(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 376 818**

(51) Int. Cl.:
H04L 9/06 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **05110495 .8**
(96) Fecha de presentación : **08.11.2005**
(97) Número de publicación de la solicitud: **1783948**
(97) Fecha de publicación de la solicitud: **09.05.2007**

(54) Título: **Métodos de aleatorización y desaleatorización de unidades de datos.**

(73) Titular/es: **IRDETO ACCESS B.V.**
Jupiterstraat 42
2132 HD Hoofddorp, NL

(45) Fecha de publicación de la mención BOPI:
20.03.2012

(72) Inventor/es:
Van de Ven, Antonius Johannes Petrus Maria

(45) Fecha de la publicación del folleto de la patente:
20.03.2012

(74) Agente/Representante:
Tomas Gil, Tesifonte Enrique

DESCRIPCIÓN

Métodos de aleatorización y desaleatorización de unidades de datos.

La invención se refiere a un método de aleatorización de un flujo de datos, que incluye el hecho de

obtener del flujo una sucesión de primeras secuencias de bloques de datos,

invertir el orden de los bloques en cada una de las primeras secuencias de bloques para formar segundas secuencias respectivas de bloques de datos, y codificar los bloques en cada segunda secuencia de bloques usando un modo de encadenamiento de cifra en bloque, iniciado con un vector de inicialización respectivo para cada segunda secuencia de bloques.

La invención también se refiere a un sistema para codificar un flujo de datos, que incluye

una entrada para recibir el flujo en una sucesión de primeras secuencias de bloques de datos,

una pluralidad de registros y al menos una unidad lógica para la inversión del orden de los bloques en cada una de las primeras secuencias de bloques para formar segundas frecuencias respectivas de bloques de datos, y

una disposición de tratamiento para la codificación de los bloques en cada segunda secuencia de bloques que usa un modo de encadenamiento de cifra en bloque, iniciado con un vector de inicialización respectivo para cada segunda secuencia de bloques.

La invención también se refiere a un método de descodificación de un flujo de datos aleatorizados para formar un flujo de datos, que incluye el hecho de

obtener del flujo de datos aleatorizados una sucesión de secuencias de bloques de datos aleatorizados, y descodificar cada secuencia de bloques de datos aleatorizados para formar una secuencia asociada de bloques de datos desaleatorizados, mediante el uso de una cifra de descifrado en modo de encadenamiento inverso, donde, para descodificar una secuencia de bloques de datos aleatorizados,

un bloque final en la secuencia de bloques de datos desaleatorizados se obtiene mediante la aplicación de la cifra de descifrado a un bloque final en la secuencia asociada de bloques de datos aleatorizados y por aplicación de un operador que tiene como operandos al menos el resultado de la cifra de descifrado y un vector de inicialización, y donde cada bloque que precede el bloque final en la secuencia de bloques de datos desaleatorizados se obtiene mediante la aplicación de la cifra de descifrado a un bloque en la secuencia de bloques de datos aleatorizados en una posición correspondiente y por aplicación de un operador que tiene como operandos al menos el resultado de la cifra de descodificación y un bloque de datos aleatorizados en una posición contigua en la secuencia de bloques de datos aleatorizados.

La invención también se refiere a un sistema para descodificar un flujo de datos aleatorizados para formar un flujo de datos, que incluye

una entrada para recibir el flujo de datos aleatorizados como una sucesión de secuencias de bloques de datos aleatorizados, y una disposición de tratamiento para descodificar cada secuencia de bloques de datos aleatorizados para formar una secuencia asociada de bloques de datos desaleatorizados, usando un cifra de descifrado en modo de encadenamiento inverso, donde, para descodificar una secuencia de bloques de datos aleatorizados, se obtiene un bloque final de datos desaleatorizados en la secuencia por aplicación de la cifra de descifrado a un bloque final en la secuencia asociada de bloques de datos aleatorizados y la aplicación de un operador que tiene como operandos al menos el resultado de la cifra de descifrado y un vector de inicialización, y donde cada bloque precedente de datos desaleatorizados en la secuencia se obtiene por aplicación de la cifra de descifrado a un bloque en la secuencia de bloques de datos aleatorizados en una posición correspondiente y por aplicación de un operador que tiene como operandos al menos el resultado de la cifra de descifrado y un bloque de datos desaleatorizados en una posición siguiente en la secuencia de bloques de datos aleatorizados.

