con la longitud de los símbolos en bits, y la operación de elementos de campo sobre símbolos es la misma que la multiplicación en el campo finito.

Más en general, si K es un campo de extensión de GF(2) de grado *d*, entonces una operación del campo puede ser definida sobre símbolos cuyo tamaño sea divisible por *d*. Tal operación está descrita en el artículo “Un esquema de codificación resistente a borraduras basado en el XOR”, de Bloemer, Kalfane, Karpinksi, Karp, Luby y Zuckerman, publicado como Informe Técnico Número TR-95-048 del Instituto Internacional de Ciencia de la Computación en Berkeley, 1995. Este esquema usa la llamada “representación regular” del campo *K* como matrices de dimensiones *d x d* con entradas binarias.

Para estas generalizaciones, la primera transformación entrelazada divide *S*, una cadena que tiene *d*l* bits de longitud, en *d* partes de igual tamaño, donde la primera parte *S*[0] es los primeros *l* bits de *S*, S[1] es los siguientes *l* bits de *S* y *S*[*d*-1] es los últimos *l* bits de *S*. La transformación opera sobre las *d* partes de *S* y produce *d* partes que se concatenan entre sí para formar el resultado de la operación. Alternativamente, la segunda transformación entrelazada divide *S* en *d* partes de igual tamaño, donde la primera parte *S*[0] es la concatenación de cada *d*-ésimo bit de *S* a partir de la posición 0 en *S*, la segunda parte *S*[1] es la concatenación de cada *d*-ésimo bit en *S* a partir de la posición 1 en *S*, la *d*-ésima parte *S*[*d*-1] es la concatenación de cada *d*-ésimo bit de *S* a partir de la posición *L*-1 en *S*. Esta segunda transformación opera sobre las *d* partes de *S* (exactamente las mismas que la primera transformación) y produce *d* partes que se entrelazan entre sí para formar el resultado de la operación.

Obsérvese que la primera transformación entrelazada puede ser calculada aplicando la operación lógica XOR entre sí a bits consecutivos de la cadena original S, y este es un beneficio para implementaciones de software, donde habitualmente una CPU presta soporte a tales operaciones en modalidad nativa. Por otra parte, los valores de los bits en posiciones específicas en el resultado de la operación dependen de la longitud de la cadena original S, y esto es cierta desventaja si se quiere implementar la operación en hardware que dé soporte a símbolos de longitud variable, ya que la operación del hardware necesita ser distinta según la longitud del símbolo. Obsérvese que la segunda transformación entrelazada implica aplicar la operación XOR entre sí a bits no consecutivos de la cadena original, y esto es cierta desventaja para implementaciones de software donde habitualmente una CPU no da soporte a tales operaciones XOR como una operación nativa. No obstante, las operaciones de software que trabajan sobre los elementos de campo finito del símbolo directamente pueden ser implementadas bastante eficazmente en software y por tanto son posibles las implementaciones de software de la segunda transformación entrelazada. Además, para la segunda transformación entrelazada los valores de los bits en posiciones específicas en el resultado de la operación no dependen de la longitud de la cadena original S, y esto es un beneficio si se quiere implementar la operación en hardware que presta soporte a símbolos de longitud variable, ya que la operación del hardware puede ser independiente de la longitud del símbolo. Por tanto, la segunda transformación entrelazada sí tiene algunas ventajas globales sobre la primera transformación entrelazada.

Transformaciones lineales

El concepto de una “transformación lineal” puede ser definido con referencia a las transformaciones simples o entrelazadas. Para enteros *m* y *n* dados, una transformación lineal inducida por la operación correlaciona vectores de *n* símbolos con vectores de *m* símbolos usando el espacio de matrices con entradas en el campo especificado. Una matriz sobre el campo *F* es una colección bidimensional de entradas, cuyas entradas pertenecen a *F*. Si una matriz tiene *m* filas y *n* columnas, entonces se menciona usualmente como una matriz de dimensiones *m x n*. El par

(*m, n*) se llama el “formato” de la matriz. Las matrices del mismo formato pueden ser sumadas y restadas, usando la suma y la resta en el campo o anillo subyacente. Una matriz de formato (*m, n*) puede ser multiplicada por una matriz de formato (*n, k*), según se conoce usualmente.

En la operación, si *B* indica una matriz con formato (*m, n*) y B[*j, k*] indica la entrada de *B* en la posición (*j, k*), y si *S* indica el vector columna que comprende los símbolos S_1, S_2, ..., S_n, y *X* indica el vector columna que comprende los símbolos X_1, X_2, ..., X_m, entonces la transformación puede ser expresada como

$$X = B \otimes S.$$

Por tanto, es válida la siguiente relación:

para todo *j* entre 1 y *m*, $X_j = B[j,1]*S_1 \oplus B[j,2]*S_2 \oplus ... \oplus B[j,n]*S_n$

en la que “*” indica bien una transformación simple o bien una entrelazada.

La fórmula anterior describe un proceso para calcular *X* a partir de *B* y *S* en un codificador o descodificador, mencionado como un “proceso de transformación simple” que puede ser efectuado por las etapas de:

1. Fijar *j* en 1, y X_j en **0**.
2. Para valores de *k* entre 1 y *n*, hacer $X_j = X_j \oplus B[j,k]*S_k$.
3. Incrementar *j* en 1. Si *j* es mayor que *m*, entonces parar; en caso contrario, ir a la etapa 2.

Tales transformaciones lineales son de uso corriente en una amplia variedad de aplicaciones. Por ejemplo, al usar un código lineal para codificar un trozo de datos, o bloque de origen, *S* podría ser los símbolos de origen del bloque de origen a codificar, *X* podría ser la versión codificada de *S* y *B* podría ser una matriz generadora para el código. En otras aplicaciones, por ejemplo, donde el código usado es sistemático, *X* podría ser los símbolos redundantes de la codificación de *S*, mientras que *B* podría ser la matriz que describe la dependencia de los símbolos redundantes sobre los símbolos de origen.

Como sabrán los expertos en la técnica, se conocen procedimientos para realizar las operaciones descritas anteriormente, ya sea mediante la provisión de instrucciones ejecutadas con un procesador de propósito general, mediante hardware diseñado específicamente para realizar tales operaciones, o mediante una combinación de ambos. En todos los casos, el coste de las operaciones, en términos del número de instrucciones requeridas, la cantidad de hardware requerida, el coste del hardware, la energía eléctrica consumida por la operación y / o el tiempo requerido para realizar la operación es generalmente mayor cuando se usan campos finitos más grandes. En particular, en el caso del campo GF(2), las operaciones requeridas son equivalentes a operaciones AND y XOR bit a bit, que están ampliamente provistas dentro de procesadores de propósito general, y son sencillas, rápidas y baratas para implementar en hardware allí donde se requiera. Por el contrario, las operaciones que usan campos finitos más grandes que GF(2) raramente son provistas directamente en procesadores de propósito general y requieren bien hardware especializado o bien un cierto número de instrucciones de procesador y operaciones de memoria a realizar.

Códigos de borradura y corrección de errores de múltiples campos

Se describen en la presente memoria numerosas realizaciones específicas de códigos de borradura y corrección de errores de múltiples campos, con referencia a una descripción matricial generalizada. Este enfoque se adopta solamente como una herramienta descriptiva y no representa una forma única de describir las realizaciones descritas en la presente memoria, ni debería ser interpretado para limitar el ámbito de esta invención. En la descripción generalizada, se construye una matriz cuyos elementos se toman de uno o más campos finitos. Distintos elementos pueden ser tomados desde distintos campos finitos, con la propiedad de que haya un único campo en el cual todos los campos puedan ser incrustados, y se escogen incrustaciones específicas de ese tipo. Algunos de, o todos, los símbolos de salida pueden ser idénticos a algunos de los símbolos de entrada o redundantes, o pueden ser distintos a los símbolos de entrada y redundantes, según la realización específica escogida, como se ilustrará más adelante.

Se efectúa una correspondencia biunívoca entre los símbolos de entrada del código y algunas de las columnas de la matriz. Se efectúa una correspondencia biunívoca adicional entre los símbolos redundantes del código y las columnas restantes de la matriz. Además, se designa un cierto número de filas de la matriz, igual al número de símbolos redundantes, como filas estáticas. Las filas restantes de la matriz son designadas como filas dinámicas. Se efectúa una correspondencia biunívoca entre las filas dinámicas de la matriz y los símbolos de salida del código. En esta descripción, las filas estáticas representan restricciones que se requiere mantener entre los símbolos de entrada y los redundantes, y las filas estáticas definen totalmente la relación entre símbolos de entrada y redundantes, de modo que el conocimiento de los símbolos de entrada y las filas estáticas sea suficiente para

construir los símbolos redundantes. Las filas dinámicas representan los símbolos de salida que son efectivamente enviados por el canal. En muchos códigos, los mismos símbolos de entrada y / o redundantes son enviados y esto está representado en esta descripción añadiendo una fila dinámica para cada símbolo de entrada y redundante que ha de transmitirse, teniendo dicha fila dinámica una entrada no nula en la columna correspondiente al símbolo de entrada o redundante requerido, y entradas nulas en las columnas restantes. En algunas realizaciones, la entrada no nula es la identidad. En otras realizaciones, esta entrada no nula no necesariamente debe ser el elemento de identidad.

Una matriz de la forma descrita anteriormente puede ser usada para determinar un procedimiento de codificación de datos para su transmisión desde un origen a un destino por un canal de comunicaciones, comprendiendo el procedimiento generar una pluralidad de símbolos redundantes a partir de un conjunto ordenado de símbolos de entrada, en el que cada símbolo redundante se genera en base a un conjunto de restricciones lineales sobre uno o más de los símbolos de entrada y otros símbolos redundantes, con coeficientes sobre campos finitos, correspondiendo dichas restricciones lineales a las filas estáticas de la descripción matricial; generar una pluralidad de símbolos de salida a partir del conjunto combinado de símbolos de entrada y redundantes, en el que cada símbolo de salida se genera como una combinación lineal de uno o más entre el conjunto combinado de símbolos de entrada y redundantes, con coeficientes escogidos de campos finitos, correspondiendo dichas restricciones lineales a las filas dinámicas de la descripción matricial, y enviar al menos algunos entre la pluralidad de símbolos de salida generados.

Por el contrario, un procedimiento que comprenda las etapas anteriores puede ser descrito en términos de una matriz de la clase descrita anteriormente, en la cual las filas estáticas corresponden a las restricciones lineales sobre uno o más de los símbolos de entrada y los símbolos redundantes, y las filas dinámicas corresponden a las combinaciones lineales de los símbolos de entrada y redundantes que se usan para formar los símbolos de salida. En la práctica, las realizaciones del procedimiento descrito anteriormente pueden no implicar la representación o construcción, explícita o implícita, de la matriz descrita.

Como es bien sabido, en el caso de que todos los elementos de la matriz estén tomados del campo GF(2), entonces una gran clase de códigos bien conocidos de corrección de errores y corrección de borraduras puede ser descrita de esta manera. Por ejemplo, para el caso de códigos de Control de Paridad de Baja Densidad (LDPC), que incluyen, por ejemplo, los descritos en el artículo titulado “Diseño, evaluación y comparación de cuatro códecs de FEC de grandes bloques, LDPC, LDGM, LDGM Escalera y LDGM Triángulo, más un códec de FEC de bloques pequeños de Reed-Solomon”, de V. Roca y C. Neumann, publicado como Informe de Investigación INRIA RR-5225, junio de 2004, disponible en www.inrialpes.fr (mencionado en adelante en la presente memoria como “Roca”), la matriz generalizada puede ser construida a partir de la matriz de control de paridad, designando a cada fila de la matriz de control de paridad como una fila estática y añadiendo una fila dinámica adicional para cada símbolo de entrada y redundante, según lo descrito anteriormente. Otro ejemplo podría usar los códigos de reacción en cadena de etapa única, descritos en Luby I y Luby II, en los cuales el número de filas estáticas en la matriz es cero y las filas dinámicas comprenden una matriz estándar de reacción en cadena. Otro ejemplo es el uso de códigos MSCR, en cuyo caso la descripción generalizada aquí es equivalente a la presentación matricial estándar de tales códigos.

Otros códigos sobre campos más grandes también pueden ser descritos de esta manera. Por ejemplo, los códigos de Reed-Solomon, tales como los obtenidos a partir de matrices de Vandermonde, en las cuales los símbolos de entrada son los símbolos de origen, la matriz generalizada es igual a la matriz de Vandermonde y todas las filas son dinámicas, donde, en este caso, cada entrada es un elemento de campo finito, proveniente de un campo que tiene al menos tantos elementos en su grupo multiplicativo como filas y columnas hay en total, p. ej., el campo finito GF(256) cuando el número de filas y columnas en total es menor que 256. Otro ejemplo son los códigos sistemáticos de Reed-Solomon sobre un campo finito tal como GF(256), que se obtienen de matrices de Vandermonde, en cuyo caso los símbolos de entrada son los símbolos de origen, los símbolos redundantes son los símbolos de paridad, y la matriz es las filas correspondientes a los símbolos de paridad dentro de la forma sistemática de la matriz de Vandermonde, con todas las filas de ese tipo consideradas estáticas, y se añaden filas dinámicas adicionales para cada símbolo de origen y de paridad, según lo descrito anteriormente, ya que estos son exactamente los símbolos enviados por el canal.

Como es bien sabido por los expertos en la técnica de los códigos correctores de errores y borraduras, las propiedades deseables de la corrección de errores y borraduras incluyen la baja complejidad de codificación, la baja complejidad de descodificación, la baja probabilidad de errores de descodificación y el bajo límite inferior de errores. La complejidad de un código es una medida de los recursos de cálculo requeridos para codificar o descodificar el código. La baja complejidad es de especial valor en aplicaciones donde la codificación y la descodificación ha de ser realizada por dispositivos restringidos en cuanto a recursos, tales como los terminales móviles, los dispositivos de electrónica de consumo, los dispositivos de almacenamiento o los dispositivos que pueden procesar muchas operaciones de codificación o descodificación simultáneamente. La complejidad de cálculo es una función, en parte, de la densidad de la matriz usada para codificar y descodificar el código, y del tamaño del campo finito del cual se toman los elementos matriciales. Las matrices densas dan generalmente como resultado una mayor complejidad y

esto ha llevado a muchos diseños de códigos basados en matrices ralas, por ejemplo, los códigos de Control de Paridad de Baja Densidad y los códigos de reacción en cadena. Loa campos finitos más grandes también dan como resultado una mayor complejidad, lo que ha llevado a muchos diseños de código basados en campos pequeños, más usualmente, el GF(2).

La probabilidad de errores en este contexto es la probabilidad de que no sea posible la descodificación completamente exitosa. La probabilidad de errores para un código dado de corrección de errores o borraduras es una función de la información recibida por el canal, y del algoritmo específico usado para la descodificación. En el caso de códigos de corrección de borraduras, la probabilidad de errores es uno toda vez que se reciban menos símbolos que el número de símbolos de entrada. Los códigos ideales de borradura tienen la propiedad de que la probabilidad de errores es cero toda vez el número de símbolos recibidos es mayor o igual que el número de símbolos de entrada. Otros códigos tienen probabilidad no nula de fallo en este caso.

Se sabe que los códigos ideales de borradura pueden ser construidos usando matrices densas, en particular, los códigos de Reed-Solomon. Sin embargo, en el caso de códigos de Reed-Solomon el tamaño del campo requerido es una función del tamaño del código, que es la suma del número de símbolos de entrada y redundantes, y este hecho, junto con la densidad de la matriz, da como resultado una complejidad de cálculo generalmente alta, especialmente según crece el tamaño del código. Además, en el caso de códigos de baja densidad, se sabe que los campos finitos más grandes pueden ser usados para reducir la probabilidad de errores para códigos de corrección de errores (como está demostrado, por ejemplo, en el artículo “Códigos de Control de Paridad de Baja Densidad sobre GF(q)”, de M. C. Davey y D. J. C. MacKay, que ha aparecido en las Cartas de Comunicaciones del IEEE, volumen 2, número 6, páginas 165 a 167, 1998) y para códigos de borradura. Adicionalmente, se sabe que la introducción de un número pequeño de filas o columnas de matrices de alta densidad en un código de baja densidad puede mejorar la probabilidad de errores, proporcionando un compromiso entre la probabilidad de errores y la complejidad [códigos MSCR y códigos de reacción en cadena]. Sin embargo, una desventaja de todos los códigos de ese tipo es que siempre hay un equilibrio significativo entre la baja complejidad y la baja probabilidad de errores.

Para muchos códigos de FEC, es decir, códigos de LDPC y códigos de reacción en cadena y códigos de MSCR, según se reciben más símbolos de salida que el número de símbolos de entrada, disminuye exponencialmente la probabilidad de errores para la descodificación exitosa, en alguna medida. El límite inferior de errores de un código de ese tipo es la probabilidad de errores en la cual la recepción de símbolos de salida adicionales disminuye la probabilidad de errores a una velocidad mucho más lenta que cuando el número de símbolos de salida recibidos supera por primera vez el número de símbolos de entrada. Se sabe que el uso de un número pequeño de filas o columnas de alta densidad y / o el uso de un campo finito más grande para la matriz puede dar como resultado un menor límite inferior de errores, con el coste de una mayor complejidad de cálculo. Una desventaja de muchos códigos conocidos de corrección de errores y borraduras con baja complejidad es que el límite inferior de errores es mayor que lo deseable.

En la presente memoria, se describen procedimientos novedosos para la construcción de códigos de corrección de errores y de corrección de borraduras, que abordan algunas de las desventajas mencionadas anteriormente. Los procedimientos para la codificación y descodificación eficaces de tales códigos se presentan con relación a realizaciones específicas descritas en la presente memoria a modo de ejemplo.

La elección de campos para los elementos matriciales a partir de un conjunto de más de un campo posible, según lo descrito en la presente memoria, permite el diseño de códigos que retienen la baja complejidad de cálculo de los códigos sobre campos pequeños, con la baja probabilidad de errores y el límite inferior de errores de códigos sobre campos más grandes, y por tanto representa una ventaja significativa en el estado actual de la técnica.

En una realización preferida que será descrita en más detalle más adelante, para la mayoría de las filas las entradas se escogen de GF(2), y para el resto de las filas las entradas se escogen de GF(256). En otra realización, para cada fila se escoge exactamente una entrada de GF(256) y los elementos restantes se escogen de GF(2).

Hay otras muchas posibles realizaciones del uso de elementos a partir de más de un campo, que dan como resultado una mejora en el equilibrio entre la complejidad de cálculo y la probabilidad de errores y el límite inferior de errores, en comparación con códigos conocidos en la técnica en los cuales todos los elementos son seleccionados a partir del mismo campo.

Según se usa en la presente memoria, el término “fichero” se refiere a datos cualesquiera que están almacenados en uno o más orígenes, y que han de ser entregados como una unidad a uno o más destinos. Por tanto, un documento, una imagen y un fichero procedentes de un servidor de ficheros o dispositivo de almacenamiento de ordenador, son todos ejemplos de “ficheros” que pueden ser entregados. Los ficheros pueden ser de tamaño conocido (tal como una imagen de un mega-octeto almacenada en un disco rígido) o pueden ser de tamaño desconocido (tal como un fichero tomado de la salida de un origen de flujo transmisor). De cualquier modo, el fichero

es una secuencia de símbolos de entrada, donde cada símbolo de entrada tiene una posición en el fichero y un valor.

Según se usa en la presente memoria, el término “flujo” se refiere a datos cualesquiera que están almacenados o son generados en uno o más orígenes, y que se entregan a una velocidad especificada en cada momento, en el orden en que se generan, a uno o más destinos. Los flujos pueden ser de velocidad fija o de velocidad variable. Por tanto, un flujo de vídeo de MPEG, un flujo de audio de AMR y un flujo de datos usado para controla un dispositivo remoto, son todos ejemplos de “flujos” que pueden ser entregados. La velocidad del flujo en cada momento puede ser conocida (tal como 4 megabits por segundo) o desconocida (tal como un flujo de velocidad variable donde la velocidad en cada momento no se conoce de antemano). En cualquier caso, el flujo es una secuencia de símbolos de entrada, donde cada símbolo de entrada tiene una posición en el flujo y un valor.

La transmisión es el proceso de transmitir datos desde uno o más remitentes a uno o más destinatarios a través de un canal, a fin de entregar un fichero o flujo. Un remitente también es mencionado a veces como el codificador. Si un remitente está conectado con cualquier número de destinatarios por un canal perfecto, los datos recibidos pueden ser una copia exacta del fichero o flujo de entrada, ya que todos los datos serán recibidos correctamente. Aquí suponemos que el canal no es perfecto, que es el caso para la mayoría de los canales del mundo real. De las muchas imperfecciones del canal, dos imperfecciones de interés son la borradura de datos y la incompletitud (que puede tratarse como un caso especial de borradura de datos). La borradura de datos ocurre cuando el canal pierde o descarta datos. La incompletitud de datos ocurre cuando un destinatario no comienza a recibir datos hasta que algunos de los datos ya lo han pasado de largo, el destinatario deja de recibir datos antes de que la transmisión acabe, el destinatario escoge recibir solamente una parte de los datos transmitidos y / o el destinatario, intermitentemente, detiene e inicia de nuevo la recepción de datos. Como ejemplo de incompletitud de datos, un remitente satelital en movimiento podría estar transmitiendo datos que representan un fichero o flujo de entrada, e iniciar la transmisión antes de que un destinatario esté a su alcance. Una vez que el destinatario está a su alcance, los datos pueden ser recibidos hasta que el satélite salga del alcance, punto en el cual el destinatario puede redirigir su antena satelital (tiempo durante el cual no está recibiendo datos) para comenzar a recibir los datos acerca del mismo fichero o flujo de entrada que está siendo transmitido por otro satélite que ha quedado a su alcance. Como debería ser evidente a partir de la lectura de esta descripción, la incompletitud de datos es un caso especial de la borradura de datos, dado que el destinatario puede tratar la incompletitud de datos (y el destinatario tiene los mismos problemas) como si el destinatario estuviera al alcance todo el tiempo, pero el canal perdiera todos los datos hasta el punto donde el destinatario comenzó a recibir datos. Además, como es bien sabido en el diseño de sistemas de comunicación, los errores detectables pueden ser considerados equivalentes a borraduras, descartando sencillamente todos los bloques de datos o símbolos que tienen errores detectables.

En algunos sistemas de comunicación, un destinatario recibe datos generados por múltiples remitentes, o por un remitente que usa múltiples conexiones. Por ejemplo, para acelerar una descarga, un destinatario podría conectarse simultáneamente con más de un remitente para transmitir datos referidos al mismo fichero. Como otro ejemplo, en una transmisión de multi-difusión, múltiples flujos de datos de multi-difusión podrían ser transmitidos para permitir a los destinatarios conectarse con uno o más de estos flujos para hacer coincidir la velocidad de transmisión compuesta con el ancho de banda del canal que los conecta con el remitente. En todos los casos de ese tipo, una preocupación es asegurar que todos los datos transmitidos sean de uso independiente para un destinatario, es decir, que los múltiples datos de origen no sean redundantes entre los flujos, incluso cuando las velocidades de transmisión son abismalmente distintas para los distintos flujos y cuando hay patrones arbitrarios de pérdida.

En general, un canal de comunicación es aquel que conecta el remitente y el destinatario para la transmisión de datos. El canal de comunicación podría ser un canal en tiempo real, donde el canal desplaza datos desde el remitente al destinatario según el canal obtiene los datos, o el canal de comunicación podría ser un canal de almacenamiento que almacena algunos de, o todos, los datos en su tránsito desde el remitente al destinatario. Un ejemplo de esto último es el almacenamiento en disco o en algún otro dispositivo de almacenamiento. En ese ejemplo, un programa o dispositivo que genera datos puede ser pensado como el remitente, que transmite los datos a un dispositivo de almacenamiento. El destinatario es el programa o dispositivo que lee los datos desde el dispositivo de almacenamiento. Los mecanismos que el remitente usa para poner los datos en el dispositivo de almacenamiento, el dispositivo de almacenamiento en sí mismo y los mecanismos que el destinatario usa para obtener los datos desde el dispositivo de almacenamiento forman colectivamente el canal. Si hay una posibilidad de que esos mecanismos o el dispositivo de almacenamiento puedan perder datos, entonces eso sería tratado como borradura de datos en el canal de comunicación.

Cuando el remitente y el destinatario están separados por un canal de comunicación en el cual los símbolos pueden ser borrados, es preferible no transmitir una copia exacta de un fichero o flujo de entrada, sino transmitir, en cambio, datos, generados a partir del fichero o flujo de entrada (que podría incluir todo, o partes de, el fichero o flujo de entrada en sí mismo), que asistan en la recuperación de borraduras. Un codificador es un circuito, dispositivo, módulo o segmento de código que gestiona esa tarea. Una manera de ver el funcionamiento del codificador es que el codificador genera símbolos de salida a partir de símbolos de entrada, donde una secuencia de valores de

símbolos de entrada representa el fichero de entrada o un bloque del flujo. Cada símbolo de entrada tendría por tanto una posición, en el fichero de entrada o el bloque del flujo, y un valor. Un descodificador es un circuito, dispositivo, módulo o segmento de código que reconstruye los símbolos de entrada a partir de los símbolos de salida recibidos por el destinatario. En la codificación de múltiples etapas, el codificador y el descodificador están adicionalmente divididos en sub-módulos, realizando cada uno una tarea distinta.

En realizaciones de sistemas de codificación de múltiples etapas, el codificador y el descodificador pueden ser adicionalmente divididos en sub-módulos, realizando cada uno una tarea distinta. Por ejemplo, en algunas realizaciones, el codificador comprende lo que se menciona en la presente memoria como un codificador estático y un codificador dinámico. Según se usa en la presente memoria, un "codificador estático" es un codificador que genera un cierto número de símbolos redundantes a partir de un conjunto de símbolos de entrada, en donde el número de símbolos redundantes está determinado antes de la codificación. Los ejemplos de códigos de codificación estática incluyen los códigos de Reed-Solomon, los códigos Tornado, los códigos de Hamming, los códigos de Control de Paridad de Baja Densidad (LDPC), etc. El término "descodificador estático" se usa en la presente memoria para referirse a un descodificador que puede descodificar datos que fueron codificados por un codificador estático.

Según se usa en la presente memoria, un "codificador dinámico" es un codificador que genera símbolos de salida a partir de un conjunto de símbolos de entrada y, posiblemente, un conjunto de símbolos redundantes. En una realización preferida descrita aquí, el número de posibles símbolos de salida es mayor, en varios órdenes de magnitud, que el número de símbolos de entrada, y el número de símbolos de salida a generar no debe ser necesariamente fijo. Un ejemplo de un codificador dinámico de ese tipo es un codificador de reacción en cadena, tal como los codificadores descritos en Luby I y Luby II. El término "descodificador dinámico" se usa en la presente memoria para referirse a un descodificador que puede descodificar datos que fueron codificados por un codificador dinámico.

Las realizaciones de codificación de múltiples campos no necesariamente deben estar limitadas a ningún tipo específico de símbolo de entrada. Habitualmente, los valores para los símbolos de entrada son seleccionados entre un alfabeto de 2^M símbolos, para algún entero positivo M. En tales casos, un símbolo de entrada puede ser representado por una secuencia de M bits de datos procedentes del fichero o flujo de entrada. El valor de M está a menudo determinado en base, por ejemplo, a los usos de la aplicación, al canal de comunicación y / o al tamaño de los símbolos de salida. Adicionalmente, el tamaño de un símbolo de salida está a menudo determinado en base a la aplicación, el canal y / o el tamaño de los símbolos de entrada. En algunos casos, el proceso de codificación podría ser simplificado si los valores de símbolos de salida y los valores de símbolos de entrada tuvieran el mismo tamaño (es decir, fueran representables por el mismo número de bits o seleccionados a partir del mismo alfabeto). Si es ese el caso, entonces el tamaño del valor del símbolo de entrada está limitado cuando el tamaño del valor del símbolo de salida está limitado. Por ejemplo, puede desearse poner símbolos de salida en paquetes de tamaño limitado. Si algunos datos acerca de una clave asociada a los símbolos de salida fueran a transmitirse a fin de recuperar la clave en el receptor, el símbolo de salida, preferiblemente, sería lo bastante pequeño para asimilar, en un paquete, el valor del símbolo de salida y los datos acerca de la clave.

Como ejemplo, si un fichero de entrada es un fichero de múltiples mega-octetos, el fichero de entrada podría ser descompuesto en miles, decenas de miles o cientos de miles de símbolos de entrada, codificando cada símbolo de entrada miles, cientos o solamente unos pocos octetos. Como otro ejemplo, para un canal de Internet basado en paquetes, un paquete con una carga útil con un tamaño de 1.024 octetos podría ser adecuado (un octeto tiene 8 bits). En este ejemplo, suponiendo que cada paquete contiene un símbolo de salida y 8 octetos de información auxiliar, un tamaño de símbolo de salida de 8.128 bits ((1.024 – 8)*8) sería adecuado. Por tanto, el tamaño del símbolo de entrada podría ser escogido como M = (1.024 – 8) * 8, o 8.128 bits. Como otro ejemplo, algunos sistemas de distribución de vídeo usan la norma de paquetes de MPEG, donde la carga útil de cada paquete comprende 188 octetos. En ese ejemplo, suponiendo que cada paquete contiene un símbolo de salida y 4 octetos de información auxiliar, un tamaño de símbolo de salida de 1.472 bits ((188 – 4) * 8) sería adecuado. Por tanto, el tamaño del símbolo de entrada podría ser escogido como M = (188 – 4) * 8, o 1.472 bits. En un sistema de comunicación de propósito general que use codificación de múltiples etapas, los parámetros específicos de la aplicación, tales como el tamaño del símbolo de entrada (es decir, M, el número de bits codificados por un símbolo de entrada), podrían ser variables fijadas por la aplicación.

Como otro ejemplo, para un flujo que se envía usando paquetes de origen de tamaño variable, el tamaño de símbolos podría ser escogido algo pequeño, de modo que cada paquete de origen pueda ser cubierto con un número entero de símbolos de entrada que tengan un tamaño compuesto, a lo sumo, levemente mayor que el paquete de origen.

Cada símbolo de salida tiene un valor. En una realización preferida, que consideramos más adelante, cada símbolo de salida también tiene asociado al mismo un identificador llamado su "clave". Preferiblemente, la clave de cada símbolo de salida puede ser fácilmente determinada por el destinatario, para permitir al destinatario distinguir un

símbolo de salida de otros símbolos de salida. Preferiblemente, la clave de un símbolo de salida es distinta a las claves de todos los otros símbolos de salida. Hay diversas formas de generación de claves, expuestas en la técnica anterior. Por ejemplo, Luby I describe diversas formas de generación de claves que pueden ser empleadas en realizaciones descritas en la presente memoria.

La codificación de múltiples campos y múltiples etapas es específicamente útil allí donde hay una expectativa de borradura de datos, o donde el destinatario no comienza y termina la recepción exactamente cuando una transmisión comienza y termina. Esta última condición se menciona en la presente memoria como “incompletitud de datos”. Con respecto a sucesos de borradura, la codificación en múltiples etapas comparte muchos de los beneficios de la codificación de reacción en cadena descrita en Luby I. En particular, los códigos de múltiples etapas pueden ser códigos fuente, o códigos sin tasa, en cuyo caso pueden ser generados símbolos de salida distintos, muchas veces más que símbolos de entrada haya, para un conjunto de símbolos de entrada de valor fijo, y puede usarse cualquier número adecuado de símbolos de salida distintos para recuperar los símbolos de entrada, con un grado deseado de precisión. Estas condiciones no afectan adversamente al proceso de comunicación cuando se usa la codificación de múltiples etapas y múltiples campos, porque los símbolos de salida generados con la codificación de múltiples etapas y múltiples campos son aditivos para la información. Por ejemplo, si se pierden cien paquetes debido a una ráfaga de ruido que provoca borradura de datos, pueden recogerse unos cientos de paquetes extra después de la ráfaga, para reemplazar la pérdida de los paquetes borrados. Si se pierden miles de paquetes porque un receptor no se sintonizó con un transmisor cuando empezó a transmitir, el receptor podría tan solo recoger esos miles de paquetes procedentes de cualquier otro periodo de transmisión, o incluso desde otro transmisor. Con la codificación de múltiples etapas y múltiples campos, un receptor no está restringido a recoger ningún conjunto específico de paquetes, por lo que puede recibir algunos paquetes desde un transmisor, conmutar a otro transmisor, perder algunos paquetes, perder el comienzo o el fin de una transmisión dada y recuperar aún un fichero o bloque de entrada de un flujo. La capacidad de incorporarse a, y abandonar, una transmisión sin coordinación entre receptor y transmisor ayuda a simplificar el proceso de comunicación.

En algunas realizaciones, la transmisión de un fichero o flujo usando codificación de múltiples etapas y múltiples campos puede incluir la generación, formación o extracción de símbolos de entrada a partir de un fichero o bloque de entrada de un flujo, el cálculo de símbolos redundantes, la codificación de símbolos de entrada y redundantes en uno o más símbolos de salida, donde cada símbolo de salida es generado en base a su clave, independientemente de todos los otros símbolos de salida, y la transmisión de los símbolos de salida a uno o más destinatarios por un canal. Adicionalmente, en algunas realizaciones, la recepción (y reconstrucción) de una copia del fichero o bloque de entrada de un flujo, usando codificación de múltiples etapas y múltiples campos, puede incluir la recepción de algún conjunto o subconjunto de símbolos de salida, desde uno o más flujos de datos, y la descodificación de los símbolos de entrada a partir de los valores y claves de los símbolos de salida recibidos.

Los códigos adecuados de borradura de FEC, según se describen en la presente memoria, pueden ser usados para superar las dificultades precitadas, y hallarán uso en un buen número de campos que incluyen los sistemas y servicios de difusión de multimedios y de multi-difusión. Un código de borradura de FEC, mencionado en adelante en la presente memoria como “un código de reacción en cadena de múltiples etapas y múltiples campos” tiene propiedades que satisfacen muchos de los requisitos actuales y futuros de tales sistemas y servicios.

Algunas propiedades básicas de los códigos de reacción en cadena de múltiples etapas y múltiples campos son que, para condiciones cualesquiera de pérdida de paquetes, y para la entrega de ficheros de origen de cualquier tamaño relevante, o flujos de cualquier velocidad relevante: (a) el sobregasto de recepción de cada dispositivo receptor individual (“RD”) está minimizado; (b) el tiempo total de transmisión necesario para entregar ficheros de origen a cualquier número de los RD puede ser minimizado y (c) la calidad del flujo entregado a cualquier número de los RD puede ser maximizada para el número de símbolos de salida enviados, con respecto al número de símbolos de entrada, con una selección adecuada de planificaciones de transmisión. Los RD podrían ser dispositivos de mano, incrustados en un vehículo, portátiles (es decir, móviles pero no habitualmente en movimiento cuando se usan) o fijos en una ubicación.

La cantidad de memoria de trabajo necesaria para descodificar es baja y aún puede brindar las propiedades anteriores, y la magnitud del cálculo necesario para codificar y descodificar es mínima. En este documento, proporcionamos una descripción sencilla y fácil de implementar de algunas variaciones de códigos de reacción en cadena de múltiples etapas y múltiples campos.

Los códigos de reacción en cadena de múltiples etapas y múltiples campos con códigos fuente, es decir, pueden ser generados sobre la marcha tantos paquetes de codificación como sean necesarios, conteniendo cada uno símbolos únicos de codificación que son igualmente útiles para recuperar un fichero o bloque de origen de un flujo. Hay muchas ventajas para el uso de códigos fuente ante otros tipos de códigos de FEC. Una ventaja es que, independientemente de las condiciones de pérdidas de paquetes y de la disponibilidad de RD, los códigos fuente minimizan el número de paquetes de codificación que cada RD necesita recibir para reconstruir un fichero o bloque de origen de un flujo. Esto es verdad incluso en condiciones extremas de pérdida de paquetes y cuando, por

ejemplo, los RD móviles se encienden, o están disponibles, solamente de manera intermitente durante una larga sesión de descarga de ficheros.

Otra ventaja es la capacidad de generar exactamente tantos paquetes de codificación como sean necesarios, tomando la decisión de cuántos paquetes de codificación generar sobre la marcha mientras la transmisión está en marcha. Esto puede ser útil si, por ejemplo, hay retro-alimentación desde los RD que indica si recibieron o no suficientes paquetes de codificación para recuperar un fichero o bloque de origen de un flujo. Cuando las condiciones de pérdida de paquetes son menos severas que lo esperado, la transmisión puede ser terminada temprano. Cuando las condiciones de pérdida de paquetes son más severas que lo esperado, o los RD no están disponibles más a menudo que lo esperado, la transmisión puede ser extendida sin fisuras.

Otra ventaja es la capacidad de multiplexar inversamente. El multiplexado inverso es cuando un RD es capaz de combinar los paquetes de codificación recibidos, generados en remitentes independientes, para reconstruir un fichero o bloque de origen de un flujo. Un uso práctico del multiplexado inverso se describe más adelante con referencia a la recepción de paquetes de codificación desde distintos remitentes.

Allí donde la pérdida futura de paquetes, y las condiciones de disponibilidad y aplicación de los RD son difíciles de predecir, es importante escoger una solución de FEC que sea tan flexible como sea posible, para funcionar bien en condiciones impredecibles. Los códigos de reacción en cadena de múltiples etapas proporcionan un grado de flexibilidad no igualado por otros tipos de códigos de FEC.

Una ventaja adicional de los códigos de múltiples etapas y múltiples campos es que la probabilidad de errores y el límite inferior de errores de los códigos son mucho menores que los de códigos previamente conocidos, con complejidad de cálculo equivalente. Igualmente, la complejidad de cálculo de los códigos de reacción en cadena de múltiples etapas y múltiples campos es mucho menor que la de códigos previamente conocidos con una probabilidad de errores y / o límite inferior de errores equivalentes.

Otra ventaja de los códigos de reacción en cadena de múltiples etapas y múltiples campos es que los parámetros tales como el tamaño de símbolos y los tamaños de campos pueden ser escogidos flexiblemente para lograr cualquier equilibrio deseado entre la complejidad de cálculo y la probabilidad de errores y / o el límite máximo de errores.

Se describirán ahora aspectos de la invención con referencia a las figuras.

Panorama general del sistema

La Fig. 1 es un diagrama de bloques de un sistema de comunicaciones 100 que usa codificación de múltiples etapas. En el sistema de comunicaciones 100, un fichero de entrada 101, o un flujo de entrada 105, se proporciona a un generador de símbolos de entrada 110. El generador de símbolos de entrada 110 genera una secuencia de uno o más símbolos de entrada (IS(0), IS(1), IS(2), ...) a partir del fichero o flujo de entrada, teniendo cada símbolo de entrada un valor y una posición (indicado en la Fig. 1 como un entero entre paréntesis). Como se ha explicado anteriormente, los posibles valores para los símbolos de entrada, es decir, su alfabeto, son habitualmente un alfabeto de 2^M símbolos, de modo que cada símbolo de entrada codifica M bits del fichero o flujo de entrada. El valor de M está generalmente determinado por el uso del sistema de comunicación 100, pero un sistema de propósito general podría incluir un tamaño de símbolo ingresado para el generador de símbolos de entrada 110, de modo que M pueda variar entre un uso y otro. La salida del generador de símbolos de entrada 110 se proporciona a un codificador 115.

