(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 812 229**

(51) Int. CI.:

G10L 19/038 (2013.01)
H03M 7/30 (2006.01)
G10L 19/00 (2013.01)
G06T 9/00 (2006.01)
G10L 19/07 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **28.06.2017 E 17178315 (2)**

(97) Fecha y número de publicación de la concesión europea: **22.07.2020 EP 3413309**

(54) Título: **Almacenamiento eficiente de registros de códigos cifrados estructurados múltiples**

(30) Prioridad:

07.06.2017 EP 17174664

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
16.03.2021

(73) Titular/es:

NOKIA TECHNOLOGIES OY (100.0%)
Karakaari 7
02610 Espoo, FI

(72) Inventor/es:

VASILACHE, ADRIANA y
RÄMÖ, ANSSI

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Almacenamiento eficiente de registros de códigos cifrados estructurados múltiples

Campo

Las realizaciones de esta invención se refieren a la codificación, en particular a la codificación de voz y audio.

Antecedentes

Los algoritmos de baja complejidad para la codificación de voz y audio constituyen un activo muy relevante, por ejemplo, para comunicaciones con base en terminales móviles. Debido al bajo almacenamiento y la baja complejidad, a la vez que se preserva la eficiencia de la codificación, los códigos cifrados estructurados pueden ser preferidos en diversos códecs de voz y audio de última generación, como por ejemplo el códec del Servicio de Habla Mejorado (EVS) que se estandarizará dentro del Proyecto de Asociación de Tercera Generación (3GPP).

Los códigos cifrados utilizados dentro de estos códecs de voz y audio pueden estar con base, por ejemplo, en estructuras de entramado, como se describe en la referencia “Multiple-scale leader-lattice VQ with application to LSF quatization” por A. Vasilache, B. Dumitrescu y I. Tabus, Procesamiento de señales, 2002, vol. 82, páginas 563-586.

Es posible definir códigos cifrados de entramado como una unión de clases líder, cada una de las cuales se caracteriza por un vector líder. Un vector líder es un vector n-dimensional (con n que denota un número entero), cuyos componentes (por ejemplo, positivos) están ordenados (por ejemplo, de forma decreciente). La clase líder correspondiente al vector líder entonces consiste en el vector líder y todos los vectores obtenidos a través de todas las permutaciones firmadas del vector líder (con algunas posibles restricciones). También es posible que una, algunas o todas las clases líder estén asociadas respectivamente con una o más escalas, y el código cifrado entramado se forma entonces como una unión de clases líder escaladas y/o sin escala.

Un vector de entrada puede, por ejemplo, codificarse (por ejemplo, en cuantificación) encontrando el vector de código vecino más cercano en el código cifrado, es decir, el vector de código que tiene la distancia más pequeña con respecto al vector de entrada. Un identificador de este vector de código (por ejemplo, un índice asignado a este vector de código) puede servir como una representación codificada del vector de entrada. La solicitud de patente de Estados Unidos US2009/304296 divulga un método para comprimir los vectores propios de un clasificador de Función Discriminante Cuadrática Modificada mediante la aplicación de cuantificación vectorial flexible, por lo que cada vector propio se divide en diversos sub-vectores. Sin embargo, en lugar de almacenar los valores originales de los sub-vectores, solo se almacenan los índices de los sub-vectores en los códigos cifrados generados. La 18ª Conferencia Europea de Procesamiento de Señal (EUSIPCO-2010), del 23 al 27 de agosto de 2010, publica “Fast low bit rate lattice entropy coding for speech and audio coding” que divulga un método de codificación de vectores largos de coeficientes de transformación a partir de códecs de voz y audio, en el cual el vector largo se divide en diversos sub-vectores y cada sub-vector se cuantifica con un cuantificador entramado. La publicación de solicitud de patente PCT WO 2009/022193 divulga un método para cuantificar los coeficientes MDCT de una señal de voz o audio mediante la cual los coeficientes MDCT se dividen en diversos sub-vectores y cada sub-vector se cuantifica de acuerdo con un esquema de asignación de bits acumulativo.

Resumen de algunas realizaciones de la invención

Aunque el uso de códigos cifrados estructurados ya reduce la cantidad de memoria y la complejidad computacional requerida para codificar un vector de entrada, siempre es deseable una reducción adicional de los requisitos de memoria, especialmente para la memoria con base en ROM en la cual los códigos cifrados estructurados se almacenan de manera típica.

De acuerdo con la solicitud, se proporciona un aparato que comprende medios para formar un vector de código base combinando componentes de vector de un sub-vector señalado por un primer puntero con componentes de vector de un sub-vector señalado por un segundo puntero, en donde el sub-vector señalado por el primer puntero y el sub-vector señalado por el segundo puntero están contenidos en una tabla que comprende una pluralidad de sub-vectores, en donde cada entrada de la tabla es un sub-vector y cada sub-vector tiene componentes vectoriales de vector de código base, en donde el vector de código base es una clase líder de un código cifrado estructurado para un códec de audio o de voz que opera en una pluralidad de modos operativos o tasas de codificación operativas, en donde la clase líder comprende un vector líder y permutaciones del vector líder, en donde el primer puntero y el segundo puntero están contenidos en una tabla adicional en donde una entrada de la tabla adicional comprende el primer puntero y el segundo puntero, en donde un valor del primer puntero que señala a un sub-vector en la tabla y un valor del segundo puntero que señala a un sub-vector en la tabla son enteros del módulo n, en donde n está relacionado con el valor máximo del tipo de datos asignados para el almacenamiento del valor del primer puntero y el valor del segundo puntero, y en donde cuando el valor del primer puntero es mayor o igual que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por el valor del primer

puntero, y en donde cuando el valor del primer puntero es menor que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por la suma del valor del primer puntero a n.

Los componentes del vector pueden ser un grupo de componentes del vector contiguos.

El uso de enteros del módulo n para el valor del primer puntero y el valor del segundo puntero puede depender de un modo operativo particular o tasa de codificación operativa del códec de audio/voz.

El tipo de datos puede ser un tipo de datos de caracteres sin signo, y en donde n es 256.