La invención también se refiere a un aparato para el envío y la recepción de datos.

La invención también se refiere a un programa informático.

Ejemplos respectivos de tales métodos y sistemas son conocidos gracias a la WO 95/10906. En el método conocido, los datos digitales se dividen en paquetes de N bloques, X(1), X(2), ... X(N), donde cada bloque tiene 2^m bits. La secuencia de bloques se invierte antes de la operación de encriptación en X(N), X(N-1), ..., X(1). Esta secuencia de bloques se codifica por el algoritmo de encriptación E de la siguiente manera (donde ^ se utiliza para indica un operador OR (XOR) exclusivo).

$$Y(1) = E[X(N) \oplus IV]$$
$$Y(i) = E[X(N-i+1) \oplus Y(i-1)] \text{ for } i > 1 \text{ and } i \leq N.$$

La secuencia de estos bloques codificados se invierte de nuevo, de tal modo que la secuencia Y(N), Y(N-1), ... Y(1) se transfiere al receptor.

En el lado de receptor, los bloques de datos originales se obtienen mediante el algoritmo de descodificación D de la siguiente manera:

$$X(i) = D[Y(N-i+1) \oplus Y(N-i)] \text{ for } i = 1,2,\ldots,N-1$$
$$X(N) = D[Y(1)] \oplus IV.$$

El método usado en el sistema conocido se indica como encadenamiento de bloque de cifra inversa o método RCBC. Muestra la ventaja de que se requiere una memoria tampón en el receptor para almacenar sólo dos bloques de datos.

Un problema del método y sistema conocidos es que requiere un tampón en el lado emisor con la capacidad de almacenar N bloques para implementar la inversión de la secuencia de bloques. Esto llega a ser un problema cuando hay muchos remitentes de datos aleatorizados en un sistema de comunicación de datos, o cuando un dispositivo tiene que funcionar como emisor y receptor de datos.

Es un objeto de la presente invención proporcionar métodos, sistemas, un aparato y programa informático de los tipos indicado en los párrafos iniciales que se puedan implementar más eficazmente mientras proporcionan un nivel aceptable de protección de contenido.

Este objeto se consigue por el método de codificación de un flujo de datos según la invención, que *se caracteriza en que*, para una sucesión de primeras secuencias de bloques incluidos en una unidad de datos en el flujo, al menos un vector de inicialización para la codificación de una segunda secuencia de bloques formados a partir de una primera secuencia de bloques en la unidad se genera en dependencia de al menos un bloque en una primera secuencia precedente de bloques de la unidad.

Debido a que la unidad de datos incluye una sucesión de primeras secuencias de bloques, cada primera secuencia de bloques está formada por bloques menores, lo que significa que se requiere menos memoria tampón para invertir el orden de los bloques. Esto es posible con un nivel aceptable de seguridad ya que al menos dos de las segundas secuencias de bloques son efectivamente encadenadas. Este encadenamiento se debe al hecho de que al menos un vector de inicialización -cada uno excepto el primero en caso de seguridad máxima- es requerido para codificar una segunda secuencia de bloques formados a partir de una primera secuencia de bloques y se genera en función de al menos un bloque en una primera secuencia precedente de bloques de la unidad.

En una forma de realización, los vectores de inicialización respectivos para la codificación de los bloques en cada segunda secuencia de bloques formados a partir de una primera secuencia de bloques se generan en función de al menos un bloque de datos anterior a un último bloque en la misma primera secuencia.

Esto tiene como efecto el hecho de conseguir una mayor variación en los vectores de inicialización. Incluso los bloques de una primera de las primeras secuencias de bloques en la sucesión incluida en una unidad se aleatorizan usando un vector de inicialización que tiene una gran probabilidad de ser único. La variación se asegura mediante la generación del vector de inicialización en dependencia de al menos un bloque de datos precedente a un último bloque de datos en la misma primera secuencia. Debido a la inversión del orden de los bloques en cada primera secuencia, uno o más bloques de datos en dependencia del cual se genera el vector de inicialización se vuelve disponible durante la descodificación antes de que el descodificador requiera el vector de inicialización. Así, la singularidad del vector de inicialización para cada primera secuencia en la sucesión de primeras secuencias incluidas en la unidad es alcanzable con una probabilidad relativamente alta sin tener que proveer para el receptor un vector de inicialización nuevo para cada primera secuencia.