El generador de claves estáticas 130 produce un flujo de claves estáticas S_0, S_1, ... El número de las claves estáticas generadas está generalmente limitado y depende de la realización específica del codificador 115. La generación de claves estáticas será descrita posteriormente en más detalle. El generador de claves dinámicas 120 genera una clave dinámica para cada símbolo de salida a ser generado por el codificador 115. Cada clave dinámica se genera de modo que una gran fracción de las claves dinámicas, para el mismo fichero o bloque de entrada de un flujo, sean únicas. Por ejemplo, Luby I describe realizaciones de generadores de claves que pueden ser usados. Las salidas del generador de claves dinámicas 120 y el generador de claves estáticas 130 son proporcionadas al codificador 115.

A partir de cada clave I proporcionada por el generador de claves dinámicas 120, el codificador 115 genera un símbolo de salida, con un valor B(I), a partir de los símbolos de entrada proporcionados por el generador de símbolos de entrada. El funcionamiento del codificador 115 será descrito en más detalle más adelante. El valor de cada símbolo de salida se genera en base a su clave, a alguna función de uno o más de los símbolos de entrada y, posiblemente, a uno o más símbolos redundantes que hubieran sido calculados a partir de los símbolos de entrada. La colección de símbolos de entrada y símbolos redundantes que dan origen a un símbolo de salida específico se menciona en la presente memoria como los “símbolos asociados” del símbolo de salida, o solamente sus “asociados”. La selección de la función (la “función de valor”) y los asociados se hace de acuerdo a un proceso

descrito en más detalle más adelante. Habitualmente, pero no siempre, M es el mismo para los símbolos de entrada y los símbolos de salida, es decir, ambos codifican para el mismo número de bits.

En algunas realizaciones, el número K de símbolos de entrada es usado por el codificador 115 para seleccionar los asociados. Si K no se conoce de antemano, tal como allí donde la entrada es un fichero de transmisión por flujo, K puede ser solamente una estimación. El valor de K podría también ser usado por el codificador 115 para asignar almacenamiento para símbolos de entrada y símbolos intermedios cualesquiera, generados por el codificador 115.

El codificador 115 proporciona símbolos de salida a un módulo de transmisión 140. Al módulo de transmisión 140 también se proporciona la clave de cada símbolo de salida de ese tipo, a partir del generador de claves dinámicas 120. El módulo de transmisión 140 transmite los símbolos de salida y, según el procedimiento de formación de claves usado, el módulo de transmisión 140 también podría transmitir algunos datos acerca de las claves de los símbolos de salida transmitidos, por un canal 145, a un módulo de recepción 150. Se supone que el canal 145 es un canal de borradura, pero ese no es un requisito para el funcionamiento adecuado del sistema de comunicación 100. Los módulos 140, 145 y 150 pueden ser componentes adecuados de hardware cualesquiera, componentes de software, medios físicos o cualquiera combinación de los mismos, mientras el módulo de transmisión 140 esté adaptado para transmitir símbolos de salida, y datos necesarios cualesquiera sobre sus claves, al canal 145, y el módulo de recepción 150 esté adaptado para recibir símbolos y, potencialmente, algunos datos acerca de sus claves desde el canal 145. El valor de K, si se usa para determinar los asociados, puede ser enviado por el canal 145, o puede ser fijado por adelantado por acuerdo del codificador 115 y el descodificador 155.

Como se ha explicado anteriormente, el canal 145 puede ser un canal en tiempo real, tal como un trayecto a través de Internet o un enlace de difusión desde un transmisor de televisión a un destinatario de televisión, o una conexión telefónica desde un punto a otro, o el canal 145 puede ser un canal de almacenamiento, tal como un CD-ROM; un controlador de disco, una sede de la Red o similares. El canal 145 podría incluso ser una combinación de un canal en tiempo real y un canal de almacenamiento, tal como un canal formado cuando una persona transmite un fichero de entrada desde un ordenador personal a un Proveedor de Servicios de Internet (ISP) por una línea telefónica, el fichero de entrada es almacenado en un servidor de la Red y es posteriormente transmitido a un destinatario por Internet.

Debido a que se supone que el canal 145 es un canal de borradura, el sistema de comunicaciones 100 no supone una correspondencia biunívoca entre los símbolos de salida que salen del módulo de recepción 150 y los símbolos de salida que van al módulo de transmisión 140. De hecho, allí donde el canal 145 comprende una red de paquetes, el sistema de comunicaciones 100 incluso podría no ser capaz de suponer que el orden relativo de dos o más paquetes cualesquiera se preserve en tránsito a través del canal 145. Por lo tanto, la clave de los símbolos de salida se determina usando uno o más de los esquemas de formación de claves descritos anteriormente, y no está necesariamente determinada por el orden en el cual los símbolos de salida salen del módulo de recepción 150.

El módulo de recepción 150 proporciona los símbolos de salida a un descodificador 155, y cualquier dato que el módulo de recepción 150 reciba acerca de las claves de estos símbolos de salida se proporciona a un regenerador de claves dinámicas 160. El regenerador de claves dinámicas 160 regenera las claves dinámicas para los símbolos de salida recibidos y proporciona estas claves dinámicas al descodificador 155. El generador de claves estáticas 163 regenera las claves estáticas S_0, S_1, ... y las proporciona al descodificador 155. El generador de claves estáticas tiene acceso al generador de números aleatorios 135, usado durante ambos procesos de codificación y de descodificación. Esto puede ser en forma de acceso al mismo dispositivo físico si los números aleatorios son generados en tal dispositivo, o en forma de acceso al mismo algoritmo para la generación de números aleatorios, para lograr un comportamiento idéntico. El descodificador 155 usa las claves proporcionadas por el regenerador de claves dinámicas 160 y el generador de claves estáticas 163, junto con los correspondientes símbolos de salida, para recuperar los símbolos de entrada (nuevamente IS(0), IS(1), IS(2), ...). El descodificador 155 proporciona los símbolos de entrada recuperados a un re-ensamblador de ficheros de entrada 165, que genera una copia 170 del fichero de entrada 101 o del flujo de entrada 105.

Una propiedad de los símbolos de salida producidos por un codificador de reacción en cadena es que un receptor es capaz de recuperar el fichero o bloque original del flujo original tan pronto como hayan sido recibidos suficientes símbolos de salida. Específicamente, para recuperar los K símbolos de entrada originales con alta probabilidad, el receptor necesita aproximadamente K+A símbolos de salida. La razón A / K se llama el “sobregasto relativo de recepción”. El sobregasto relativo de recepción depende el número K de símbolos de entrada, y de la fiabilidad del descodificador. Luby I, Luby II y Shokrollahi I proporcionan revelaciones de sistemas y procedimientos que pueden ser empleados en ciertas realizaciones. Ha de entenderse, sin embargo, que estos sistemas y procedimientos no se requieren a la presente invención, y también pueden ser usadas muchas otras variaciones, modificaciones o alternativas.

Un codificador

La Fig. 2 es un diagrama de bloques de una realización específica del codificador 115 mostrado en la Fig. 1. El codificador 115 comprende un codificador estático 210, un codificador dinámico 220 y un calculador de redundancia 230. El codificador estático 210 recibe las siguientes entradas: a) símbolos de entrada originales IS(0), IS(1), ..., IS(K-1) proporcionados por el generador de símbolos de entrada 110, y almacenados en un almacén temporal de símbolos de entrada 205; b) el número K de símbolos de entrada originales; c) las claves estáticas S_0, S_1, proporcionadas por el generador de claves estáticas 130; y d) un número R de símbolos redundantes. Al recibir estas entradas, el codificador estático 205 calcula R símbolos redundantes RE(0), RE(1), ..., RE(R-1), como se describirá más adelante. Habitualmente, pero no siempre, los símbolos redundantes tienen el mismo tamaño que los símbolos de entrada. En una realización específica, los símbolos redundantes generados por el codificador estático 210 son almacenados en el almacén temporal de símbolos de entrada 205. El almacén temporal de símbolos de entrada 205 puede ser solamente lógico, es decir, el fichero o bloque del flujo puede estar físicamente almacenado en un lugar y las posiciones de los símbolos de entrada dentro del almacén temporal de símbolos 205 podrían solamente ser renombramientos de las posiciones de estos símbolos dentro del fichero o bloque original del flujo.

El codificador dinámico recibe los símbolos de entrada y los símbolos redundantes, y genera símbolos de salida, como se describirá en mayor detalle más adelante. En una realización en la cual los símbolos redundantes son almacenados en el almacén temporal de símbolos de entrada 205, el codificador dinámico 220 recibe los símbolos de entrada y los símbolos redundantes desde el almacén temporal de símbolos de entrada 205.

El calculador de redundancia 230 calcula el número R de símbolos redundantes a partir del número K de símbolos de entrada. Este cálculo se describe en mayor detalle más adelante.

Panorama general del codificador estático

El funcionamiento general del codificador estático 210 se muestra con referencia a las Figs. 3 y 4. La Fig. 3 es un diagrama de flujo simplificado que ilustra una realización de un procedimiento para codificar estáticamente. En una etapa 305, una variable j, que rastrea cuántos símbolos redundantes han sido generados, se fija en cero. Luego, en una etapa 310, se calcula un primer símbolo redundante RE(0) como una función F_0 de al menos algunos de los símbolos de entrada IS(0), ..., IS(K-1). Luego, en una etapa 315, se incrementa la variable j. Luego, en una etapa 320, se comprueba si todos los símbolos redundantes ha sido regenerados o no (es decir, ¿es j mayor que R-1?). Si lo es, entonces el flujo acaba. En otro caso, el flujo avanza a la etapa 325. En la etapa 325, se calcula RE(j) como una función F_j de los símbolos de entrada IS(0), ..., IS(K-1) y de los símbolos redundantes previamente generados RE(0), RE(j-1), donde F_j no necesariamente debe ser una función que dependa de cada uno de los símbolos de entrada, o de cada uno de los símbolos redundantes. Las etapas 315, 320 y 325 se repiten hasta que hayan sido calculados R símbolos redundantes.

Con referencia nuevamente a las Figs. 1 y 2, en algunas realizaciones, el codificador estático 210 recibe una o más claves estáticas S_0, S_1, ..., desde el generador de claves estáticas 130. En estas realizaciones, el codificador estático 210 usa las claves estáticas para determinar algunas de, o todas, las funciones F_0, F_1, ..., F_{j-j}. Por ejemplo, la clave estática So puede ser usada para determinar la función F_0, la clave estática S_1 puede ser usada para determinar la función F_1, etc. O bien, una o más de las claves estáticas S_0, S_1, ..., pueden ser usadas para determinar la función F_0, una o más de las claves estáticas S_0, S_1, ... pueden ser usadas para determinar la función F_1, etc. En otras realizaciones, no se necesita ninguna clave estática, y por tanto no es necesario el generador de claves estáticas 130.

Con referencia ahora a las Figs. 2 y 3, en algunas realizaciones, los símbolos redundantes generados por el codificador estático 210 pueden ser almacenados en el almacén temporal de símbolos de entrada 205. La Fig. 4 es una ilustración simplificada del funcionamiento de una realización del codificador estático 210. En particular, el codificador estático 210 genera el símbolo redundante RE(j) como una función Fj de los símbolos de entrada IS(0), ..., IS(K-1), RE(0), ..., RE(j-1), recibidos desde el almacén temporal de símbolos de entrada 205, y lo almacena de nuevo en el almacén temporal de símbolos de entrada 205. La forma exacta de las funciones F_0, F_1, ..., F_{R-1} depende de la aplicación específica. Habitualmente, pero no siempre, las funciones F_1, F_1, ..., F_{R-1} incluyen una operación lógica OR exclusivo de algunos de, o todos, sus correspondientes argumentos. Según se ha descrito anteriormente, estas funciones pueden o no emplear efectivamente claves estáticas generadas por el generador de claves estáticas 130 de la Fig. 1. Por ejemplo, en una realización específica descrita más adelante, las primeras pocas funciones implementan un código de Hamming y no hacen uso alguno de las claves estáticas S_0, S_1, .., mientras que las funciones restantes implementan un código de Control de Paridad de Baja Densidad y hacen uso explícito de las claves estáticas.

Panorama general del codificador de múltiples etapas

Con referencia nuevamente a la Fig. 2, el codificador dinámico 220 recibe los símbolos de entrada IS(0), ..., IS(K-1) y los símbolos redundantes RE(0), ..., RE(R-1) y una clave I para cada símbolo de salida que ha de generar. La colección que comprende los símbolos de entrada y los símbolos redundantes originales será mencionada como la

colección de “símbolos de entrada dinámicos” a continuación. La Fig. 5 es un diagrama de bloques simplificado de una realización de un codificador dinámico, que incluye un selector de ponderaciones 510, un asociador 515, un selector de funciones de valor 520 y un calculador 525. Según se muestra en la Fig. 5, los K+R símbolos de entrada dinámicos son almacenados en un almacén temporal de símbolos dinámicos 505. En efecto, el codificador dinámico 500 realiza la acción ilustrada en la Fig. 6, esto es, generar un valor de símbolo de salida B(l) como alguna función de valor de símbolos de entrada seleccionados.

La Fig. 7 es un diagrama de bloques simplificado de una realización específica de un codificador estático. El codificador estático 600 comprende un calculador de parámetros 605, un codificador de control de paridad de baja densidad (LDPC) 610 y un codificador de control de paridad de alta densidad (HDPC) 620. El codificador de LDPC 610 está acoplado para recibir los símbolos de entrada IS(0), ..., IS(K-1) desde un almacén temporal de símbolos de entrada 625, el número K de símbolos de entrada y el parámetro E. En respuesta. el codificador de LDPC 610 genera E símbolos redundantes LD(0), ..., LD(E-1), de acuerdo al código de LDPC. Luego, el codificador de HDPC 620 está acoplado para recibir la pluralidad de K+E símbolos IS(0), ..., IS(K-1), LD(0), ..., LD(E-1) y el parámetro D, para generar D símbolos redundantes HA(0), HA(1), ..., HA(D-1), de acuerdo al código de HDPC.

La Fig. 8 ilustra el funcionamiento de una realización que emplea el codificador estático mostrado en la Fig. 7.

La Fig. 9 es un diagrama de flujo simplificado que ilustra una realización de un calculador de parámetros, tal como el calculador de parámetros 605 de la Fig. 7, que calcula el parámetro D y E según lo descrito anteriormente, cuando el código de HDPC es un código de Hamming. Primero, en una etapa 705, el parámetro D se inicializa en uno. Luego, en la etapa 710, se determina si $2^D - D - 1$ es menor o no que K. Si no lo es entonces el flujo avanza a la etapa 730. Si lo es, el flujo avanza a la etapa 720, donde se incrementa el parámetro D. Luego, el flujo avanza de nuevo a la etapa 710. Una vez que D ha sido determinado, entonces, en la etapa 730, el parámetro E se calcula como R – D – 1.

La Fig. 10 es un diagrama de flujo simplificado de un codificador de ese tipo, de acuerdo a una realización de la presente invención, que será descrita ahora. Primero, en la etapa 805, se inicializa una variable i con cero. La variable i rastrea el número de símbolos redundantes ya generados. En la etapa 810, se calcula un número t como el más pequeño entero impar mayor o igual a K / 2. En la etapa 815, los valores P_1, P_2, ..., Pt son generados en base a K, t y una clave estática Si. Los valores P_1, P_2, ..., P_t indican las posiciones de los símbolos de entrada que serán usados para generar un símbolo redundante. En una realización específica, un asociador tal como el asociador 515 de la Fig. 5 se usa para generar P_1, P_2, ..., P_t. En particular, el valor t puede ser proporcionado como la entrada W(l), el valor K puede ser proporcionado como la entrada K+R y la clave estática S_i puede ser proporcionada como la entrada de la clave l. Debería observarse que muchos valores distintos de t producirían similares efectos de codificación, y por tanto esta elección específica es solamente un ejemplo. En la etapa 820, el valor de RE(i) se calcula como la operación lógica XOR de los valores $IS(P_1)$, $IS(P_2)$, ..., $IS(P_t)$. En la etapa 825, la variable i se incrementa en uno para preparar el cálculo del próximo símbolo redundante y, en la etapa 830, se determina si todos los símbolos redundantes han sido calculados o no. Si no lo han sido, entonces el flujo vuelve a la etapa 815.

La Fig. 11 es un diagrama de bloques simplificado que ilustra una realización de un descodificador de acuerdo a la presente invención. El descodificador 900 puede ser usado, por ejemplo, para implementar el descodificador 155 de la Fig. 1.

El descodificador 900 comprende un descodificador dinámico 905 y un descodificador estático 910. Los símbolos de entrada y los símbolos redundantes recuperados por el descodificador dinámico 905 son almacenados en un almacén temporal de reconstrucción 915. Al completar la descodificación dinámica, el descodificador estático 910 intenta recuperar todo símbolo de entrada no recuperado por el descodificador dinámico 905, si lo hubiera. En particular, el descodificador estático 910 recibe símbolos de entrada y símbolos redundantes desde el almacén temporal de reconstrucción 915.

La Fig. 12 es un diagrama de flujo simplificado que ilustra una realización de un procedimiento para descodificar, de acuerdo a la presente invención. En la etapa 1005, Q símbolos de salida son recibidos por el descodificador. El valor de Q puede depender del número de símbolos de entrada y del codificador dinámico específico usado. El valor de Q también puede depender del grado deseado de precisión con el cual el descodificador puede recuperar los símbolos de entrada. Por ejemplo, si se desea que el descodificador pueda recuperar todos los símbolos de entrada con una alta probabilidad, entonces Q debería ser escogido mayor que el número de símbolos de entrada. En particular, en algunas aplicaciones, cuando el número de símbolos de entrada es grande, Q puede ser menos que un 3% más grande que el número de símbolos de entrada originales. En otras aplicaciones, cuando el número de símbolos de entrada es pequeño, Q puede ser al menos un 10% mayor que el número de símbolos de entrada. Específicamente, Q puede ser escogido como el número K de símbolos de entrada más un número A, donde A se escoge para asegurar que el descodificador pueda regenerar todos los símbolos de entrada con alta probabilidad. La determinación del número A se describe en mayor detalle más adelante. Si es aceptable que el descodificador sea incapaz de descodificar todos los símbolos de entrada (bien algunas veces, o bien siempre), entonces Q puede ser

menor que K+A, igual a K o incluso menor que K. Es claro que un objetivo de un sistema de codificación global será a menudo disminuir el número Q tanto como sea posible, manteniendo a la vez buenas garantías probabilísticas del éxito del proceso de descodificación con respecto al grado deseado de precisión.

En la etapa 1010, el descodificador dinámico 905 regenera símbolos de entrada y símbolos redundantes a partir de los Q símbolos de salida recibidos. Ha de entenderse que las etapas 1005 y 1010 pueden ser realizadas de manera esencialmente simultánea. Por ejemplo, el descodificador dinámico 905 puede comenzar a regenerar símbolos de entrada y símbolos redundantes antes de que el descodificador reciba Q símbolos de salida.

Después de que el descodificador dinámico 905 ha procesado Q símbolos de salida, se determina si los símbolos de entrada han sido recuperados o no con un grado deseado de precisión. El grado deseado de precisión puede ser, por ejemplo, todos los símbolos de entrada, o algún número, porcentaje, etc., menor que todos los símbolos de entrada. Si es así, entonces el flujo acaba. Si no es así, entonces el flujo avanza a la etapa 1020. En la etapa 1020, el descodificador estático 910 intenta recuperar todo símbolo de entrada que el descodificador dinámico 905 fue incapaz de recuperar. Después de que el codificador estático 910 ha procesado los símbolos de entrada y los símbolos redundantes recuperados por el codificador dinámico 950, entonces el flujo acaba.

La Fig. 13 es un diagrama de flujo simplificado que ilustra otra realización de un procedimiento para descodificar de acuerdo a la presente invención. Esta realización es similar a la descrita con respecto a la Figa. 11, e incluye las etapas 1005, 1010, 1015 y 1025 en común. Pero, después de la etapa 1025, el flujo avanza a la etapa 1030, en la cual se determina si los símbolos de entrada han sido recuperados o no con un grado deseado de precisión. Si es así, entonces el flujo acaba. Si no es así, entonces el flujo avanza a la etapa 1035. En la etapa 1035, se reciben uno o más símbolos de salida adicionales. Luego, el flujo avanza de nuevo a la etapa 1010, de modo que el descodificador dinámico 905 y / o el descodificador estático 910 puedan intentar recuperar los restantes símbolos de entrada no recuperados.

La Fig. 14 es un diagrama de flujo simplificado que ilustra otra realización más de un procedimiento para la descodificación de acuerdo a la presente invención. En la etapa 1055, los símbolos de salida son recibidos por el descodificador y, en la etapa 1060, el descodificador dinámico 905 regenera símbolos de entrada y símbolos redundantes a partir de los símbolos de salida recibidos. Luego, en la etapa 1065, se determina si la descodificación dinámica debería ser finalizada o no. Esta determinación puede estar basada en uno o más del número de símbolos de salida procesados, el número de símbolos de entrada recuperados, la velocidad actual a la cual están siendo recuperados los símbolos de entrada adicionales, el tiempo empleado procesando símbolos de salida, etc.

En la etapa 1065, si se determina que la descodificación dinámica no ha de ser detenida, entonces el flujo avanza de nuevo a la etapa 1055. Pero, si en la etapa 1065 se determina finalizar la descodificación dinámica, entonces el flujo avanza a la etapa 107. En la etapa 1070, se determina si los símbolos de entrada han sido recuperados o no con un grado deseado de precisión. Si es así, entonces el flujo acaba. Si no es así, entonces el flujo avanza a la etapa 1075. En la etapa 1075, el descodificador estático 910 intenta recuperar todo símbolo de entrada que el descodificador dinámico 905 fue incapaz de recuperar. Después de que el codificador estático 910 ha procesado los símbolos de entrada y los símbolos redundantes recuperados por el descodificador dinámico 905, el flujo acaba.

La Fig. 15 muestra una realización del descodificador dinámico de acuerdo a la presente invención. El descodificador dinámico 1100 incluye componentes similares a los del codificador dinámico 500 mostrado en la Fig. 5. El descodificador 1100 es similar a las realizaciones de los descodificadores de reacción en cadena descritos en Luby I y Luby II. El descodificador dinámico 1100 comprende un selector de ponderación 510, un asociador 515, un selector de función de valor 520, un almacén temporal de símbolos de salida 1105, un reductor 1115, un reconstructor 1120 y un almacén temporal de reconstrucción 1125.

La Fig. 16 es un diagrama de bloques simplificado que ilustra una realización de un descodificador estático. Esta realización puede ser usada cuando los datos son codificados con un codificador estático tal como el descrito con referencia a la Fig. 7. El descodificador estático 1200 comprende un descodificador de LDPC 1205 y un descodificador de Hamming 1210. El descodificador de LDPC 1205 recibe símbolos de entrada y símbolos redundantes desde un almacén temporal de reconstrucción 1215, e intenta reconstruir aquellos símbolos del almacén temporal de reconstrucción 1215, no recuperados después de la etapa de descodificación del descodificador dinámico. En algunas realizaciones, el almacén temporal de reconstrucción 1215 es el almacén temporal de reconstrucción 1125 (Fig. 15).

Muchas variaciones de los descodificadores de LDPC y los descodificadores de HDPC son bien conocidas para los expertos en la técnica, y pueden ser empleadas en diversas realizaciones de acuerdo a la presente invención. En una realización específica, el descodificador de HDPC se implementa usando un algoritmo de eliminación Gaussiana. Muchas variaciones de algoritmos de eliminación Gaussiana son bien conocidas para los expertos en la técnica, y pueden ser empleadas en diversas realizaciones de acuerdo a la presente invención.

Una variación de la codificación de HDPC

Otro tipo de codificación de HDPC se describe ahora. En esta realización de la codificación de HDPC, la operación matemática para crear símbolos redundantes a partir de un conjunto dado de datos se basa en operaciones en un campo finito.

En esta realización de la codificación de HDPC, los elementos de un campo finito se usan para obtener símbolos redundantes HD[0], ..., HD[D-1]. Estos símbolos son obtenidos definiendo un proceso de multiplicación entre los símbolos IS[0], ..., IS[K-1], LD[0], ..., LD[E-1]. Estos símbolos se obtienen definiendo un proceso de multiplicación entre los símbolos IS[0], ..., IS[K-1], LD[0], ... LD[E-1] y los elementos del campo finito, según lo descrito anteriormente.

Codificación de HDPC

Cuando se usa un código de HDPC, el código podría ser descrito por una matriz generadora sobre un campo finito GF(2M). Allí donde el código es sistemático, que es el caso en una realización preferida, la matriz generadora puede ser descrita usando solamente la relación entre los K+E símbolos de entrada IS[0], ..., IS[K-1], LD[0], ..., LD[E-1] y los símbolos redundantes HD[0], ..., HD[D-1]. Esta matriz, llamada *G*, es de formato Dx(K+E). Si *X* indica el vector columna que comprende los símbolos HD[0], ... HD[D-1] y *S* indica el vector columna que comprende los símbolos IS[0], ..., IS[K-1], LD[0], ..., LD[E-1], entonces tenemos $X = G \oplus S$. Se describen más adelante realizaciones más específicas para la matriz *G* y diversos procedimientos para el cálculo eficaz de los símbolos.

Variaciones

Los códigos de reacción en cadena de múltiples etapas, según lo descrito anteriormente, no son códigos sistemáticos, es decir, no todos los símbolos originales de origen de un bloque de origen están necesariamente entre los símbolos de codificación que se envían. Sin embargo, los códigos de FEC sistemáticos son útiles para un sistema o servicio de descarga de ficheros, y muy importantes para un sistema o servicio de flujos de transmisión. Según se muestra en la implementación más adelante, puede hacerse que un código modificado sea sistemático y mantenga aún el código fuente y otras propiedades descritas.

Un motivo por el cual es fácil construir una amplia variedad de servicios suplementarios usando códigos de múltiples etapas es que puede combinar los símbolos de codificación recibidos desde múltiples remitentes para reconstruir un fichero o flujo de origen, sin coordinación entre los remitentes. El único requisito es que los remitentes usen distintos conjuntos de claves para generar los símbolos de codificación que envían al codificar paquetes según el código. Las formas de lograr esto incluyen la designación de distintas gamas del espacio de claves a usar por parte de cada remitente de ese tipo, o la generación de claves aleatoriamente en cada remitente.

Como ejemplo del uso de esta capacidad, considérese proporcionar un servicio suplementario a un servicio de descarga de ficheros que permita a los códigos de reacción en cadena de múltiples etapas, que no recibieron suficientes paquetes de codificación para reconstruir un fichero de origen a partir de la sesión de descarga de ficheros, solicitar paquetes de codificación adicionales, a enviar desde un remitente sustituto, p. ej., mediante una sesión de HTTP. El remitente sustituto genera símbolos de codificación a partir del fichero de origen y los envía, por ejemplo, usando HTTP, y todos estos símbolos de codificación pueden ser combinados con los recibidos desde la sesión de descarga de ficheros para recuperar el fichero de origen. El uso de este enfoque permite a distintos remitentes proporcionar servicios incrementales de entrega de ficheros de origen, sin coordinación entre los remitentes, y asegurando que cada receptor individual solamente necesita recibir un número mínimo de paquetes de codificación para recuperar cada fichero de origen.

La descodificación de códigos de reacción en cadena de múltiples etapas, según lo descrito anteriormente, puede requerir un sobregasto relativamente grande cuando el número de símbolos de origen es pequeño, por ejemplo, del orden de cientos a unos pocos miles de símbolos de origen. En tal caso, se prefiere un descodificador diferente, por ejemplo, un descodificador divulgado en Shokrollahi III. Según se muestra en la implementación más adelante, puede diseñarse un algoritmo de descodificación modificado para la clase de códigos divulgados en la presente memoria, que usa características de los códigos y los conceptos divulgados en Shokrollahi III, y proporciona una baja probabilidad de errores de descodificación para números muy pequeños de símbolos de origen, manteniendo a la vez la eficacia en la descodificación.

Implementaciones de diversas etapas de códigos de múltiples etapas y múltiples campos

Definición del esquema de FEC

Un paquete que usa estas técnicas podría ser representado con información de cabecera, tal como un Identificador de Carga Útil de FEC, de cuatro octetos, que comprende un Número de Bloque de Origen (SBN) (identificador

entero de 16 bits para el bloque de origen al que se refieren los símbolos de codificación dentro del paquete) y un Identificador de Símbolo de Codificación (ESI) (identificador entero de 16 bits para los símbolos de codificación dentro del paquete). Una interpretación adecuada del Número de Bloque de Origen y del Identificador de Símbolo de Codificación está definida en las Secciones B más adelante. La información de Transmisión de Objetos de FEC podría comprender el Identificador de Codificación de FEC, una Longitud de Transferencia (*F*) y los parámetros *T, Z, N* y *A* definidos más adelante. Los parámetros *T* y *Z* son enteros sin signo de 16 bits, *N* y *A* son enteros sin signo de 8 bits. Si es necesario, podrían usarse otros tamaños de enteros.

Se define un esquema de codificación de FEC para la corrección anticipada de errores, en las secciones a continuación. Define dos distintos formatos de Identificador de Carga Útil de FEC, uno para los paquetes de origen de FEC y otro para los paquetes de reparación de FEC, pero también son posibles variaciones para códigos no sistemáticos.

El Identificador de carga útil de FEC de Origen podría comprender un Número de Bloque de Origen (SBN) (identificador entero de 16 bits para el bloque de origen al que se refieren los símbolos de codificación dentro del paquete) y un Identificador de Símbolo de Codificación (ESI) (identificador entero de 16 bits para los símbolos de codificación dentro del paquete), mientras que el Identificador de Carga Útil de FEC de Reparación podría comprender un Número de Bloque de Origen (SBN) (identificador entero de 16 bits para el bloque de origen al que se refieren los símbolos de reparación dentro del paquete), un Identificador de Símbolo de Codificación (ESI) (identificador entero de 16 bits para los símbolos de reparación dentro del paquete), y una Longitud de Bloque de Origen (SBL) (16 bits, que representan el número de símbolos de origen en el bloque de origen. La interpretación del Número de Bloque de Origen, el Identificador de Símbolo de Codificación y la Longitud de Bloque de Origen se define más adelante.

La información de Transmisión de Objetos de FEC podría comprender el Identificador de Codificación de FEC, la máxima longitud de bloque de origen, en símbolos, y el tamaño de símbolos, en octetos. El tamaño de símbolos y la máxima longitud de bloque de origen podrían comprender un campo de cuatro octetos del Tamaño de Símbolos (*T*) (16 bits que representan el tamaño de un símbolo de codificación, en octetos), y una Longitud Máxima de Bloque de Origen (16 bits que representan la máxima longitud de un bloque de origen, en símbolos).

Las secciones a continuación especifican el código sistemático de corrección anticipada de errores MSCR. Los códigos MSCR de múltiples campos son códigos fuente, es decir, tantos símbolos de codificación como se necesiten pueden ser generados por el codificador sobre la marcha, a partir de los símbolos de origen de un bloque. El descodificador es capaz de recuperar el bloque de origen a partir de cualquier conjunto de símbolos de codificación, solo levemente mayores en número al número de símbolos de origen. El código descrito en este documento es un código sistemático, es decir, los símbolos originales de origen son enviados no modificados desde el remitente al receptor, así como un cierto número de símbolos de reparación.

B. 1 Definiciones, símbolos y abreviaturas

B. 1. 1 Definiciones

Para los fines de esta descripción, se aplican los siguientes términos y definiciones.

Bloque de origen: un bloque de K símbolos de origen que se consideran conjuntamente para los fines de codificación de MSCR.

Símbolo de origen: la más pequeña unidad de datos usada durante el proceso de codificación. Todos los símbolos de origen dentro de un bloque de origen tienen el mismo tamaño.

Símbolo de codificación: un símbolo que está incluido en un paquete de datos. Los símbolos de codificación comprenden los símbolos de origen y los símbolos de reparación. Los símbolos de reparación generados a partir de un bloque de origen tienen el mismo tamaño que los símbolos de origen de ese bloque de origen.

Código sistemático: un código en el cual los símbolos de origen están incluidos como parte de los símbolos de codificación enviados para un bloque de origen.

Símbolo de reparación: los símbolos de codificación enviados para un bloque de origen que no son los símbolos de origen. Los símbolos de reparación son generados en base a los símbolos de origen.

Símbolos intermedios: símbolos generados a partir de los símbolos de origen usando un proceso de codificación inversa. Los símbolos de reparación son luego generados directamente a partir de los símbolos intermedios. Los símbolos de codificación no incluyen los símbolos intermedios, es decir, los símbolos intermedios no están incluidos en los paquetes de datos.

Símbolo: una unidad de datos. El tamaño, en octetos, de un símbolo es conocido como el tamaño del símbolo.

Grupo de símbolos de codificación: un grupo de símbolos de codificación que se envían juntos, es decir, dentro del mismo paquete cuya relación con los símbolos de origen puede ser obtenida a partir de un único Identificador de Símbolo de Codificación.

Identificador de Símbolo de Codificación: información que define la relación entre los símbolos de un grupo de símbolos de codificación y los símbolos de origen

Paquete de codificación: paquetes de datos que contienen símbolos de codificación

Sub-bloque: un bloque de origen se descompone a veces en sub-bloques, cada uno de los cuales es suficientemente pequeño para ser descodificado en la memoria de trabajo. Para un bloque de origen que comprende K símbolos de origen, cada sub-bloque comprende K sub-símbolos, estando cada símbolo del bloque de origen compuesto por un sub-símbolo de cada sub-bloque.

Sub-símbolo: parte de un símbolo. Cada símbolo de origen está compuesto por tantos sub-símbolos como sub-bloques hay en el bloque de origen.

Paquete de origen: paquetes de datos que contienen símbolos de origen.

Paquete de reparación: paquetes de datos que contienen símbolos de reparación.

B. 1. 2. Símbolos

i, j, x, h, a, b, d, v, m	representan enteros positivos
techo(*x*)	indica el más pequeño entero positivo que es mayor o igual a *x*
escoger(*i,j*)	indica el número de maneras en que *j* objetos pueden ser escogidos entre *i* objetos sin repetición
suelo(*x*)	indica el mayor entero positivo que es menor o igual a *x*
i % j	indica *i* módulo *j*
X ^ Y	indica, para cadenas *X* e *Y* de bits de igual longitud, la operación lógica 'o' exclusivo bit a bit de *X* e *Y*
A	indica un parámetro de alineación de símbolo. Los tamaños de símbolo y de sub-símbolo están restringidos a ser múltiplos de *A*.
$\mathbf{A}^T$	indica la matriz traspuesta de la matriz **A**
$\mathbf{A}^{-1}$	indica la matriz inversa de la matriz **A**
K	indica el número de símbolos en un único bloque de origen
K_{MAX}	indica el número máximo de símbolos de origen que pueden estar en un único bloque de origen. Fijado en 8.192. Obsérvese que podrían ser usados otros valores.
L	indica el número de símbolos de pre-codificación para un único bloque de origen
S	indica el número de símbolos de LDPC para un único bloque de origen
H	indica el número de semi-símbolos para un único bloque de origen
C	indica una formación de símbolos intermedios, *C*[0], *C*[1], *C*[2], ..., *C*[*L*-1]
C'	indica una formación de símbolos de origen, *C*[0], *C*[1], *C*[2], ..., *C*[*K*-1]
X	un valor entero no negativo
V_0, V_1	dos formaciones de enteros de 4 octetos, V_0[0], V_0[1], ... V_0[255]; V_1[0], V_1[1], ..., V_1[255]
Rand[*X, i, m*]	un generador de números seudo-aleatorios
Deg[*v*]	un generador de grados
LTEnc[*K*, **C**, (*d, a, b*)]	un generador de símbolos de codificación LT
Trip[*K, X*]	una función generadora triple
G	el número de símbolos dentro de un grupo de símbolos de codificación
N	el número de sub-bloques dentro de un bloque de origen
T	el tamaño de símbolo en octetos. Si el bloque de origen está dividido en sub-bloques, entonces $T = T' \cdot N$.
T'	el tamaño de sub-símbolo, en octetos. Si el bloque de origen no está dividido en sub-bloques, entonces *T'* no es relevante.
F	el tamaño de fichero, para la descarga de ficheros, en octetos
I	el tamaño de sub-bloque en octetos
P	para descarga de ficheros, el tamaño de carga útil de cada paquete, en octetos, que se usa en una derivación preferida de los parámetros de transporte de la descarga de ficheros. Para el flujo de transmisión, el tamaño de carga útil de cada paquete de reparación, en octetos, que se usa en una derivación preferida de los parámetros de transporte del flujo de transmisión.
Q	Q = 65.521, es decir, Q es el mayor primo más pequeño que 2^{16}. Obsérvese que podrían usarse otros valores en lugar de 2^{16}.
Z	el número de bloques de origen, para la descarga de ficheros
J(K)	el índice sistemático asociado a *K*
G	indica cualquier matriz generadora
$\mathbf{I}_S$	indica la matriz identidad de dimensiones *S*x*S*
$\mathbf{0}_{SxH}$	indica la matriz de ceros de dimensiones *S*x*H*

B. 1. 3 Abreviaturas

Para los fines del presente documento, se aplican las siguientes abreviaturas:

ESI: Identificador de Símbolo de Codificación
LDPC: Control de Paridad de Baja Densidad
LT: Transformación de Luby
SBN: Número de Bloque de Origen
SBL: Longitud de bloque de Origen (en unidades de símbolos)

B. 2. Panorama general

El código de corrección anticipada de errores MSCR puede ser aplicado a aplicaciones tanto de entrega de ficheros como de flujo de transmisión. Los aspectos del código MSCR que son específicos para cada una de estas aplicaciones se exponen en las Secciones B.3 y B.4 de este documento.

Un componente del código MSCR sistemático es el codificador básico descrito en la Sección B.5. En primer lugar, se describe cómo obtener valores para un conjunto de símbolos intermedios a partir de los símbolos de origen originales, de modo que el conocimiento de los símbolos intermedios sea suficiente para reconstruir los símbolos de origen. En segundo lugar, el codificador produce símbolos de reparación, cada uno de los cuales es el OR exclusivo de un cierto número de los símbolos intermedios. Los símbolos de codificación son la combinación de los símbolos de origen y de reparación. Los símbolos de reparación se producen de tal forma que los símbolos intermedios y, por lo tanto, también los símbolos de origen puedan ser recuperados a partir de cualquier conjunto suficientemente grande de símbolos de codificación.

Este documento define el codificador sistemático de código MSCR. Son posibles un cierto número de posibles algoritmos de descodificación. Un eficaz algoritmo de descodificación se proporciona en la Sección B.6.

La construcción de los símbolos intermedios y de reparación se basa, en parte, en un generador de números seudo-aleatorios descrito en la Sección B.5. Este generador está basado en un conjunto fijo de 512 números aleatorios que están disponibles tanto para el remitente como para el receptor. Un conjunto ejemplar de números son los proporcionados en los Apéndices B.1 y B.2.

Finalmente, la construcción de los símbolos intermedios a partir de los símbolos de origen está gobernada por un "índice sistemático". Un conjunto ejemplar de valores para el índice sistemático se muestra en el Apéndice A para tamaños de bloque de origen desde 4 símbolos de origen hasta K_{MAX} = 8.192 símbolos de origen.