El uno o más vectores de código base pueden ser un conjunto de vectores de código base los cuales definen los códigos cifrados estructurados, y el conjunto de vectores base pueden ser clases líder, cada clase líder puede comprender un vector líder diferente y permutaciones de dicho vector líder.

La tabla puede formar parte de un códec de Servicio de Habla Mejorado de Voz de Evolución a Largo Plazo del Proyecto de Asociación de Tercera Generación.

De acuerdo con un aspecto adicional se proporciona un método que comprende formar un vector de código base combinando componentes de vector de un sub-vector señalado por un primer puntero con componentes de vector de un sub-vector señalado por un segundo puntero, en donde el sub-vector señalado por el primer puntero y el sub-vector señalado por el segundo puntero están contenidos en una tabla que comprende una pluralidad de sub-vectores, en donde cada entrada de la tabla es un sub-vector y cada sub-vector tiene componentes vectoriales de un vector de código base, en donde el vector de código base es una clase líder de códigos cifrados estructurados para un códec de audio o de voz que opera en una pluralidad de modos operativos o tasas de codificación operativas, en donde la clase líder comprende un vector líder y permutaciones del vector líder, en donde el primer puntero y el segundo puntero están contenidos en una tabla adicional en donde una entrada de la tabla adicional comprende el primer puntero que señala y el segundo puntero, en donde un valor del primer puntero que señala a un sub-vector en la tabla y un valor del segundo puntero que señala a un sub-vector en la tabla son enteros del módulo n, en donde n está relacionado con el valor máximo del tipo de datos asignados para el almacenamiento del valor del primer puntero y el valor del segundo puntero, y en donde cuando el valor del primer puntero es mayor o igual que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por el valor del primer puntero, y en donde cuando el valor del primer puntero es menor que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por la suma del valor del primer puntero a n.

Los componentes del vector pueden ser un grupo de componentes del vector contiguos.

El tipo de datos puede ser un tipo de datos de caracteres sin signo, y en donde n es 256.

El uno o más vectores de código base pueden ser un conjunto de vectores de código base los cuales definen los códigos cifrados estructurados, y el conjunto de vectores base puede ser de clases líder, cada clase líder puede comprender un vector líder diferente y permutaciones de dicho vector líder.

Breve descripción de las figuras

Para una mejor comprensión de la presente solicitud y de cómo puede llevarse a cabo la misma, se hará referencia a modo de ejemplo a los dibujos adjuntos en los cuales:

La Figura 1a muestra una ilustración esquemática de un aparato de acuerdo con una realización de la invención;

La Figura 1b muestra un medio de almacenamiento tangible de acuerdo con una realización de la invención;

La Figura 2 muestra un sistema de códec de audio de acuerdo con algunas realizaciones; y

La Figura 3 muestra un diagrama de flujo de un método de acuerdo con una realización de la invención.

Descripción detallada de realizaciones

La Figura 1 ilustra esquemáticamente componentes de un aparato 1 de acuerdo con una realización de la invención. El aparato 1 puede ser, por ejemplo, un dispositivo electrónico que, por ejemplo, es capaz de codificar al menos una de las señales de voz, audio y video, o un componente de dicho dispositivo. El aparato 1 está configurado en particular para determinar un vector de código para codificar un vector de entrada. El aparato 1 puede estar realizado, por ejemplo, como un módulo. Ejemplos no limitativos del aparato 1 son un teléfono móvil, un asistente digital personal, un reproductor multimedia portátil (audio y/o video) y un ordenador (por ejemplo, un ordenador portátil o de escritorio).

El aparato 1 comprende un procesador 10, el cual se puede incorporar, por ejemplo, como un microprocesador, un Procesador de Señal Digital (DSP) o un Circuito Integrado de Aplicación Específica (ASIC), por nombrar solo algunos ejemplos no limitantes. El procesador 10 ejecuta un código de programa almacenado en la memoria 11 de programa y usa la memoria 12 principal como una memoria de trabajo, por ejemplo para almacenar al menos temporalmente resultados intermedios, pero también para almacenar, por ejemplo, bases de datos predefinidas y/o pre calculadas. Algunas o todas las memorias 11 y 12 también pueden incluirse en el procesador 10. Las memorias 11 y/o 12 pueden incorporarse, por ejemplo, como Memoria de Solo Lectura (ROM), Memoria de Acceso Aleatorio (RAM), por nombrar algunos ejemplos no limitantes. Una o ambas memorias 11 y 12 pueden estar conectadas de manera fija al procesador 10 o desmontables del procesador 10, por ejemplo en la forma de una tarjeta de memoria o dispositivo de memoria.

El procesador 10 controla además una interfaz 13 de entrada/salida (E/S), a través de la cual el procesador recibe o proporciona información a otras unidades funcionales.

Como se describirá a continuación, el procesador 10 es al menos capaz de ejecutar el código de programa para determinar un vector de código para codificar un vector de entrada. Sin embargo, el procesador 10 puede, por supuesto, poseer capacidades adicionales. Por ejemplo, el procesador 10 puede ser capaz de al menos una codificación de voz, audio y video, por ejemplo, con base en valores de entrada muestreados. El procesador 10 puede adicional o alternativamente ser capaz de controlar el funcionamiento de un dispositivo portátil de comunicación y/o multimedia.

El aparato 1 de la Figura 1 puede comprender además componentes tales como una interfaz de usuario, por ejemplo, para permitir que un usuario del aparato 1 interactúe con el procesador 10, o una antena con circuitos de radiofrecuencia (RF) asociados para permitir que el aparato 1 realice comunicación inalámbrica.

Los circuitos formados por los componentes del aparato 1 pueden implementarse solo en hardware, parcialmente en hardware y en software, o solo en software, como se describe adicionalmente al final de esta especificación.

Se apreciaría que las estructuras esquemáticas descritas en las Figuras 1 a 2, y las etapas del método que se muestran en la Figura 3 representan solo una parte de un códec de audio y específicamente parte de un cuantificador y descuantificador incorporado dentro de un aparato codificador/decodificador o como se muestra a manera de ejemplo implementado en el aparato que se muestra en la Figura 1.