En una forma de realización, cada vector de inicialización para codificar una segunda secuencia de bloques formados a partir de una primera secuencia de bloques en la unidad se genera en dependencia de al menos un bloque en cada una de cualquiera de las primeras secuencias precedentes de los bloques de la unidad.

Así, el encadenamiento entre las segundas secuencias se maximiza, en la medida en que la última primera secuencia de bloques de datos no se puede obtener en claro sin haber obtenido previamente cualquiera de las primeras secuencias de bloques precedentes en la sucesión de las primeras secuencias incluidas en la unidad.

Una forma de realización incluye la recepción un paquete de datos comprendiendo un membrete y una carga útil, donde la unidad es formada por la carga útil.

Esta forma de realización es ventajosa debido a que la carga útil puede ser aleatorizada sin tener que tamponarla completamente primero.

En una forma de realización, la cifra es una cifra de bloque configurada para funcionar en bloques básicos de un tamaño predeterminado, donde los bloques en al menos las segundas secuencias de datos corresponden en tamaño al tamaño de bloque básico.

En una forma de realización, si la unidad se constituye de la sucesión de primeras secuencias de bloques y de una cantidad de sucesión de datos del mismo tamaño que al menos un múltiple del tamaño del bloque básico, la cantidad de datos es rellenada en un tamaño igual a un múltiple del tamaño de un bloque básico para formar una primera secuencia final de al menos dos bloques,

los últimos dos bloques de la primera secuencia final de bloques son intercambiados y la orden de los bloques en la primera secuencia final de bloques se reserva para formar una segunda secuencia final de bloques de datos, los bloques de la segunda secuencia final de bloques se codifican usando la cifra del modo de encadenamiento de bloques, inicializado por un vector de inicialización generado en función de al menos un bloque en una primera secuencia precedente de bloques de la unidad.

El método se adapta así para implementar una forma de robo de texto cifrado. Esta es una forma relativamente segura de asegurar que la unidad entera está aleatorizada. Además, permite el uso de primeras secuencias formadas por un número predeterminado de bloques para codificar una primera sección de la unidad.

En una forma de realización, si una unidad siguiente en el flujo se constituye por cero o más primeras secuencias de un número predeterminado de bloques y por una cantidad de datos del mismo tamaño o inferior al tamaño de un bloque básico,

la cantidad de datos es rellenada hasta un tamaño igual al tamaño de un bloque básico para formar un bloque final, el bloque final es cifrado usando el modo de encadenamiento de cifra en bloque, iniciado por un vector de inicialización generado en función de al menos un bloque en una primera secuencia precedente de bloques de la unidad.

Así, no se necesita transmitir la cantidad de sucesión de datos en claro, incluso si es más pequeño que el tamaño de bloque básico para el cual se configura la cifra.

En una variante de esta forma de realización, el vector de inicialización se genera mediante la realización de una operación criptográfica, preferiblemente una descodificación que es una inversión de la cifra, en un vector basado en al menos un vector que es independiente de cualquier bloque en cualquier primera secuencia precedente de bloques de la unidad.

El efecto es que una variación en el vector de inicialización se puede conseguir usando el mismo vector usado para generar un vector de inicialización para bloques aleatorizados de datos en una unidad precedente. Así, una cantidad inferior de vectores se debe transmitir al descodificador, mientras que la seguridad sigue siendo relativamente buena. El uso de un descifrado que es una inversión de la cifra tiene como efecto de que se utiliza la configuración del hardware y/o software del desaleatorizador que ya está presente para la descodificación.

Según otro aspecto, el sistema para la codificación de un flujo de datos según la invención *se caracteriza en que* el sistema se instala, para una sucesión de primeras secuencias de bloques incluidas en una unidad de datos en el flujo, para generar al menos un vector de inicialización para codificar una segunda secuencia de bloques formados a partir de una primera secuencia de bloques en la unidad que depende de al menos un bloque en una primera secuencia precedente de bloques de la unidad.

Debido a su eficiencia, el sistema se adapta muy bien para ser incluido en un procesador dedicado a la codificación.

Preferiblemente, el sistema se configura para ejecutar un método según la invención.

Según otro aspecto, el método de desaleatorización de un flujo de datos aleatorizados según la invención *se caracteriza en que*, para una sucesión de secuencias de bloques de datos aleatorizados incluidos en una unidad de datos en el flujo de datos aleatorizados, al menos un vector de inicialización para la desaleatorización de una secuencia de bloques de datos aleatorizados se genera en función de al menos un bloque de datos desaleatorizados en una secuencia de bloques de datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente de bloques de datos aleatorizados de la unidad.