B. 3. Descarga de ficheros

B. 3. 1. Construcción de bloques de origen

B. 3. 1. 1. General

A fin de aplicar el codificador MSCR a un fichero de origen, el fichero puede ser descompuesto en $Z \geq 1$ bloques, conocidos como *bloques de origen*. El codificador MSCR se aplica independientemente a cada bloque de origen. Cada bloque de origen está identificado por un único Número de Bloque de Origen (SBN) entero, donde el primer bloque de origen tiene SBN cero, el segundo tiene SBN uno, etc. Cada bloque de origen está dividido en un número, *K*, de *símbolos de origen* de *T* octetos de tamaño cada uno. Cada símbolo de origen está identificado por un único Identificador de Símbolo de Codificación (ESI) entero, donde el primer símbolo de origen de un bloque de origen tiene ESI cero, el segundo tiene ESI uno, etc.

Cada bloque de origen con *K* símbolos de origen está dividido en $N \geq 1$ sub-bloques, que son lo bastante pequeños para ser descodificados en la memoria de trabajo. Cada sub-bloque está dividido en *K* sub-símbolos de tamaño *T'*.

Obsérvese que el valor de *K* no es necesariamente el mismo para cada bloque de origen de un fichero y que el valor de *T'* puede no ser necesariamente el mismo para cada sub-bloque de un bloque de origen. Sin embargo, el tamaño de símbolo *T* es el mismo para todos los bloques de origen de un fichero y el número de símbolos, *K*, es el mismo para todo sub-bloque de un bloque de origen. La división exacta del fichero en bloques de origen y sub-bloques se describe en B.3.1.2 más adelante.

La Fig. 17 muestra un bloque de origen ejemplar colocado en una formación bidimensional, donde cada entrada es un sub-símbolo de *T'* octetos, cada fila es un sub-bloque y cada columna es un símbolo de origen. En este ejemplo, el valor de *T'* es el mismo para cada sub-bloque. El número mostrado en cada entrada de sub-símbolo indica su orden original dentro del bloque de origen. Por ejemplo, el sub-símbolo con número *K* contiene los octetos $T' \cdot K$ a $T' \cdot (K+1)-1$ del bloque de origen. Entonces, el símbolo de origen *i* es la concatenación del *i*-ésimo sub-símbolo proveniente de cada uno de los sub-bloques, que corresponde a los sub-símbolos del bloque de origen con número i, $K+i$, $2 \cdot K+i$, ..., $(N-1) \cdot K+i$.

B. 3. 1. 2 División de bloques de origen y de sub-bloques

La construcción de bloques de origen y de sub-bloques está determinada en base a cinco parámetros de entrada, *F, A, T, Z* y *N* y una función División[]. Los cinco parámetros de entrada están definidos de la siguiente manera:

F el tamaño del fichero, en octetos

A un parámetro de alineación de símbolos, en octetos
T el tamaño de símbolo, en octetos, que es, preferiblemente, un múltiplo de *A*
Z el número de bloques de origen
N el número de sub-bloques en cada bloque de origen

Estos parámetros podrían fijarse de modo que techo(techo(*F/T)/Z*) ≤ K_{MAX}. Un ejemplo de algunas derivaciones adecuadas de estos parámetros se proporciona en la Sección B.3.4.

La función División[] toma un par de enteros (*I, J*) como entrada y obtiene cuatro enteros (I_L, I_S, J_L, J_S) como salida. Específicamente, el valor de División[*I, J*] es una secuencia de cuatro enteros (I_L, I_S, J_L, J_S), donde I_L = techo(*I/J*), I_S = suelo(*I/J*), $J_L = I - I_S \cdot J$ y $J_S = J - J_L$. División[] obtiene parámetros para dividir un bloque de tamaño *I* en *J* bloques de tamaño aproximadamente igual. Específicamente, J_L bloques de longitud I_L y J_S bloques de longitud I_S.

El fichero de origen podría ser dividido en bloques de origen y sub-bloques de la siguiente manera:

Sea

$$K_t = \text{techo}(F/T)$$

$$(K_L, K_S, Z_L, Z_S) = \text{División}[Kt, Z]$$
$$(T_L, T_S, N_L, N_S) = \text{División}[T/A, N]$$

Luego, el fichero podría ser dividido en $Z = Z_L + Z_S$ bloques de origen contiguos, teniendo cada uno de los primeros Z_L bloques de origen una longitud de $K_L \cdot T$ octetos, y teniendo cada uno de los restantes Z_S bloques de origen $K_S \cdot T$ octetos.

Si $K_t \cdot T > F$, entonces, con fines de codificación, el último símbolo podría ser rellenado al final con $K_t \cdot T - F$ octetos nulos.

A continuación, cada bloque de origen podría ser dividido en $N = N_L + N_S$ sub-bloques contiguos, comprendiendo cada uno de los primeros N_L sub-bloques *K* sub-símbolos contiguos de tamaño $T_L \cdot A$, y comprendiendo cada uno de los restantes N_S sub-bloques *K* sub-símbolos contiguos de tamaño $T_S \cdot A$. El parámetro de alineación de símbolos *A* asegura que los sub-símbolos tienen siempre un múltiplo de *A* octetos.

Finalmente, el *m*-ésimo símbolo de un bloque de origen comprende la concatenación del *m*-ésimo sub-símbolo procedente de cada uno de los *N* sub-bloques.

B. 3. 2. Construcción de paquetes de codificación

B. 3. 2. 1. General

Cada paquete de codificación contiene un Número de Bloque de Origen (SBN), un Identificador de Símbolo de Codificación (ESI) y uno o más símbolos de codificación. Cada bloque de origen se codifica independientemente de los otros. Los bloques de origen están numerados consecutivamente desde cero. Los valores del Identificador de Símbolo de Codificación, entre 0 y *K*-1, identifican los símbolos de origen. Los Identificadores de Símbolos de Codificación desde K en adelante identifican símbolos de reparación.

B. 3. 2. 2 Construcción de paquetes de codificación

Cada paquete de codificación, preferiblemente, contiene bien símbolos de origen (paquete de origen) o bien contiene símbolos de reparación (paquete de reparación). Un paquete puede contener cualquier número de símbolos procedentes del mismo bloque de origen. En el caso en que el último símbolo en el paquete incluya octetos de relleno añadidos con fines de codificación de FEC, entonces estos octetos no necesariamente deben ser incluidos en el paquete. En otro caso, solamente podrían incluirse símbolos enteros.

El Identificador de Símbolo de Codificación, *X*, llevado en cada paquete de origen es el Identificador de Símbolo de Codificación del primer símbolo de origen llevado en el paquete. Los posteriores símbolos de origen en el paquete tienen Identificadores de Símbolos de Codificación, *X*+1 a *X*+*G*-1, en orden secuencial, donde *G* es el número de símbolos en el paquete.

De manera similar, el Identificador de Símbolo de Codificación, *X*, colocado en un paquete de reparación es el Identificador de Símbolo de Codificación del primer símbolo de reparación en el paquete de reparación y los posteriores símbolos de reparación en el paquete tienen Identificadores de Símbolos de Codificación *X*+1 a *X*+*G*-1, en orden secuencial, donde *G* es el número de símbolos en el paquete.

Obsérvese que no es necesario que el receptor conozca el número total de paquetes de reparación. Los *G* tríos de símbolos de reparación (*d*[0], *a*[0], *b*[0]), ..., (*d*[*G*-1], *a*[*G*-1], *b*[*G*-1]) para los símbolos de reparación colocados en un paquete de reparación con ESI X se calculan usando el generador de Tríos definido en B.5.3.4 de la siguiente manera:

Para cada *i* = 0, ..., *G*-1
(*d*[i], *a*[i], *b*[i] = Trip[*K,X+i*]

Los *G* símbolos de reparación a colocar en el paquete de reparación con ESI *X* se calculan en base a los tríos de símbolos de reparación, según lo descrito en la Sección B.5.3, usando los símbolos intermedios **C** y el codificador de LT LTenc[*K*, **C**, (*d*[i], *a*[i], *b*[i])].

B. 3. 3. Transporte

Esta sección describe el intercambio de información entre el codificador / descodificador MSCR y cualquier protocolo de transporte que hace uso de la corrección anticipada de errores MSCR para la entrega de ficheros.

El codificador y descodificador MSCR para la entrega de ficheros requiere la siguiente información del protocolo de transporte: el tamaño de fichero, *F*, en octetos, el parámetro de alineación de símbolos, *A*, el tamaño de símbolo, *T*, en octetos, que es un múltiplo de *A*, el número de bloques de origen, *Z*, y el número de sub-bloques en cada bloque de origen, *N*. El codificador MSCR para la entrega de ficheros requiere adicionalmente el fichero a codificar, *F* octetos.

El codificador MSCR suministra al protocolo de transporte información de paquetes de codificación, para cada paquete, el SBN, el ESI y el símbolo, o los símbolos, de codificación. El protocolo de transporte podría comunicar esta información de forma transparente al descodificador MSCR.

B. 3. 4. Detalles de ejemplos específicos para parámetros

B. 3. 4. 1 Algoritmo de obtención de parámetros

Esta sección proporciona ejemplos para la obtención de los cuatro parámetros de transporte, *A, T, Z* y *N* que proporcionan buenos resultados. Estos se basan en los siguientes parámetros de entrada:

F	el tamaño de fichero, en octetos
W	un objetivo en el tamaño del sub-bloque, en octetos
P	el máximo tamaño de carga útil del paquete, en octetos, que se supone un múltiplo de *A*
A	el factor de alineación de símbolos, en octetos
K_{MAX}	el máximo número de símbolos de origen por bloque de origen
K_{MIN}	un objetivo mínimo en el número de símbolos por bloque de origen
G_{MAX}	un número máximo deseado de símbolos por paquete

En base a las entradas anteriores, se calculan los parámetros de transporte *T, Z* y *N* de la siguiente manera:

Sean

$$G = \min \{\text{techo}(P \cdot K_{MIN} / F),\ P/A,\ G_{MAX}\} - \text{el número aproximado de símbolos por paquete}$$

$$T = \text{suelo}(P/A \cdot G)) \cdot A$$

$$K_t = \text{techo}(F/T) - \text{el número total de símbolos en el fichero}$$

$$Z = \text{techo}\ (K_t / K_{MAX})$$

$$N = \min\{\text{techo}(\text{techo}(K_t/Z) \cdot T/W),\ T/A\}$$

Los valores de *G* y *N* obtenidos anteriormente deberían ser considerados como cotas inferiores. Puede ser ventajoso aumentar estos valores, por ejemplo, a la más cercana potencia de dos. En particular, el algoritmo anterior no garantiza que el tamaño de símbolo, *T*, divida al máximo tamaño de paquete, *P*, y por tanto puede no ser posible usar los paquetes de tamaño exactamente P. Si, en cambio, se escoge *G* para que sea un valor que divida a *P/A*, entonces el tamaño de símbolo, *T*, será un divisor de P y pueden usarse paquete de tamaño *P*.

Valores adecuados para los parámetros de entrada podrían ser *W* = 256 KB, *A* = 4, K_{MIN} = 4 y G_{MAX} = 1.

B. 3. 4. 2 Ejemplos

El algoritmo anterior lleva a parámetros de transporte según se muestra en la Fig. 18, suponiendo que se usan los valores anteriores para W, A, K_{MIN} y G_{MAX}, y con $P = 512$.

B. 4. Transmisión por flujo

B. 4. 1. Construcción de bloques de origen

Un bloque de origen es construido por el protocolo de transporte, por ejemplo, según lo definido en este documento, haciendo uso del código Sistemático de Corrección Anticipada de Errores MSCR. El tamaño de símbolo, T, a usar para la construcción de bloques de origen y la construcción de símbolos de reparación son proporcionados por el protocolo de transporte. El parámetro T podría fijarse de modo que el número de símbolos de origen en cualquier bloque de origen sea a lo sumo K_{MAX}.

Un ejemplo de parámetros que funcionan bien se presenta en la sección B. 4. 4.

B. 4. 2. Construcción de paquetes de codificación

Según lo descrito en B.4.3., cada paquete de reparación contiene los SBN, ESI, SBL y uno o más símbolos de reparación. El número de símbolos de reparación contenidos dentro de un paquete de reparación se calcula a partir de la longitud del paquete. Los valores de ESI colocados en los paquetes de reparación y los tríos de símbolos de reparación usados para generar los símbolos de reparación se calculan según lo descrito en la Sección B.3.2.2.

B.4.3. Transporte

Esta sección describe el intercambio de información entre el codificador / descodificador MSCR y cualquier protocolo de transporte que hace uso de la corrección anticipada de errores MSCR para la transmisión por flujo. El codificador MSCR para la transmisión por flujo podría usar la siguiente información proveniente del protocolo de transporte para cada bloque de origen: el tamaño de símbolo, T, en octetos, el número de símbolos en el bloque de origen, K, el Número de Bloque de Origen (SBN) y los símbolos de origen a codificar, $K \cdot T$ octetos. El codificador MSCR suministra al protocolo de transporte información de paquetes de codificación que comprende, para cada paquete de reparación, el SBN, el ESI, la SBL y el símbolo, o los símbolos, de reparación. El protocolo de transporte podría comunicar esta información de forma transparente al descodificador MSCR.

B. 4. 4. Selección de parámetros

Puede usarse un cierto número de procedimientos para la selección de parámetros. Algunos de ellos se describen a continuación en detalle.

B. 4. 4. 1 Algoritmo de obtención de parámetros

Esta sección explica una obtención del parámetro de transporte, T, en base a los siguientes parámetros de entrada:

B	el máximo tamaño de bloque de origen, en octetos
$Pmax$	el máximo tamaño de Información de Paquete de Origen, sin relleno
P_r	el tamaño de Información de Paquete de Origen del x-ésimo percentil, sin relleno (es decir, el menor número, n, tal que se espere que el x% de los paquetes tengan Información de Paquete de Origen de tamaño n o menor. En una realización, el valor de x es 30.
A	el factor de alineación de símbolos, en octetos
K_{MAX}	el número máximo de símbolos de origen por bloque de origen
K_{MIN}	un objetivo mínimo en el número de símbolos por bloque de origen
G_{MAX}	un máximo número deseado de símbolos por paquete de reparación

Un requisito para estas entradas es que techo(B/P) $\leq K_{MAX}$. En base a las entradas precedentes, el parámetro de transporte T se calcula de la manera siguiente:

Sea $G = \min\{\max\{\text{techo}(P \cdot K_{MIN}/B), \text{suelo}(P_X/P_{max})\}, P/A, G_{MAX}\}$ – el número de símbolos por SPI

$$T = \text{suelo}(P/(A \cdot G)) \cdot A$$

El valor de T obtenido anteriormente debería ser considerado como una guía para el valor efectivo de T usado. Puede ser ventajoso asegurar que T divida a P, o puede ser ventajoso fijar el valor de T más pequeño para minimizar el desperdicio cuando se usan símbolos de reparación de tamaño completo para recuperar símbolos de

origen parciales al final de paquetes de origen perdidos (mientras el número máximo de símbolos de origen en un bloque de origen no supere K_{MAX}). Además, la elección de T puede depender de la distribución de tamaños de paquete de origen, p. ej., si todos los paquetes de origen tienen el mismo tamaño entonces es ventajoso escoger T de modo que el tamaño efectivo de carga útil de un paquete de reparación P', donde P' es un múltiplo de T, sea igual a (o tan pocos octetos como sea posible mayor que) el número de octetos que cada paquete de origen ocupa en el bloque de origen.

Valores adecuados para los parámetros de entrada podrían ser $A = 16$, $K_{MIN} = 4$ y $G_{MAX} = 4$.

B. 4. 4. 2 Ejemplos

El algoritmo anterior lleva a parámetros de transporte como los mostrados en la Fig. 19, suponiendo los valores anteriores para A, K_{MIN} y G_{MAX}, y suponiendo $P = 1.424$.

B. 5. Codificador sistemático MSCR de múltiples campos

B. 5. 1. Panorama general de la codificación

El codificador sistemático MSCR se usa para generar *símbolos de reparación* a partir de un bloque de origen que comprende *K símbolos de origen*.

Los símbolos son las unidades de datos fundamentales del proceso de codificación y descodificación. Para cada bloque (sub-bloque) de origen, todos los símbolos (sub-símbolos) tienen el mismo tamaño. La operación atómica realizada sobre símbolos (sub-símbolos), tanto para la codificación como para la descodificación, es la operación lógica 'o' exclusivo.

Indiquemos con $C[0], ..., C[K-1]$ los K símbolos de origen.
Indiquemos con $C[0], ..., C[L-1]$ los L símbolos intermedios.

La primera etapa de codificación es generar un número, $L > K$, de símbolos intermedios a partir de los K símbolos de origen. En esta etapa, se generan K tríos de origen $(d[0], a[0], b[0]), ..., (d[K-1], a[K-1], b[K-1])$ usando el generador Trip[], según lo descrito en la Sección B.5.4.4. Los K tríos de origen están asociados a los K símbolos de origen y se usan luego para determinar los L símbolos intermedios $C[0], ..., C[L-1]$ a partir de los símbolos de origen, usando un proceso de codificación inversa. Este proceso puede luego ser realizado por un proceso de descodificación MSCR.

Ciertas "relaciones de pre-codificación", preferiblemente, valen dentro de los L símbolos intermedios. La sección B.5.2. describe estas relaciones y cómo se generan los símbolos intermedios a partir de los símbolos de origen.

Una vez que los símbolos intermedios han sido generados, se producen los símbolos de reparación y uno o más símbolos de reparación se colocan como un grupo en un único paquete de datos. Cada grupo de símbolos de reparación está asociado a un Identificador de Símbolo de Codificación (ESI) y a un número, G, de símbolos de codificación. El ESI se usa para generar un trío de tres enteros, (d, a, b) para cada símbolo de reparación, usando nuevamente el generador Trip[] según se describe en la Sección B.5.4.4. Esto se hace según se describe en las Secciones B.3 y B.4, usando los generadores descritos en la Sección B.5.4. Luego, se usa cada trío (d, a, b) para generar el correspondiente símbolo de reparación a partir de los símbolos intermedios, usando el generador $LTEnc[K, C[0], ..., C[L-1], (d,a,b)]$ descrito en la Sección B.5.4.3.

B.5.2.1 General

La primera etapa de codificación es una etapa de pre-codificación para generar los L símbolos intermedios $C[0], ..., C[L-1]$ a partir de los símbolos de origen $C'[0], ..., C'[K-1]$. Los símbolos intermedios están definidos de manera única por dos conjuntos de restricciones:

1. Los símbolos intermedios están relacionados con los símbolos de origen por un conjunto de *tríos de símbolos de origen*. La generación de los tríos de símbolos de origen está definida en la Sección B.5.2.2, usando el generador Trip[], según lo descrito en la Sección B.5.4.4.

2. Un conjunto de relaciones de pre-codificación vale dentro de los mismos símbolos intermedios.

Estas están definidas en la Sección B.5.2.3. La generación de los L símbolos intermedios está luego definida en la Sección 5.2.4.

B.5.2.2 Tríos de símbolos de origen

Cada uno de los K símbolos de origen está asociado a un trío ($d[i]$, $a[i]$, $b[i]$) para $0 \leq i < K$. Los tríos de símbolos de origen se determinan, usando el generador Triple definido en la Sección B.5.4.4, como:

Para cada i, $0 \leq i < K$
($d[i]$, $a[i]$, $b[i]$) = Trip[K, i]

B.5.2.3 Relaciones de pre-codificación

Las relaciones de pre-codificación entre los L símbolos intermedios se definen expresando los últimos L-K símbolos intermedios en términos de los primeros K símbolos intermedios.

Los últimos L-K símbolos intermedios $C[K]$, ..., $C[L\text{-}1]$ comprenden *S símbolos LDPC* y *H símbolos HDPC*. Los valores de S y H se determinan a partir de K según se describe más adelante. Entonces $L = K + S + H$.

Sean

X	el más pequeño entero positivo tal que $X \cdot (X\text{-}1) >= 2 \cdot K$.
S	el más pequeño entero primo tal que $S \geq$ techo$(0{,}01 \cdot K) + X$
H	el más pequeño entero tal que escoger(H,techo(H/2)) $\geq K + S$
H'	= techo(H/2)
L	$= K + S + H$
$C[0]$, ..., $C[K\text{-}1]$	indicadores de los primeros K símbolos intermedios
C[K], ..., $C[K+S\text{-}1]$	indicadores de los S símbolos LDPC, inicializados con cero
$C[K+S]$, ..., $C[L\text{-}1]$	indicadores de los H símbolos HDPC, inicializados con cero

Se definen los S símbolos LDPC como los valores de $C[K]$, ..., $C[K+S\text{-}1]$ al final del siguiente proceso:

Para $i = 0$, ..., K-1 hacer

$$a = 1 + (\text{suelo}(i/S) \,\%\, (S\text{-}1))$$

$$b = i \,\%\, S$$

$$C[K + b] = C[K + b] \,\hat{}\, C[i]$$

$$b = (b + a) \,\%\, S$$

$$C[K + b] = C[K + b] \,\hat{}\, C[i]$$

$$\text{b} = (\text{b} + \text{a}) \,\%\, \text{S}$$

$$\text{C[K + b]} = \text{C[K + b]} \,\hat{}\, \text{C[i]}$$

Para la construcción de los H símbolos HDPC, el sistema usa el campo GF(256). El campo puede ser representado con respecto al polinomio irreducible $f = x^8 + x^4 + x^3 + x^2 + 1$ sobre el campo GF(2). Indiquemos con α el elemento x módulo f. Como es bien sabido para los medianamente expertos en la técnica, el elemento α es primitivo, es decir, las 255 primeras potencias de α coinciden con los 255 elementos no nulos de GF(256). En una realización, el sistema escoge $K+S$ enteros $a[0]$, ..., $a[K+S\text{-}1]$, e indiquemos con $\beta[0]$, ..., $\beta[K+S\text{-}1]$ los elementos $\alpha^{a[0]}$, ..., $\alpha^{a[K+S-1]}$. Además, escogemos H enteros adicionales $b[0]$, ..., $b[H\text{-}1]$ e indicamos con $\Gamma[0]$, ..., $\Gamma[H\text{-}1]$ los elementos $\alpha^{b[0]}$, ..., $\alpha^{b[H-1]}$. Las realizaciones preferidas adicionales de la presente invención especificarán elecciones específicas para estos enteros. Sin embargo, debería observarse que hay muchas elecciones equivalentes de estos enteros. Sea $g[i] = i \,\hat{}\, (\text{suelo}(i/2)$ para todos los enteros positivos i. Obsérvese que $g[i]$ es la secuencia de Gray, en la cual cada elemento difiere del anterior en una única posición de bit. Además, indiquemos con $g[j, k]$ el j-ésimo elemento, $j = 0$, 1, 2, ..., de la sub-secuencia de $g[i]$ cuyos elementos tienen exactamente k bits no nulos en su representación binaria. Como es bien sabido para los expertos en la técnica, la secuencia $g[j, k]$ tiene la propiedad de que las representaciones binarias de $g[j, k]$ y $g[j+1, k]$ difieren en exactamente dos posiciones. Indicamos estas posiciones con $p[j, k, 1]$ y $p[j, k, 2]$.

Los valores de los símbolos HDPC se definen como los valores de $C[K+S]$, ..., $C[L\text{-}1]$ después del siguiente proceso.

Inicializamos un símbolo U con 0. El tamaño de este símbolo es el mismo que el tamaño común de los símbolos de origen, LDPC y HDPC.

Luego, para una variable *h* que oscila entre 0 y *K*+*S*-2, realizamos lo siguiente: la variable *U* se actualiza como *U* = *U**β[*h*]^*C*[*h*]. Al mismo tiempo, fijamos *C*[*K*+*S*+*p*[*j*,*H'*,1]] = *C*[*K*+*S*+*p*[*j*,*H'*,1]] ^ *U*, y *C*[*K*+*S*+*p*[*j*,*H'*,2]] = *C*[*K*+*S*+*p*[*j*,*H'*,2]] ^ *U*.

En una etapa adicional, transformamos *U* en *U**β[*K*+*S*-1] ^ *C*[*K*+*S*-1].

Luego, para una variable *h* que oscila entre 0 y *H*-1, actualizamos *C*[*K*+*S*+*h*] = *C*[*K*+*S*+*h*] ^ Γ[*h*]**U*. Esto completa la descripción del proceso de codificación HDPC.

En una realización preferida, el sistema escoge los siguientes enteros *a*[0], ..., *a*[*K*+*S*-1], y *b*[0], ..., *b*[*H*-1]: *a*[0]=*a*[1]= ... =*a*[*K*+*S*-1] = 1 y *b*[0]=1, *b*[1]=2, ..., *b*[*i*] = *i*+1, etc. Ventajosamente, en esta realización preferida, la construcción de los símbolos HDPC puede ser realizada usando solamente la acción del elemento primitivo, α, junto con operaciones de OR exclusivo bit a bit entre símbolos. La elección del polinomio irreducible dado anteriormente admite una implementación sumamente eficaz de la acción de α, reduciendo por ello la complejidad de cálculo del algoritmo de construcción de HDPC. Como será evidente a los expertos en la técnica, el algoritmo de construcción descrito anteriormente puede ser fácilmente adaptado para realizar las operaciones de descodificación requeridas dentro de un descodificador de código de múltiples etapas, realizando así también la precitada reducción en la complejidad de cálculo en el descodificador.

B.5.2.4 Símbolos intermedios

B.5.2.4.1 Definición

Dados los *K* símbolos de origen *C*'[0], *C*'[1], ..., *C*'[*K*-1], los L símbolos intermedios *C*[0], *C*[1], ..., *C*[*L*-1] son los valores de símbolos, definidos de forma única, que satisfacen las siguientes condiciones:

1. Los *K* símbolos *de origen C*'[0], *C*'[1], ..., *C*'[*K*-1] satisfacen las *K* restricciones *C*'[*i*] = LTEnc[*K*, (*C*[0], ..., *C*[*L*-1]), (*d*[*i*], *a*[*i*], *b*[*i*])], para todo *i*, 0 ≤ *i* < *K*.

2. Los *L* símbolos intermedios *C*[0], *C*[1], ..., *C*[*L*-1] satisfacen las relaciones de pre-codificación definidas en B.5.2.3.

Esta sub-sección describe un posible procedimiento para el cálculo de los *L* símbolos intermedios *C*[0], *C*[1], ..., *C*[*L*-1] que satisfaga las restricciones en B.5.2.4.1.

La matriz generadora **G**, para un código que genera *N* símbolos de salida a partir de *K* símbolos de entrada, es una matriz de dimensiones *NxK* sobre GF(2), donde cada fila corresponde a uno de los símbolos de salida y cada columna a uno de los símbolos de entrada, y donde el *i*-ésimo símbolo de salida es igual a la suma de aquellos símbolos de entrada cuya columna contiene una entrada no nula en la fila *i*.

Indiquemos con

C el vector columna de los *L* símbolos intermedios, *C*[0], *C*[1], ..., *C*[*L*-1],
D el vector columna que comprende *S*+*H* símbolos cero, seguidos por los *K* símbolos de origen *C*'[0], *C*'[1], ..., *C*'[*K*-1]

Entonces las restricciones anteriores definen una matriz de dimensiones *LxL* sobre GF(2), **A**, tal que:

$$\mathbf{A} \cdot \mathbf{C} = \mathbf{D}$$

La matriz **A** puede ser construida de la siguiente manera:

Sean:

$\mathbf{G}_{LDPC}$ la matriz generadora de dimensiones *S x K* de los símbolos LDPC. Entonces,
$\mathbf{G}_{LDPC}$ $(C[0], ..., C[K-1])^T = (C[K], ..., C[K+S-1])^T$
$\mathbf{G}_{HDPC}$ la matriz generadora de dimensiones *H x (K+S)* de los Semi-símbolos. Entonces,
$\mathbf{G}_{HDPC} \otimes (C[0], ..., C[S+K-1])^T = (C[K+S], ..., C[K+S+H-1])^T$
$\mathbf{I}_S$ la matriz identidad de dimensiones *S x S*
$\mathbf{I}_H$ la matriz identidad de dimensiones *H x H*
$\mathbf{0}_{SxH}$ la matriz cero de dimensiones *S x H*
$\mathbf{G}_{LT}$ la matriz generadora de dimensiones *KxL* de los símbolos de codificación generados por el Codificador LT. Entonces,
$\mathbf{G}_{LT} \cdot (C[0], ..., C[L-1])^T = (C'[0], C'[1], ..., C'[K-1])^T$

es decir, $\mathbf{G}_{LT\,i,j}$ = 1 si y solo si *C*[i] está incluido en los símbolos que son sometidos a la operación lógica XOR para producir LTEnc[*K*, *C*[0], ..., *C*[*L*-1]), (*d*[*i*], *a*[*i*], *b*[*i*])].

Entonces:

Las primeras *S* filas de **A** son iguales a $\mathbf{G}_{LDPC}\,|\mathbf{I}_S|\mathbf{Z}_{SxH}$.
Las siguientes *H* filas de **A** son iguales a $\mathbf{G}_{HDPC}\,|\mathbf{I}_H$.
Las restantes *K* filas de **A** son iguales a $\mathbf{G}_{LT}$.

La matriz A está ilustrada en la Fig. 20. Los símbolos intermedios pueden luego ser calculados como:

$$\mathbf{C = A^{-1} \cdot D}$$

Los tríos de origen se generan de modo que, para cualquier *K*, la matriz **A** tiene rango completo y es, por lo tanto, invertible. Este cálculo puede ser realizado aplicando un proceso de descodificación MSCR a los *K* símbolos de origen *C*'[0], *C*'[1], ..., *C*'[*K*-1] para producir los *L* símbolos intermedios *C*[0], *C*[1], ..., *C*[*L*-1].

Para generar eficazmente los símbolos intermedios a partir de los símbolos de origen, podría usarse una implementación eficaz del descodificador, tal como la descrita en la Sección B.6. Los tríos de símbolos de origen están diseñados para facilitar la descodificación eficaz de los símbolos de origen usando ese algoritmo.

B.5.3. Segunda etapa de codificación: Codificación de reacción en cadena

En la segunda etapa de codificación, el símbolo de reparación con ESI *X* se genera aplicando el generador LTEnc[*K*, (*C*[0], *C*[1], ..., *C*[*L*-1]), (*d, a, b*)] definido en la Sección B.5.4 a los *L* símbolos intermedios *C*[0], *C*[1], ..., *C*[*L*-1], usando el trío (*d, a, b*) = Trip[*K, X*] generado de acuerdo a las Secciones B.3.2.2 y B.4.2.

B.5.4. Generadores

B.5.4.1 Generador aleatorio

El generador de números aleatorios Rand[*X, i, m*] se define de la siguiente manera, donde *X* es un entero no negativo, *i* es un entero no negativo y *m* es un entero positivo, y el valor producido es un entero entre 0 y *m*-1. Sean V_0 y V_1 formaciones de 256 entradas cada una, donde cada entrada es un entero sin signo de 4 octetos. Se proporcionan formaciones adecuadas de números aleatorios en los Apéndices B.1 y B.2, a modo de ejemplo solamente, y que no deberían ser interpretados para limitar el ámbito de la invención Dadas esas hipótesis, Rand[*X, i, m*] = (V_0[(*X*+ *i*) % 256] ^ V_1[(suelo(*X*/256)+ *i*) % 256]) % *m*. Según se usa en la presente memoria, a menos que se indique otra cosa, debería suponerse que “aleatorio” incluye “seudo-aleatorio” y “esencialmente aleatorio”.

B.5.4.2 Generador de grados

El generador de grados Deg[*v*] se define de la siguiente manera, donde *v* es un entero que es al menos 0 y menor que 2^{20} = 1.048.576.

En la Fig. 1, hallar el índice *j* tal que *f*[*j*-1] < *v* < *f*[*j*]

$$\text{Deg}[v] = d[j]$$

B.5.4.3 Generador de símbolos de codificación de reacción en cadena

El generador de símbolos de codificación LTEnc[*K*, (*C*[0], *C*[1], ..., *C*[*L*-1]), (*d, a, b*)] toma las siguientes entradas:

K es el número de símbolos (o sub-símbolos) de origen para el bloque (sub-bloque) de origen. Sea *L* obtenido de *K* según lo descrito en la Sección B.5.2, y sea *L'* el más pequeño entero primo mayor o igual a *L*.

(*C*[0], *C*[1], ..., *C*[*L*-1]) es la formación de L símbolos (sub-símbolos) intermedios generados según lo descrito en la Sección B.5.2.

(*d, a, b*) es un trío de origen determinado usando el generador Triple definido en la Sección B.5.3.4, por lo que *d* es un entero que indica un grado de símbolo de codificación, *a* es un entero entre 1 y *L'*-1 inclusive, y *b* es un entero entre 0 y *L'*-1 inclusive.

El generador de símbolos de codificación produce un único símbolo de codificación como salida, de acuerdo al siguiente algoritmo:

Mientras ($b \geq$ L) hacer b = (b + a) % L'
LTEnc[K, (C[0], C[1], ..., C[L-1]), (d, a, b)] = C[b].
Para j = 1, ..., min (d-1, L-1) hacer
b = (b + a) % L'
Mientras ($b \geq L$) hacer b = (b + a) % L'
LTEnc[K, (C[0], C[1], ..., C[L-1]), (d, a, b)]) $LTEne$[K, (C[0], C[1], ..., C[L-1]), (d, a, b)] ^ C[b]

B.5.4.4 Generador de tríos

El generador de tríos Trip[K, X] toma las siguientes entradas:

K El número de símbolos de origen
X Un Identificador de símbolo de codificación

Sean

L determinado a partir de K, según lo descrito en la Sección B.5.2
L' el primo más pequeño que sea mayor o igual a L
Q = 65.521, el mayor primo más pequeño que 2^{16}
$J(K)$ el índice sistemático asociado a K. El índice sistemático es un número escogido de modo que el proceso a continuación, junto con lo restante procesado para la construcción de la matriz **A** descrita en la presente memoria, dé como resultado una matriz B que sea invertible. Se proporcionan índices sistemáticos adecuados en el Apéndice A, a modo de ejemplo solamente, y no deberían ser interpretados para limitar el ámbito de la invención.

La salida del generador de tríos es un trío (d, a, b) determinado de la siguiente manera:

1. A = (53.591 + $J(K)$·997) % Q
2. B = 10.267·($J(K)$+1) % Q
3. Y = (B + X·A) % Q
4. v = Rand[Y, 0, 2^{20}]
5. d = Deg[v]
6. a = 1 + Rand[Y, 1, L'-1]
7. b = Rand[Y, 2, L']

B. 6 Implementaciones del descodificador de FEC

B. 6. 1 General

Esta sección describe un eficaz algoritmo de descodificación para los códigos MSCR descritos en esta especificación. Obsérvese que cada símbolo de codificación recibido puede ser considerado como el valor de una ecuación entre los símbolos intermedios. A partir de estas ecuaciones simultáneas, y las relaciones de pre-codificación conocidas entre los símbolos intermedios, cualquier algoritmo para resolver ecuaciones simultáneas puede descodificar con éxito los símbolos intermedios y por tanto los símbolos de origen. Sin embargo, el algoritmo escogido tiene un efecto mayor sobre la eficacia de cálculo de la descodificación.

B. 6. 2 Descodificación de un bloque de origen

B.6.2.1 General

Se supone que el descodificador conoce la estructura del bloque de origen que ha de descodificar, incluyendo el tamaño de símbolo, T, y el número K de símbolos en el bloque de origen.

A partir de los algoritmos descritos en las Secciones B.5, el descodificador MSCR puede calcular el número total L = K+S+H de símbolos de pre-codificación y determinar cómo fueron generados a partir del bloque de origen a descodificar. En esta descripción se supone que los símbolos de codificación recibidos para el bloque de origen a descodificar son pasados al descodificador. Además, para cada símbolo de codificación de ese tipo se supone que el número y el conjunto de símbolos intermedios, cuyo 'o' exclusivo sea igual al símbolo de codificación, son pasados al descodificador. En el caso de símbolos de origen, los tríos de símbolos de origen descritos en la Sección B.5.2.2 indican el número y el conjunto de símbolos intermedios que se suman para dar cada símbolo de origen.

Sea $N \geq K$ el número de símbolos de codificación recibidos para un bloque de origen y sea $M = S+H+N$. La siguiente matriz **A** de dimensiones $M \times L$ puede ser obtenida a partir de la información pasada al descodificador para el bloque de origen a descodificar. Sea **C** el vector columna de los L símbolos intermedios, y sea **D** el vector columna de M símbolos con valores conocidos para el receptor, donde los últimos $S+H$ de los M símbolos son símbolos de valor cero que corresponden a símbolos LDPC y HDPC (estos son símbolos de control para los símbolos de LDPC y HDPC, y no los símbolos LDPC y HDPC en sí mismos), y los restantes N de los M símbolos son los símbolos de codificación recibidos para el bloque de origen. Entonces, **A** es la matriz que satisface $\mathbf{A} \cdot \mathbf{C} = \mathbf{D}$, donde aquí · indica la multiplicación matricial sobre GF(256). La matriz **A** tiene una estructura de bloques, según se muestra en la Figura 23. La estructura de bloques comprende una matriz **F** con N filas y L columnas, una matriz **E** con S filas y $L-S-H$ columnas, una matriz identidad **I** de S por S, una matriz **O** con S filas y H columnas que son todas cero, una matriz **B** con H filas y $L-H$ columnas, y una matriz identidad **J** de H por H. La sub-matriz **B** tiene entradas definidas sobre el campo GF(256), mientras que las matrices **E** y **F** tienen entradas 0 / 1, es decir, entradas en el campo GF(2). La matriz **F** define el proceso de codificación dinámica, la matriz **E** define el proceso de codificación de LDPC descrito anteriormente, y la matriz **B** define el proceso de codificación de HDPC. En particular, $\mathbf{F}[i,j] = 1$ si el símbolo intermedio correspondiente al índice j es sometido a una operación lógica de 'O' exclusivo con el símbolo de codificación correspondiente al índice i en la codificación. Para todo otro i y j, $\mathbf{F}[i,j] = 0$. De manera similar, $\mathbf{E}[i,j] = 1$ si el símbolo intermedio correspondiente al índice j es sometido a una operación lógica de 'O' exclusivo con el símbolo de LDPC correspondiente al índice i. Finalmente, $\mathbf{B}[i,j] = \beta$ si el resultado de la acción de β sobre los símbolos intermedios correspondientes al índice j es sometido a una operación lógica de 'O' exclusivo en el símbolo de HDPC correspondiente al índice i.

La descodificación de un bloque de origen es equivalente a la descodificación de **C** a partir de **A** y **D** conocidas. Es claro que **C** puede ser descodificada si y solo si el rango de **A** sobre GF(256) es L. Una vez que **C** ha sido descodificada, los símbolos de origen faltantes pueden ser obtenidos usando los tríos de símbolos de origen para determinar el número y el conjunto de símbolos intermedios que son sometidos a la operación lógica 'O' exclusivo para obtener cada símbolo de origen faltante.

La primera etapa en la descodificación de **C** es formar una planificación de descodificación. En esta etapa **A** es convertida, usando la eliminación Gaussiana (usando operaciones de fila y reordenamientos de fila y columna) y después de descartar $M - L$ filas, en la matriz identidad de L por L. La planificación de descodificación comprende la secuencia de operaciones de fila y de reordenamientos de fila y columna durante el proceso de eliminación Gaussiana, y solamente depende de **A** y no de **D**. La descodificación de C a partir de **D** puede tener lugar simultáneamente con la formación de la planificación de descodificación, o la descodificación puede tener lugar después, en base a la planificación de descodificación.