En la Figura 2 se muestra el funcionamiento general de los códecs de audio empleados por las realizaciones. Los sistemas generales de codificación/decodificación de audio comprenden tanto un codificador como un decodificador, como se ilustra esquemáticamente en la Figura 2. Sin embargo, se entenderá que algunas realizaciones pueden implementar uno del codificador o decodificador, o tanto el codificador como el decodificador. La Figura 2 ilustra un sistema 102 con un codificador 104 y, en particular, se describe en el presente documento un codificador que implementa códigos cifrados estructurados, un canal 106 de almacenamiento o medios y un decodificador 108. Podría entenderse que, como se describió anteriormente, algunas realizaciones pueden comprender o implementar uno del codificador 104 o decodificador 108 o tanto el codificador 104 como el decodificador 108.

El codificador 104 comprime una señal 110 de audio de entrada que produce un flujo 112 de bits, los cuales en algunas realizaciones pueden almacenarse o transmitirse a través de un canal 106 multimedia. El flujo 112 de bits puede recibirse dentro del decodificador 108. El decodificador 108 descomprime el flujo 112 de bits y produce una señal 114 de audio de salida. La tasa de bits del flujo 112 de bits y la calidad de la señal 114 de audio de salida en relación con la señal 110 de entrada son las características principales las cuales definen el rendimiento del sistema 102 de codificación.

El codificador 104 y el decodificador 108 pueden comprender un cuantificador 151 y un descuantificador 152 para cuantificar y descuantificar coeficientes y parámetros del proceso de codificación de audio. Por ejemplo, los parámetros del proceso de codificación de audio pueden incluir, pero no se limitan a, representaciones transformadas de Coeficientes de Predicción Lineal (LPC). Por ejemplo, dichas representaciones transformadas de LPCs pueden comprender al menos uno de los siguientes; frecuencias espectrales de Línea (LSF), Pares Espectrales de Línea (LSP) o Pares Espectrales de Inmitancia (ISP). Lo anterior puede representarse como vectores que comprenden diversos componentes del vector.

Un código cifrado estructurado del cuantificador 151 y el descuantificador 152 puede definirse mediante un conjunto de vectores de código base definiendo cada vector de código base del conjunto de vectores de código base, un conjunto adicional respectivo de vectores de código base. En un código cifrado estructurado, el conjunto adicional respectivo de vectores de código base puede formarse a partir del vector de código base con el cual está asociado. En consecuencia, el código cifrado estructurado puede representarse en el cuantificador 151 y el descuantificador 152 mediante una tabla que comprende el conjunto de vectores de código base los cuales definen dicho código cifrado estructurado.

En las realizaciones, la tabla que comprende el conjunto de vectores de código base que definen el código cifrado estructurado puede almacenarse eficientemente en el aparato 1 como una tabla que tiene una pluralidad de vectores de código sub-base. Cada vector de código sub-base comprende un conjunto contiguo de componentes a partir de un vector de código base del código cifrado estructurado con un menor número de componentes.

Se puede dar cuenta de que se puede usar menos memoria para almacenar los vectores de código base los cuales definen el código cifrado estructurado al observar que los vectores de código base comprenden patrones de conjuntos contiguos de componentes de vector los cuales se repiten en todo el conjunto de vectores de código base. En realizaciones, estos patrones repetitivos pueden explotarse teniendo vectores de código sub-base cuyos componentes son los patrones de componentes repetitivos presentes a través del conjunto de vectores de código base.

A modo de explicación adicional usando un ejemplo de fuente C, un código cifrado estructurado puede definirse mediante los siguientes vectores de código base. Debe observarse que cada vector de código base en este ejemplo particular tiene una dimensión vectorial de seis.

```
int no_líder[][] = { …
{    1,    0,    0,    1,    0,    0, }
{    2,    0,    0,    1,    0,    0, }
{    4,    2,    0,    1,    0,    0, }
{    2,    0,    0,    2,    2,    0, }
{    4,    2,    0,    2,    0,    0, }
{    7,    5,    2,    2,    2,    0, }
{    8,    7,    2,    2,    2,    0, }
{    5,    4,    0,    0,    0,    0, }
{    7,    5,    2,    0,    0,    0, }
{    1,    0,    0,    1,    0,    0, }    …} ;
```

En este caso, la tabla de vectores de código sub-base puede basarse en una dimensión de vector de tres, la mitad que la de los vectores de código base del código cifrado estructurado, y puede comprender los siguientes vectores de código sub-base.

```
líderes char_cortos[][] = {
{          1,    0,    0 },
{          2,    0,    0 },
{          4,    0,    0 },
{          2,    1,    0 },
{          4,    1,    0 },
{          2,    2,    0 },
{          3,    2,    0 },
{          4,    2,    0 },
{          5,    2,    0 },
{          6,    3,    0 },
{          7,    3,    0 },
{          8,    3,    0 },
{          9,    3,    0 },
… } ;
```

En este ejemplo particular, el primer vector de código base utilizado para definir el código cifrado estructurado puede representarse como el primer vector de código sub-base {1, 0, 0} en la tabla de vectores de código sub-base. En otras palabras, el primer grupo de componentes 0 a 2 y el segundo grupo de componentes 3 a 5 del primer vector de código base utilizado para definir el código cifrado estructurado pueden representarse ambos por el primer vector de código sub-base.

Tomando otro caso usando este ejemplo particular, el segundo vector de código base usado para definir el código cifrado estructurado puede representarse por el segundo vector de código sub-base {2, 0, 0} seguido por el primer vector de código sub-base {1, 0, 0}.

El mapeo el cual especifica cómo se forma cada vector de código base a partir de la tabla de vectores de código sub-base puede estar contenido como una tabla de punteros, por lo cual cada entrada en la tabla de punteros especifica los vectores de código sub-base necesarios para formular un vector de código base particular en la forma de diversos punteros a entradas en la tabla de vectores de código sub-base. El número de punteros en cada entrada en la tabla de punteros es proporcional al número de vectores de código sub-base requeridos para formular el vector de código base asociado con esa entrada particular en la tabla de punteros.