El método es apropiado para desaleatorizar un flujo de datos aleatorizados que se puede obtener por aplicación de un método de aleatorización de un flujo de datos según la invención. Debido a que las secuencias de bloques de datos en una sucesión se descodifican separadamente y en orden, no es tan grave si los bloques dentro de una secuencia se reciben desordenados (p. ej. en orden inverso), ya que las secuencias son más cortas que toda la sucesión de secuencias. Porque al menos un vector de inicialización para descodificar una secuencia de bloques de datos aleatorizados se

genera en función de al menos un bloque de datos desaleatorizados en una secuencia de bloques de datos desaleatorizados obtenidos por la descodificación de una secuencia precedente de bloques de datos aleatorizados de la unidad, al menos dos secuencias de bloques de datos aleatorizados son encadenados, haciendo que la descodificación ilícita sea más difícil.

En una forma de realización, vectores de inicialización respectivos para descodificar cada secuencia de bloques de datos aleatorizados se generan en función de al menos un bloque de datos obtenidos aplicando la cifra de descifrado a un bloque en la secuencia de bloques de datos aleatorizados precedente al bloque final de datos aleatorizados en la misma secuencia y aplicando un operador que tiene como operandos al menos el resultado de la cifra de descifrado y un bloque de datos aleatorizados en una posición siguiente en la misma secuencia de bloques de datos aleatorizados.

Esto tiene la ventaja de que un flujo de muchas unidades no requiere transferir muchos vectores de inicialización del codificador al descodificador para lograr una variedad suficiente en los vectores de inicialización.

En una forma de realización, cada vector de inicialización para descodificar una secuencia de bloques de datos aleatorizados en la unidad se genera en función de al menos un bloque de datos desaleatorizados de cada una de cualquier secuencia de bloques de datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente de bloques de datos aleatorizados en la unidad.

Así, en efecto, todas las secuencias en la sucesión incluida en la unidad son encadenadas.

Una forma de realización incluye recibir un paquete de datos que comprende una cabecera y una carga, donde la unidad es formada por el flujo.

En una forma de realización, la cifra de descifrado es un bloque configurado para operar en bloques básicos de un tamaño predeterminado, donde los bloques de las secuencias de bloques de datos aleatorizados corresponde en tamaño al tamaño de bloque básico.

En una forma de realización, si la unidad se constituye de la sucesión de secuencias de bloques de datos aleatorizados y de una cantidad de sucesión de datos de igual tamaño a un múltiple entero del tamaño del bloque básico y una fracción del tamaño del bloque básico, la cantidad de datos se rellena con datos predeterminados del mismo tamaño que un múltiple del tamaño del bloque básico para formar una secuencia final de bloques de datos desaleatorizados, una final de la secuencia final de bloques de datos desaleatorizados se forma aplicando la cifra de descifrado a un bloque que precede inmediatamente un bloque final de la secuencia final de datos aleatorizados, aplicando un operador XOR que tiene como operandos el resultado de una cifra de descifrado y el bloque final de la secuencia final de bloques de datos cifrados, y eliminando una parte del resultado del operador XOR que corresponde en tamaño a los datos predeterminados, cada uno de cualquiera de los bloques que preceden los dos bloques finales de cada secuencia de bloques de datos desaleatorizados se forma aplicando la cifra de descifrado a un bloque en una posición correspondiente en la primera secuencia final de bloques de datos aleatorizados y aplicando un operador XOR que tiene como operandos el resultado de la cifra de descifrado y un bloque de datos desaleatorizados en una posición siguiente en la secuencia final de bloques de datos aleatorizados, y

un bloque precedente al bloque final en la secuencia final de bloques de datos desaleatorizados se obtiene mediante la aplicación de la cifra de descifrado a un bloque formado por concatenación de la parte eliminada cuyo tamaño corresponde a los datos predeterminados y al bloque final de la secuencia final de bloques de datos aleatorizados, y por aplicación de un operador XOR que tiene como operandos el resultado de la cifra de descifrado y un vector de inicialización generado en función de al menos un bloque de datos desaleatorizados obtenidos por desaleatorización de la secuencia precedente de bloques de datos aleatorizados en la unidad.

Esto es una implementación de robo de texto cifrado por el lado de descodificador.