La correspondencia entre la planificación de descodificación y la descodificación de C es la siguiente. Sea $c[0] = 0$, $c[1] = 1$, ..., $c[L-1] = L-1$ y $d[0] = 0$, $d[1] = 1$, ..., $d[M-1] = M-1$ inicialmente.

Cada vez que la fila i de **A** es sometida a una operación lógica de 'O' exclusivo con la fila i' en la planificación de descodificación, entonces, en el proceso de descodificación, el símbolo $D[d[i]]$ es sometido a una operación lógica de 'O' exclusivo con el símbolo $D[d[i']]$. Llamamos a esta operación una operación de fila de GF(2).

Cada vez que un múltiplo α (para algún α en GF(256)) de la fila i de **A** es sometido a una operación lógica de 'O' exclusivo con la fila i' en la planificación de descodificación, entonces, en el proceso de descodificación, el símbolo $\alpha * D[d[i]]$ es sometido a una operación lógica de 'O' exclusivo con el símbolo $D[d[i']]$. Llamamos a esta operación una operación de fila de GF(256). Obsérvese que una operación de fila de GF(2) es un caso particular de una operación de fila de GF(256) en la cual el elemento α es 1.

Cada vez que la fila i es intercambiada con la fila i' en la planificación de descodificación, entonces, en el proceso de descodificación el valor de $d[i]$ es intercambiado con el valor de $d[i']$.

Cada vez que la columna j es intercambiada con la columna j' en la planificación de descodificación, entonces, en el proceso de descodificación el valor de $c[j]$ es intercambiado con el valor de $c[j']$.

A partir de esta correspondencia, es claro que el número total de operaciones lógicas de 'O' exclusivo de símbolos en la descodificación del bloque de origen está relacionado con el número de operaciones de fila (no intercambios) en la eliminación Gaussiana. Dado que A es la matriz identidad de L por L después de la eliminación Gaussiana y después de descartar las últimas $M - L$ filas, es claro, al final de la descodificación con éxito, que los L símbolos $D[d[0]]$, $D[d[1]]$, ..., $D[d[L-1]]$ son los valores de los L símbolos $C[c[0]]$, $C[c[1]]$, ..., $C[c[L-1]]$.

El orden en el cual se realiza la eliminación Gaussiana para formar la planificación de descodificación no tiene relevancia sobre si la descodificación es exitosa o no. Sin embargo, la velocidad de la descodificación depende sumamente del orden en el cual se realiza la eliminación Gaussiana. (Además, es crucial mantener una representación rala de **A**, aunque esto no se describe aquí). También es claro que es más eficaz realizar

operaciones de fila de GF(2), antes que operaciones de fila de GF(256). Por lo tanto, al realizar la eliminación Gaussiana, es mejor pivotar sobre las filas de la matriz **A** que tienen elementos tomados del campo GF(2). También es ventajoso dejar la eliminación de las filas de la matriz correspondiente a los símbolos HDPC para el final del proceso de eliminación Gaussiana. El resto de esta sección describe un orden en el cual la eliminación Gaussiana podría ser realizada, y que es relativamente eficaz.

B.6.2.2 Primera fase

Con referencia a la Figura 23, indicamos con **X** la matriz que comprende **F, E, I** y **0** según lo ilustrado en la Figura 24a.

En la primera fase de la eliminación Gaussiana la matriz **X** es conceptualmente dividida en sub-matrices. Los tamaños de sub-matrices están parametrizados por enteros no negativos *i* y *u*, que se inicializan con 0. Las sub-matrices de **X** son:

(1) La sub-matriz definida por la intersección de las primeras *i* filas y las primeras *i* columnas. Esta es la matriz identidad al final de cada etapa en la fase.

(2) La sub-matriz definida por la intersección de las primeras *i* filas y todas menos las primeras *i* columnas y últimas *u* columnas. Todas las entradas de esta sub-matriz son cero.

(3) La sub-matriz definida por la intersección de las primeras *i* columnas y todas menos las primeras *i* filas. Todas las entradas de esta sub-matriz son cero.

(4) La sub-matriz **U** definida por la intersección de todas las filas y las últimas *u* columnas.

(5) La sub-matriz **V** formada por la intersección de todas menos las primeras *i* columnas y las últimas *u* columnas de todas menos las primeras *i* filas.

La Fig. 22 ilustra las sub-matrices de **X**. En el comienzo de la primera fase **V = X**. En cada etapa, se escoge una fila de **X**. El siguiente gráfico definido por la estructura de **V** se usa en la determinación de cuál fila de **X** se escoge. Las columnas que intersecan a **V** son los nodos en el gráfico, y las filas que tienen exactamente 2 unos en **V** son los bordes de un gráfico que conectan las dos columnas (nodos) en las posiciones de los dos unos. Un componente de este gráfico es un conjunto máximo de nodos (columnas) y bordes (filas) tal que haya un trayecto entre cada par de nodos / bordes en el gráfico. El tamaño de un componente es el número de nodos (columnas) en el componente. El gráfico está indicado por *Y* en lo siguiente.

Hay a lo sumo *L* etapas en la primera fase. La fase acaba cuando **V** desaparece, o bien se convierte en la matriz nula. En cada etapa, se escoge una fila de **X** de la siguiente manera:

Si todas las entradas de **V** son cero, entonces no se escoge ninguna fila y la primera fase acaba.

En otro caso, sea *r* el mínimo entero tal que al menos una fila de **X** tiene exactamente *r* unos en **V**.

Si $r = 1$, entonces escoger la fila con exactamente un 1 en **V**.

Si $r = 2$, entonces escoger cualquier fila con exactamente 2 unos en **V** que sea parte de un componente de tamaño máximo en el gráfico definido por *Y*.

Si $r > 2$, entonces escoger una fila con exactamente *r* unos en **V**, con ponderación original mínima entre todas tales filas.

Después de que se escoge la fila en esta etapa, la primera fila de **X** que interseca a **V** es intercambiada con la fila escogida, de modo que la fila escogida sea la primera fila que interseca a **V**. Las columnas de **X** entre las que intersecan a **V** son reordenadas de modo que uno de los *r* unos en la fila escogida aparezca en la primera columna de **V** y de modo que los restantes $r-1$ unos aparezcan en las últimas columnas de **V**. Luego, la fila escogida es sometida a una operación lógica de 'O' exclusivo con todas las otras filas de **X** debajo de la fila escogida que tengan un uno en la primera columna de **V**. En otras palabras, realizamos una operación de fila de GF(2) en esta etapa. Finalmente, se incrementa *i* en 1 y se incrementa *u* en $r-1$, lo que completa la etapa.

Indiquemos con *v* el número de columnas de la matriz **V** al final de esta fase. Después de permutar las columnas de la matriz **B** de modo que las columnas de **V** correspondan a las últimas *v* columnas de **X**, la matriz **X** tendrá la forma dada en la Figura 24b.

B.6.2.3 Segunda fase

Modificamos la matriz **U** para que comprenda adicionalmente las últimas *v* filas de la matriz **X**, y reemplazamos *u*, en consecuencia, por *u+v*. La sub-matriz **U** es dividida adicionalmente en las primeras *i* filas, $\mathbf{U}_{superior}$, y las restantes *N+S-i* filas, $\mathbf{U}_{inferior}$, según lo ilustrado en la Fig. 25. La eliminación Gaussiana se realiza en la segunda fase en $\mathbf{U}_{inferior}$. Después de esta etapa, la matriz $\mathbf{U}_{inferior}$ tendrá la forma dada en la Fig. 26, es decir, después de una permutación de las filas y columnas, la intersección de las primeras *s* filas con las primeras *s* columnas es una matriz identidad, llamada **I**, las últimas *m* filas son cero y la intersección de las primeras *s* filas con las últimas *u-s* columnas forma la matriz **W**. Obsérvese que *s+m* es igual al número *N+S-i* de filas de la matriz $\mathbf{U}_{inferior}$. Si el valor de *s* es *u*, entonces puede omitirse la próxima fase. Si el valor de *m* es mayor que *H-v*, entonces se devuelve un error de descodificación, dado que el rango de la matriz **A** es menor que *L* en este caso. Las últimas *m* filas de la matriz **X** son descartadas, de modo que, después de esta fase, **A** tiene la forma dada en la Fig. 27. En esta figura, $\mathbf{B}_1$, ..., $\mathbf{B}_3$ son matrices con *H* filas cada una y entradas en GF(256). Luego, se realizan operaciones de fila de GF(256) en las matrices $\mathbf{B}_1$ y $\mathbf{B}_2$ para anularlas. Esto puede hacerse de una entre dos formas. En un primer procedimiento, las primeras *i* filas de **A** se usan para anular la matriz $\mathbf{B}_1$, por medio de operaciones de fila de GF(256). Las siguientes *s* filas de **A** son luego usadas para anular la matriz $\mathbf{B}_2$. En un segundo procedimiento, las filas *i* a *i+s-l* inclusive son usadas para anular las primeras *s* columnas de $\mathbf{U}_{superior}$, por medio de operaciones de fila de GF(2), y luego las primeras *i+s* filas de **X** se usan para anular tanto $\mathbf{B}_1$ como $\mathbf{B}_2$, por medio de operaciones de fila de GF(256). Como es evidente para los medianamente expertos en la técnica, el algoritmo del procedimiento descrito anteriormente para la construcción de los símbolos de HDPC lleva a un algoritmo similar para anular la matriz $\mathbf{B}_1$ (en el primer procedimiento), o tanto $\mathbf{B}_1$ como $\mathbf{B}_2$ (en el segundo procedimiento). Este algoritmo requiere el cálculo de la acción de un elemento de GF(256) sobre un símbolo, solamente una vez por columna matricial, más una vez por fila de **H**. Por tanto, el segundo procedimiento descrito anteriormente da como resultado menos operaciones, en general, para anular las matrices $\mathbf{B}_1$ y $\mathbf{B}_2$.

Después de esta etapa, la matriz **A** tiene la forma dada en la Fig. 28. La matriz **T** tiene *H* filas y *u-s* columnas. La eliminación Gaussiana se realiza sobre la matriz **T** para transformarla en una matriz identidad, seguida por *H-u+s* filas. Si esto no es posible, es decir, si el rango de **T** es más pequeño que *u-s*, entonces se indica un error de descodificación. Al final de esta etapa la matriz **A** tiene la forma dada en la Fig. 29, después de descartar las últimas *H-u+s* filas. En esta figura, **I** indica una matriz identidad de *s* por *s*, y **J** indica una matriz identidad de *u-s* por *u-s*.

B.6.2.4 Tercera fase

Después de la segunda fase, las partes de **A** que necesitan ser anuladas para acabar la conversión de **A** en la matriz identidad de *L* por *L* son W y todas las *u* columnas de $\mathbf{U}_{superior}$, en el caso en que haya sido seguido el primer procedimiento de anulación de $\mathbf{B}_1$ y $\mathbf{B}_2$, o **W** y las últimas *u-s* columnas de $\mathbf{U}_{superior}$, en el caso en que haya sido seguido el segundo procedimiento de anulación de $\mathbf{B}_1$ y $\mathbf{B}_2$. En el primer caso, dado que la matriz **W** es generalmente de tamaño pequeño, puede ser anulada usando operaciones elementales de fila de GF(2). Después de esta etapa, la matriz **A** tiene la forma dada en la Fig. 30. En ambos casos, la parte restante de la matriz a anular es ahora rectangular. En el primer caso tiene un tamaño de *i* filas y *u* columnas, en el segundo caso tiene un tamaño de *i+s* filas y *u-s* columnas. En lo que sigue usaremos *i'* para el número de filas en esta matriz y *u'* para el número de columnas, e indicaremos la matriz con $\hat{\mathbf{U}}$.

El número de filas *i'* de la sub-matriz restante $\hat{\mathbf{U}}$ es generalmente mucho mayor que el número de columnas *u'*. Hay varios procedimientos que pueden usarse para anular $\hat{\mathbf{U}}$ eficazmente. En un procedimiento, se calcula la siguiente matriz **U'** de pre-cálculo, en base a las últimas u filas y columnas de **A**, que indicamos con $\mathbf{I}_u$, y luego **U'** se usa para anular $\hat{\mathbf{U}}$. Las *u* filas de $\mathbf{I}_u$ se dividen en techo(*u/z*) grupos de z filas cada uno, para algún entero z. Luego, para cada grupo de z filas se calculan todas las combinaciones no nulas de las z filas, dando como resultado 2^z-1 filas (esto puede hacerse con 2^z-z-1 operaciones lógicas de 'O' exclusivo por grupo, dado que las combinaciones de peso uno de Hamming que aparecen en $\mathbf{I}_u$ no necesitan ser recalculadas). Por tanto, la matriz **U'** de pre-cálculo resultante tiene techo(*u/z*)·2^z-1 filas y *u* columnas. Obsérvese que **U'** no es formalmente una parte de la matriz **A**, pero será usada posteriormente para anular $\mathbf{U}_{superior}$. En una realización preferida, z=8.

Para cada una de las *i'* filas de $\hat{\mathbf{U}}$, para cada grupo de z columnas en la sub-matriz $\hat{\mathbf{U}}$ de esta fila, si el conjunto de z entradas de columna en $\hat{\mathbf{U}}$ no son todas cero, entonces la fila de la matriz de pre-cálculo **U'** que coincide con el patrón en las z columnas es sometida a una operación lógica de 'O' exclusivo con la fila, anulando de tal modo aquellas z columnas en la fila, con el coste de someter a la operación lógica de 'O' exclusivo una fila de **U'** con la fila.

Después de esta fase **A** es la matriz identidad de *L* por *L* y se ha formado con éxito una planificación completa de descodificación. Entonces, puede ser ejecutada la correspondiente descodificación, que comprende someter a la operación lógica 'O' exclusivo los símbolos de codificación conocidos, para recuperar los símbolos intermedios en base a la planificación de descodificación.

Los tríos asociados a todos los símbolos de origen se calculan de acuerdo a B.5.2.2. Los tríos para los símbolos de origen recibidos se usan en la descodificación. Los tríos para los símbolos de origen faltantes se usan para determinar cuáles símbolos intermedios necesitan ser sometidos a la operación lógica 'O' exclusivo para recuperar los símbolos de origen faltantes.

Codificadores / descodificadores de reacción en cadena de etapa única y múltiples campos

Los códigos de etapa única y múltiples campos (MFSS) tienen propiedades útiles que se divulgan o sugieren en la presente memoria. Se describen en la presente memoria disposiciones novedosas para códigos MFSS, codificadores y descodificadores. En una realización, los datos son codificados para su transmisión desde un origen a un destino, en la cual cada símbolo de salida es generado como una combinación lineal de uno o más de los símbolos de entrada, con coeficientes tomados de campos finitos y, para cada símbolo de salida:

- selección de acuerdo a un proceso aleatorio de un entero mayor que cero, *d*, conocido como el grado del símbolo de salida,

- selección de acuerdo a un proceso aleatorio de un conjunto de tamaño *d* de símbolos de entrada, siendo conocido este conjunto de símbolos de entrada como el conjunto vecino del símbolo de salida,

- selección de un conjunto de campos finitos, de modo que, para al menos un símbolo de salida, este conjunto contenga al menos dos campos finitos,

- selección, para cada símbolo de entrada en el conjunto vecino del símbolo de salida, de un campo finito a partir del conjunto seleccionado de posibles campos finitos,

- selección, para cada símbolo de entrada en el conjunto vecino del símbolo de salida, de acuerdo a un proceso aleatorio, de un elemento no nulo del campo finito seleccionado anteriormente.

El proceso aleatorio para seleccionar los grados de los símbolos de salida puede ser un proceso descrito en Luby I y Luby II, en el cual el grado se selecciona de acuerdo a una distribución de grados. El proceso aleatorio para seleccionar los símbolos de entrada, para asociar a cada símbolo de salida, puede ser un proceso descrito en Luby I y Luby II, en el cual los símbolos de entrada se seleccionan aleatoriamente y uniformemente. Según se usa en la presente memoria, "aleatorio" puede incluir "seudo-aleatorio", "aleatorio sesgado" y similares.

El conjunto de posibles campos finitos puede ser el conjunto { GF(2), GF(256) }.

El proceso para seleccionar el campo finito puede estar basado en un parámetro d_j, de modo que, para símbolos de salida de grado menor que d_j, el campo GF(2) se escoge para todos los símbolos de entrada en el conjunto vecino del símbolo de salida, y para símbolos de salida de grado d_j o mayor, entonces se escoge el campo GF(256) para al menos uno de, algunos de o todos, los miembros del conjunto vecino del símbolo de salida y el campo GF(2) se escoge para los elementos restantes del conjunto vecino, si los hubiera.

El proceso para seleccionar el elemento de campo finito en el campo seleccionado puede ser el sencillo proceso aleatorio en el cual se escoge un elemento uniformemente al azar entre los elementos no nulos del campo.

Un descodificador que recibe datos codificados por un codificador MFSS, según lo descrito anteriormente, podría descodificar los símbolos de salida para regenerar los símbolos de entrada, formando una representación matricial del código de acuerdo al procedimiento descrito anteriormente, sin incluir esta matriz ninguna fila estática, e incluyendo una fila dinámica para cada símbolo de salida del código, y aplicando luego la Eliminación Gaussiana para hallar la inversa de la matriz, asegurando que en cada etapa del proceso de Eliminación Gaussiana se escojan filas pivote de grado mínimo.

Como quedará claro para los medianamente expertos en la técnica, muchas de las propiedades bien conocidas de los códigos descritos en Luby I y Luby II son igualmente aplicables a los códigos descritos anteriormente y, en particular, la elección de una distribución de grados adecuada puede asegurar que, con alta probabilidad, el proceso de Eliminación Gaussiana es capaz de identificar una fila de grado restante uno y, por tanto, el proceso de descodificación funciona como un proceso de reacción en cadena, según lo descrito en Luby I y Luby II.

Este código de MFSS tiene varias ventajas adicionales sobre los códigos conocidos en la técnica. En primer lugar, la inclusión de elementos del campo GF(256) reduce significativamente la probabilidad de que cualquier símbolo de salida recibido dado no sea aditivo en información con respecto a los símbolos de salida previamente recibidos. Como resultado, la probabilidad de errores de descodificación de este código es mucho menor que los códigos

anteriores. Por ejemplo, en algunos casos, la probabilidad de fallo de los códigos descritos en Luby I y Luby II es mejorada.

Una ventaja de este código sobre otros códigos basados en grandes campos es que los símbolos de salida de bajo grado serán generalmente procesador primero por el proceso de Eliminación Gaussiana y, como resultado, la inclusión de elementos de GF(256) no necesita ser considerada hasta más tarde en el proceso de descodificación. Dado que las operaciones sobre GF(256) son relativamente caras en comparación con las operaciones sobre GF(2), esto da como resultado una complejidad de cálculo sumamente reducida, en comparación con códigos donde muchos de, o todos, los símbolos se construyen usando elementos de GF(256) u otros campos finitos grandes.

Una ventaja adicional sobre otros códigos basados en grandes campos es que, para aquellos símbolos de salida generados usando el campo más grande, solamente un elemento del conjunto vecino tiene un coeficiente que está tomado del campo más grandes y, como resultado, solamente se requiere una operación entre un símbolo y un elemento de campo finito para cada símbolo de salida de ese tipo. Esto da como resultado una baja complejidad de cálculo global.

Se sabe que el uso de códigos internos y códigos externos para codificar símbolos de entrada, usando dos (o más) procedimientos de codificación, lleva a un sencillo esquema de código que brinda beneficios a menudo hallados en códigos más complejos. Con el uso de códigos internos y códigos externos, los símbolos de origen son codificados primero usando uno de los códigos y la salida del primer codificador se proporciona a un codificador que codifica de acuerdo al otro código, y ese resultado es emitido como los símbolos de salida. El uso de un MFSS, por supuesto, es distinto al uso de códigos internos / externos. Por una parte, los símbolos de salida son obtenidos de conjuntos vecinos de códigos de entrada. En muchas de las realizaciones descritas en la presente memoria, cada símbolo de salida es una combinación lineal de símbolos de entrada. Con códigos de múltiples etapas, cada símbolo de salida podría ser una combinación lineal de símbolos de entrada y / o símbolos redundantes y / o intermedios.

Códigos densos de múltiples campos y codificadores / descodificadores para tales códigos

En una variación de las revelaciones descritas anteriormente, la representación matricial del código es una matriz densa. Como es bien sabido, los códigos de corrección de errores pueden ser construidos a partir de matrices densas sobre campos finitos. Por ejemplo, puede construirse una matriz generalizada en la cual no han ninguna fila estática y cada fila dinámica comprende elementos de $GF(2^q)$, con cada elemento escogido al azar. Puede construirse luego un código de tasa fija en el cual cada símbolo de salida corresponde a una de las filas dinámicas, y se genera como la combinación lineal de aquellos símbolos de entrada para los cuales hay un elemento no nulo en la correspondiente columna de esta fila de la matriz, usando estos elementos como coeficientes en el proceso de combinación lineal.

Es bien sabido para los expertos en la técnica que la probabilidad de que una matriz escogida al azar con K filas y $K+A$ columnas, con coeficientes que son escogidos independientemente y aleatoriamente en $GF(2^q)$, tenga un rango que sea más pequeño que K, es a lo sumo 2^{-qA}. Por lo tanto, la probabilidad de errores de descodificación de un código con K símbolos de entrada y K/R símbolos de salida, en el cual los símbolos de salida son generados independientemente y aleatoriamente a partir de los símbolos de entrada, usando coeficientes escogidos al azar de $GF(2^q)$ es a lo sumo 2^{-qA}, si el número de símbolos codificados recibidos es $K+A$.

En el caso de $q = 1$, el código descrito anteriormente tiene la ventaja de una complejidad de cálculo razonable, dado que todas las operaciones están dentro del campo GF(2) y por tanto corresponden a operaciones XOR convencionales. Sin embargo, en este caso la cota inferior de la probabilidad de fallo de 2^{-A}, una vez que han sido recibidos A símbolos adicionales, es mucho más alta que lo deseable.

En el caso de q=8, el código descrito anteriormente tiene la ventaja de una menor probabilidad de fallo (acotada por 2^{-8A} para A símbolos adicionales recibidos). Sin embargo, en este caso todas las operaciones están dentro del campo GF(256) y por tanto son relativamente caras en términos de cálculo.

Una realización adicional permite que se consigan probabilidades de errores de descodificación cercanas a las conseguibles usando grandes valores de q, con complejidad de cálculo cercana a la conseguible con valores pequeños de q. En esta realización, los símbolos de salida son generados como combinaciones lineales de símbolos de entrada, con coeficientes tomados bien de $GF(2^p)$ o bien de $GF(2^q)$, donde $p < q$. En una realización específica, se generan exactamente $(K-2p/q)/R$ símbolos de salida usando coeficientes de $GF(2^p)$ y los restantes $2p/(qR)$ símbolos de salida se generan usando coeficientes de $GF(2^q)$.

Los datos recibidos en un destino pueden ser descodificados determinando las relaciones lineales entre los símbolos de salida recibidos y los símbolos de entrada del código, y resolviendo este conjunto de relaciones lineales para determinar los símbolos de entrada.

La probabilidad de errores de descodificación de este código es a lo sumo la del código en el cual todos los coeficientes se escogen del campo $GF(2^p)$ y puede ser significativamente menor, según el número de símbolos generados usando coeficientes del campo mayor $GF(2^q)$. Sin embargo, dado que la mayoría de los símbolos de salida son generados usando coeficientes de $GF(2^p)$, la complejidad de cálculo de la codificación es solo levemente mayor que la de un código en el cual todos los símbolos son generados usando coeficientes de $GF(2^p)$. Además, el procedimiento de descodificación puede ser dispuesto de modo que los símbolos generados con coeficientes de $GF(2^p)$ sean procesados primero y por tanto la mayoría de las operaciones de descodificación sean realizadas con operaciones exclusivamente en $GF(2^p)$. Como resultado, la complejidad de cálculo del procedimiento de descodificación es análogamente cercana a la de los códigos construidos usando solamente $GF(2^p)$. En una realización preferida específica, p=1 y q=8.

Algunas propiedades de algunos códigos de múltiples campos

En la mayoría de los ejemplos descritos anteriormente, los símbolos de entrada y de salida se codifican para el mismo número de bits y cada símbolo de salida se coloca en un paquete (siendo un paquete una unidad de transporte que bien es recibida en su totalidad o bien se pierde en su totalidad). En algunas realizaciones, el sistema de comunicación es modificado de modo que cada paquete contenga varios símbolos de salida. El tamaño de un valor de símbolo de salida se fija luego en un tamaño determinado por el tamaño de los valores de símbolos de entrada en la división inicial del fichero o los bloques del flujo en símbolos de entrada, en base a un cierto número de factores. El proceso de descodificación permanece esencialmente sin cambios, excepto porque los símbolos de salida llegan en manojos según es recibido cada paquete.

La determinación de los tamaños de los símbolos de entrada y de los símbolos de salida está usualmente dictada por el tamaño del fichero o bloque del flujo, y el sistema de comunicación por el cual han de transmitirse los símbolos de salida. Por ejemplo, si un sistema de comunicación agrupa bits de datos en paquetes de un tamaño definido, o agrupa bits de otras maneras, el diseño de los tamaños de símbolos comienza con el tamaño del paquete o del agrupamiento. A partir de allí, un diseñador determinará cuándos símbolos de salida serán llevados en un paquete o grupo, y eso determina el tamaño del símbolo de salida. Para mayor simplicidad, el diseñador fijaría probablemente el tamaño del símbolo de entrada igual al tamaño del símbolo de salida, pero si los datos de entrada componen un tamaño distinto de símbolo de entrada, más conveniente, puede usarse.

El proceso de codificación descrito en lo que antecede produce un flujo de paquetes que contiene símbolos de salida basados en el fichero o bloque original del flujo. Cada símbolo de salida en el flujo es generado independientemente de todos los otros símbolos de salida, y no hay ninguna cota inferior o superior para el número de símbolos de salida que pueden ser creados. Se asocia una clave a cada símbolo de salida. Esa clave, y algunos contenidos del fichero de entrada o el bloque del flujo, determinan el valor del símbolo de salida. Los símbolos de salida consecutivamente generados no necesitan tener claves consecutivas, y en algunas aplicaciones sería preferible generar aleatoriamente la secuencia de claves, o generar de manera seudo-aleatoria la secuencia.

La descodificación de múltiples etapas tiene una propiedad de que un bloque de K símbolos de entrada de igual tamaño puede ser recuperado a partir de K+A símbolos de salida en promedio, con muy alta probabilidad, donde A es pequeño en comparación con K. Por ejemplo, en la realización preferida descrita en primer lugar anteriormente, cuando K = 100, la Fig. 31 muestra la probabilidad de no lograr descodificar a partir de K+A símbolos de salida, escogidos aleatoriamente entre los primeros 120 símbolos de salida generados, y la tabla de la Fig. 32 muestra la probabilidad de no lograr descodificar a partir de K+A símbolos de salida escogidos aleatoriamente entre los primeros 110 símbolos de salida generados.

Dado que los símbolos de salida específicos son generados en un orden aleatorio o seudo-aleatorio, y que la pérdida de símbolos de salida específicos en tránsito está generalmente desvinculada de los valores de los símbolos, hay solamente una pequeña varianza en el número efectivo de símbolos de salida necesarios para recuperar el fichero o bloque de entrada. En muchos casos, donde una colección específica de K+A símbolos de salida no es suficiente para descodificar un bloque, el bloque es aún recuperable si el receptor puede recibir más símbolos de salida desde uno o más orígenes.

Debido a que el número de símbolos de salida está solamente limitado por la resolución de I, pueden ser generados bastantes más de K+A símbolos de salida. Por ejemplo, si I es un número de 32 bits, podrían ser generados 4 mil millones de símbolos de salida distintos, mientras que el fichero o bloque del flujo podría incluir K=50.000 símbolos de entrada. En algunas aplicaciones, solamente un pequeño número de esos 4 mil millones de símbolos de salida pueden ser generados y transmitidos, y es casi una certeza que un fichero de entrada o bloque de un flujo puede ser recuperado con una fracción muy pequeña de los posibles símbolos de salida, y una excelente probabilidad de que el fichero o bloque de entrada pueda ser recuperado con poco más de K símbolos de salida (suponiendo que el tamaño del símbolo de entrada es el mismo que el tamaño del símbolo de salida).

En algunas aplicaciones, puede ser aceptable no poder descodificar todos los símbolos de entrada, o poder descodificar todos los símbolos de entrada, pero con una probabilidad relativamente baja. En tales aplicaciones, un receptor puede dejar de intentar de descodificar todos los símbolos de entrada después de recibir K+A símbolos de salida. O bien, el receptor puede dejar de recibir símbolos de salida después de recibir menos de K+A símbolos de salida. En algunas aplicaciones, el receptor puede incluso recibir solamente K o menos símbolos de salida. Por tanto, ha de entenderse que en algunas realizaciones de la presente invención el grado deseado de precisión no necesariamente debe ser la recuperación completa de todos los símbolos de entrada.

Además, en algunas aplicaciones donde es aceptable la recuperación incompleta, los datos pueden ser codificados de modo que no todos los símbolos de entrada puedan ser recuperados, o de modo que la recuperación completa de los símbolos de entrada requiera la recepción de muchos más símbolos de salida que el número de símbolos de entrada. Una tal codificación requeriría en general menos gasto de cálculo y puede por tanto ser una manera aceptable de disminuir el gasto de cálculo de la codificación.

Ha de entenderse que los diversos bloques funcionales en las figuras descritas anteriormente pueden ser implementados por una combinación de hardware y / o software, y que en implementaciones específicas algo de, o toda, la funcionalidad de algunos de los bloques puede ser combinada. De manera similar, también ha de entenderse que los diversos procedimientos descritos en la presente memoria pueden ser implementados por una combinación de hardware y / o software.

La descripción anterior es ilustrativa y no restrictiva. Muchas variaciones de la invención devendrán evidentes para los expertos en la técnica tras la revisión de esta divulgación. El ámbito de la invención, por lo tanto, debería estar determinado, no con referencia a la descripción anterior, sino que, en cambio, debería estar determinado con referencia a las reivindicaciones adjuntas, junto con su ámbito completo de equivalentes.

APÉNDICE A. VALORES PARA ÍNDICES SISTEMÁTICOS *J(K)*

Para cada valor de *K*, el índice sistemático *J(K)* está diseñado para que tenga la propiedad de que el conjunto de tríos de símbolos de origen (*d*[0], *a*[0], *b*[0]), ..., (*d*[*L*-1], *a*[*L*-1], *b*[*L*-1]) sea tal que los *L* símbolos intermedios estén definidos de manera única, es decir, la matriz A en la Sección B.5.2.4.2 tenga rango completo y sea por lo tanto invertible. La siguiente es una lista de índices sistemáticos adecuados para valores de K entre 4 y 8.192 inclusive, y se proporciona a modo de ejemplo. El orden de los valores está en orden de lectura, es decir, desde el primer número en la primera línea hasta el último número en la primera línea, seguidos por el primer número en la segunda línea, y así sucesivamente.

22, 1486, 2151, 2239, 4286, 3887, 2016, 3247, 4531, 490, 2223,
705, 1243, 2241, 3457, 3030, 1918, 2349, 455, 1745, 182,
1130, 29, 381, 1088, 23, 1147, 353, 951, 161, 56,
421, 401, 1141, 557, 75, 611, 409, 828, 1514, 2722,
1044, 718, 768, 1696, 2145, 2515, 57, 43, 136, 438,
815, 540, 88, 270, 364, 236, 158, 14, 49, 160,
203, 49, 93, 104, 83, 567, 322, 143, 77, 29,
70, 37, 333, 120, 5, 5, 5, 408, 183, 356,
110, 233, 4, 251, 90, 1170, 322, 245, 664, 1057,
95, 512, 184, 1551, 1477, 778, 1038, 809, 719, 1025,
768, 1127, 689, 76, 1367, 172, 1313, 1557, 311, 311,
559, 1145, 624, 234, 172, 913, 1555, 1555, 1134, 692,
687, 692, 1501, 692, 1106, 1040, 195, 25, 1127, 1234,
1404, 1125, 798, 1511, 75, 947, 671, 1049, 385, 1273,
834, 1116, 1386, 1386, 399, 84, 1563, 365, 1563, 611,
611, 611, 1020, 819, 1688, 1688, 1550, 19, 1600, 1600,
796, 19, 1469, 1571, 1571, 1117, 1059, 476, 789, 1117,
1207, 1117, 1163, 1163, 512, 1719, 1719, 1719, 1719, 100,
623, 623, 623, 1567, 1567, 623, 876, 876, 876, 876,
816, 816, 814, 814, 431, 323, 1, 220, 1286, 1286,
623, 1286, 1286, 1286, 228, 1642, 1642, 1642, 259, 1410,
259, 259, 259, 353, 122, 826, 522, 826, 485, 1260,
1061, 1061, 1260, 1260, 1260, 1678, 353, 870, 870, 870,
714, 145, 3, 991, 211, 1644, 1644, 288, 288, 857,
1413, 1413, 1677, 857, 242, 242, 1565, 1565, 1677, 1810,
714, 714, 714, 714, 714, 1757, 714, 714, 964, 666,
964, 964, 1432, 219, 219, 908, 365, 679, 80, 1286,
1432, 1432, 579, 966, 62, 62, 62, 75, 62, 62,
62, 1261, 62, 62, 62, 62, 62, 894, 690, 690,
456, 538, 538, 1115, 1999, 1999, 102, 102, 1137, 1115,
102, 579, 579, 783, 783, 96, 1535, 1073, 1073, 612,
612, 1386, 1197, 690, 690, 690, 690, 690, 690, 690,
690, 968, 244, 244, 1812, 244, 968, 160, 656, 160,
160, 160, 656, 263, 263, 815, 146, 1099, 1074, 1099,
1099, 1099, 972, 1876, 222, 1994, 222, 327, 327, 1400,
1441, 1898, 1292, 1400, 1066, 1066, 1755, 54, 54, 54,
54, 54, 54, 54, 54, 1791, 1791, 1781, 554, 1047,
1047, 1047, 1047, 1047, 1047, 631, 1219, 289, 289, 1321,
1191, 6, 1781, 184, 184, 6, 2030, 1704, 1704, 1704,
1704, 1088, 1088, 1088, 1088, 1223, 1223, 2053, 2053, 8,
1704, 2053, 2053, 8, 1704, 1969, 770, 1969, 1969, 1371,
1371, 1353, 1810, 1810, 1810, 652, 652, 652, 652, 652,
652, 652, 652, 652, 423, 652, 1566, 1566, 1566, 1566,
1566, 1566, 1566, 1566, 1617, 965, 965, 965, 1267, 1267,

1267, 1267, 1892, 1699, 1699, 1267, 981, 1091, 981, 981,
2053, 2053, 510, 510, 510, 510, 927, 1843, 1843, 2203,
234, 234, 865, 1800, 865, 944, 927, 927, 1693, 1967,
1379, 1379, 1379, 1379, 1379, 843, 1441, 499, 1441, 160,
601, 1828, 675, 675, 601, 675, 675, 160, 2338, 203,
1138, 1138, 1244, 1244, 1244, 1244, 346, 323, 850, 850,
870, 870, 927, 927, 927, 408, 677, 677, 677, 677,
1166, 762, 677, 131, 965, 965, 965, 965, 233, 965,
540, 540, 540, 540, 540, 540, 540, 540, 540, 540,
411, 411, 81, 411, 411, 411, 943, 943, 91, 1067,
943, 1067, 304, 304, 949, 1046, 916, 751, 751, 1046,
576, 1193, 576, 576, 1193, 1208, 375, 1089, 1089, 375,
945, 945, 540, 540, 540, 540, 540, 540, 540, 540,
540, 540, 81, 41, 571, 571, 571, 723, 723, 723,
723, 723, 723, 723, 589, 589, 403, 403, 403, 1044,
980, 488, 698, 132, 1187, 1219, 1219, 1219, 1219, 1219,
851, 231, 929, 929, 929, 929, 929, 929, 929, 929,
929, 929, 929, 929, 8, 8, 8, 8, 540, 973,
973, 973, 973, 1, 983, 1144, 1071, 1071, 37, 37,
37, 173, 173, 173, 173, 665, 173, 173, 918, 918,
1310, 1297, 1297, 1297, 699, 662, 547, 882, 1234, 1234,
29, 29, 1228, 459, 547, 365, 34, 34, 34, 34,
34, 34, 34, 34, 317, 317, 317, 317, 317, 15,
242, 242, 242, 242, 437, 242, 292, 292, 292, 385,
753, 1061, 1061, 1061, 1061, 37, 906, 37, 1191, 1191,
1012, 1327, 869, 869, 266, 769, 41, 266, 120, 506,
506, 506, 943, 506, 506, 715, 1155, 142, 142, 142,
375, 375, 375, 375, 375, 375, 375, 375, 375, 375,
690, 375, 375, 375, 548, 939, 94, 94, 94, 939,
939, 939, 939, 94, 1219, 1219, 1219, 1219, 1219, 1219,
1253, 359, 666, 666, 94, 666, 666, 666, 94, 666,
666, 666, 633, 477, 968, 498, 968, 968, 865, 8,
165, 1219, 1219, 165, 165, 165, 842, 842, 1219, 1219,
343, 343, 987, 920, 987, 987, 987, 987, 139, 522,
522, 522, 522, 522, 640, 640, 640, 640, 173, 173,
365, 487, 487, 487, 965, 120, 120, 965, 120, 120,
120, 120, 965, 120, 1269, 1269, 965, 965, 1187, 1187,
1187, 1187, 481, 481, 1209, 663, 823, 823, 1143, 46,
46, 46, 46, 46, 46, 46, 46, 46, 46, 46,
46, 46, 46, 46, 46, 46, 46, 939, 757, 757,
757, 757, 902, 902, 1446, 902, 351, 1446, 351, 1446,
1222, 1222, 716, 1222, 1222, 1222, 1222, 833, 833, 833,
912, 912, 912, 912, 94, 912, 915, 808, 1163, 808,
1421, 443, 443, 443, 443, 443, 443, 432, 905, 366,
366, 366, 366, 408, 408, 408, 408, 408, 408, 1156,
1156, 145, 1156, 973, 973, 476, 476, 867, 634, 1077,
1077, 375, 375, 375, 375, 375, 375, 1371, 1371, 1371,
1371, 1371, 1371, 1371, 1371, 1371, 522, 383, 148, 1267, 522,
522, 1298, 674, 1298, 939, 674, 674, 674, 674, 1298,
1298, 1298, 1298, 197, 197, 197, 197, 120, 918, 918,
918, 918, 918, 47, 47, 1065, 1308, 1065, 1065, 1308,