Los vectores de código sub-base señalados por una entrada particular de la tabla de punteros se combinan en un solo vector de código base.

Volviendo al ejemplo de la fuente C anterior como explicación, podemos formular la tabla de punteros como

```
líder corto_p_idx[][] = {
{          0,          0 },
{          1,          0 },
{          7,          0 },
{          1,          5 },
{          7,          2 },
… } ;
```

En este caso, cada entrada está formada por dos enteros, con cada entero que señala a una entrada en la tabla de vectores de código sub-base. En el ejemplo anterior, el primer vector de código base que define el código cifrado estructurado se da como la primera entrada en la tabla de punteros. En este caso, el primer entero señala a la entrada “0” de la tabla de vectores de código sub-base, lo que denota que los primeros tres componentes 0 a 2 del vector de código base son el vector de código sub-base asociado con la primera entrada de la tabla de vectores de código sub-base. El segundo entero también señala a la entrada “0” de la tabla de vectores de código sub-base, lo que también denota que los segundos tres componentes 3 a 5 del vector de código base es el vector de código base asociado con la primera entrada de la tabla de sub-vectores.

Por lo tanto, debe apreciarse que la tabla de vectores de código base utilizada para definir un código cifrado estructurado puede almacenarse en la forma de una tabla de vectores de código base junto con una tabla de puntero adicional que comprende entradas las cuales señalan a entradas de vectores de código base en la tabla de vectores de código sub-base. Esto puede resultar en el beneficio técnico de requerir menos memoria para almacenar los vectores de código base utilizados para definir el código cifrado estructurado, ya que la tabla de vectores de código sub-base junto con la tabla de puntero adicional requiere menos almacenamiento.

Además, el procesador 10 en el aparato 1 puede estar dispuesto para realizar el método de formulación de un vector de código base combinando vectores de código sub-base de la tabla de vectores de código sub-base, donde los vectores de código sub-base se usan para formular un vector de código base particular que se especifica mediante una entrada de la tabla de punteros como se describe anteriormente. Por consiguiente, la Figura 3 muestra las etapas de procesamiento que puede realizar un procesador como el que se representa como 10 en la Figura 1.

La etapa 301 de procesamiento muestra la etapa de recuperar una entrada particular a partir de la tabla de punteros los cuales especifican el vector de código base.

La etapa 303 de procesamiento muestra la etapa de recuperar una pluralidad de vectores de código sub-base a partir de la tabla de vectores de código sub-base. Cada vector de código sub-base se recupera usando un puntero dado por la entrada de la tabla de punteros de la etapa 301.

La etapa 305 de procesamiento muestra la etapa de combinar los vectores de código sub-base recuperados en un vector de código base. El orden de colocación de los vectores de código sub-base para formar el vector de código base está dado por el orden de los punteros tal como lo proporciona la entrada de la tabla de punteros.

Normalmente, la memoria utilizada para contener tablas de cuantificación y similares puede tomar la forma de Memoria de Solo Lectura (ROM).

Por ejemplo, para una implementación particular del Servicio de Habla Mejorado (EVS) estandarizado de 3GPP que emplea un total de 406 estructuras cuantificadoras, el consumo de ROM de la tabla es del orden de 4.872 K palabras de 16 bits cuando se usan las tablas completas para almacenar el conjunto de vectores de código base. Sin embargo, esta cifra se reduce a 3.797 k palabras de 16 bits cuando se utiliza la estructura anterior de la tabla de vectores de código sub-base y la tabla de punteros que la acompaña.

Con el fin de reducir aún más los requisitos de memoria para almacenar las tablas de cuantificación anteriores, la tabla de punteros la cual se representa como *líder_p_idx* en el ejemplo específico de código fuente C dado anteriormente, se muestra actualmente como que almacena cada miembro usando el tipo de datos *corto* el cual es una palabra de 16 bits con signo que tiene un posible rango de valores a partir de -32,768 a 32,767. Alternativamente, en algunas realizaciones, la tabla de punteros puede usar el tipo de datos *char sin asignar* (*uchar*) para almacenar los miembros de la tabla. En este caso, cada miembro o número de la tabla de punteros se almacena como un número de 8 bits sin signo y, en consecuencia, cada número está limitado a un posible rango de valores de 0 a 255. Esto sería la mitad del almacenamiento requerido para la tabla de punteros (*líder_p_idx*) en comparación con el uso de un tipo de almacenamiento de palabras de 16 bits.

Es ventajoso apreciar en este punto que diversos códecs de audio/voz tal como el códec EVS estandarizado del Proyecto de Asociación de Tercera Generación (3GPP) están diseñados para operar en una gran cantidad de tasas de bits/codificación y modos operativos, y que cada modo operativo y la tasa de bits de origen puede requerir un código cifrado estructurado particular adaptado a las condiciones operativas particulares. Dentro del contexto de la descripción anterior, un códec de audio/voz que funciona de esta manera (tal como el códec EVS) puede utilizar diversos códigos cifrados estructurados diferentes, y cada código cifrado estructurado puede requerir un número diferente de vectores de código base para definir dicho código cifrado estructurado.

Tomando, por ejemplo, la tabla *corta líder* de arriba la cual contiene los vectores de código sub-base los cuales se usan para formar los vectores de código base para definir un código cifrado estructurado particular. En algunos modos de funcionamiento, se puede requerir acceso a las entradas en la tabla de vectores de código sub-base (*líder_corto*) la cual excede 255. Para estos casos, el valor *uchar* el cual señala a la entrada en la tabla de vectores de código sub-base (*líder_corto*) no podría señalar directamente a cualquier entrada en esa tabla la cual es mayor a 255.

Para estos modos de funcionamiento (o tasas de bits de codificación) del códec de audio, la tabla de punteros (*líder_p_idx*) puede disponerse para usar un módulo aritmético con el fin de almacenar valores superiores al número máximo permitido por el tipo de datos asignado al valor del puntero. En otras palabras, el valor asignado al puntero puede “ajustarse” cuando el valor del puntero excede el rango máximo de su tipo de datos asignado. Por ejemplo, utilizando la situación anterior en la cual el tipo de datos utilizado para almacenar los valores de los punteros es del tipo *uchar*, los números pueden almacenarse de acuerdo con la aritmética del módulo 256, de modo que los punteros los cuales deben señalar a las entradas en la tabla de vectores de código sub-base mayores que 256 se asignan los valores 0, 1, 2.... de puntero.