En una forma de realización, si una unidad siguiente está constituida por cero o más secuencias de un número de bloques predeterminado y por una sucesión de una cantidad de datos cuyo tamaño es inferior al tamaño de un bloque básico,

la cantidad de datos se rellena hasta un tamaño igual al tamaño de un bloque básico para formar un bloque final,

el bloque final se descifra usando la cifra en modo de encadenamiento de bloque, inicializada por el vector de inicialización generado dependiendo de al menos un bloque en al menos una de cualquiera de las secuencias de bloques de datos desaleatorizados obtenidos por desaleatorización de una secuencia de bloques precedente de datos aleatorizados en la unidad.

El uso de una secuencia de número de bloques predeterminado tiene el efecto el hecho de que el número de bloques por secuencia no necesita estar en comunicación con el descodificador. En el lugar donde los límites de la unidad también están predeterminados, por ejemplo en el caso de que la unidad esté formada por la carga útil de un paquete, cualquier cantidad restante de datos de tamaño inferior al tamaño del bloque básico de la cifra, puede ser

enviada aún al desaleatorizador en forma encriptada. No es necesario aumentar el tamaño de la unidad aleatorizada en comparación con la unidad desaleatorizada.

En una variante, el vector de inicialización se genera creando una operación criptográfica, preferiblemente la cifra de descodificación, en un vector basado en al menos un vector independiente de cualquier bloque en cualquier bloque precedente de datos desaleatorizados que se puede obtener por desaleatorización de una secuencia precedente de bloques de datos aleatorizados en la unidad.

Esto significa que el desaleatorizador no necesita recibir ni almacenar muchos vectores de los cuales derivan los vectores de inicialización. El vector del que derivan los vectores de inicialización puede mantenerse constante con respecto al número de unidades de datos del flujo. La operación criptográfica asegura que nunca se usa directamente en forma de operando para un operador lógico, tal como un operador OR exclusivo.

Según otro aspecto, el sistema de desaleatorización de un flujo de datos aleatorizados para formar un flujo de datos según la invención *se caracteriza en que* el sistema está configurado, para una sucesión de secuencias de bloques de datos aleatorizados incluidos en una unidad de datos que están en el flujo de datos aleatorizados, para generar al menos un vector de inicialización para descodificar una secuencia de bloques de datos aleatorizados que depende de al menos un bloque de datos desaleatorizados en una secuencia de bloques de datos desaleatorizados obtenida por desaleatorización de una secuencia precedente de bloques de datos desaleatorizados de la unidad.

Preferiblemente, el sistema se configura para llevar a cabo un método de desaleatorización según la invención.

Según otro aspecto de la invención, se provee un aparato proporcionado para enviar y recibir datos, incluyendo un dispositivo dispuesto para aplicar un método de aleatorización de un flujo de datos según la invención y un método de desaleatorización de un flujo de datos aleatorizados según la invención.

Debido a que los métodos complementarios de aleatorización y desaleatorización pueden ser implementados con los mismos requisitos de memoria tampón, este aparato no necesita tener registros para almacenar bloques que no sean usados en una de las dos operaciones. Así, una implementación económica en el hardware es posible, siendo especialmente útil para un dispositivo dispuesto para funcionar enviando y recibiendo datos, por ejemplo una entrada entre dos redes.

Según otro aspecto de la invención, se proporciona un programa informático que incluye un conjunto de instrucciones capaces, cuando se incorporan en una máquina legible media, de hacer que un sistema que tiene capacidades de procesamiento de información aplique un método para codificar un flujo de datos según la invención o un método de desaleatorización de un flujo de datos aleatorizados según la invención.

La invención será explicada ahora con más detalle en referencia a los dibujos anexos, donde

La fig. 1 ilustra un sistema para implementar métodos de aleatorización y desaleatorización de flujos de datos emitidos,

la fig. 2 ilustra una forma de realización de un método para aleatorizar un flujo de datos,

la fig. 3 ilustra un método para implementar el robo de textos de cifrado en un método de aleatorización de flujo de datos,

la fig. 4 ilustra cómo se manipulan bloques parciales individuales que forman finilla extremidad de un paquete de carga útil de Flujo de Transporte MPEG-2,

La fig. 5 ilustra una forma de realización de un método de desaleatorización de un flujo de datos,

La fig. 6 ilustra una implementación de robo de datos de texto cifrado en un desaleatorizador, y

La fig. 7 ilustra cómo se manipulan bloques parciales individuales que forman la extremidad de un paquete de carga de Flujo de Transporte MPEG-2 en un desaleatorizador,

La fig. 1 ilustra la aplicación de un método de aleatorización de transmisión de datos. Al menos parte del sistema ilustrado usa técnicas conocidas *per se* a partir del "Digital Video Broadcasting (DVB); Implementation Guidelines of the DVB Simulcrypt Standard", Informe Técnico ETSI-TR 102 035 v. 1.1.1, European Telecommunication Standards Institute, 2002.