1308, 1308, 1308, 1308, 1308, 1415, 1415, 417, 417, 417,
417, 417, 417, 1206, 613, 537, 1437, 1437, 476, 476,
476, 476, 1437, 669, 669, 558, 303, 303, 902, 902,
902, 902, 125, 902, 10, 1501, 1472, 535, 535, 535,
535, 535, 535, 977, 977, 977, 977, 409, 1309, 1309,
1309, 1309, 1309, 1309, 1309, 1309, 1309, 1309, 1309, 1309,
1309, 796, 796, 796, 716, 617, 694, 973, 973, 973,
973, 1054, 1054, 1054, 1054, 662, 662, 447, 447, 447,
447, 447, 447, 634, 634, 634, 634, 634, 634, 634,
37, 1459, 949, 1459, 1459, 1459, 1459, 634, 565, 565,
777, 634, 777, 977, 977, 1057, 1057, 657, 657, 657,
657, 1057, 1057, 1057, 657, 657, 657, 657, 657, 657,
657, 657, 657, 657, 657, 1437, 1500, 263, 263, 263,
70, 570, 570, 70, 70, 70, 70, 1497, 1497, 1497,
1497, 883, 364, 883, 883, 1128, 1390, 1390, 1390, 1390,
1032, 1032, 1032, 1338, 1338, 1338, 1338, 1338, 1338, 1166,
1166, 1166, 1166, 1166, 1166, 983, 983, 983, 983, 71,
983, 983, 983, 1137, 1137, 1137, 1137, 1137, 1137, 1137,
1137, 1137, 1137, 1137, 871, 571, 571, 571, 571, 918,
571, 918, 571, 571, 571, 118, 118, 118, 118, 118,
118, 809, 809, 1437, 1437, 1437, 1437, 1437, 1437, 537,
206, 206, 206, 206, 537, 537, 537, 537, 206, 537,
537, 498, 120, 498, 120, 566, 566, 1219, 566, 1219,
566, 566, 1219, 1219, 1219, 1219, 1219, 1219, 1219, 1219,
1219, 1219, 1219, 95, 550, 1367, 1367, 1367, 1367, 1705,
1230, 1230, 1407, 1230, 1230, 363, 363, 1539, 1539, 1539,
1539, 1539, 1539, 1380, 1380, 1380, 1380, 1152, 1455, 1345,
1345, 366, 366, 1308, 1308, 1308, 1308, 1308, 1308, 1308,
1308, 1219, 1219, 1219, 1219, 871, 871, 581, 581, 581,
581, 581, 581, 1, 1, 121, 1, 1, 1, 121,
1249, 1, 1249, 1249, 441, 1249, 1249, 1249, 1249, 1249,
1495, 1653, 1249, 1249, 1249, 615, 1064, 1249, 615, 615,
1249, 615, 221, 615, 615, 615, 98, 723, 723, 723,
723, 723, 723, 1533, 1533, 106, 106, 106, 106, 1232,
1232, 1864, 1232, 1232, 1232, 1864, 912, 918, 273, 273,
273, 273, 273, 918, 273, 273, 273, 273, 273, 273,
273, 273, 273, 273, 273, 1348, 1348, 1348, 1348, 1348,
1466, 1348, 1466, 1466, 160, 1460, 1460, 1460, 1460, 1460,
1792, 1792, 473, 473, 473, 1460, 473, 1792, 1792, 1009,
1009, 1009, 1009, 106, 106, 1434, 902, 1434, 1434, 1387,
1387, 1387, 1387, 1387, 1387, 1042, 1512, 1512, 1512, 1512,
1047, 1512, 1512, 1588, 1588, 1550, 662, 927, 628, 103,
222, 103, 103, 1313, 980, 980, 662, 980, 980, 980,
980, 980, 980, 632, 1107, 1048, 632, 1107, 1107, 1620,
632, 1067, 1620, 963, 963, 94, 94, 94, 514, 1725,
1257, 1676, 1676, 1676, 1676, 1437, 1437, 1417, 1417, 681,
1437, 1064, 1064, 1117, 1117, 1064, 1064, 1117, 1117, 1117,
1064, 1117, 1117, 453, 677, 871, 871, 871, 871, 871,
871, 871, 313, 313, 677, 1259, 1259, 118, 1259, 1259,
1707, 441, 441, 1766, 1766, 1766, 1766, 1766, 1766, 1766,
1766, 1766, 1766, 1766, 1766, 1767, 1767, 1767, 935, 1767,

935, 549, 549, 549, 159, 873, 1868, 1868, 1868, 1598,
1868, 1140, 1587, 1587, 1587, 1587, 1587, 1140, 1140, 1457,
1457, 1457, 1457, 1937, 1253, 1937, 1937, 1253, 1937, 1195,
1937, 986, 986, 1821, 1821, 1797, 41, 1439, 1011, 1011,
1011, 1011, 1011, 1011, 1011, 1439, 1439, 1381, 1381, 1381,
1381, 1200, 1200, 1200, 1200, 1200, 1200, 858, 858, 1077,
1669, 1669, 1669, 108, 108, 968, 690, 108, 968, 973,
343, 1313, 1313, 1313, 1313, 1313, 1313, 949, 949, 949,
949, 949, 949, 716, 949, 716, 716, 1581, 1431, 1431,
1431, 1431, 1431, 1431, 1431, 1581, 1581, 1431, 1431, 1581,
1581, 664, 1581, 845, 396, 664, 664, 650, 650, 650,
814, 814, 1973, 461, 461, 461, 492, 1997, 306, 1114,
865, 865, 1820, 865, 865, 865, 1104, 952, 952, 1104,
1104, 1257, 1306, 1306, 968, 952, 952, 175, 175, 175,
175, 1066, 1632, 452, 902, 902, 902, 902, 1067, 243,
243, 1067, 243, 1360, 408, 408, 408, 498, 498, 4,
498, 498, 498, 498, 498, 1218, 1218, 1165, 6, 6,
1165, 6, 1381, 1165, 1165, 1165, 1725, 1725, 164, 1725,
1725, 1725, 54, 1725, 716, 716, 1114, 1114, 1114, 1114,
1067, 1426, 1426, 1426, 1426, 1426, 2162, 2162, 2162, 2162,
2162, 2162, 243, 243, 243, 243, 1095, 243, 1095, 243,
243, 243, 243, 243, 243, 1982, 243, 243, 243, 243,
347, 347, 347, 347, 347, 347, 1823, 1069, 1823, 1069,
1671, 1745, 2179, 376, 623, 2179, 376, 376, 2179, 376,
2179, 2179, 2179, 376, 756, 571, 2159, 2159, 1125, 2159,
1125, 1125, 2159, 2159, 1322, 180, 977, 180, 180, 977,
180, 977, 180, 565, 180, 977, 2124, 857, 2124, 857,
2124, 2124, 857, 2124, 842, 842, 842, 842, 842, 842,
62, 62, 1390, 62, 1390, 1420, 62, 62, 62, 62,
1055, 1055, 1055, 1055, 1055, 1055, 175, 1055, 1055, 1055,
1055, 1055, 175, 1055, 918, 918, 718, 233, 233, 718,
1263, 1263, 1263, 1263, 757, 2048, 70, 1723, 1723, 1723,
2154, 2154, 1532, 233, 1532, 1532, 1532, 1532, 1532, 1532,
1532, 1532, 888, 888, 888, 888, 888, 888, 689, 689,
1546, 1546, 1546, 1546, 1546, 1546, 1896, 1896, 662, 662,
450, 264, 1430, 1430, 1430, 1430, 1430, 1430, 1430, 1430,
1430, 1430, 1430, 1430, 1430, 30, 30, 1710, 1710, 1710,
2234, 2101, 498, 432, 498, 498, 1677, 947, 947, 1677,
1677, 498, 947, 498, 1283, 498, 1677, 498, 530, 533,
960, 960, 960, 960, 960, 960, 1099, 960, 960, 1215,
475, 960, 960, 960, 1215, 960, 960, 960, 960, 960,
960, 960, 1553, 1309, 1553, 1553, 1309, 1309, 94, 1647,
94, 94, 94, 94, 94, 1647, 935, 935, 140, 508,
508, 1511, 95, 95, 95, 95, 1306, 1169, 1431, 1431,
1431, 1431, 1176, 1176, 1176, 1176, 1176, 1176, 1176, 1416,
1839, 1839, 1839, 1839, 2324, 1839, 2425, 1744, 723, 2506,
2506, 2506, 2506, 2506, 2506, 2506, 1823, 1823, 1079, 130,
1838, 1838, 508, 1838, 1838, 1838, 1838, 1838, 424, 424,
424, 424, 424, 424, 424, 424, 424, 424, 424, 582,
582, 1867, 582, 582, 582, 582, 2326, 1091, 1091, 1091,
1091, 1091, 1462, 1462, 1462, 1462, 1462, 1462, 850, 1462,

850, 850, 850, 1413, 2374, 2374, 740, 740, 740, 740,
572, 572, 1084, 415, 2426, 2426, 1134, 2426, 2426, 2426,
2426, 2426, 270, 270, 270, 270, 270, 270, 270, 270,
270, 270, 270, 270, 1165, 1165, 975, 32, 975, 975,
32, 975, 975, 975, 975, 975, 975, 975, 975, 975,
975, 975, 157, 1066, 160, 669, 160, 160, 160, 160,
461, 834, 834, 834, 292, 36, 156, 27, 27, 27,
27, 27, 27, 27, 27, 27, 645, 690, 690, 1237,
690, 1106, 1106, 690, 949, 550, 949, 550, 949, 550,
949, 949, 949, 949, 949, 949, 949, 949, 949, 949,
949, 949, 273, 273, 273, 273, 273, 273, 273, 273,
273, 273, 273, 273, 273, 273, 273, 273, 273, 273,
273, 273, 273, 273, 353, 353, 306, 306, 306, 396,
355, 355, 355, 192, 355, 355, 918, 918, 918, 918,
918, 918, 918, 918, 970, 970, 970, 970, 970, 970,
970, 490, 970, 970, 490, 490, 970, 490, 490, 490,
807, 273, 374, 118, 374, 374, 1237, 675, 1237, 675,
1144, 1144, 1144, 675, 273, 273, 1251, 1251, 1251, 1251,
1251, 1251, 1251, 1251, 1251, 1251, 1251, 1251, 1251, 1251,
11, 11, 1157, 1157, 1157, 1157, 784, 784, 784, 784,
784, 951, 1259, 951, 1012, 1012, 1012, 1012, 808, 1012,
812, 812, 812, 812, 812, 812, 1206, 1206, 1206, 1206,
734, 482, 482, 482, 1043, 1313, 166, 166, 166, 166,
675, 166, 166, 675, 166, 675, 675, 675, 675, 675,
675, 675, 675, 166, 675, 675, 166, 166, 968, 968,
968, 968, 968, 968, 1311, 1311, 538, 538, 538, 538,
538, 538, 582, 582, 582, 582, 347, 347, 347, 347,
347, 347, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 645, 645, 21, 29, 29, 1312, 1312,
1380, 1380, 1411, 1120, 1411, 1411, 306, 306, 806, 806,
669, 669, 669, 681, 1253, 1253, 876, 876, 1036, 1036,
1036, 1036, 1036, 1036, 233, 233, 233, 233, 233, 233,
233, 233, 233, 233, 233, 233, 271, 271, 271, 271,
271, 271, 871, 871, 871, 871, 871, 871, 1268, 1268,
1268, 1268, 1268, 1268, 1268, 1268, 1268, 1268, 1268, 1268,
1268, 1268, 1194, 1125, 1125, 1125, 353, 353, 353, 353,
353, 353, 865, 865, 913, 865, 865, 913, 865, 865,
865, 913, 913, 913, 409, 409, 409, 409, 409, 409,
409, 409, 954, 954, 954, 954, 954, 954, 159, 159,
159, 159, 585, 585, 585, 585, 585, 585, 585, 585,
585, 585, 585, 585, 585, 585, 585, 585, 585, 585,
585, 585, 585, 585, 585, 585, 585, 103, 529, 529,
529, 529, 529, 529, 529, 529, 529, 529, 529, 529,
529, 529, 529, 529, 529, 529, 871, 694, 310, 871,
871, 310, 310, 310, 310, 1526, 1194, 1194, 1194, 1194,
698, 1194, 1194, 1194, 770, 770, 770, 1103, 1367, 1367,
1367, 361, 1367, 1367, 850, 850, 1420, 1420, 1420, 1420,
1420, 1420, 353, 353, 353, 353, 353, 353, 514, 514,
514, 353, 353, 353, 353, 723, 514, 11, 723, 514,
11, 11, 723, 723, 538, 538, 538, 538, 538, 538,
538, 538, 538, 538, 538, 538, 1379, 1379, 1523, 850,

850, 850, 321, 850, 850, 850, 850, 1523, 1523, 1523,
1523, 1523, 1523, 1523, 1444, 1550, 1550, 1550, 70, 690,
690, 690, 415, 415, 415, 415, 415, 518, 725, 725,
415, 415, 415, 415, 1036, 353, 1036, 353, 353, 353,
1012, 1036, 353, 1036, 353, 353, 1671, 1671, 935, 935,
935, 935, 159, 935, 935, 935, 935, 935, 832, 832,
1283, 1283, 1283, 1283, 121, 121, 121, 121, 121, 121,
121, 121, 756, 756, 1380, 756, 756, 756, 1413, 1413,
1413, 1413, 1413, 1413, 308, 308, 481, 1219, 662, 662,
1512, 1512, 1512, 986, 21, 21, 1144, 1144, 21, 21,
920, 585, 585, 920, 920, 920, 920, 920, 871, 871,
871, 871, 582, 1333, 353, 353, 353, 353, 353, 353,
1244, 1244, 1244, 1244, 1244, 1244, 1244, 1244, 1244, 1244,
316, 316, 918, 918, 918, 918, 918, 918, 918, 918,
918, 1032, 1508, 671, 671, 671, 671, 819, 1437, 1437,
55, 960, 960, 960, 970, 481, 481, 1114, 1114, 1114,
1114, 481, 1114, 481, 1556, 1114, 1556, 1556, 627, 627,
627, 627, 627, 627, 627, 627, 627, 627, 871, 871,
871, 871, 871, 871, 871, 871, 871, 871, 871, 871,
1160, 1160, 1448, 361, 361, 396, 396, 361, 994, 994,
994, 994, 717, 717, 663, 663, 1016, 663, 663, 663,
663, 496, 663, 663, 663, 663, 161, 161, 230, 230,
230, 1348, 230, 230, 1149, 230, 230, 1348, 1348, 1348,
1348, 1348, 1197, 1197, 1197, 1197, 1429, 1429, 1429, 1429,
36, 36, 1707, 1707, 1707, 1707, 1707, 1707, 1707, 1707,
1586, 1586, 1174, 1174, 1174, 1586, 681, 1707, 681, 681,
1314, 1314, 1314, 1314, 1512, 1314, 1314, 1512, 1314, 31,
1512, 1314, 1314, 1512, 1314, 1314, 1314, 1314, 1434, 1434,
1434, 1434, 1434, 1434, 1434, 1434, 353, 1592, 1592, 1592,
1592, 1592, 30, 62, 62, 62, 62, 62, 1485, 1485,
1485, 740, 807, 1485, 963, 963, 963, 963, 832, 832,
963, 963, 1444, 1591, 1444, 1444, 1444, 1444, 1444, 1591,
1591, 1591, 842, 1586, 842, 842, 1586, 842, 842, 1586,
1586, 1776, 1586, 1586, 968, 968, 968, 968, 968, 968,
968, 968, 968, 968, 968, 968, 968, 968, 992, 1362,
992, 992, 983, 983, 983, 983, 983, 983, 1521, 1521,
1521, 1521, 1521, 1165, 1158, 1158, 91, 91, 91, 91,
91, 91, 91, 820, 91, 91, 1069, 1069, 1069, 1069,
91, 91, 91, 91, 91, 91, 91, 91, 91, 91,
91, 91, 91, 91, 1825, 1825, 1432, 1432, 1432, 1432,
1432, 1432, 1432, 1432, 1432, 1432, 30, 30, 30, 30,
30, 30, 30, 30, 1675, 1508, 1675, 1508, 1620, 1620,
1620, 1620, 1620, 1620, 1620, 1620, 1620, 1620, 1620, 1620,
1620, 1620, 1221, 1620, 1221, 935, 334, 334, 334, 231,
334, 334, 952, 952, 952, 952, 554, 952, 1164, 952,
952, 952, 1338, 952, 1832, 1832, 1832, 1832, 1832, 1832,
355, 355, 355, 355, 355, 355, 355, 355, 355, 355,
355, 355, 1741, 383, 1741, 1741, 1748, 1748, 1748, 1748,
1748, 1748, 550, 550, 550, 550, 550, 550, 550, 550,
550, 550, 550, 1669, 1669, 1669, 1669, 1669, 550, 1669,
1669, 550, 1588, 1588, 1588, 1588, 1588, 1588, 1588, 1588,

1588, 1588, 1669, 1669, 368, 368, 368, 368, 1313, 1313,
368, 1313, 1313, 1313, 1313, 1313, 1221, 1221, 1221, 1221,
1221, 1221, 1221, 1221, 1221, 1221, 1221, 1221, 1221, 1221,
1221, 1221, 1221, 1221, 1221, 1221, 1221, 1221, 1221, 1221,
1221, 1221, 1671, 1671, 1671, 1671, 832, 832, 963, 963,
963, 963, 1822, 963, 1381, 963, 522, 963, 963, 522,
6, 6, 6, 6, 6, 6, 1009, 1009, 1009, 1009,
1839, 1839, 1839, 1707, 1839, 1839, 1707, 1839, 1839, 1707,
1839, 1707, 1827, 1916, 1089, 1457, 1827, 1089, 64, 64,
64, 64, 64, 64, 64, 64, 22, 22, 22, 22,
740, 1087, 740, 740, 35, 35, 740, 740, 1221, 1221,
1706, 1706, 1706, 1221, 1011, 1706, 1706, 1011, 1706, 1011,
1706, 1706, 1706, 1706, 1706, 1706, 1706, 1011, 1011, 1403,
265, 265, 121, 1099, 121, 1099, 582, 2018, 1972, 1373,
1373, 1972, 415, 415, 1256, 1256, 1256, 1936, 1936, 1256,
1797, 1797, 1797, 740, 740, 935, 935, 1797, 935, 1797,
935, 935, 1797, 1797, 1797, 1797, 1797, 1797, 1797, 1797,
6, 1797, 1797, 1797, 1797, 1797, 1797, 1797, 1795, 1795,
1727, 1727, 1727, 1727, 1727, 582, 375, 375, 375, 375,
1176, 67, 81, 1915, 81, 81, 81, 81, 740, 740,
740, 740, 1564, 1564, 1564, 1564, 158, 158, 136, 136,
136, 136, 1598, 1598, 1598, 1598, 1875, 1505, 1598, 1875,
1773, 1773, 1773, 175, 461, 780, 1952, 1952, 312, 312,
1219, 1219, 1219, 312, 312, 1952, 1947, 1727, 492, 30,
492, 492, 492, 492, 492, 388, 492, 492, 388, 388,
388, 388, 492, 492, 1382, 1382, 424, 424, 424, 424,
424, 424, 424, 424, 1592, 554, 424, 1592, 424, 424,
1592, 1592, 516, 516, 516, 516, 516, 516, 625, 625,
625, 625, 625, 625, 625, 625, 625, 625, 625, 625,
625, 625, 625, 625, 625, 625, 625, 625, 625, 625,
625, 625, 625, 625, 625, 625, 625, 625, 1809, 183,
1809, 183, 183, 183, 1809, 1562, 1562, 1562, 1562, 1562,
1562, 1562, 1562, 1779, 1779, 1779, 1779, 1779, 1779, 1779,
1779, 1158, 1158, 1158, 1158, 1592, 1592, 63, 63, 63,
63, 1448, 1448, 538, 751, 538, 538, 751, 751, 538,
538, 538, 751, 751, 751, 751, 751, 538, 751, 538,
538, 538, 538, 538, 538, 1089, 1089, 1089, 1089, 1089,
1089, 1089, 1089, 948, 948, 948, 948, 948, 948, 948,
948, 948, 948, 948, 948, 948, 1617, 1617, 1617, 1617, 1617,
1617, 1958, 1958, 1958, 1958, 1958, 1958, 1958, 1958, 1958, 1958,
1958, 409, 409, 1362, 1661, 1362, 1362, 2283, 850, 2283,
2283, 2283, 2283, 865, 1464, 1592, 2174, 1592, 784, 784,
784, 1060, 1060, 1060, 1060, 1060, 1060, 241, 1060, 241,
1060, 736, 736, 442, 442, 442, 442, 442, 442, 442,
442, 442, 442, 442, 442, 271, 271, 271, 271, 271,
271, 271, 271, 271, 271, 353, 353, 1222, 2224, 1222,
1222, 1222, 1222, 1222, 1222, 1222, 1222, 755, 755, 755,
755, 755, 755, 1331, 1331, 1331, 1331, 1331, 1331, 1331,
1331, 1569, 1569, 1569, 1569, 1569, 1569, 1219, 2138, 1219,
1219, 35, 265, 409, 409, 409, 35, 22, 22, 812,
812, 154, 1023, 812, 409, 1916, 1916, 1916, 1916, 1916,
1916, 2047, 2047, 2047, 2047, 1448, 1448, 299, 299, 299,

299, 299, 299, 299, 299, 299, 1745, 299, 299, 299,
299, 299, 299, 1764, 1764, 1764, 1764, 1764, 1764, 1764,
1764, 1764, 1764, 1764, 1764, 1514, 1514, 552, 552, 552,
552, 1687, 1687, 1687, 1687, 1687, 1687, 1687, 1687, 1687,
1687, 1687, 1687, 1687, 1687, 172, 172, 172, 172, 172,
172, 2115, 2115, 2115, 2115, 196, 1858, 196, 1858, 1858,
1858, 1858, 1858, 364, 364, 364, 364, 364, 364, 364,
364, 1517, 364, 646, 646, 646, 646, 646, 646, 646,
646, 128, 128, 128, 128, 128, 128, 1163, 1163, 427,
427, 427, 427, 1371, 1371, 1371, 1868, 1868, 1868, 960,
1868, 1868, 1371, 960, 694, 1371, 425, 694, 694, 694,
694, 425, 1540, 694, 694, 1466, 1466, 1466, 1466, 1466,
1466, 1604, 1604, 334, 334, 334, 334, 334, 334, 30,
30, 30, 1851, 83, 83, 1629, 1851, 1629, 784,
1629, 1629, 1629, 1629, 1629, 1629, 1629, 391, 178, 178,
2044, 398, 1947, 398, 398, 398, 398, 1641, 1641, 1641,
1641, 1641, 1641, 1641, 1641, 30, 30, 30, 30, 30,
1779, 30, 1779, 1779, 1779, 1707, 1707, 1128, 1128, 1128,
1128, 1128, 1128, 1128, 1128, 1128, 1128, 1569, 420, 1569,
420, 1569, 420, 420, 420, 420, 420, 420, 420, 420,
420, 1592, 2086, 1429, 1347, 2011, 2011, 2161, 1448, 1448,
1448, 1448, 1448, 1448, 1448, 1448, 1448, 1457, 1457, 1457,
1457, 1457, 1457, 1457, 1457, 1457, 1457, 1457, 1457, 1457,
1457, 998, 998, 998, 998, 749, 749, 749, 2305, 295,
2441, 2441, 2441, 159, 159, 886, 886, 886, 886, 159,
2201, 886, 886, 886, 159, 2201, 886, 2201, 2201, 886,
2201, 2346, 2131, 2131, 2131, 514, 2075, 2075, 2075, 2075,
2075, 1188, 1188, 1444, 1444, 1444, 1444, 2263, 2263, 2263,
2263, 2263, 2263, 1125, 1125, 933, 933, 933, 933, 933,
933, 364, 933, 1128, 933, 933, 933, 818, 1926, 1926,
1926, 1128, 2440, 1128, 1128, 1128, 1128, 1128, 1128, 1128,
1128, 1128, 1128, 1128, 1128, 1128, 1128, 1128, 1128, 1128,
1128, 2174, 1651, 1651, 1651, 1651, 1651, 1651, 2174, 2174,
1651, 1651, 1651, 1651, 1651, 1651, 1651, 1651, 1651, 22,
22, 22, 22, 2263, 22, 22, 2263, 2263, 22, 22,
22, 2548, 2548, 2665, 1998, 2427, 2053, 920, 920, 920,
920, 920, 36, 13, 19, 1790, 2471, 1197, 1197, 2502,
2502, 266, 266, 266, 266, 266, 266, 521, 521, 521,
521, 521, 521, 521, 521, 521, 521, 521, 521, 521,
521, 521, 521, 521, 521, 521, 521, 521, 521, 1418,
1418, 2232, 2232, 2232, 2232, 2232, 2232, 2806, 2806, 2806,
2806, 2878, 2806, 2806, 2878, 174, 2636, 2636, 2878, 1450,
2878, 2806, 2878, 253, 253, 253, 253, 253, 253, 979,
979, 979, 979, 979, 979, 979, 979, 979, 979, 979,
979, 552, 552, 295, 295, 295, 295, 295, 295, 1707,
2465, 1707, 1707, 1707, 978, 1707, 1707, 1707, 2465, 1707,
1707, 132, 132, 2842, 130, 132, 132, 130, 130, 130,
132, 132, 130, 132, 132, 132, 132, 1495, 1495, 1973,
1973, 1973, 1973, 723, 912, 2482, 2482, 2482, 723, 89,
89, 89, 89, 89, 89, 1671, 89, 89, 89, 1767,
89, 1009, 1009, 1177, 1177, 1177, 1177, 1177, 1177, 1177,

1177, 1177, 1177, 1177, 1177, 1177, 1177, 1177, 1177, 1177,
1744, 2076, 2076, 2076, 2076, 2811, 2076, 2076, 2076, 2811,
2076, 607, 2076, 2076, 2076, 2076, 2076, 2076, 2076, 607,
2076, 2076, 2076, 2076, 2076, 2306, 2306, 2306, 2306, 2306,
2306, 2306, 2306, 2306, 2306, 770, 1514, 1514, 770, 1514,
1514, 1421, 1616, 2110, 2110, 2110, 2110, 2110, 2110, 2110,
2110, 2110, 2110, 127, 127, 2521, 2521, 2521, 2521, 2521,
2521, 2521, 2521, 2521, 2521, 1588, 1588, 1505, 1505, 272,
272, 1505, 1505, 1505, 1505, 1505, 1505, 2961, 695, 695,
1174, 1174, 695, 2135, 2135, 838, 838, 2155, 2155, 2155,
2155, 2155, 2122, 2122, 2122, 465, 465, 583, 583, 583,
583, 583, 583, 583, 583, 583, 583, 583, 2451, 1845, 1845,
1617, 2451, 1845, 978, 978, 978, 978, 978, 978, 978,
978, 1197, 1197, 1197, 1197, 1197, 1197, 1197, 1197, 1197,
1197, 1197, 1197, 1197, 1197, 1197, 1197, 3017, 1197, 1197,
1197, 3017, 1197, 3017, 1197, 1197, 1197, 1197, 1197, 1197,
1197, 2863, 2863, 2863, 2863, 2863, 2863, 2863, 2863, 2863,
2863, 852, 852, 1168, 1168, 1168, 1168, 1168, 1168, 2748,
1168, 1168, 1168, 1495, 1495, 1495, 1495, 1495, 1669, 1669,
1669, 1329, 1329, 1329, 1329, 1329, 1329, 1727, 1727, 2441,
1532, 1532, 1532, 1727, 2441, 1727, 1727, 550, 550, 550,
550, 550, 550, 550, 550, 550, 550, 550, 550, 2263, 550,
2263, 550, 550, 550, 550, 353, 353, 353, 353, 353,
353, 2245, 1592, 1612, 2245, 1834, 2245, 1612, 1822, 1612,
2826, 2245, 1592, 477, 477, 477, 477, 477, 477, 477,
477, 477, 477, 477, 477, 2724, 2724, 353, 209, 486,
353, 1485, 2113, 353, 353, 2113, 1485, 353, 353, 353,
1485, 1485, 2113, 2113, 2113, 2811, 2811, 2811, 2811, 2811,
2811, 2308, 1587, 3336, 1587, 1401, 1285, 1285, 1285, 1099,
1099, 1099, 1099, 1099, 1099, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 2071, 2071, 1188, 1188, 1188, 1188, 1188,
1188, 1188, 1188, 2494, 2494, 2963, 2963, 2963, 2963, 2963,
2963, 2963, 2963, 2963, 2963, 2963, 2963, 2963, 2963, 398,
398, 398, 398, 398, 398, 3153, 3102, 3153, 169, 2089,
2089, 2273, 1530, 1530, 3307, 1222, 1222, 1222, 1222, 1222,
1222, 1222, 1222, 1222, 1222, 1222, 1222, 1222, 1222, 1222,
1222, 1222, 1222, 1222, 1222, 1222, 1222, 1222, 1222, 1118,
1118, 492, 492, 492, 492, 492, 1489, 1489, 1489, 555,
555, 2645, 555, 555, 555, 555, 555, 555, 555, 555,
555, 555, 555, 555, 555, 2656, 2656, 2656, 2656, 2656,
1015, 1015, 1015, 2619, 2619, 2619, 2619, 2619, 2619, 1401,
1118, 1401, 2780, 1118, 1118, 1778, 1778, 1778, 1778, 1778,
1778, 1778, 1778, 1778, 1778, 1778, 1778, 1778, 1778, 1778,
1778, 1778, 1778, 1263, 1263, 1263, 1263, 1263, 1263, 1263,
1263, 1263, 1263, 1263, 1263, 1263, 1263, 1263, 3649, 1263, 2860,
2860, 2924, 1592, 1592, 3223, 1490, 9, 2620, 954, 1490,
954, 785, 785, 2627, 772, 772, 2627, 772, 772, 2347,
2347, 2347, 2347, 2347, 2347, 2534, 2534, 2534, 2534, 2534,
2534, 2534, 2534, 2534, 2534, 1260, 1260, 1260, 1260, 1260,
1260, 647, 647, 647, 647, 647, 647, 647, 647, 647,

647, 647, 647, 2282, 2282, 2282, 2282, 2282, 2282, 2282,
2282, 2282, 2282, 2282, 2282, 960, 960, 960, 960, 960,
960, 960, 960, 960, 960, 960, 960, 960, 960, 960,
960, 960, 960, 2263, 2263, 287, 287, 287, 287, 287,
287, 1797, 1797, 572, 1797, 1179, 1179, 1179, 1179, 1179,
1179, 1179, 1179, 1179, 1179, 1179, 1179, 1179, 1179, 1179,
1179, 3355, 1179, 784, 784, 784, 364, 1381, 1381, 1381,
364, 647, 647, 647, 647, 647, 647, 647, 647, 647,
647, 647, 647, 647, 647, 647, 647, 647, 647, 647,
647, 647, 647, 647, 647, 112, 112, 112, 2626, 1490,
2179, 2041, 2041, 2041, 2041, 481, 2094, 2094, 2094, 514,
2094, 2455, 2455, 582, 306, 306, 306, 306, 306, 306,
306, 306, 306, 306, 2076, 3103, 3103, 3103, 3103, 488,
488, 488, 488, 488, 488, 488, 488, 488, 488, 488,
488, 82, 82, 241, 241, 241, 241, 241, 241, 3535,
241, 241, 241, 241, 241, 241, 241, 241, 241, 241,
241, 3535, 241, 241, 3535, 241, 241, 241, 241, 241,
241, 241, 241, 891, 285, 285, 285, 1188, 1188, 1188,
1188, 1188, 1188, 3011, 1325, 1325, 567, 1325, 3549, 1367,
1367, 1367, 1367, 1367, 1367, 1367, 1367, 1367, 1367, 1367,
1367, 2665, 23, 23, 3017, 3017, 3017, 3017, 3017, 3017,
3017, 3017, 3017, 3017, 3017, 1423, 2897, 1423, 1423, 918,
918, 166, 166, 166, 166, 166, 166, 2815, 166, 166,
166, 2282, 2282, 2282, 2282, 2282, 2282, 2282, 2282, 2282,
2282, 2282, 2282, 2619, 2619, 2161, 2161, 2161, 2161, 2696,
2814, 3336, 3336, 2814, 2696, 2452, 2452, 2452, 3482, 2452,
1392, 2452, 2452, 1229, 1229, 2627, 303, 2263, 2263, 3346,
838, 838, 3346, 838, 3346, 838, 2583, 2583, 2583, 3232,
3232, 1596, 1720, 1720, 1720, 2044, 2044, 409, 2044, 2044,
2044, 2044, 2044, 409, 21, 409, 2044, 409, 2044, 21,
21, 21, 21, 21, 21, 3268, 3268, 1727, 1727, 3994,
3268, 1335, 1335, 1335, 1335, 3547, 3547, 393, 1069, 3547,
1260, 465, 465, 2041, 2656, 2656, 2656, 2656, 2656, 2656,
2656, 2656, 2656, 1355, 2185, 1260, 1260, 1260, 1260, 1260,
1260, 1260, 1260, 1260, 1571, 1778, 1778, 1778, 1571, 1260,
1260, 2294, 2294, 2294, 2294, 2294, 2294, 2294, 2294, 2294,
2294, 2294, 2294, 516, 2214, 1811, 1811, 1811, 1811, 1811,
1811, 420, 420, 420, 420, 420, 420, 420, 420, 420,
420, 420, 420, 420, 420, 420, 420, 1086, 1086, 1086,
1086, 1104, 1610, 1610, 1610, 1610, 1610, 1404, 1404, 1404,
1404, 1404, 1404, 1404, 1404, 1404, 1404, 1404, 3811, 4368,
4368, 1338, 1338, 1338, 1338, 1338, 1338, 1260, 1260, 550,
1260, 1260, 550, 920, 2161, 920, 2161, 920, 920, 3641,
3357, 3641, 3641, 3641, 3641, 3641, 3357, 3357, 3641, 3641,
3641, 3641, 3641, 3641, 998, 3641, 3641, 3641, 3641, 3641,
3641, 3641, 3641, 3641, 3641, 3641, 3641, 231, 231, 231,
231, 231, 231, 1457, 1457, 1457, 1457, 3525, 1457, 3525,
1457, 1457, 1457, 1457, 1457, 1457, 409, 409, 409, 1423,
409, 4075, 409, 2968, 2968, 2968, 2968, 2968, 2968, 2968,
2968, 2968, 2968, 3676, 3676, 2386, 3676, 3676, 3676, 3676,
3676, 1129, 1129, 1129, 1129, 1129, 1129, 1129, 1129, 1129,

1129, 1129, 1129, 1650, 1650, 1650, 1650, 4246, 4246, 4246,
4246, 1650, 1650, 1650, 4246, 4246, 4246, 4246, 4246, 4246,
4246, 2224, 2224, 2224, 425, 2220, 826, 663, 663, 663,
2423, 2707, 2707, 2707, 2707, 2707, 2707, 2707, 2707, 2707,
2707, 2707, 2707, 2707, 2707, 2707, 2707, 2707, 2707, 2707,
2707, 2707, 2707, 2707, 2707, 3178, 2519, 2519, 3178, 3178,
3178, 51, 3178, 51, 3178, 4507, 4507, 2437, 2437, 2521,
231, 231, 2521, 2854, 2854, 3019, 3019, 3611, 3611, 3019,
918, 3019, 3545, 3545, 3545, 3545, 3545, 3611, 3611, 3611,
3611, 2514, 2514, 2514, 2514, 2514, 2514, 4151, 4151, 4151,
4151, 4151, 4151, 586, 586, 4343, 2664, 2664, 2664, 586,
2664, 2664, 2664, 2679, 2664, 2664, 2664, 1017, 1423, 1423,
1423, 1017, 1423, 1017, 1017, 1423, 2665, 1888, 1888, 788,
788, 788, 788, 788, 788, 1888, 788, 788, 1888, 788,
788, 871, 871, 954, 871, 1477, 4453, 4241, 4453, 1477,
4453, 4453, 4241, 4453, 4453, 4241, 838, 4241, 4453, 4453,
4241, 4241, 3726, 838, 4241, 3726, 1477, 1477, 1477, 4968,
3726, 4968, 4968, 1477, 4968, 2444, 2444, 2444, 2444, 2444,
2444, 1429, 1429, 1429, 1429, 1429, 1429, 893, 893, 893,
893, 893, 893, 4101, 2263, 4101, 4101, 2263, 4101, 2263,
2263, 4173, 4173, 4173, 4173, 4173, 4173, 4620, 3269, 3461,
4394, 3830, 3830, 423, 423, 423, 423, 423, 423, 423,
423, 423, 423, 423, 423, 3220, 3220, 3220, 3220, 3220,
2081, 1599, 3780, 1599, 3780, 2311, 2311, 2632, 2632, 2632,
2632, 2632, 2632, 3360, 1158, 3360, 3085, 3085, 1158, 3360,
3085, 3085, 3085, 2464, 3085, 2464, 3085, 2464, 2464, 3085,
3085, 3085, 3085, 3085, 3085, 734, 3285, 920, 920, 920,
920, 364, 364, 364, 364, 364, 364, 364, 364, 364,
364, 364, 364, 364, 364, 364, 364, 364, 364, 4978,
4978, 4131, 4131, 4131, 4131, 2903, 303, 303, 303, 1189,
303, 1189, 1189, 1189, 303, 1189, 303, 2263, 2263, 2599,
2599, 2599, 2599, 2599, 3029, 1060, 1060, 1060, 1060, 1727,
1727, 1727, 1727, 1727, 1727, 1727, 1727, 1727, 1727, 1727,
1727, 1727, 1727, 1727, 1727, 1727, 1727, 1727, 1727, 1727,
1727, 1727, 1727, 1727, 1727, 2831, 2728, 2831, 3421, 3421,
3421, 2582, 2582, 2582, 2582, 1333, 1965, 465, 465, 465,
465, 5182, 5182, 5182, 5182, 5182, 5182, 73, 5182, 398,
398, 3619, 398, 398, 398, 398, 398, 398, 398, 4554,
4554, 5110, 5110, 5110, 5110, 4554, 5110, 5110, 5110, 5110,
5110, 5110, 5110, 2907, 2907, 2907, 2907, 2907, 2907, 2907,
2907, 2907, 2907, 2907, 2907, 2907, 5110, 2907, 2907, 2907, 2907,
2907, 784, 784, 784, 3138, 3138, 784, 3138, 3138, 3138,
1710, 3138, 1710, 3138, 3138, 1710, 1710, 1367, 1367, 4114,
4114, 1614, 4114, 4114, 4114, 5082, 1778, 1778, 1778, 2562,
2562, 780, 780, 780, 780, 2843, 2843, 4642, 3355, 3355,
3355, 3355, 3355, 3355, 4478, 3355, 3355, 2665, 2665, 4274,
4274, 4274, 2665, 4274, 4274, 4274, 4274, 4274, 4274, 4274,
4274, 325, 325, 325, 325, 325, 325, 1222, 1222, 1222,
1222, 2665, 2665, 2665, 2665, 3351, 2665, 2665, 2665, 2359,
1053, 2359, 2294, 2359, 3599, 3599, 2359, 2359, 2359, 2831,
2831, 3672, 2831, 3672, 1335, 2160, 2160, 2160, 4009, 4240,