El valor real del puntero se puede obtener agregando el módulo al valor del puntero almacenado. Entonces, continuando con el ejemplo anterior, el valor real del puntero se puede obtener agregando 256 al valor almacenado en la tabla de punteros.

Debe apreciarse que el mecanismo anterior de usar aritmética del módulo para extender el rango de posibles valores de puntero de la tabla de punteros depende del modo operativo (o tasas de bits de operación) del códec de audio/voz. Es decir, para algunos modos operativos (o tasas de bits operativas) se puede requerir acceso a regiones de la tabla de vectores de código sub-base que van más allí del rango de números del tipo de datos asignados utilizado para almacenar el valor del puntero.

Otras realizaciones pueden extender el valor máximo efectivo permitido por el tipo de datos asignado a los punteros asegurando que cada entrada de la tabla de punteros siga una regla o patrón específicos. Por ejemplo, los valores relativos de los punteros en cada entrada de la tabla pueden usarse para indicar si los punteros para la entrada particular pueden tratarse como un valor el cual está dentro del rango de números del tipo de datos, o un valor el cual representa un número más allí del rango de números estándar del tipo de datos y, por lo tanto, requiere un procesamiento adicional para obtener el valor verdadero.

La señalización requerida para indicar si el valor del puntero en la tabla de punteros puede tratarse (o no tratarse) como un número normal dentro del rango de números del tipo de datos puede incorporarse intrínsecamente en la estructura de las tablas y códigos cifrados. De este modo, no se genera una señalización adicional en la sobrecarga. En otras palabras, los vectores de código base y, por lo tanto, los vectores de código sub-base de la tabla de vectores de código sub-base de la tabla, pueden estructurarse de tal manera que cada entrada en la tabla de punteros tenga una regla/patrón particular si los valores de puntero deben tratarse como un valor dentro del rango de números del tipo de datos, y otro patrón/regla si los valores de puntero deben tratarse como un valor el cual se encuentra fuera del rango de números del tipo de datos.

Por ejemplo, se puede lograr una regla/patrón de este tipo diseñando el código cifrado estructurado de tal manera que (para cada vector de código base del código cifrado estructurado) cuando los valores de puntero en una entrada de la tabla de punteros se tratan como números normales dentro del rango de números del tipo de datos, el primer vector de código sub-base tiene un valor del puntero asignado a este en la tabla de punteros el cual es mayor o igual al valor del puntero asignado al segundo vector sub-base. Por el contrario, cuando los valores de puntero en una entrada de la tabla de punteros deben tratarse como números fuera del rango de números del tipo de datos, se puede usar un patrón/regla diferente para almacenar los valores de puntero. Por ejemplo, el patrón/regla puede ser tal que cuando el primer valor del puntero del vector del código sub-base sea menor que el segundo valor del puntero del vector del código sub-base, los valores del puntero en la entrada de la tabla deben tratarse como valores numéricos los cuales se encuentran fuera del rango de números del tipo de datos.

Por ejemplo, usando el ejemplo de código fuente C anterior, y a modo de explicación adicional, la tabla de vectores de código sub-base *líderes_cortos* puede estructurarse de tal modo que cada vector de código base del código cifrado estructurado *no_líder* tenga una entrada en la tabla de punteros *líder_p_idx* como se indica a continuación. La tabla de punteros *líder_p_idx* son del tipo de datos *uchar*

líder char sin *signo_p_idx* [] = {
{ 0, 0 }, /* valores de puntero actuales */
{ 1, 0 }, /* valores de puntero actuales */
{ 7, 0 }, /* valores de puntero actuales */

{ 100, 65 }, /* valores de puntero actuales */
{ 230, 100 }, /* valores de puntero actuales */
{ 4, 100 }, /* primer puntero dado corresponde a 256+4, 100*/
{ 18, 145 }, /*primer puntero corresponde a 256+18, 145*/
…
{ 7, 2 }, /* valores de puntero actuales */

Se puede ver que algunas entradas en la tabla siguen el patrón/regla como se discutió anteriormente de que el primer valor del puntero es igual o mayor que el segundo valor del puntero. Esta condición solo puede ser válida para valores de puntero los cuales se encuentran dentro del rango de números del tipo de datos asignado para almacenar el valor del puntero. Por ejemplo, como se discutió anteriormente, si el tipo de datos es *char sin signo*, entonces el rango dinámico de los valores del puntero estará entre 0 y 255. Por lo tanto, los punteros almacenados asociados con estas entradas particulares se tratan como números normales, ya que no es necesario para realizar la aritmética del módulo 256 con el fin de obtener el valor real del puntero.

Al inspeccionar la tabla anterior *líder_p_idx* se puede ver que algunas de las entradas tienen un primer valor del puntero el cual es menor que el segundo valor del puntero. Como se discutió anteriormente, esta disposición particular de punteros denota que el valor almacenado del primer puntero en la entrada de la tabla de punteros particular denota que el primer puntero debe tratarse como un valor el cual se encuentra fuera del rango de números del tipo de datos *uchar*, en otras palabras el valor del puntero es mayor que 255. El valor real del puntero se puede obtener agregando el módulo al valor del puntero almacenado. Continuando con el ejemplo anterior, se puede obtener el real del primer puntero agregando 256 al valor almacenado en la tabla de punteros.

En algunas realizaciones, dicho conjunto de vectores de código base que define el código cifrado estructurado puede representar clases líder, en donde cada clase líder comprende un vector líder diferente y permutaciones de dicho vector líder. Por lo tanto, dicho vector líder y las permutaciones de dicho vector líder pueden representar el conjunto adicional de vectores de código base definidos por un vector de código base del conjunto de vectores de código base. Como un ejemplo, un vector líder es un vector n-dimensional (con n que denota un número entero), cuyos componentes (positivos) están ordenados (por ejemplo, de manera decreciente). La clase líder correspondiente al vector líder consiste entonces en el vector líder y todos los vectores obtenidos a través de todas las permutaciones con signo del vector líder (con algunas posibles restricciones).