Un proveedor de contenidos 1 proporciona un flujo de paquetes de transporte de texto común, es decir desaleatorizado, datos de contenido, a una unidad de multiplexación 2. Un generador 3 de mensaje de control de derechos (ECM) proporciona a la unidad multiplexada 2 un flujo de soporte de paquetes ECMs. Un generador 4 de mensaje de gestión de derechos (EMM) proporciona a la unidad multiplexada 2 un flujo de soporte de paquetes EMMs. Los flujos son multiplexados en un único flujo de MPEG-2 de paquetes de Flujo de transporte (TS), y proporcionados a un aleatorizador 5. La sintaxis de los paquetes MPEG-2 TS se describe de manera más completa en el estándar internacional

ISO/IEC 13818-1. El aleatorizador 5 implementa un método de aleatorización de las cargas de los paquetes MPEG-2 TS que se describirán más detalladamente a continuación. Éste recibe las palabras de control (CWs) de un generador CW 6, que se usan como claves para una cifra de bloque.

El generador CW 6 proporciona los CWs al generador ECM 3, que las codifica bajo una clave de sesión obtenida a partir de un Sistema de Autorización de Suscriptor (SAS) 7. El SAS proporciona claves de sesión con autorizaciones para suscritores individuales al generador EMM 4. El generador EMM incluye estas claves de información de autorización en el EMMs destinada a asegurar testigos provistos para suscritores individuales. Tales testigos de seguridad pueden incluir agentes de software implementados usando características tales como el oscurecimiento de código para el análisis de prevención de las rutinas contenidos en éstos. Otros ejemplos de testigos de seguridad incluyen dispositivos que incluyen procesadores provistos con características protectoras para prevenir el acceso a datos almacenados y/o análisis de rutinas cableadas en éstos.

El flujo aleatorizado de datos pasa del codificador 5 a un modulador 8, y desde ahí al transmisor 9. El transmisor 9 emite el flujo de datos aleatorizados en una red de emisión 10, por ejemplo un satélite, un cable o red terrestre, o una red comprendiendo una cantidad de estas redes. En una forma de realización alternativa, el flujo de paquetes codificados MPEG-2 TS se encapsula en otros paquetes de datos, y emite, multidifunde o unidifunde una red de datos, tal como una red de datos basada en el Protocolo Internet.

Con fines ilustrativos, un Descodificador Integrado de Recepción (IRD) primario 11 se muestra en la Fig 1. El IRD primario 11 incluye un adaptador de red 12 para recibir los datos transmitidos por la red de emisión 10. Un demodulador 14 hace que el flujo de datos aleatorizados esté disponible para un aleatorizador/desaleatorizador 14. Esta unidad se configura para llevar a cabo un método de desaleatorización de un flujo de datos y un método de aleatorización de un flujo de datos. Un procesador 15 controla la operación del IRD primario 11. Puede dirigir un flujo de datos aleatorizados generado por el codificador/descodificador 14 hacia un segundo adaptador de red 16 conectando el IRD primario 11 a una red local 17. La red local 17 puede ser una red de hogar, por ejemplo basada en el estándar IEEE 1394. Se provee un receptor secundario 18 con un adaptador de red correspondiente y un chip desaleatorizador. Otros componentes no se muestran en detalle. El chip desaleatorizador20 opera de la misma manera que el aleatorizador/desaleatorizador14. Por esta razón, no se describirá más detalladamente.

Volviendo al aleatorizador/desaleatorizador 14, este componente desaleatorizará paquetes MPEG-2TS aleatorizados usando CWs provistos por un Subsistema de Acceso Condicional (CASS) 21. El CASS 21 se conecta con un dispositivo de procesamiento seguro 22, por ejemplo una tarjeta electrónica, que proporciona las claves para descifrar ECMs para un primer descodificador 23. El primer descodificador 23 obtiene los CWs pasados al aleatorizador/desaleatorizador 14. Los EMMS se descifran con un segundo descodificador 24, para proporcionar al dispositivo de procesamiento seguro 22 la información necesaria para obtener las claves de servicio. El procesador dirige una unidad desmultiplexada 25 para salvar los paquetes MPEG-2 TS que transportan los ECMs y los que transportan los EMMs para proporcionarlos al CASS 21.