4240, 4240, 4240, 4240, 4240, 4009, 4240, 4240, 4240, 4009,
4240, 4009, 4240, 2160, 2160, 2386, 681, 681, 2386, 3270,
3270, 2631, 2631, 2631, 2631, 2631, 2631, 2434, 2434, 2434,
2434, 2434, 2434, 2434, 2434, 2434, 2434, 2434, 2434, 2434,
2434, 2434, 2434, 2434, 2434, 2434, 1514, 2434, 2434, 2434,
2434, 2434, 2434, 2434, 2434, 2434, 4465, 3926, 3926, 3926,
3926, 3926, 3926, 3926, 3926, 1275, 266, 266, 1275, 1275,
1275, 266, 1275, 266, 1275, 5397, 1879, 1879, 1879, 1879,
1879, 1331, 681, 681, 1331, 1275, 1275, 5208, 5208, 5208,
5208, 3399, 3399, 3399, 5208, 327, 327, 327, 327, 327,
327, 496, 496, 496, 496, 496, 496, 496, 496, 496,
496, 496, 496, 691, 4587, 4642, 691, 3767, 2607, 2607,
4527, 4527, 3767, 3946, 3946, 960, 2194, 2194, 2194, 2194,
960, 4282, 55, 55, 55, 920, 550, 920, 920, 957,
920, 822, 822, 5031, 5031, 5031, 5031, 5031, 5031, 5031,
4850, 5031, 5031, 5164, 5332, 3277, 3277, 3277, 3277, 3277,
3277, 3277, 3277, 3277, 3277, 3277, 3277, 3277, 3277, 3277,
3277, 2116, 2116, 2116, 4467, 4418, 4467, 1896, 1596, 1596,
1596, 1596, 1596, 1596, 1896, 1896, 4129, 4129, 4129, 4129,
4129, 4129, 4129, 4129, 4129, 4129, 4129, 1780, 1780, 1780,
1780, 1367, 1367, 1367, 2162, 1367, 2162, 2162, 2162, 2162,
2162, 2162, 2162, 238, 238, 238, 238, 238, 238, 238,
238, 238, 238, 238, 238, 238, 238, 238, 3830, 3830, 3533,
3533, 3533, 3732, 3533, 3533, 3533, 3533, 3533, 3533, 3533,
3533, 3533, 3533, 3533, 3533, 3533, 3533, 3533, 3533, 3533,
3533, 3533, 3533, 3470, 3533, 4588, 355, 3591, 3591, 355,
355, 355, 355, 355, 355, 355, 355, 355, 355, 355,
355, 355, 355, 355, 355, 3268, 1877, 1877, 1877, 4001,
4001, 4001, 4001, 4001, 4001, 4001, 4001, 4001, 4001, 4001,
4001, 4001, 4001, 4001, 4001, 4001, 4001, 656, 656, 4713,
4713, 656, 4713, 4713, 4713, 322, 322, 322, 322, 322,
322, 4218, 4218, 4218, 4218, 5182, 5182, 2845, 5182, 5182,
5182, 4518, 4518, 4518, 2069, 4518, 4518, 4518, 4518, 4518,
2069, 2069, 2069, 2069, 2069, 4650, 759, 4650, 759, 759,
4650, 552, 552, 552, 552, 552, 552, 2331, 2331, 2331,
2331, 2331, 2331, 2331, 2331, 2331, 2331, 2331, 6187, 6187, 5172,
5842, 5842, 5842, 5842, 5842, 5842, 5842, 5842, 5842, 2162,
2162, 2162, 2162, 2162, 5031, 2162, 2162, 2162, 2162, 2162,
2162, 3832, 2386, 5945, 6126, 3832, 5945, 5945, 5945, 1283,
1283, 1283, 1283, 1283, 1283, 1283, 1283, 1283, 1592, 4352,
4352, 3349, 3349, 6917, 6917, 3349, 6917, 3349, 2751, 3349,
3349, 6135, 2837, 2837, 6135, 6135, 6135, 2794, 6135, 4780,
4780, 4780, 4780, 4780, 4780, 4780, 4780, 4780, 4780, 4780,
4780, 1060, 1060, 1060, 1060, 1060, 2764, 1060, 1060, 1060,
1060, 1489, 308, 1423, 1423, 1423, 1423, 1423, 1423, 1423,
1423, 1423, 1423, 1423, 1423, 1423, 1423, 1423, 1423, 1423,
1423, 1423, 1423, 1423, 1423, 1423, 4558, 4330, 1654, 1654, 1654,
1654, 5061, 5061, 5061, 5061, 1425, 5061, 1425, 1425, 971,
971, 971, 716, 971, 971, 2814, 2616, 2814, 2814, 308,
308, 308, 308, 308, 308, 308, 308, 5380, 785, 5380,
5380, 5380, 5380, 5380, 1592, 5380, 5380, 5380, 5380, 5380,

(continuación)

5380, 5380, 3464, 4468, 4468, 2244, 2244, 2244, 2244, 2244,
2244, 2244, 2244, 2244, 2244, 3880, 3880, 3880, 3880, 3880,
3880, 60, 60, 60, 60, 60, 60, 734, 734, 5647,
5647, 5647, 5647, 5647, 5647, 3145, 6661, 3145, 3145, 3145,
3145, 3145, 3145, 3145, 3145, 2071, 2071, 2071, 2071, 2071,
2071, 4680, 4680, 676, 676, 676, 676, 676, 676, 676,
676, 676, 676, 2641, 2641, 2641, 2641, 2641, 2641, 6532,
6532, 6532, 6532, 6532, 6532, 5098, 5098, 5098, 5098, 1628,
5098, 1592, 1592, 1592, 1592, 1592, 1592, 1592, 1592, 55,
6221, 6221, 6221, 6221, 6221, 3977, 550, 550, 550, 550,
550, 3977, 550, 550, 3977, 4055, 3651, 3651, 3651, 3651,
550, 21, 21, 2639, 2639, 2639, 2639, 2639, 2639, 863,
863, 863, 863, 863, 1086, 4557, 4557, 4557, 4557, 5449,
5449, 277, 1497, 1497, 277, 1497, 1497, 277, 277, 1497,
277, 2432, 2432, 2432, 2432, 2432, 2432, 2432, 2432, 2603,
2603, 3041, 3041, 3041, 3041, 2603, 3041, 3041, 2702, 2702,
2603, 2603, 2603, 2603, 2603, 2603, 3041, 3041, 3041, 2702,
3041, 3041, 3041, 6043, 6043, 6043, 6043, 6043, 6043, 2365,
784, 784, 2365, 2365, 2365, 2365, 2365, 2365, 784, 784,
784, 2365, 784, 784, 4718, 4718, 4718, 4718, 4718, 4718,
4718, 55, 681, 681, 5658, 5658, 5658, 5658, 5658, 681,
5658, 681, 5658, 5658, 5658, 681, 681, 5658, 5658, 5658,
6510, 6510, 6510, 7236, 5014, 5014, 5014, 5014, 5014, 5014,
5014, 5014, 4116, 4116, 4116, 4116, 4116, 4116, 4116, 4116,
4116, 4116, 4885, 5997, 4885, 4885, 5997, 5997, 4885, 4885,
4885, 7266, 7266, 7266, 7266, 7266, 7266, 4885, 4885, 5997,
4885, 4885, 5997, 4885, 4885, 4885, 4885, 4885, 4885, 4885,
4885, 4885, 327, 327, 327, 327, 327, 327, 327, 327,
4650, 4650, 4650, 2396, 4650, 4650, 4650, 4650, 4650, 4650,
4650, 3676, 4650, 4650, 3676, 4650, 4650, 4650, 6481, 7188,
6481, 7188, 1195, 4368, 5475, 4518, 4518, 4518, 4518, 4518,
4518, 4518, 4518, 4518, 425, 425, 425, 425, 4272, 4272,
6009, 6009, 4650, 4650, 4650, 4650, 4650, 4650, 1267, 1267,
1267, 1267, 1229, 1229, 1229, 1229, 1229, 1229, 1229, 1229,
1229, 1229, 1229, 1229, 3357, 1229, 1229, 1229, 1229, 3357,
6476, 6476, 6476, 6476, 6476, 6476, 6476, 6476, 6305, 6305,
6305, 6305, 6305, 6305, 6305, 6305, 6305, 6305, 6305, 6305,
8903, 8903, 8903, 8903, 8903, 36, 8903, 36, 36, 36,
36, 36, 36, 36, 36, 36, 8903, 425, 425, 425,
425, 425, 425, 425, 425, 425, 425, 425, 425, 425, 425, 425,
425, 425, 6380, 6380, 6380, 367, 6380, 2877, 1916, 1916,
4512, 4195, 4195, 4512, 4195, 4195, 4195, 4512, 4512, 4771,
4771, 4771, 7692, 7692, 7692, 7692, 7692, 7692, 1839, 1839,
1839, 1839, 5201, 1839, 5201, 5201, 5201, 1953, 1413, 1413,
5740, 5957, 5740, 5740, 5740, 5740, 5957, 5957, 5957, 5740,
3017, 4290, 759, 759, 759, 1066, 1066, 759, 6569, 6569,
6569, 7782, 6569, 6569, 6569, 6569, 7782, 7782, 3651, 3651,
3651, 3651, 427, 427, 427, 3651, 427, 427, 427, 427,
427, 427, 2622, 2107, 2107, 2107, 7035, 7035, 7035, 7334,
7334, 7035, 7334, 7334, 7035, 7035, 7035, 7035, 7035, 7035,
7035, 6186, 3300, 3300, 7031, 3300, 3300, 7031, 3300, 3300,

(continuación)

3300,	3300,	3300,	3300,	7031,	3300,	3300,	3300,	6835,	6835,
5164,	5164,	5164,	7092,	7092,	5164,	5164,	5164,	5164,	5164,
3722,	3722,	869,	869,	869,	869,	869,	869,	869,	869,
869,	869,	5122,	5122,	5122,	5122,	5122,	4840,	5122,	5122,
5122,	5122,	5122,	5122,	5122,	5122,	4840,	4840,	5122,	5122,
3743,	5122,	5658,	5658,	5658,	5658,	2388,	5475,	4805,	4805,
4805,	4805,	2597,	2597,	2597,	6009,	6009,	6009,	3606,	6009,
1778,	1778,	1778,	1778,	1778,	1778,	1778,	1778,	1778,	1778,
4645,	1778,	1778,	1778,	1778,	2751,	4364,	4364,	4094,	4364,
4094,	4094,	1222,	1222,	1222,	1222,	1222,	1222,	5041,	5041,
5110,	5110,	5110,	5110,	1335,	5110,	5110,	5110,	5110,	5110,
383,	5110,	3610,	3610,	3610,	3610,	3610,	3610,	3610,	3610,
7965,	7965,	7965,	7965,	7965,	3610,	7965,	3610,	2814,	2814,
2814,	8731,	2814,	2814,	2814,	2814,	1335,	1026,	1026,	1026,
6683,	2129,	6683,	6683,	2129,	2129,	827,	827,	827,	827,
827,	827,	827,	827,	827,	827,	827,	827,	827,	827,
827,	827,	827,	827,	827,	827,	827,	827,	5723,	5017,
5723,	5723,	5723,	5723,	5723,	5017,	4055,	4055,	3756,	3756,
3756,	3756,	3756,	3756,	3756,	3756,	3756,	3756,	616,	6933,
7871,	7871,	7871,	7871,	7871,	7871,	5846,	5846,	5846,	5846,
2019,	2019,	2019,	2019,	2019,	2019,	5261,	5261,	5261,	5261,
5261,	5261,	1737,	1737,	2041,	1737,	6532,	6532,	6532,	5676,
442,	442,	442,	1859,	1859,	442,	3101,	3101,	3101,	3101,
4143,	4143,	4143,	4143,	4143,	4143,	4143,	4143,	4143,	4143,
4143,	4143,	5726,	3102,	3102,	3102,	3102,	3102,	4880,	4880,
4880,	4880,	4880,	4880,	2595,	2595,	7052,	7052,	7052,	4885,
7052,	7052,	7052,	7052,	5177,	7052,	7052,	7052,	691,	691,
691,	691,	2000,	2000,	2000,	2000,	2000,	2000,	5999,	2000,
2000,	2000,	2000,	2000,	2000,	2000,	3692,	3692,	3692,	3692,
3692,	3692,	6111,	6111,	3692,	6111,	3692,	3692,	5741,	5741,
5741,	5741,	5741,	5741,	5741,	5741,	5741,	5741,	8670,	10473,
8670,	8670,	8670,	10473,	8670,	4035,	4035,	8670,	5261,	8670,
4035,	4035,	5261,	4035,	4035,	8670,	4035,	5261,	5261,	4035,
6497,	10666,	734,	734,	734,	734,	734,	734,	9318,	9318,
8980,	8980,	9155,	9155,	9155,	9318,	5991,	9318,	314,	9318,
2378,	2378,	3102,	2378,	3102,	2378,	2854,	5416,	2889,	2889,
2889,	2889,	2889,	2889,	2889,	2889,	2889,	2889,	2889,	2889,
2889,	2889,	2889,	2889,	2931,	2931,	2889,	2889,	2889,	2889,
4240,	4240,	4240,	4240,	4240,	2162,	9862,	2162,	7246,	7246,
7246,	7246,	7434,	7246,	7246,	7246,	7246,	7246,	7246,	7246,
7246,	7246,	7246,	7246,	7246,	7246,	4333,	4333,	4333,	4333,
4333,	4333,	4333,	4333,	4333,	4333,	5244,	5244,	5244,	5244,
5244,	5244,	51,	51,	51,	51,	51,	51,	51,	51,
51,	51,	51,	51,	51,	51,	6447,	6810,	6810,	6447,
2921,	2921,	2046,	2046,	3424,	2046,	2046,	2046,	6599,	1911,
6599,	6599,	1911,	1911,	6599,	1911,	6599,	1911,	327,	327,
327,	327,	327,	327,	327,	327,	2678,	4094,	4094,	4094,
4094,	4094,	5633,	5633,	5633,	5985,	7152,	7152,	7152,	7152,
7152,	7152,	8279,	5283,	5283,	32,	5283,	32,	32,	32,
32,	32,	32,	32,	32,	32,	32,	32,	32,	32,
32,	32,	5283,	32,	32,	32,	5283,	32,	32,	32,

(continuación)

32, 32, 3634, 3634, 3634, 3634, 1158, 3634, 3634, 3634,
3634, 3634, 3634, 3634, 3634, 3634, 3101, 3101, 3101, 1392,
1392, 1392, 3101, 3101, 3101, 1392, 4929, 4929, 4929, 4929,
4929, 4929, 4929, 4929, 4929, 4929, 4929, 4929, 1803, 1803,
1803, 9232, 9232, 9232, 9232, 1803, 9232, 9232, 7228, 7228,
2129, 3435, 2129, 2129, 2129, 2641, 9158, 3351, 2641, 9158,
9158, 3351, 7167, 9158, 2129, 2129, 10358, 10358, 4330, 4330,
3814, 4330, 6012, 6012, 6012, 6012, 6012, 3814, 3814, 6012,
6012, 4330, 4330, 7264, 4330, 6012, 1446, 1446, 1446, 1446,
1446, 1446, 1446, 1446, 1446, 1446, 1446, 1446, 11793, 1446,
1446, 1446, 1446, 1446, 1446, 1446, 1446, 1446, 1446, 1446,
8729, 8729, 427, 8729, 8729, 8729, 8729, 427, 8729, 8729,
8729, 8729, 8729, 8729, 8729, 8729, 7741, 7741, 7741, 7741,
4400, 4400, 4400, 4400, 4400, 4400, 4400, 4400, 4400, 4400,
4400, 4400, 4400, 4400, 4400, 4400, 11052, 5355, 5355, 11052,
11052, 11052, 11052, 5355, 11052, 11052, 11052, 11052, 11052, 11052,
11052, 11052, 11052, 11052, 3090, 2757, 3090, 3090, 5310, 5310,
7829, 7829, 4155, 4155, 5740, 5740, 4155, 5740, 4155, 5740,
4155, 4155, 4155, 4155, 4155, 4155, 4155, 4155, 5740, 5740,
3017, 3017, 3017, 3017, 5920, 2420, 4001, 5920, 4001, 4001,
1641, 1641, 6807, 6807, 6807, 6807, 6807, 6807, 6807, 6807,
6807, 6807, 4460, 4460, 4460, 4460, 4460, 4460, 4460, 4460,
9860, 9860, 9860, 9860, 9860, 9860, 9860, 9860, 9860, 9860,
6596, 6596, 6596, 6596, 5190, 5190, 2599, 3921, 2599, 2599,
6569, 6569, 8982, 8982, 8982, 8982, 8982, 8982, 8982, 8982,
10063, 1655, 10063, 10063, 11787, 11787, 11787, 11787, 11787, 11787,
85, 85, 550, 550, 550, 550, 550, 550, 550, 550,
550, 550, 4328, 1089, 10623, 10623, 10623, 10623, 10623, 10623,
10623, 10623, 10623, 10623, 8757, 8757, 5876, 6540, 4682, 4682,
2308, 2308, 6219, 2308, 2308, 4664, 3360, 3360, 3360, 3360,
3360, 3360, 7152, 7152, 6024, 6024, 6024, 6024, 6024, 6024,
6024, 6024, 6024, 6024, 6024, 6024, 5319, 5319, 5319, 5319,
713, 713, 713, 713, 713, 713, 10037, 713, 10037, 10037,
10037, 10037, 10950, 10900, 10950, 10950, 10900, 10950, 10950, 10950,
10950, 10950, 10950, 10950, 10950, 10950, 10950, 10950, 10950, 10950,
4518, 4518, 4518, 4518, 2263, 2263, 9473, 9473, 9473, 7371,
3818, 3818, 3818, 3818, 3818, 7505, 3818, 3818, 3818, 3818,
7505, 3818, 3818, 3818, 7505, 3818, 3818, 3818, 3818, 3818,
6810, 6810, 6810, 6810, 1790, 1118, 1118, 1790, 1118, 1790,
1118, 1790, 8982, 8982, 8982, 8982, 1165, 8982, 9317, 9317,
9317, 9317, 173, 173, 173, 173, 173, 173, 173, 173,
5069, 5069, 5069, 5069, 8058, 8058, 8058, 8058, 8058, 8058,
8058, 8058, 8058, 8058, 8058, 8058, 8058, 8058, 6230, 6574,
6574, 6574, 6574, 6574, 6574, 6574, 6574, 6574, 6574, 6574,
3101, 3101, 3101, 3101, 3101, 3101, 3101, 3101, 3101, 3101,
3101, 3101, 7502, 7502, 7502, 7502, 6313, 6313, 4518, 4518,
4518, 1606, 1606, 1606, 1606, 1606, 9284, 9284, 9284, 9837,
3830, 3269, 3269, 1606, 1606, 1606, 9284, 1606, 6912, 9284,
6912, 6912, 9284, 9284, 1606, 9284, 6545, 9284, 8502, 8502,
8502, 8502, 398, 398, 398, 398, 398, 398, 398, 398,
398, 398, 398, 398, 398, 398, 398, 398, 398, 398,

(continuación)

398,	398,	398,	398,	398,	398,	9158,	5275,	5275,	5275,
5275,	5275,	8169,	8169,	8169,	8169,	8169,	8169,	8962,	11967,
11967,	11967,	8962,	8962,	8962,	8962,	11967,	11967,	8962,	8962,
12618,	12618,	12618,	12618,	12618,	12618,	12618,	12618,	12618,	12618,
12618,	12618,	7414,	7414,	7414,	7414,	7414,	7414,	7414,	7414,
7414,	7414,	7414,	7414,	7414,	7414,	1426,	1426,	1426,	1426,
1426,	1426,	7020,	8968,	8731,	8968,	6251,	6251,	327,	327,
327,	327,	3649,	3649,	3649,	4765,	4765,	4765,	4765,	7688,
4765,	7688,	4765,	4765,	4765,	4765,	4765,	4765,	3267,	7688,
4682,	3115,	3115,	3115,	3115,	3115,	5929,	5929,	5929,	5929,
5929,	5929,	5929,	5929,	5929,	5929,	9332,	5929,	5321,	5321,
5321,	5321,	5321,	5321,	5321,	8753,	4410,	4410,	4410,	4410,
4410,	4410,	3300,	3300,	3300,	3300,	4069,	4069,	4069,	818,
818,	818,	4069,	4069,	4069,	4069,	4069,	10900,	656,	656,
3830,	3830,	3830,	3830,	3830,	3830,	3830,	3830,	3830,	3830,
3830,	3830,	9632,	9632,	9632,	9632,	9632,	7871,	9632,	7871,
9632,	9632,	7871,	7871,	7871,	7871,	7871,	9632,	7871,	7871,
7871,	7871,	7871,	7871,	7871,	7871,	7871,	7871,	7871,	7871,
7871,	7871,	5261,	10356,	5261,	5261,	5261,	5261,	5261,	5261,
5261,	5261,	5261,	5261,	6979,	10356,	10010,	10356,	5846,	10089,
5267,	5267,	5267,	5267,	4823,	5267,	10958,	11674,	11674,	11674,
11674,	11674,	11674,	11674,	10958,	10958,	11674,	11674,	11674,	11674,
8410,	11674,	8410,	734,	734,	734,	1885,	8410,	7020,	7020,
7020,	7020,	7020,	7020,	7020,	7020,	7020,	7020,	7020,	7020,
7020,	7020,	1055,	1055,	1055,	1055,	1055,	1055,	7167,	7167,
7167,	7167,	7167,	7167,	7167,	7167,	7167,	7167,	8443,	8443,
7471,	7471,	7471,	7471,	8443,	8443,	7471,	8443,	8443,	8443,
3269,	3269,	3269,	3269,	3269,	3269,	8286,	8286,	5925,	5925,
5925,	5925,	4901,	4901,	4901,	4901,	4901,	4901,	4901,	4901,
13343,	13343,	13343,	13343,	13343,	13343,	13343,	13343,	13343,	13343,
8111,	8111,	8111,	8111,	8952,	6978,	2484,	7742,	2484,	7742,
2484,	7742,	1953,	1953,	1953,	1953,	1953,	1953,	1953,	1953,
1953,	1953,	1953,	1953,	1953,	1953,	1953,	1953,	1953,	1953,
1953,	1953,	1953,	1964,	1201,	1964,	1501,	1501,	9004,	1501,
1501,	1501,	1501,	1501,	1501,	1501,	1501,	1501,	1501,	1501,
9497,	9497,	9497,	9497,	9497,	9497,	7751,	7751,	7751,	7751,
13738,	13738,	13738,	13738,	13738,	13738,	13738,	13738,	2387,	2387,
2387,	2387,	2387,	2387,	10089,	10245,	10245,	10245,	10245,	10245,
14486,	767,	8427,	767,	767,	8427,	1099,	8427,	8427,	8427,
8427,	8427,	767,	8427,	767,	767,	767,	767,	450,	15435,
8482,	8482,	8482,	4195,	18171,	15435,	15435,	450,	6203,	6203,
13051,	13051,	6143,	13051,	13051,	13051,	13051,	7956,	7218,	7218,
14111,	14111,	8643,	8643,	8643,	8643,	5832,	5832,	5832,	5832,
5832,	5832,	12089,	12089,	12089,	12089,	12089,	12089,	12089,	12089,
12089,	12089,	12089,	12089,	12089,	12089,	12089,	12089,	12089,	12089,
12089,	12089,	585,	585,	585,	585,	585,	585,	6468,	6468,
6468,	6468,	795,	10272,	10272,	1620,	10272,	12097,	12097,	12097,
12097,	12097,	10272,	12097,	10272,	12097,	1997,	5561,	5474,	5474,
14969,	14969,	11391,	11391,	11391,	11391,	2518,	2518,	2518,	2518,
2518,	2518,	2518,	2518,	2518,	2518,	2518,	2518,	2518,	2518,
6594,	6594,	6594,	6594,	6594,	6594,	6170,	16423,	6170,	6170,

(continuación)

16423,	16423,	16423,	6170,	16423,	16423,	6170,	16423,	5601,	5601,
5601,	5601,	5601,	5601,	5601,	5601,	1934,	5601,	5601,	5601,
5601,	5601,	5601,	5601,	5601,	5601,	5601,	5601,	5601,	8349,
8349,	8349,	10612,	3223,	10612,	10612,	10612,	10612,	10612,	7965,
10612,	3223,	3830,	3830,	3830,	3830,	3830,	3830,	10583,	10583,
10583,	10583,	10583,	10583,	516,	10583,	11222,	11222,	11222,	1373,
11222,	11222,	11222,	11222,	11222,	11222,	7871,	7871,	17470,	17470,
17470,	17470,	17470,	17470,	17470,	17431,	17431,	17431,	17470,	17470,
17470,	8502,	17470,	17431,	17470,	17470,	17431,	17431,	17431,	17470,
17470,	17431,	17470,	17431,	17470,	17470,	17431,	17470		

APÉNDICE B.1 VALORES PARA LA TABLA V_0

Estos valores representan un conjunto ejemplar de valores para la Tabla V_0 descrita en la solicitud anterior, en la Sección B.5.4.1. Cada entrada es un entero de 32 bits en representación decimal. Los valores deberían leerse desde la primera columna, de arriba abajo, luego la segunda columna, de arriba abajo, y así sucesivamente.

251291136	2581671944	1521339547	2068983570	2859178611
3952231631	3312220480	3041843489	2247491078	3284308411
3370958628	681232419	420130494	3669524410	3819792700
4070167936	307306866	10677091	1575146607	3557526733
123631495	4112503940	515623176	828029864	451874476
3351110283	1158111502	3457502702	3732001371	1740576081
3218676425	709227802	2115821274	3422026452	3592838701
2011642291	2724140433	2720124766	3370954177	1709429513
774603218	4201101115	3242576090	4006626915	3702918379
2402805061	4215970289	854310108	543812220	3533351328
1004366930	4048876515	425973987	1243116171	1641660745
1843948209	3031661061	325832382	3928372514	179350258
428891132	1909085522	1796851292	2791443445	2380520112
3746331984	510985033	2462744411	4081325272	3936163904
1591258008	1361682810	1976681690	2280435605	3685256204
3067016507	129243379	1408671665	885616073	3156252216
1433388735	3142379587	1228817808	616452097	1854258901
504005498	2569842483	3917210003	3188863436	2861641019
2032657933	3033268270	263976645	2780382310	3176611298
3419319784	1658118006	2593736473	2340014831	834787554
2805686246	932109358	2471651269	1208439576	331353807
3102436986	1982290045	4291353919	258356309	517858103
3808671154	2983082771	650792940	3837963200	3010168884
2501582075	3007670818	1191583883	2075009450	4012642001
3978944421	3448104768	3046561335	3214181212	2217188075
246043949	683749698	2466530435	3303882142	3756943137
4016898363	778296777	2545983082	880813252	3077882590
649743608	1399125101	969168436	1355575717	2054995199
1974987508	1939403708	2019348792	207231484	3081443129
2651273766	1692176003	2268075521	2420803184	3895398812
2357956801	3868299200	1169345068	358923368	1141097543
689605112	1422476658	3250240009	1617557768	2376261053
715807172	593093658	3963499681	3272161958	2626898255
2722736134	1878973865	2560755113	1771154147	2554703076

(continuación)

191939188	2526292949	911182396	2842106362	401233789
3535520147	1591602827	760842409	1751209208	1460049922
3277019569	3986158854	3569308693	1421030790	678083952
1470435941	3964389521	2687243553	658316681	1064990737
3763101702	2695031039	381854665	194065839	940909784
3232409631	1942050155	2613828404	3241510581	1673396780
122701163	424618399	2761078866	38625260	528881783
3920852693	1347204291	1456668111	301875395	1712547446
782246947	2669179716	883760091	4176141739	3629685652
372121310	2434425874	3294951678	297312930	1358307511
2995604341	2540801947	1604598575	2137802113	
2045698575	1384069776	1985308198	1502984205	
2332962102	4123580443	1014570543	3669376622	
4005368743	1523670218	2724959607	3728477036	
218596347	2708475297	3062518035	234652930	
3415381967	1046771089	3115293053	2213589897	
4207612806	2229796016	138853680	2734638932	
861117671	1255426612	4160398285	1129721478	
3676575285	4213663089	3322241130	3187422815	

APÉNDICE B.2 VALORES PARA LA TABLA V_1

Estos valores representan un conjunto ejemplar de valores para la Tabla V_1 descrita en la solicitud anterior, en la Sección B.5.4.1. Cada entrada es un entero de 32 bits en representación decimal. Los valores deberían leerse desde la primera columna, de arriba abajo, luego la segunda columna, de arriba abajo, y así sucesivamente.

807385413	3843885867	3545667983	644189126	922673938
2043073223	4201106668	332038910	226475395	3877430102
3336749796	415906198	976628269	307789415	3422391938
1302105833	19296841	3123492423	1196105631	1414347295
2278607931	2402488407	3041418372	3191691839	1971054608
541015020	2137119134	2258059298	782852669	3061798054
1684564270	1744097284	2139377204	1608507813	830555096
372709334	579965637	3243642973	1847685900	2822905141
3508252125	2037662632	3226247917	4069766876	167033190
1768346005	852173610	3674004636	3931548641	1079139428
1270451292	2681403713	2698992189	2526471011	4210126723
2603029534	1047144830	3453843574	766865139	3593797804
2049387273	2982173936	1963216666	2115084288	429192890
3891424859	910285038	3509855005	4259411376	372093950
2152948345	4187576520	2358481858	3323683436	1779187770
4114760273	2589870048	747331248	568512177	3312189287
915180310	989448887	1957348676	3736601419	204349348
3754787998	3292758024	1097574450	1800276898	452421568
700503826	506322719	2435697214	4012458395	2800540462
2131559305	176010738	3870972145	1823982	3733109044
1308908630	1865471968	1888833893	27980198	1235082423
224437350	2619324712	2914085525	2023839966	1765319556
4065424007	564829442	4161315584	869505096	3174729780
3638665944	1996870325	1273113343	431161506	3762994475
1679385496	339697593	3269644828	1024804023	3171962488

(continuación)

3431345226	4071072948	3681293816	1853869307	442160826
1779595665	3618966336	412536684	3393537983	198349622
3068494238	2111320126	1156034077	1500703614	45942637
1424062773	1093955153	3823026442	3019471560	1324086311
1033448464	957978696	1066971017	1351086955	2901868599
4050396853	892010560	3598330293	3096933631	678860040
3302235057	1854601078	1979273937	3034634988	3812229107
420600373	1873407527	2079029895	2544598006	19936821
2868446243	2498544695	1195045909	1230942551	1119590141
311689386	2694156259	1071986421	3362230798	3640121682
259047959	1927339682	2712821515	159984793	3545931032
4057180909	1650555729	3377754595	491590373	2102949142
1575367248	183933047	2184151095	3993872886	2828208598
4151214153	3061444337	750918864	3681855622	3603378023
110249784	2067387204	2585729879	903593547	4135048896
3006865921	228962564	4249895712	3535062472	
4293710613	3904109414	1832579367	1799803217	
3501256572	1595995433	1192240192	772984149	
998007483	1780701372	946734366	895863112	
499288295	2463145963	31230688	1899036275	
1205710710	307281463	3174399083	4187322100	
2997199489	3237929991	3549375728	101856048	
640417429	3852995239	1642430184	234650315	
3044194711	2398693510	1904857554	3183125617	
486690751	3754138664	861877404	3190039692	
2686640734	522074127	3277825584	525584357	
2394526209	146352474	4267074718	1286834489	
2521660077	4104915256	3122860549	455810374	
49993987	3029415884	666423581	1869181575	

REIVINDICACIONES

1. Un procedimiento de codificación de datos para su transmisión desde un origen (101, 105) a un destino (170) por un canal de comunicaciones (145), comprendiendo el procedimiento:

obtener un conjunto ordenado de símbolos de entrada que representan datos a codificar;

seleccionar una pluralidad de formaciones de valores de campo, en la que cada formación de campo es una formación de campo finito y la pluralidad de formaciones de campo incluyen al menos dos distintas formaciones de campo finito;

estando el procedimiento **caracterizado por** generar una estructura de datos que representa una matriz de coeficientes con entradas provenientes de al menos dos de entre la pluralidad de formaciones de campo, que son formaciones distintas de campo finito;

generar símbolos de salida como combinaciones lineales de símbolos de entrada, con coeficientes tomados de la estructura de datos que representa la matriz de coeficientes; y
usar los símbolos de salida generados y una codificación para los datos, para su transmisión desde el origen al destino.

2. El procedimiento de la reivindicación 1, en el que la estructura de datos que representa una matriz de coeficientes es una formación bidimensional de valores de celda, representando cada valor de celda un coeficiente de un símbolo de entrada en la generación de un símbolo de salida, de modo que, cuando un coeficiente no es cero, o cero módulo alguna base, el valor del correspondiente símbolo de salida depende del valor del correspondiente símbolo de entrada.

3. El procedimiento de la reivindicación 1, en el que la estructura de datos que representa una matriz de coeficientes es un conjunto de reglas que especifican valores de coeficientes, y en el que, además, cuando una regla indica que un coeficiente no es cero, o cero módulo alguna base, el valor del correspondiente símbolo de salida depende del valor del correspondiente símbolo de entrada.

4. El procedimiento de la reivindicación 1, en el que la generación de una estructura de datos que representa una matriz de coeficientes, con entradas de al menos dos de las formaciones de campo, que son formaciones distintas de campo finito, es una generación que usa una primera formación de campo, que es una primera formación de campo finito, y una segunda formación de campo, que es una segunda formación de campo finito, en el que la primera formación de campo finito y la segunda formación de campo finito son distintas, y en el que, además, tanto el primer campo finito como el segundo campo finito son cada uno seleccionados a partir del conjunto de campos que consiste en los Campos de Galois GF(2), GF(4), GF(16) y GF(256).

5. El procedimiento de la reivindicación 1, que comprende además:

generar una pluralidad de símbolos redundantes a partir del conjunto ordenado de símbolos de entrada, en el que cada símbolo redundante es generado en base a un conjunto de restricciones lineales sobre uno o más de los símbolos de entrada, y otros símbolos redundantes, con coeficientes sobre campos finitos; y

generar una pluralidad de símbolos de salida a partir del conjunto combinado de símbolos de entrada y redundantes, en el que cada símbolo de salida es generado como una combinación lineal de uno o más entre el conjunto combinado de símbolos de entrada y redundantes, con coeficientes escogidos de campos finitos.

6. Un procedimiento (1000, 1025, 1050) de descodificación de datos procedentes de una transmisión recibida en un destino desde un origen, por un canal de comunicaciones (145), comprendiendo el procedimiento:

recibir (1005, 1055) al menos alguno entre una pluralidad de símbolos de salida generados a partir de un conjunto ordenado de símbolos de entrada, que fueron codificados en la pluralidad de símbolos de salida, en el que cada símbolo de salida fue generado como una combinación lineal de uno o más de los símbolos de entrada, con coeficientes escogidos de una matriz de coeficientes que tiene entradas de al menos dos formaciones de campo finito, que son formaciones distintas de campo finito, y en el que al menos un coeficiente es un miembro del primer campo finito y al menos otro coeficiente es un miembro de un segundo campo finito, y no es un miembro del primer campo finito; y

regenerar (1010, 1020, 1060, 1075) el conjunto ordenado de símbolos de entrada, con un grado deseado de precisión, a partir de la recepción de cualquier número predeterminado de los símbolos de salida.

7. El procedimiento de la reivindicación 6, en el que la pluralidad de símbolos de salida incluye símbolos de salida generados a partir de un conjunto combinado de símbolos de entrada y una pluralidad de símbolos redundantes, en el que cada símbolo de salida es generado como una combinación lineal de uno o más entre un conjunto combinado de símbolos de entrada y redundantes, con coeficientes escogidos de campos finitos, y en el que la pluralidad de símbolos redundantes es generada a partir del conjunto ordenado de símbolos de entrada, en el que cada símbolo redundante es generado en base a un conjunto de restricciones lineales sobre uno o más de los símbolos de entrada y otros símbolos redundantes, con coeficientes sobre campos finitos.

8. El procedimiento de la reivindicación 6, en el que los campos finitos son tales que una primera formación de campo finito y una segunda formación de campo finito son distintas, y tanto el primer campo finito como el segundo campo finito son seleccionados cada uno entre el conjunto de campos que consiste en los Campos de Galois GF(2), GF(4), GF(16) y GF(256).

9. El procedimiento de las reivindicaciones 1 o 6, en el que el número de símbolos de salida únicos que pueden ser generados a partir del conjunto de símbolos de entrada, para cualquier conjunto de valores fijos para los símbolos de entrada, es independiente de los tamaños de formaciones de campo.

10. El procedimiento de las reivindicaciones 4 u 8, en el que la primera formación de campo finito es GF(2) y la segunda formación de campo finito es GF(256).

11. El procedimiento de las reivindicaciones 4 u 8, en el que la primera formación de campo finito es GF(2) y la segunda formación de campo finito es GF(4).

12. El procedimiento de las reivindicaciones 4 u 8, en el que la primera formación de campo finito es GF(4) y la segunda formación de campo finito es GF(16).

13. El procedimiento de las reivindicaciones 4 u 8, en el que la primera formación de campo finito es GF(16) y la segunda formación de campo finito es GF(256).

14. El procedimiento de las reivindicaciones 4 u 8, en el que la primera formación de campo finito es más pequeña que la segunda formación de campo finito.

15. El procedimiento de las reivindicaciones 4 u 8, en el que la primera formación de campo finito es más grande que la segunda formación de campo finito.

16. Un dispositivo para codificar datos para su transmisión desde un origen (101, 105) a un destino (170) por un canal de comunicaciones (145), comprendiendo el dispositivo:

medios para obtener un conjunto ordenado de símbolos de entrada que representan datos a codificar;

medios para seleccionar una pluralidad de formaciones de valores de campo, en el que cada formación de campo es una formación de campo finito y la pluralidad de formaciones de campo incluyen al menos dos formaciones distintas de campo finito;

estando el dispositivo **caracterizado por** comprender además

medios para generar una estructura de datos que representa una matriz de coeficientes con entradas de al menos dos de entre la pluralidad de formaciones de campo, que son formaciones distintas de campo finito;
medios para generar símbolos de salida como combinaciones lineales de símbolos de entrada, con coeficientes tomados de la estructura de datos que representa la matriz de coeficientes; y

medios para usar los símbolos de salida generados y una codificación para los datos, para su transmisión desde el origen al destino.

17. Un dispositivo para descodificar datos procedentes de una transmisión recibida en un destino desde un origen, por un canal de comunicaciones (145), comprendiendo el dispositivo:

medios para recibir (1005, 1055) al menos alguno entre una pluralidad de símbolos de salida generados a partir de un conjunto ordenado de símbolos de entrada que fueron codificados en la pluralidad de símbolos de salida, en el que cada símbolo de salida fue generado como una combinación lineal de uno o más de los símbolos de entrada, con coeficientes escogidos de una matriz de coeficientes que tiene entradas de al menos dos formaciones de campo finito, que son formaciones distintas de campo finito, en el que al menos un coeficiente es un miembro de un primer campo finito y al menos otro coeficiente es un miembro de un segundo campo finito, y no es un miembro del primer campo finito; y

medios para regenerar (1010, 1020, 1060, 1075) el conjunto ordenado de símbolos de entrada, con un grado deseado de precisión, a partir de la recepción de cualquier número predeterminado de los símbolos de salida.