Una unión de clases líder puede definirse por el conjunto de vectores de código base asociados con la misma representación a escala de la pluralidad de representaciones a escala y la representación a escala respectiva. Por ejemplo, una unión de clases líder puede estar asociada con un conjunto de vectores de código obtenidos mediante el escalado de los vectores de código base de la etapa asociada de vectores de código base con la escala representativa.

Dicha unión de clases líder puede considerarse como un truncamiento. Por lo tanto, si la pluralidad de representaciones de escala son n representaciones de escala, se pueden definir n uniones de clases líder, en donde cada unión de clase líder se define a través de la representación de escala respectiva y un grupo de vectores de código base del conjunto de vectores de código base asociados con la representación de escala respectiva.

En consecuencia, la pluralidad de representaciones a escala y la pluralidad de grupos de vectores de código base pueden definir una pluralidad de unión de clases líder definiendo de este modo un código cifrado, en donde, como un ejemplo, cada unión de clases líder puede considerarse como una unión de clases líder escaladas.

Los códigos cifrados utilizados dentro de estos códecs de voz y audio pueden estar con base, por ejemplo, en estructuras de entramado, como se describe en la referencia “Multiple-scale leader-lattice VQ with application to LSF quantization” por A. Vasilache, B. Dumitrescu y I. Tabus, Procesamiento de señales, 2002, vol. 82, páginas 563-586, Elsevier. Por ejemplo, un entramado D10+ puede considerarse para la cuantificación, pero también puede considerarse cualquier otra cuantización de entramado adecuada.

Como se usa en esta solicitud, el término “circuito” se refiere a todo lo siguiente:

(a) implementaciones de circuitos solo de hardware (tales como implementaciones solo en circuitos analógicos y/o digitales) y

(b) combinaciones de circuitos y software (y/o firmware), tales como (de acuerdo como corresponda):

(i) a una combinación de procesador(es) o

(ii) a porciones de procesador(es)/software (que incluyen el(los) procesador(es) de señal digital), software y memoria(s) que trabajan en conjunto para hacer que un aparato, tal como un teléfono móvil o un dispositivo de posicionamiento, realice diversas funciones y

(c) a circuitos, tales como un(os) microprocesador(es) o una porción de un(os) microprocesador(es), que requieren software o firmware para funcionar, incluso si el software o firmware no está físicamente presente.

Esta definición de 'circuitos' se aplica a todos los usos de este término en esta solicitud, incluso en cualquier reivindicación. Como un ejemplo adicional, como se usa en esta aplicación, el término “circuitos” también cubriría una implementación de simplemente un procesador (o múltiples procesadores) o una porción de un procesador y su (o sus) software y/o firmware que lo acompaña. El término “circuitos” también cubriría, por ejemplo y, si corresponde, al elemento de reivindicación particular, un circuito integrado de banda base o un circuito integrado de procesador de aplicaciones para un teléfono móvil o un dispositivo de posicionamiento.

Con respecto a los aspectos de la invención y sus realizaciones descritas en esta solicitud, se entiende que una divulgación de cualquier acción o etapa se entenderá como una divulgación de una configuración correspondiente (funcional) de un aparato correspondiente (por ejemplo una configuración del código de programa de ordenador y/o el procesador y/o algún otro medio del aparato correspondiente), de un código de programa de ordenador correspondiente definido para causar dicha acción o etapa cuando se ejecuta y/o de una configuración (funcional) correspondiente de un sistema (o partes del mismo).

También debe entenderse que la secuencia de etapas del método en los diagramas de flujo presentados anteriormente no es obligatoria, también pueden ser posibles secuencias alternativas.

La invención se ha descrito anteriormente mediante ejemplos no limitantes. En particular, debe observarse que existen formas y variaciones alternativas las cuales son obvias para una persona experta en la técnica y que pueden implementarse sin desviarse del alcance de las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un aparato (1) que comprende:

medios para formar (305) un vector de código base combinando componentes de vector de un sub-vector señalado por un primer puntero con componentes de vector de un sub-vector señalado por un segundo puntero, en donde el sub-vector señalado por el primer puntero y el sub-vector señalado por el segundo puntero están contenidos en una tabla que comprende una pluralidad de sub-vectores, en donde cada entrada de la tabla es un sub-vector y cada sub-vector tiene componentes vectoriales de un vector de código base, en donde el vector de código base es una clase líder de un código cifrado estructurado para un códec de audio o de voz que opera en una pluralidad de modos operativos o tasas de codificación operativas, en donde la clase líder comprende un vector líder y permutaciones del vector líder, en donde el primer puntero y el segundo puntero están contenidos en una tabla adicional en donde una entrada de la tabla adicional comprende el primer puntero y el segundo puntero, en donde un valor del primer puntero que señala a un sub-vector en la tabla y un valor del segundo puntero que señala a un sub-vector en la tabla son enteros del módulo n, en donde n está relacionado con el valor máximo del tipo de datos asignados para el almacenamiento del valor del primer puntero y el valor del segundo puntero, y en donde el aparato se caracteriza porque cuando el valor del primer puntero es mayor o igual que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por el valor del primer puntero, y en donde cuando el valor del primer puntero es menor que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por la suma del valor del primer puntero a n.

2. El aparato (1) de acuerdo con la reivindicación 1, en donde los componentes del vector son un grupo de componentes del vector contiguos.

3. El aparato (1) de acuerdo con las reivindicaciones 1 y 2, en donde el uso de enteros del módulo n para el valor del primer puntero y el valor del segundo puntero depende de un modo operativo particular o tasa de codificación operativa del códec de audio/voz.

4. El aparato (1) de acuerdo con las reivindicaciones 1 a 3, en donde el tipo de datos es un tipo de datos de caracteres sin signo, y en donde n es 256.