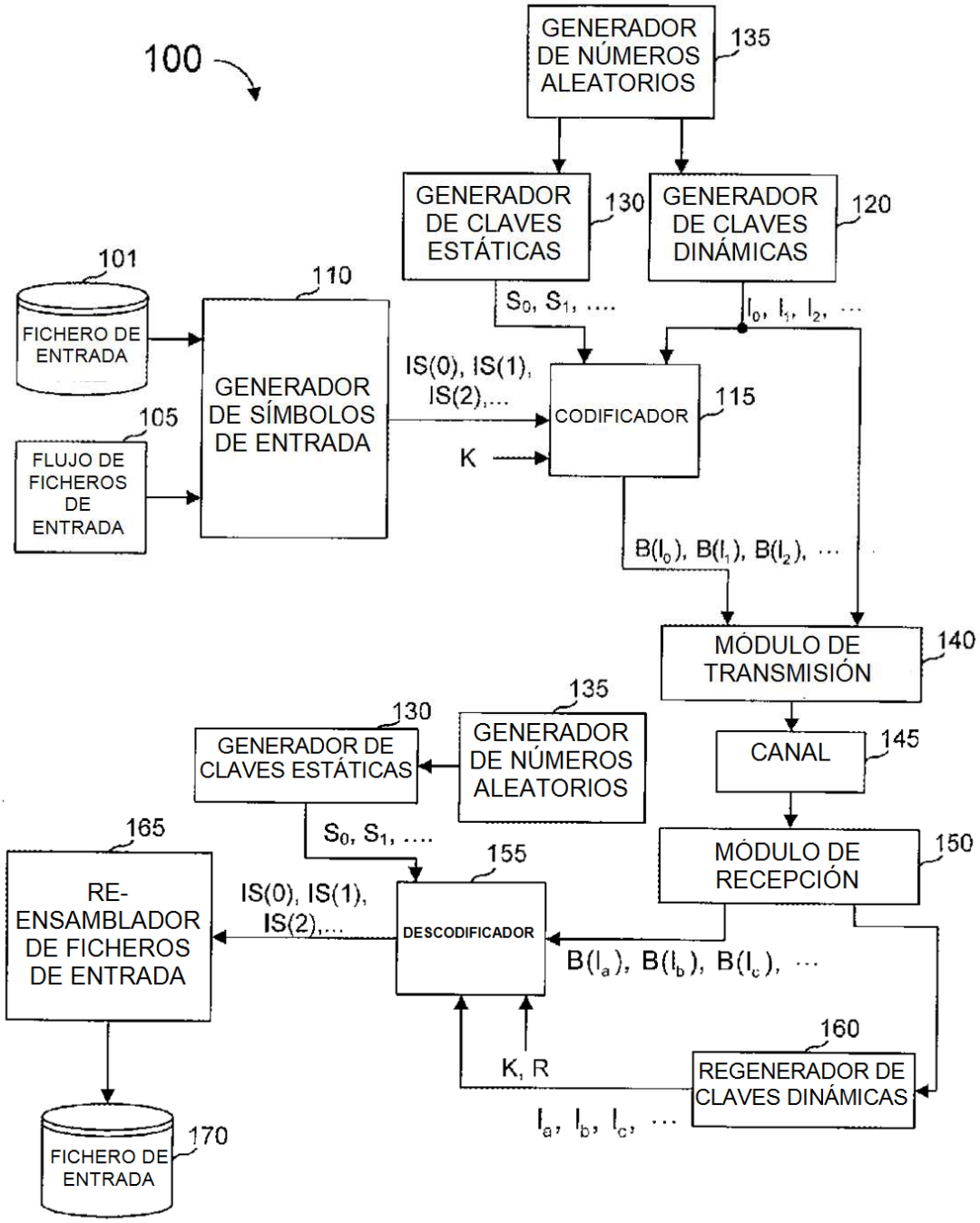
100
GENERADOR DE NÚMEROS ALEATORIOS
135
GENERADOR DE CLAVES ESTÁTICAS
130
GENERADOR DE CLAVES DINÁMICAS
120
101
FICHERO DE ENTRADA
110
GENERADOR DE SÍMBOLOS DE ENTRADA
105
FLUJO DE FICHEROS DE ENTRADA
S0, S1, ...
I0, I1, I2, ...
IS(0), IS(1), IS(2),...
CODIFICADOR
115
K
B(I0), B(I1), B(I2), ...
MÓDULO DE TRANSMISIÓN
140
135
130
GENERADOR DE CLAVES ESTÁTICAS
GENERADOR DE NÚMEROS ALEATORIOS
CANAL
145
165
S0, S1, ...
155
MÓDULO DE RECEPCIÓN
150
RE-ENSAMBLADOR DE FICHEROS DE ENTRADA
IS(0), IS(1), IS(2),...
DESCODIFICADOR
B(Ia), B(Ib), B(Ic), ...
160
K, R
REGENERADOR DE CLAVES DINÁMICAS
Ia, Ib, Ic, ...
FICHERO DE ENTRADA
170

Figura 1

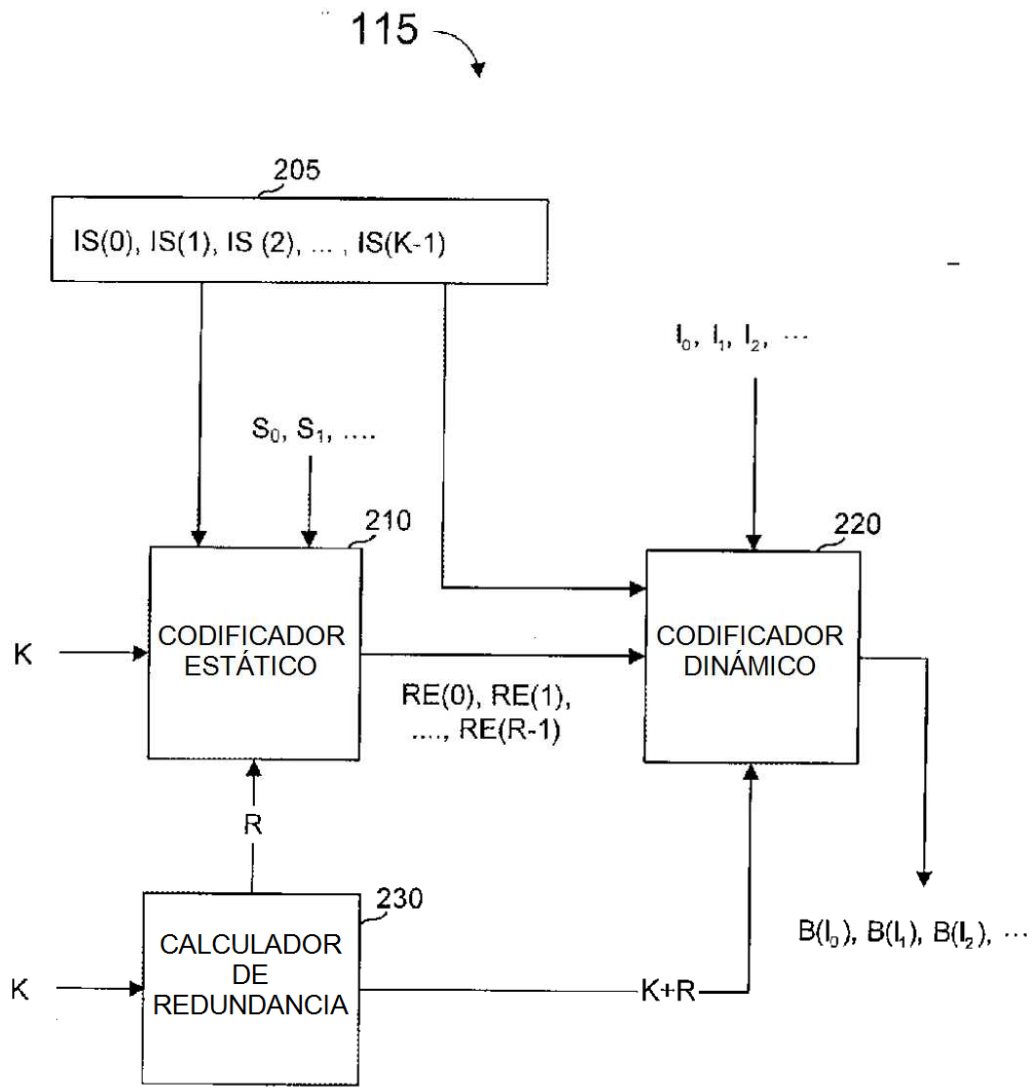
115
205
IS(0), IS(1), IS (2), ... , IS(K-1)
I_0, I_1, I_2, ...
S_0, S_1, ...
210
220
K
CODIFICADOR ESTÁTICO
RE(0), RE(1), ..., RE(R-1)
CODIFICADOR DINÁMICO
R
230
CALCULADOR DE REDUNDANCIA
K
K+R
B(I_0), B(I_1), B(I_2), ...

Figura 2

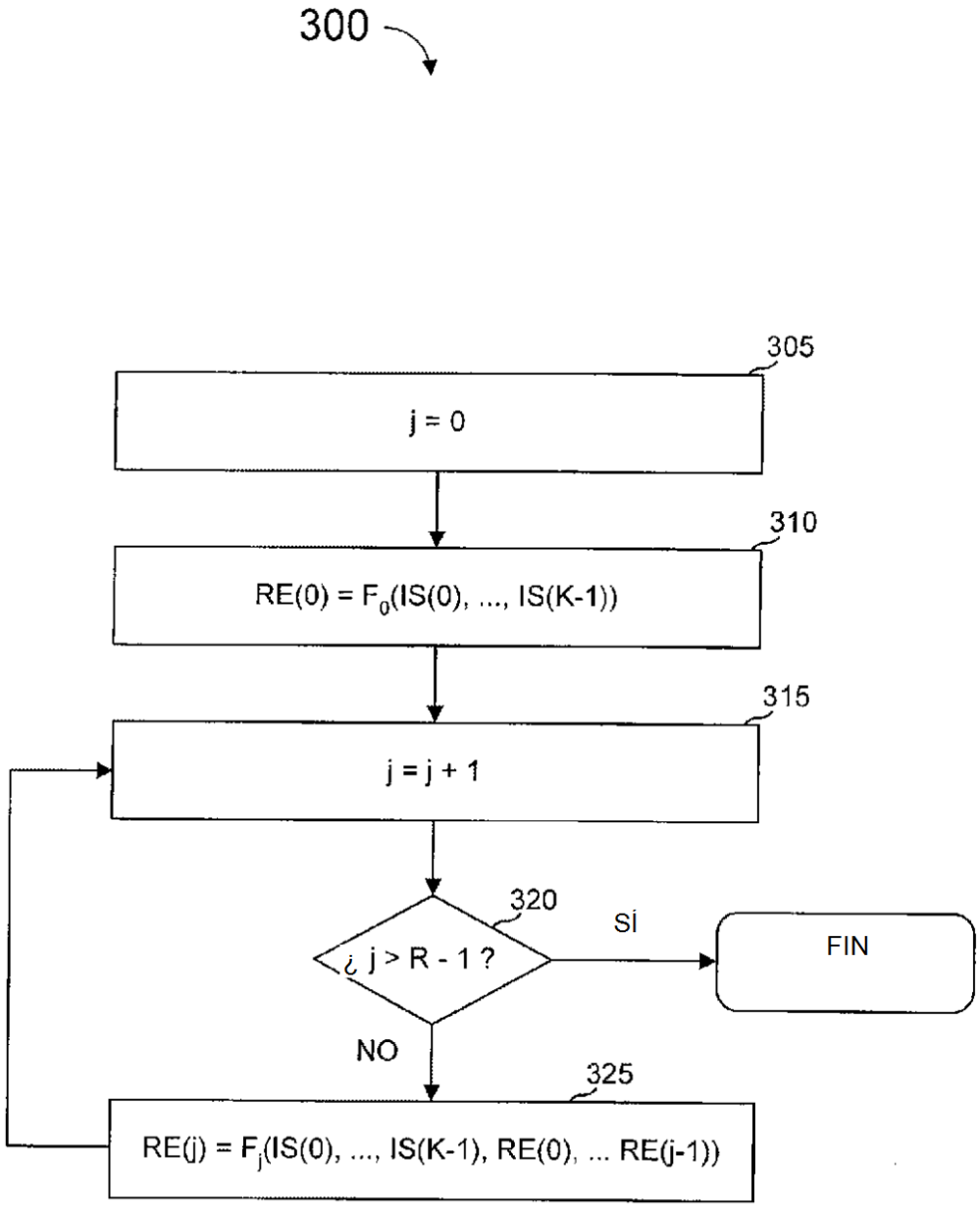
300
305
j = 0
310
RE(0) = F0(IS(0), ..., IS(K-1))
315
j = j + 1
320
¿ j > R - 1 ?
SÍ
FIN
NO
325
RE(j) = Fj(IS(0), ..., IS(K-1), RE(0), ... RE(j-1))

Figura 3

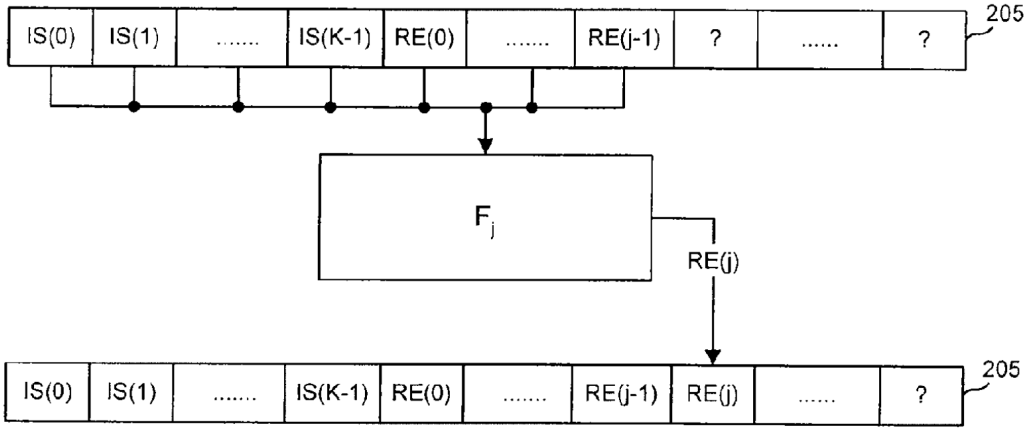

Figura 4

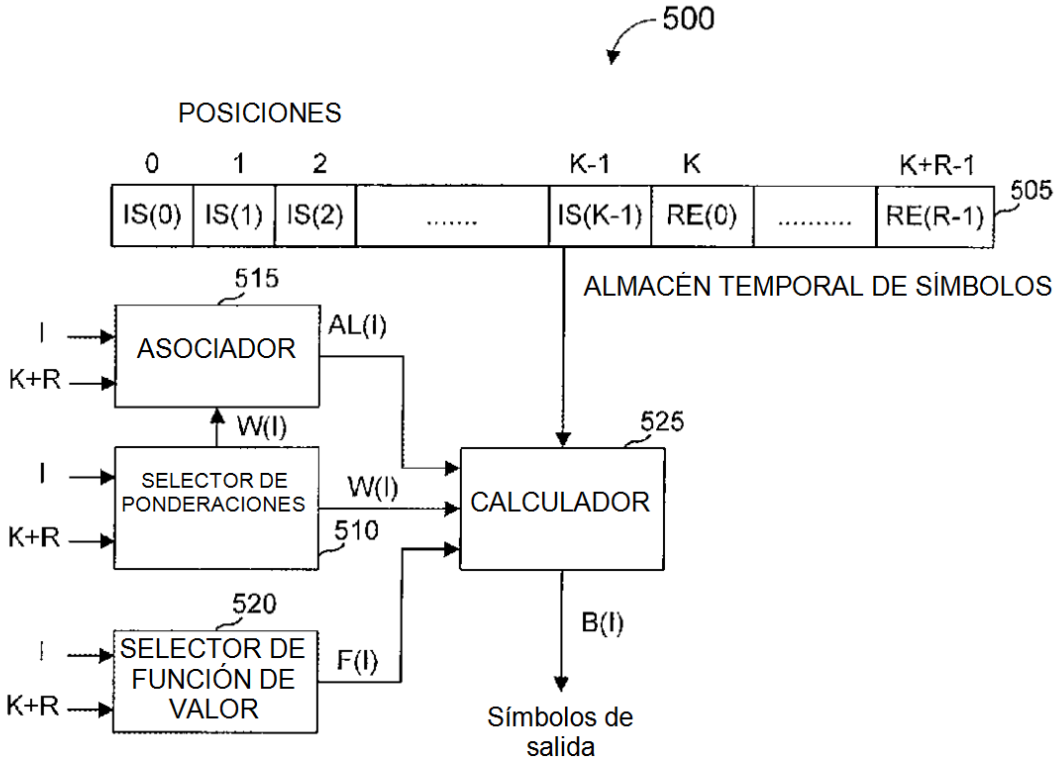

Figura 5

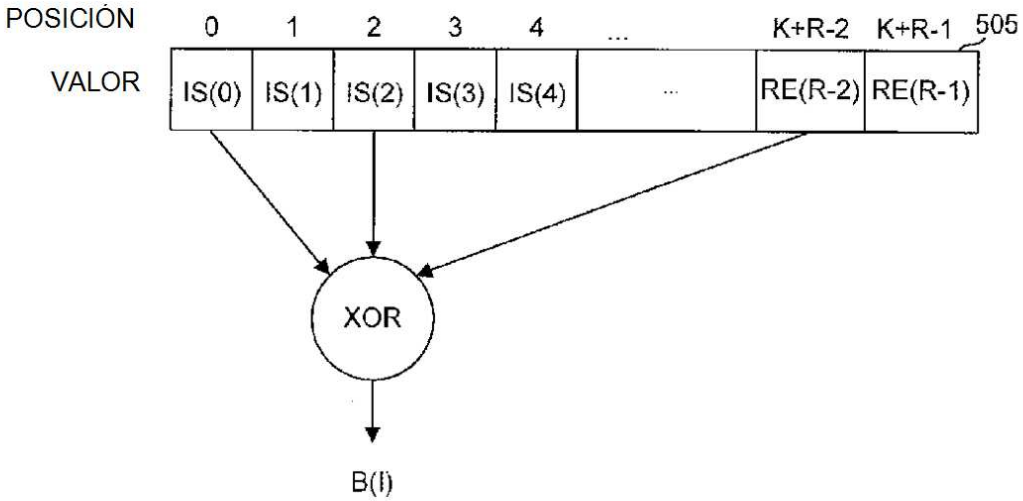

Figura 6

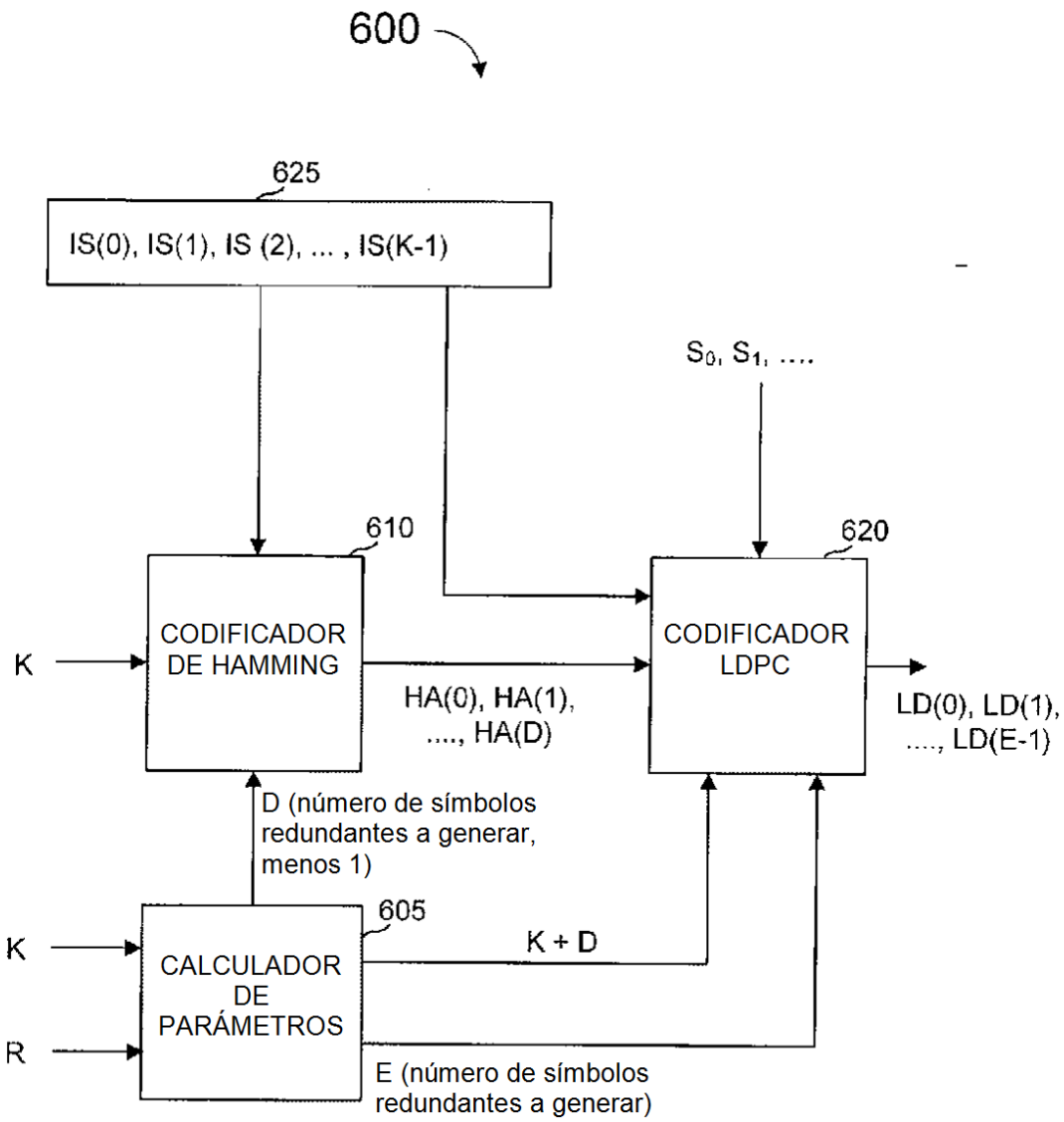
600
625
IS(0), IS(1), IS (2), ... , IS(K-1)
S0, S1, ...
610
620
CODIFICADOR DE HAMMING
CODIFICADOR LDPC
K
HA(0), HA(1), ..., HA(D)
LD(0), LD(1), ..., LD(E-1)
D (número de símbolos redundantes a generar, menos 1)
605
K
K + D
CALCULADOR DE PARÁMETROS
R
E (número de símbolos redundantes a generar)

Figura 7

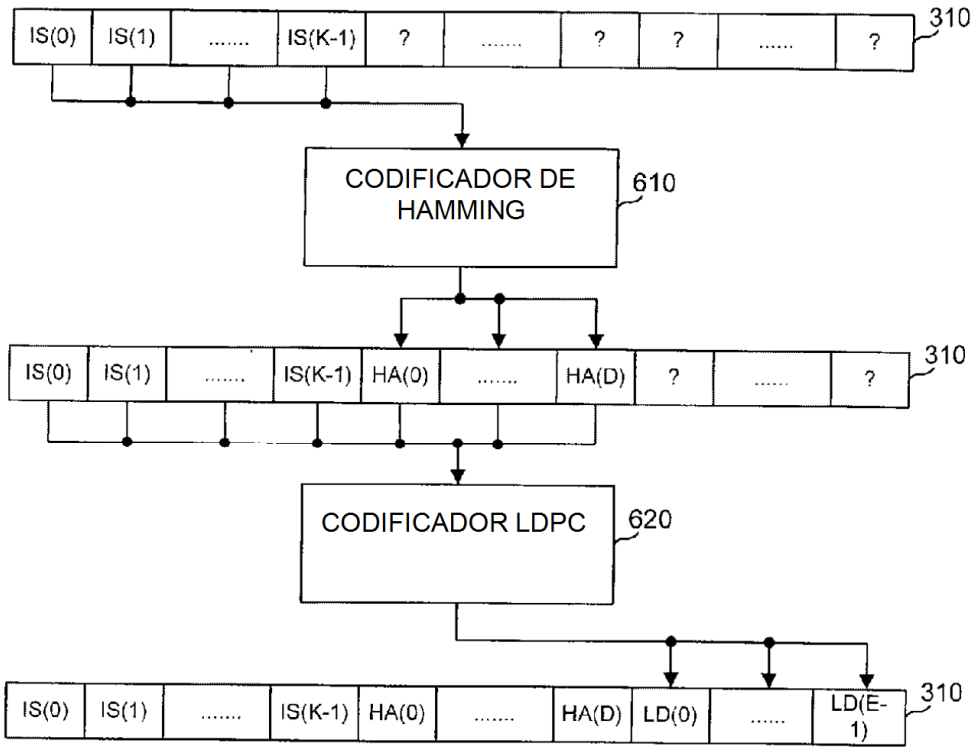
IS(0)
IS(1)
.....
IS(K-1)
?
.....
?
?
.....
?
310
CODIFICADOR DE HAMMING
610
IS(0)
IS(1)
.....
IS(K-1)
HA(0)
.....
HA(D)
?
.....
?
310
CODIFICADOR LDPC
620
IS(0)
IS(1)
.....
IS(K-1)
HA(0)
.....
HA(D)
LD(0)
.....
LD(E-1)
310

Figura 8

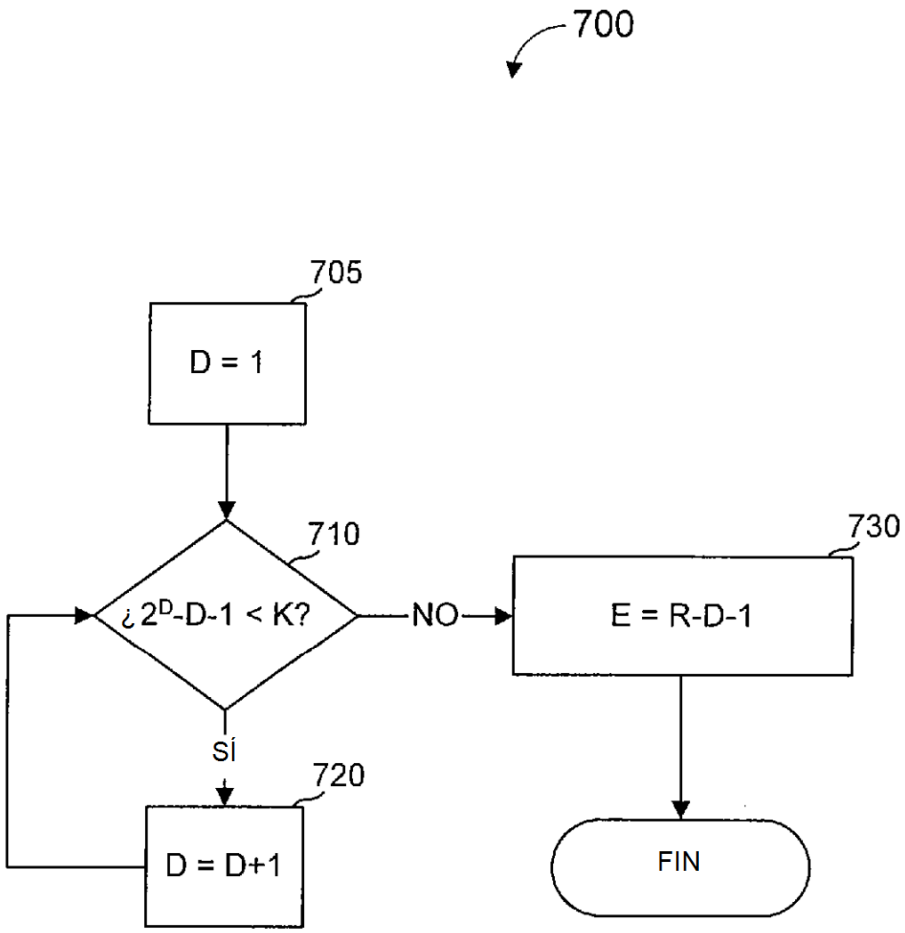

Figura 9

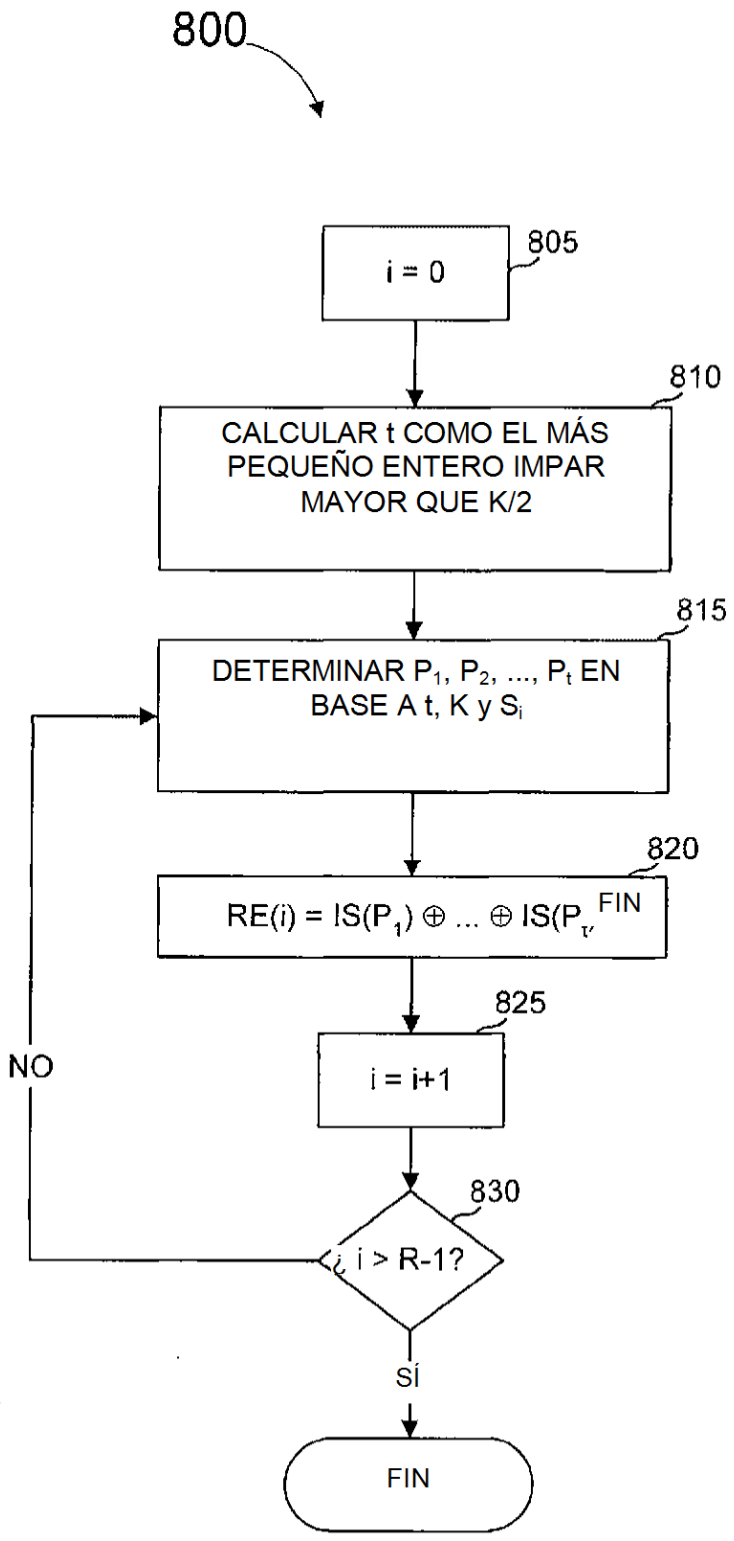
800
i = 0
805
810
CALCULAR t COMO EL MÁS PEQUEÑO ENTERO IMPAR MAYOR QUE K/2
815
DETERMINAR P1, P2, ..., Pt EN BASE A t, K y Si
820
RE(i) = IS(P1) ⊕ ... ⊕ IS(Pτ' FIN
825
i = i+1
NO
830
¿ i > R-1?
SÍ
FIN

Figura 10

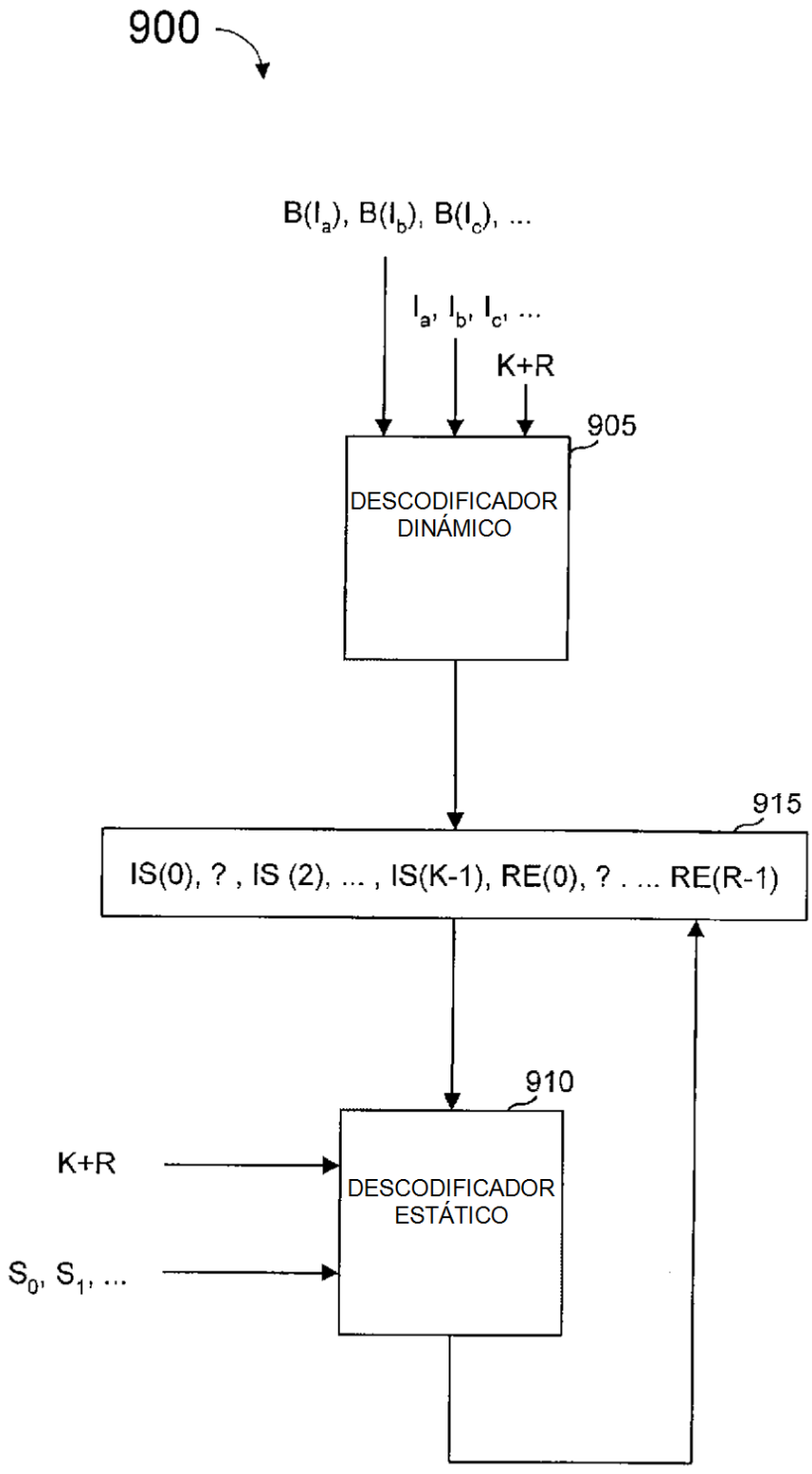
900
B(I_a), B(I_b), B(I_c), ...
I_a, I_b, I_c, ...
K+R
905
DESCODIFICADOR DINÁMICO
915
IS(0), ? , IS (2), ... , IS(K-1), RE(0), ? . ... RE(R-1)
910
K+R
S_0, S_1, ...
DESCODIFICADOR ESTÁTICO

Figura 11

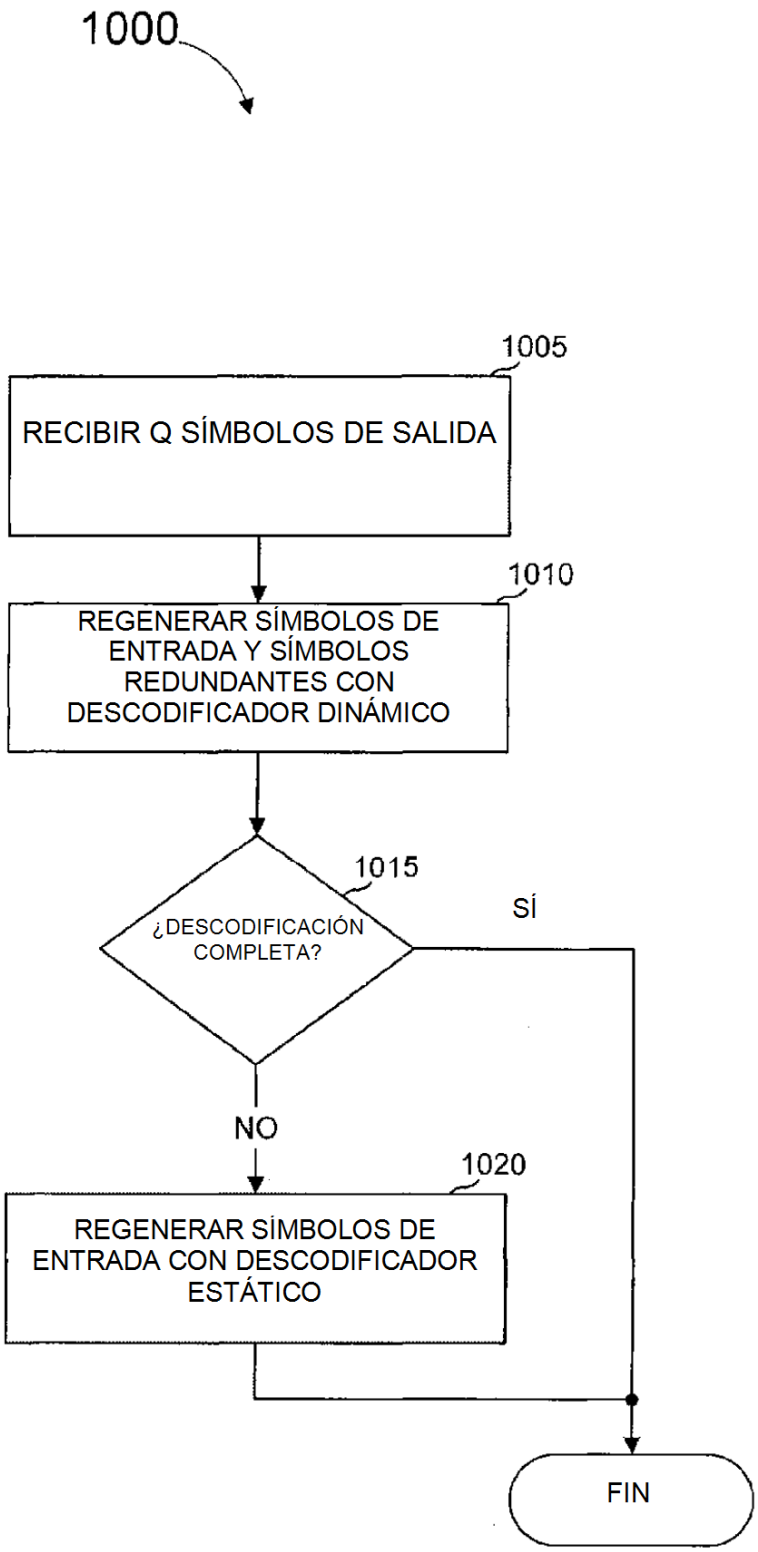
1000
1005
RECIBIR Q SÍMBOLOS DE SALIDA
1010
REGENERAR SÍMBOLOS DE ENTRADA Y SÍMBOLOS REDUNDANTES CON DESCODIFICADOR DINÁMICO
1015
¿DESCODIFICACIÓN COMPLETA?
SÍ
NO
1020
REGENERAR SÍMBOLOS DE ENTRADA CON DESCODIFICADOR ESTÁTICO
FIN