5. El aparato (1) como se reivindica en las reivindicaciones 1 a 4, en donde el uno o más vectores de código base son un conjunto de vectores de código base los cuales definen el código cifrado estructurado, y en donde el conjunto de vectores base son clases líder, en donde cada clase líder comprende un vector líder diferente y permutaciones de dicho vector líder.

6. Un método que comprende:

formar (305) un vector de código base combinando los componentes del vector de un sub-vector señalado por un primer puntero con componentes del vector de un sub-vector señalado por un segundo puntero, en donde el sub-vector señalado por el primer puntero y el sub-vector señalado por el segundo puntero están contenidos en una tabla que comprende una pluralidad de sub-vectores, en donde cada entrada de la tabla es un sub-vector y cada sub-vector tiene componentes vectoriales de un vector de código base, en donde el vector de código base es una clase líder de un código cifrado estructurado para un códec de audio o de voz que funciona en una pluralidad de modos operativos o tasas de codificación operativas, en donde la clase líder comprende un vector líder y permutaciones del vector líder, en donde el primer puntero y el segundo el puntero están contenidos en una tabla adicional en donde una entrada de la tabla adicional comprende el primer puntero y el segundo puntero, en donde un valor del primer puntero que señala a un sub-vector en la tabla y un valor del segundo puntero que señala a un sub-vector en la tabla son enteros del módulo n, en donde n está relacionado con el valor máximo del tipo de datos asignados para el almacenamiento del valor del primer puntero y el valor del segundo puntero, y en donde el método se caracteriza porque cuando el valor del primer puntero es mayor o igual que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por el valor del primer puntero, y en donde cuando el valor del primer puntero es menor que el valor del segundo puntero, el primer puntero señala a un sub-vector en la tabla en la ubicación dada por la suma del valor del primer puntero a n.

7. El método de acuerdo con la reivindicación 6, en donde los componentes del vector son un grupo de componentes del vector contiguos.

8. El método de acuerdo con las reivindicaciones 6 y 7, en donde el uso de enteros del módulo n para el valor del primer puntero y el valor del segundo puntero depende de un modo operativo particular o tasa de codificación operativa del códec de audio/voz.

9. El método de acuerdo con las reivindicaciones 6 a 8, en donde el tipo de datos es un tipo de datos de caracteres sin signo y en donde n es 256.

10. El método de acuerdo con las reivindicaciones 6 a 9, en donde el uno o más vectores de código base son un conjunto de vectores de código base los cuales definen el código cifrado estructurado, y en donde el conjunto de vectores base son clases líder, en donde cada clase líder comprende un vector líder diferente y permutaciones de dicho vector líder.

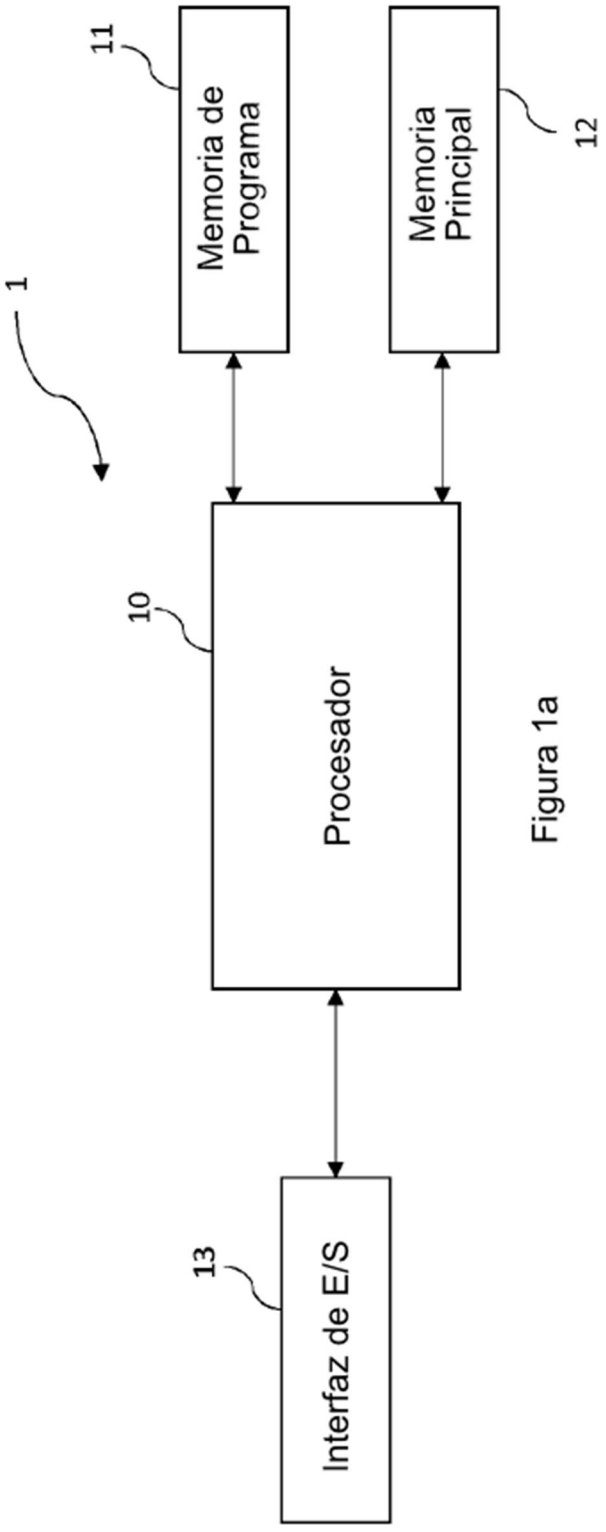

Figura 1a

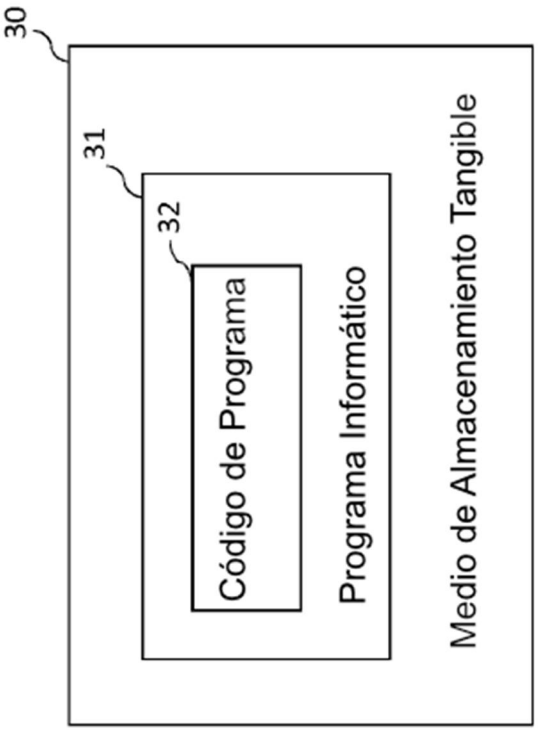

Figura 1b

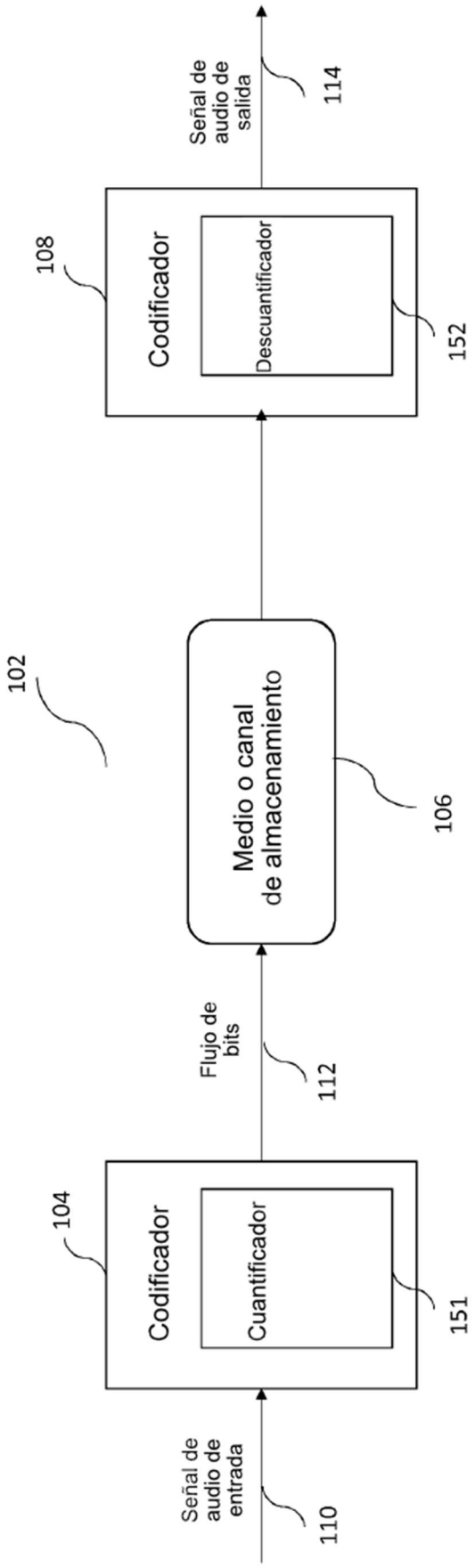
Señal de audio de entrada
110
Codificador
Cuantificador
104
151
Flujo de bits
112
Medio o canal de almacenamiento
102
106
Codificador
Descuantificador
108
152
Señal de audio de salida
114

Figura 2

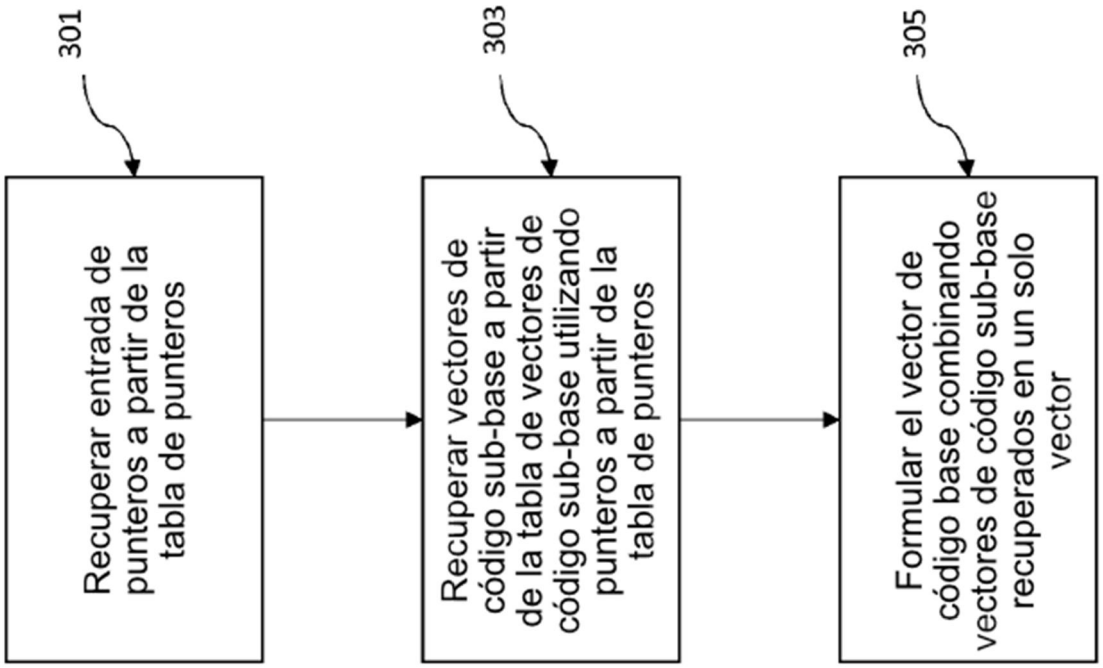
301
Recuperar entrada de punteros a partir de la tabla de punteros
303
Recuperar vectores de código sub-base a partir de la tabla de vectores de código sub-base utilizando punteros a partir de la tabla de punteros
305
Formular el vector de código base combinando vectores de código sub-base recuperados en un solo vector

Figura 3