Figura 12

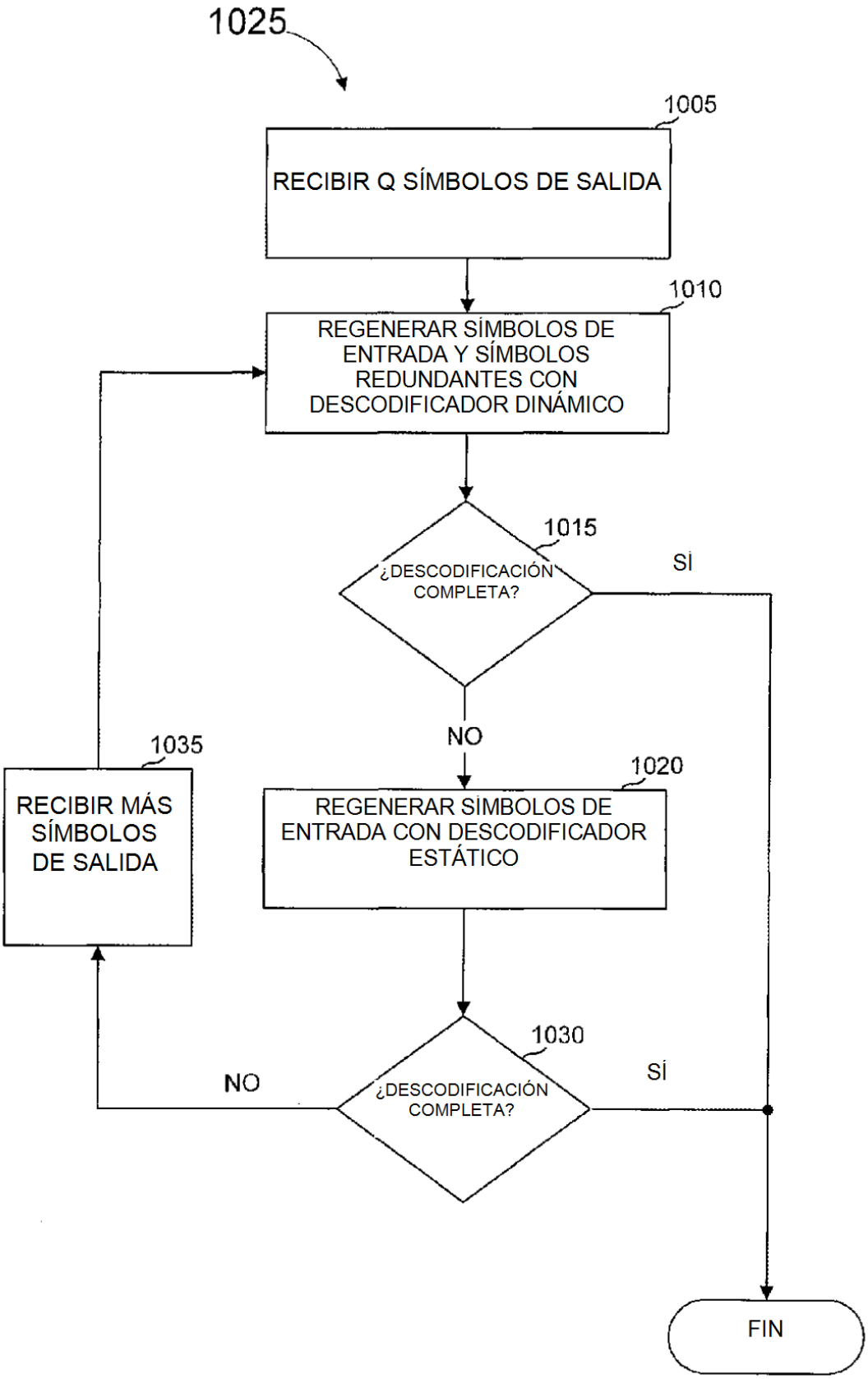
1025
1005
RECIBIR Q SÍMBOLOS DE SALIDA
1010
REGENERAR SÍMBOLOS DE ENTRADA Y SÍMBOLOS REDUNDANTES CON DESCODIFICADOR DINÁMICO
1015
¿DESCODIFICACIÓN COMPLETA?
SÍ
NO
1020
REGENERAR SÍMBOLOS DE ENTRADA CON DESCODIFICADOR ESTÁTICO
1030
¿DESCODIFICACIÓN COMPLETA?
NO
SÍ
1035
RECIBIR MÁS SÍMBOLOS DE SALIDA
FIN

Figura 13

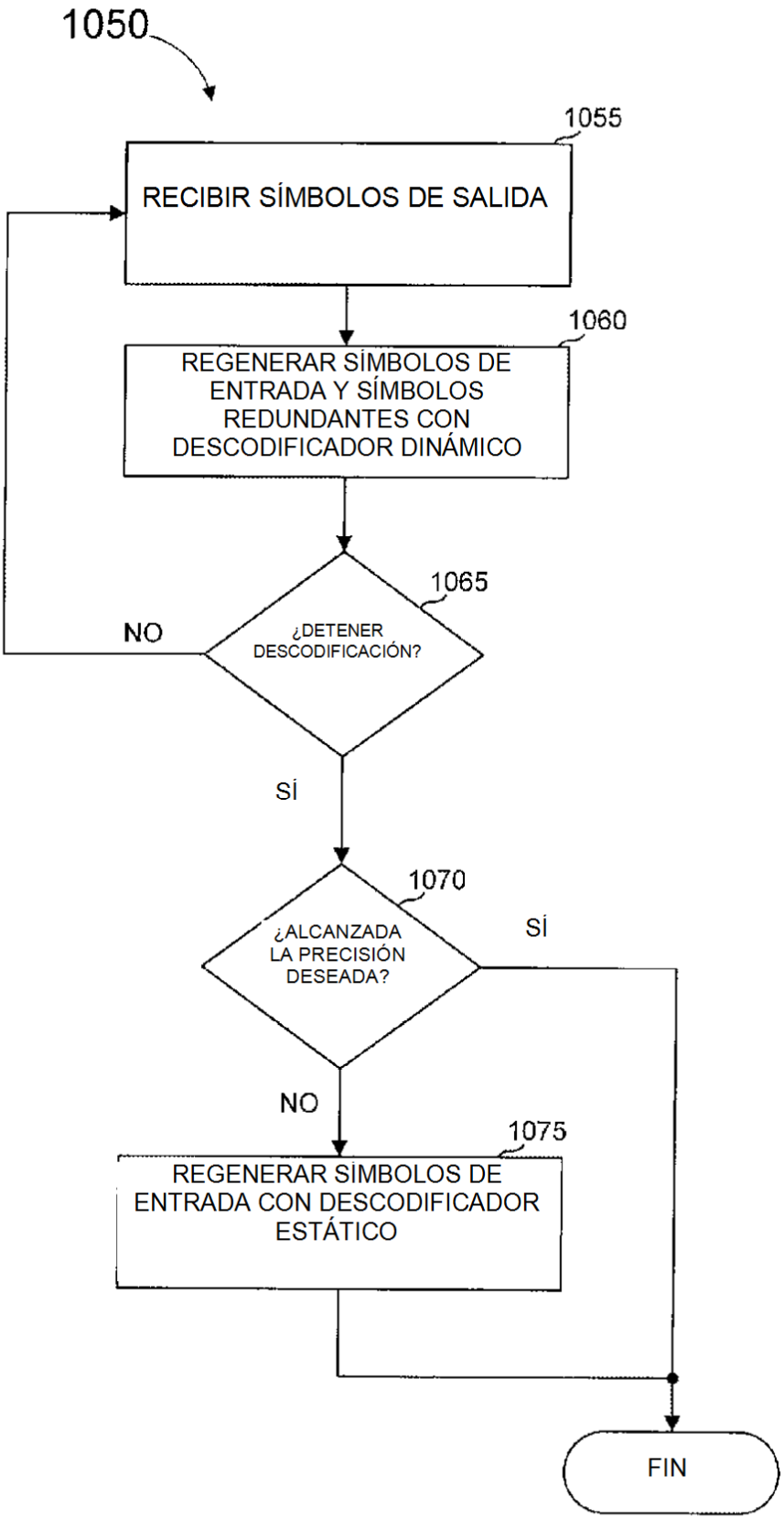
1050
1055
RECIBIR SÍMBOLOS DE SALIDA
1060
REGENERAR SÍMBOLOS DE ENTRADA Y SÍMBOLOS REDUNDANTES CON DESCODIFICADOR DINÁMICO
1065
¿DETENER DESCODIFICACIÓN?
NO
SÍ
1070
¿ALCANZADA LA PRECISIÓN DESEADA?
SÍ
NO
1075
REGENERAR SÍMBOLOS DE ENTRADA CON DESCODIFICADOR ESTÁTICO
FIN

Figura 14

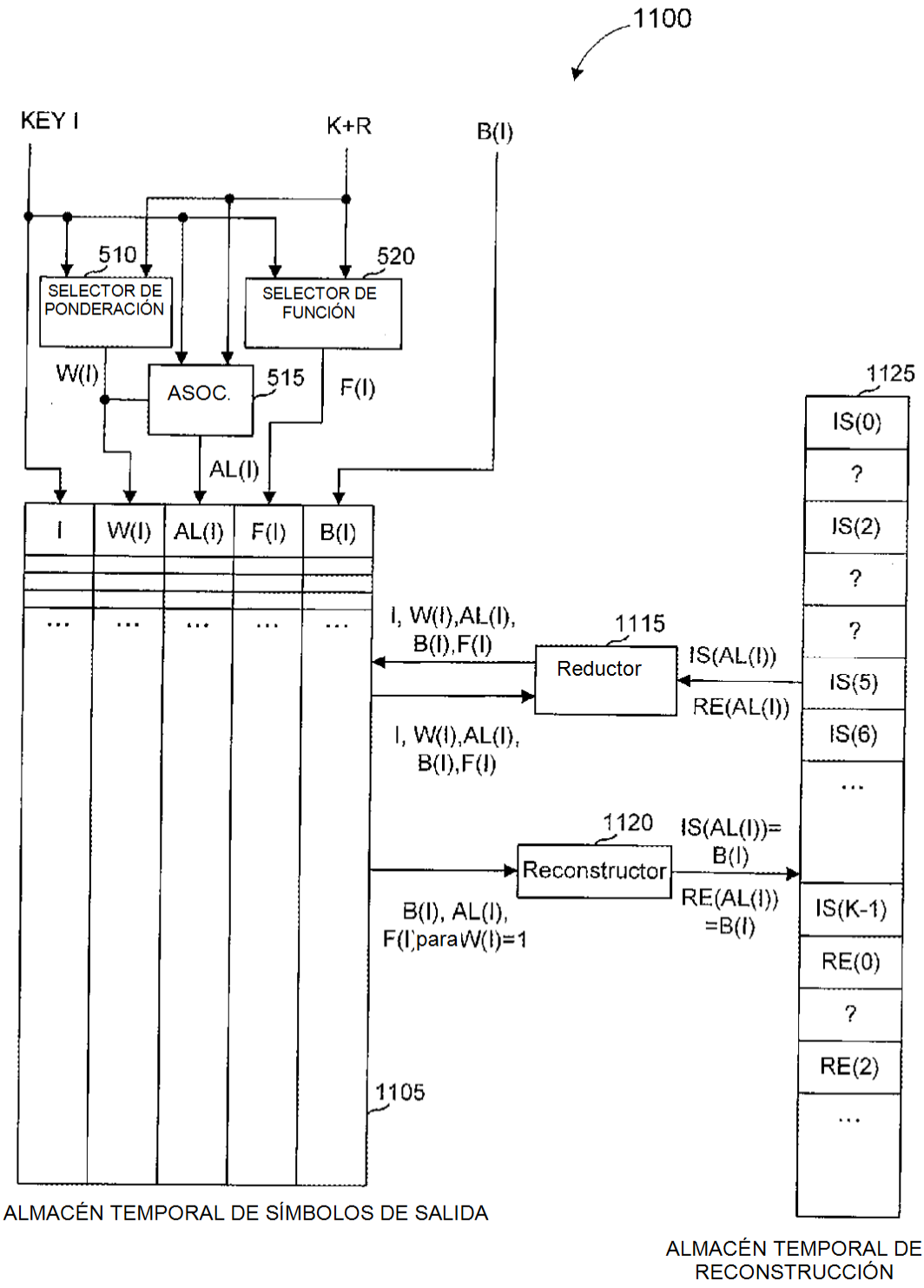
1100
KEY I
K+R
B(I)
510
SELECTOR DE PONDERACIÓN
520
SELECTOR DE FUNCIÓN
W(I)
ASOC.
515
F(I)
AL(I)
I
W(I)
AL(I)
F(I)
B(I)
I, W(I),AL(I), B(I),F(I)
1115
Reductor
IS(AL(I))
RE(AL(I))
I, W(I),AL(I), B(I),F(I)
1120
Reconstructor
IS(AL(I))= B(I)
RE(AL(I)) =B(I)
B(I), AL(I), F(I),paraW(I)=1
1105
ALMACÉN TEMPORAL DE SÍMBOLOS DE SALIDA
1125
IS(0)
?
IS(2)
?
?
IS(5)
IS(6)
IS(K-1)
RE(0)
?
RE(2)
ALMACÉN TEMPORAL DE RECONSTRUCCIÓN

Figura 15

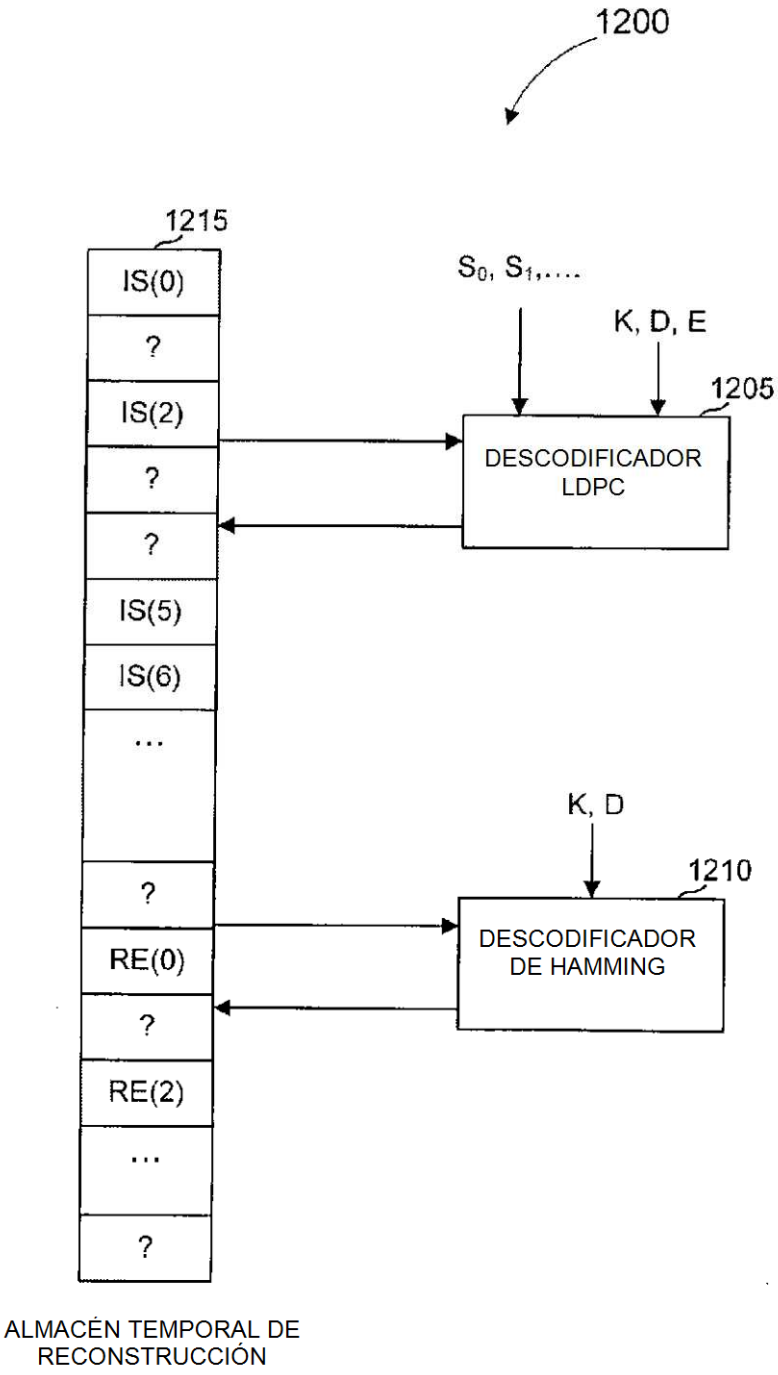
1200
1215
IS(0)
?
IS(2)
?
?
IS(5)
IS(6)
...
?
RE(0)
?
RE(2)
...
?
ALMACÉN TEMPORAL DE RECONSTRUCCIÓN
S0, S1,....
K, D, E
1205
DESCODIFICADOR LDPC
K, D
1210
DESCODIFICADOR DE HAMMING

Figura 16

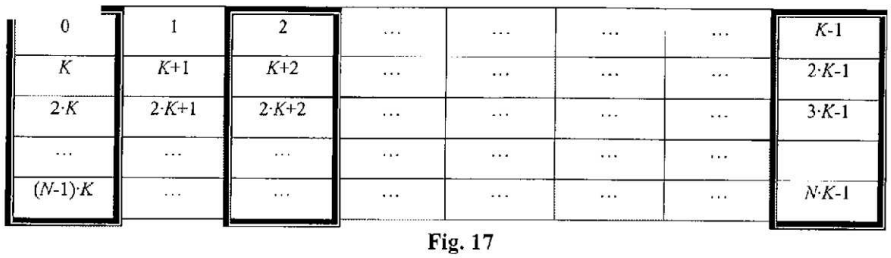

Fig. 17

Tamaño de fichero *F*	*G*	Tamaño de símbolo *T*	*G*T*	K_t	Bloques de origen *Z*	Sub-bloques *N*	K_L	K_S	$T_L \cdot A$	$T_S \cdot A$
100 KB	1	512	512	200	1	1	200	200	N/A	N/A
300 KB	1	512	512	600	1	2	600	600	128	128
1,000 KB	1	512	512	2,000	1	5	2,000	2,000	104	100
3,000 KB	1	512	512	6,000	1	12	6,000	6,000	44	40
10,000 KB	1	512	512	20,000	3	14	6,666	6,667	40	36

Fig. 18

Máximo tamaño de bloque de origen *B*	P_{30}	*G*	Tamaño de símbolo *T*
16KB	1424	1	1424
32KB	1424	1	1424
128KB	700	2	712

Fig. 19

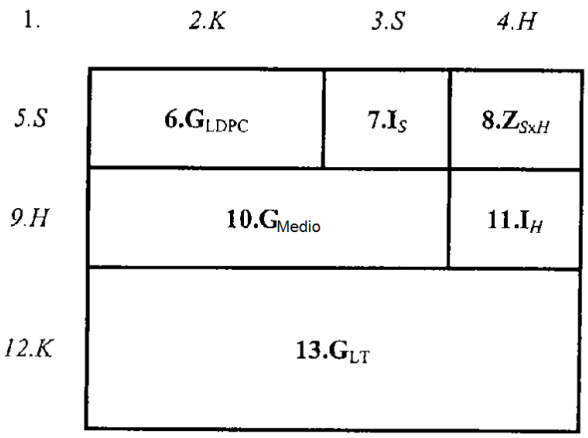

Fig. 20

Índice *j*	*f[j]*	*d[j]*
0	0	--
1	10241	1
2	491582	2
3	712794	3
4	831695	4
5	948446	10
6	1032189	11
7	1048576	40

Fig. 21

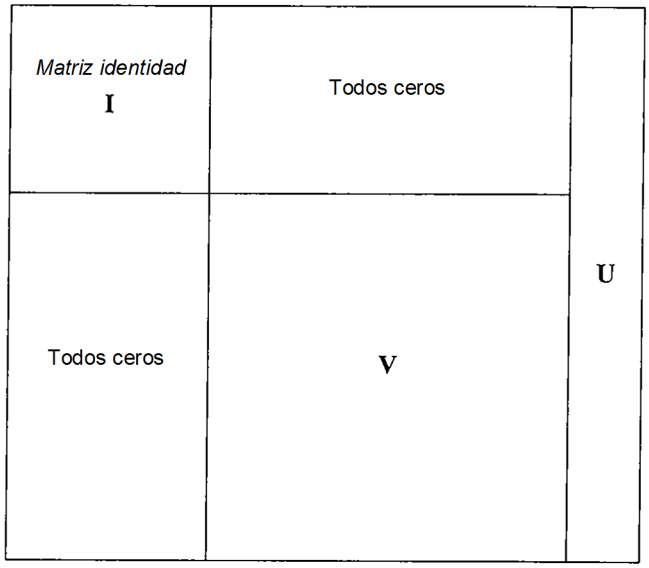

Fig. 22

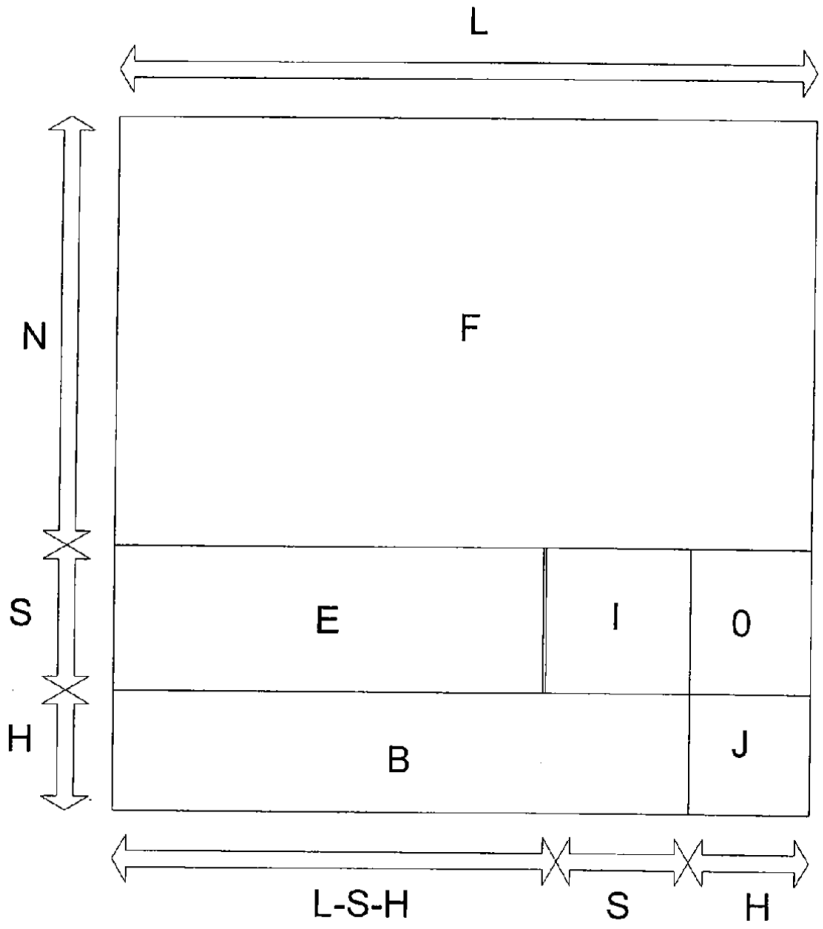

Figura 23

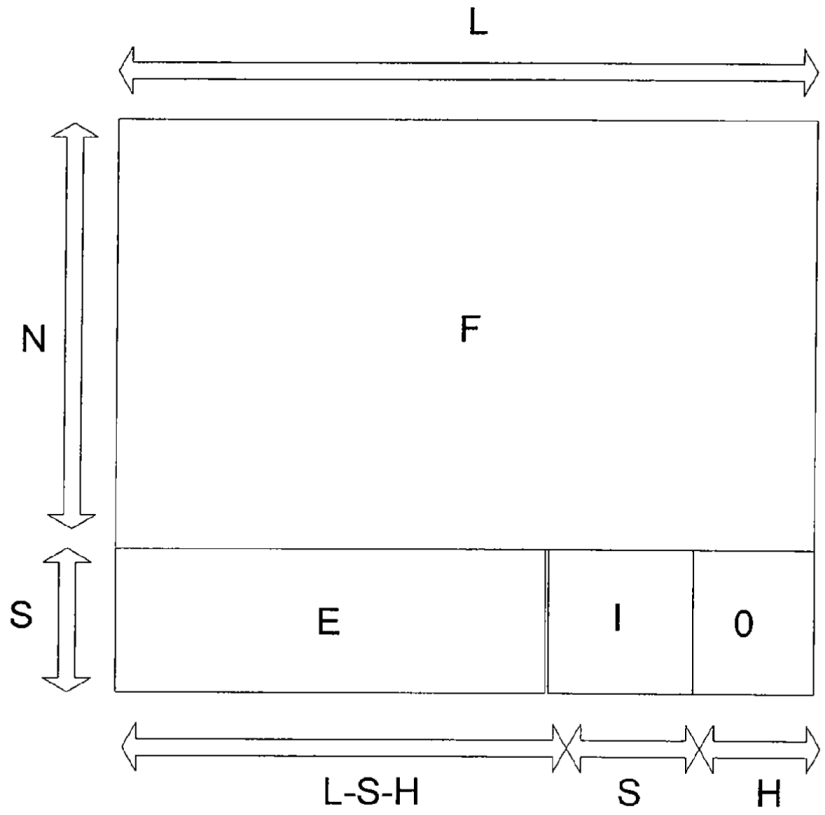

Figura 24a

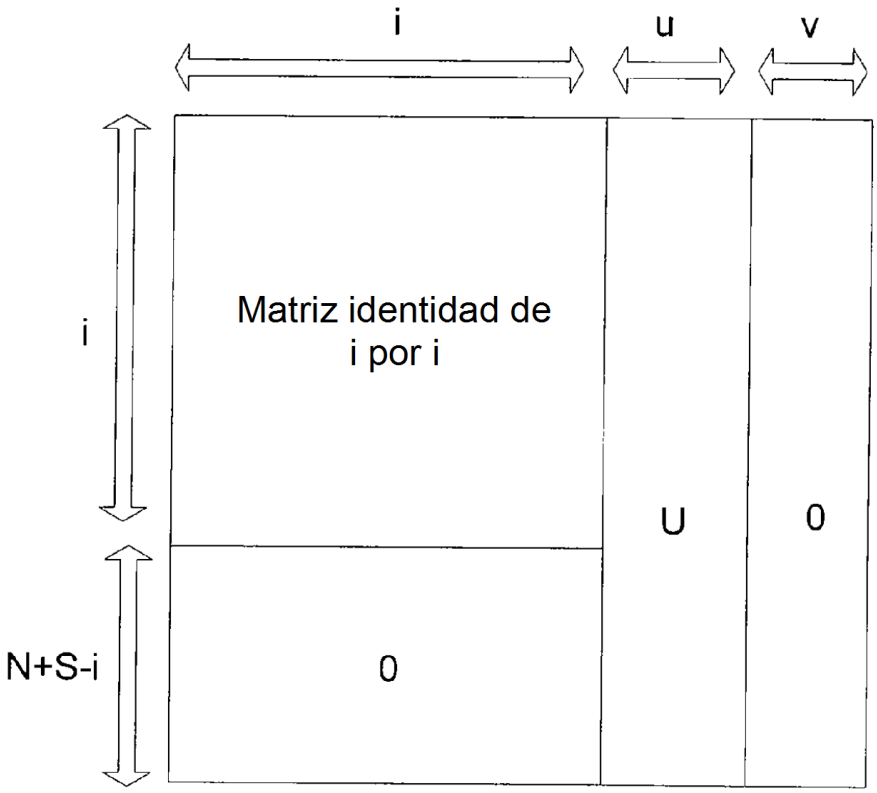

Figura 24b

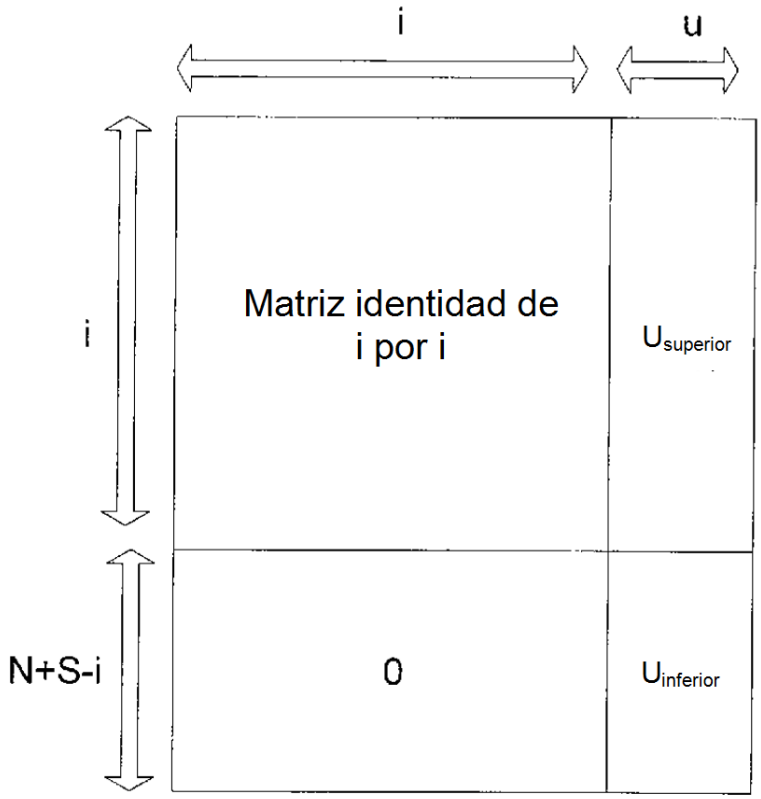

Figura 25

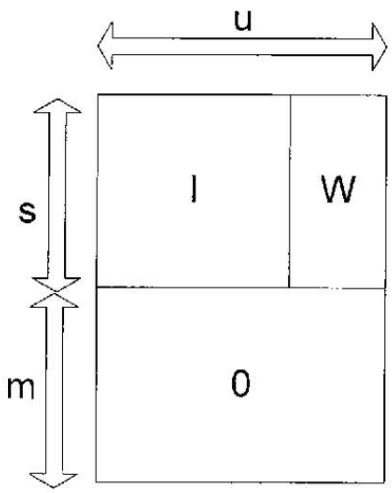

Figura 26

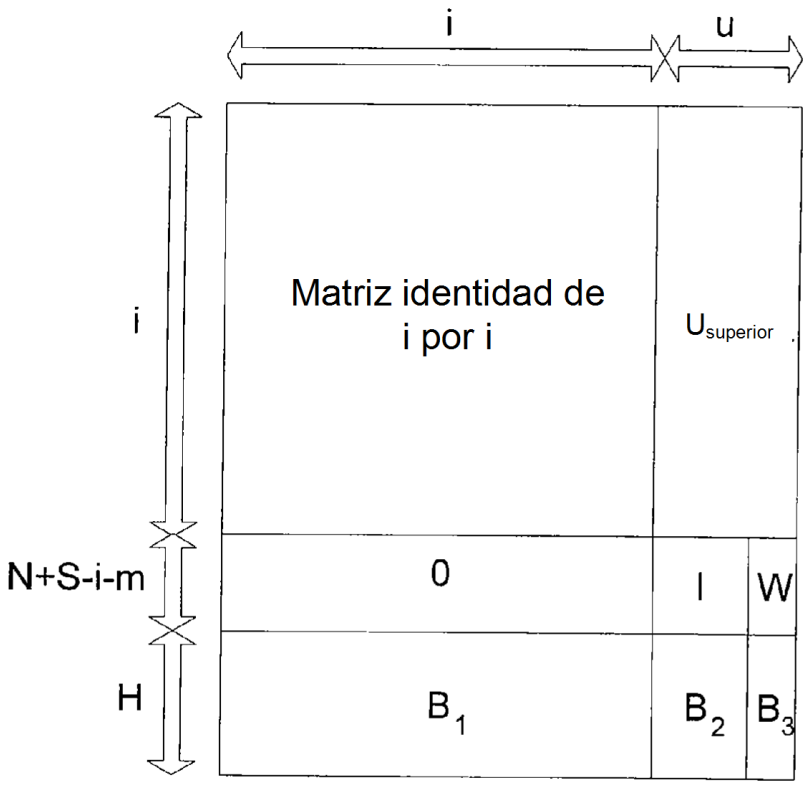

Figura 27

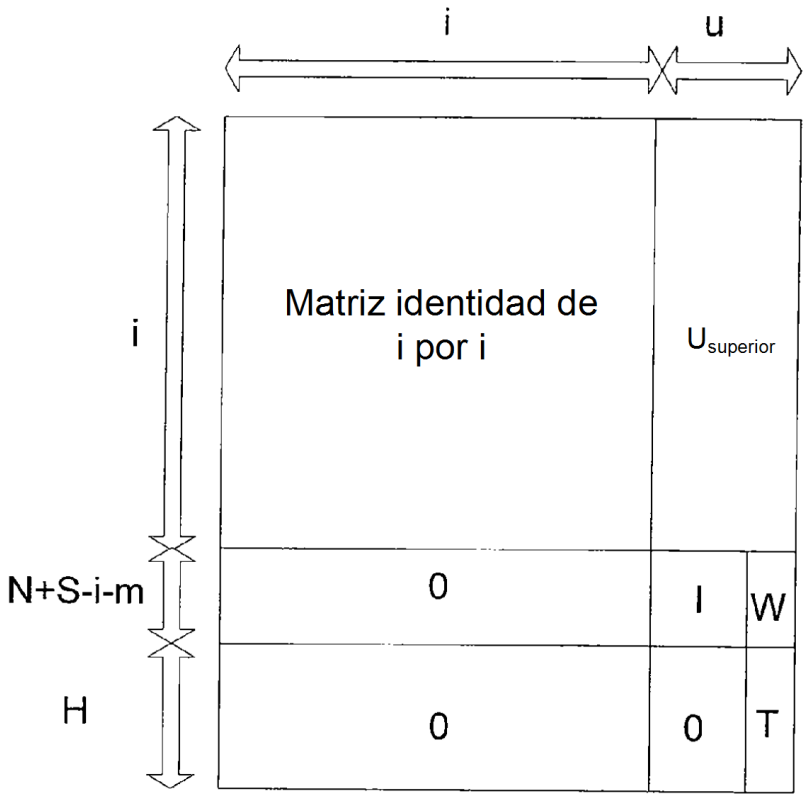

Figura 28

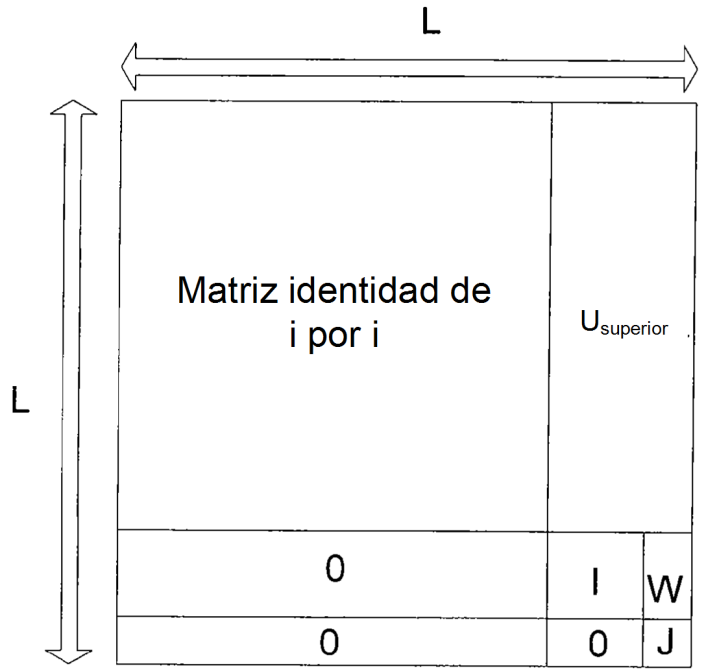

Figura 29

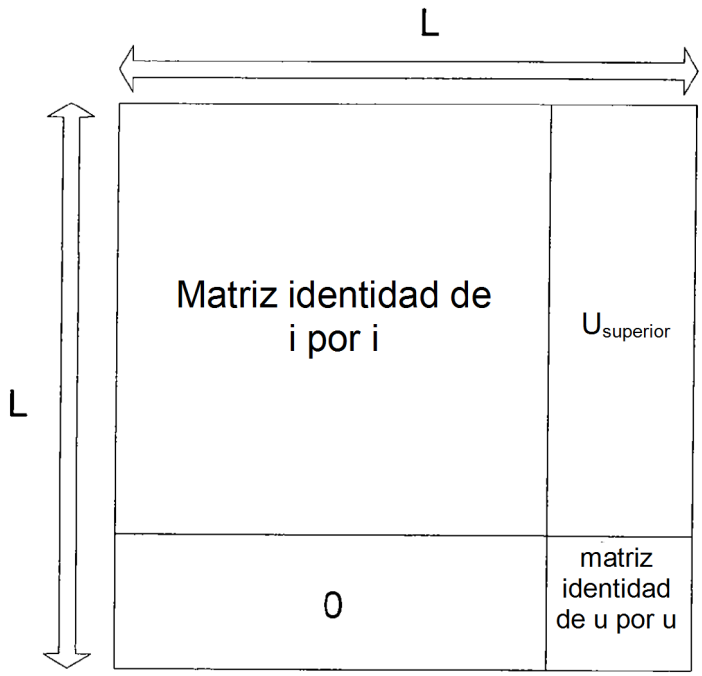
L
L
Matriz identidad de i por i
$U_{superior}$
0
matriz identidad de u por u

Figura 30

Figura 31

Probabilidades de fallo para código (120,100)	
Sobregasto de recepción (A)	Probabilidad de fallo
0	$1,7x10^{-2}$
1	$1,25x10^{-4}$
2	$2,56\ x10^{-6}$
3	$< 1\ x10^{-8}$
4	$<1\ x10^{-8}$
5	$< 1\ x10^{-8}$
6	$< 1\ x10^{-8}$
7	$< 1\ x10^{-8}$
8	$< 1\ x10^{-8}$
9	$< 1\ x\ 10^{-8}$
10	$< 1\ x10^{-8}$
11	$< 1\ x10^{-8}$
12	$< 1\ x10^{-8}$
13	$<1\ x10^{-8}$
14	$<1\ x10^{-8}$
15	0
16	0
17	0
18	0
19	0
20	0

Figura 32

Probabilidades de fallo para código (110,100)	
Sobregasto de recepción (A)	Probabilidad de fallo
0	$4,8\ x10^{-3}$
1	$3,11\ x10^{-5}$
2	$1,37\ x10^{-7}$
3	$<1\ x10^{-8}$
4	$<1\ x10^{-8}$
5	$< 1\ x10^{-8}$
6	0
7	0
8	0
9	0
10	0