La fig. 2 ilustra una forma de realización de un método de aleatorización de un flujo de datos tales como los que son obtenidos por el aleatorizador 5 y/o aleatorizador/desaleatorizador 14. En la forma de realización ilustrada, el método es realizado por una unidad de datos formada por una carga útil 26 de un paquete MPEG-2 TS 27, que comprende un membrete 28. El membrete 28 no está aleatorizado, sino dejado en claro. Se observa que el método puede ser realizado en otro tipo de paquetes, que no son definidos necesariamente por un protocolo de red de capa de transporte. Por ejemplo, el método también puede realizarse en paquetes de Flujo de Programa Elemental (PES) transportados en paquetes MPEG-2 TS 27.

Aunque el paquete MPEG-2 TS 27 tiene una longitud fija, 188 bites, no es el caso de la carga útil 26. Esto se debe a la longitud variable del membrete 28. En el ejemplo ilustrado en la fig. 2, el flujo 26 tiene este tamaño que puede ser dividido en un número entero de bloques básicos P_i de tamaño fijo, y también en un número entero de los denominados super bloques. Cada super bloque está formado por una primera secuencia de bloques básicos P_i. Una primera secuencia inicial 29 y una última primera secuencia 30 se ilustran explícitamente. Una sucesión de super bloques constituye el flujo 26. En el ejemplo, cada super bloque está formado por una primera secuencia 29, 30 de los tres bloques básicos P_i. En otra forma de realización, estaban presentes dos bloques básicos P_i por cada super bloque. Puede haber más de tres, por ejemplo, cuatro, cinco o diez bloques básicos por super bloque. Un número mayor requiere más registros en el aleatorizador 5 y/o aleatorizador/desaleatorizador 14.

Las etapas realizadas donde la carga útil 26 no tiene el mismo tamaño que un número entero de super bloques formado por un número entero de bloques básicos P_i se tratarán más abajo.

En otra variante del método (no ilustrada), la carga útil 26 se divide en dos super bloques de tamaño variable, por ejemplo, que contiene un número variable de bloques básicos P_i.

El tamaño del bloque básico se determina preferiblemente por una cifra de bloque E_k, una clave de cifra simétrica que opera en grupos de bits de tamaño fijo, los bloques básicos, con una transformación invariable. En la forma de realización ilustrada, la cifra de bloque E_k se usa para codificar bloques básicos individuales P_i bajo una CW. Algunos ejemplos de cifras adecuadas incluyen, DES, triple DES, y AES/Rijndael. De esta manera, el tamaño del bloque básico será generalmente de 128 bits.

En una primera etapa del método, el orden de los bloques básicos P_1-P_3 en la primera secuencia inicial 29 se invierte para formar una segunda secuencia 31.

En un segundo paso, la cifra de bloque E_k se usa en un modo de encadenamiento de bloque de cifra para codificar los tres bloques básicos P_1-P_3 en la primera segunda secuencia 31, que se inicializa con un vector de inicialización IV_3 asociado a la primera segunda secuencia. El índice usado para referirse al vector de inicialización IV_3 asociado a la primera segunda secuencia 31 de bloques básicos es el índice del último bloque básico P_3 de la secuencia, como será el caso a través del presente texto. El vector de inicialización IV_3 usado para codificar la primera segunda secuencia asociada 31 de los bloques básicos se forma por aplicación de un operador exclusivo OR que tiene como operando un vector de inicialización fijado a largo plazo IV_0 y el OR exclusivo o ambos bloques básicos P_1-P_2 que preceden el último bloque básico P_3 en la primera secuencia inicial 29 de los bloques básicos P_1-P_3.

El método de codificación de las cargas 26 está configurado para que el vector de inicialización fijado a largo plazo IV_0 no sea utilizado nunca directamente como un vector de inicialización. Nunca se usa en una operación XOR inmediatamente precedente a la operación de la cifra de bloque de E_k en un primer bloque básico P_i en una segunda (es decir, invertida) secuencia de bloques. Por esta razón, se puede usar en paquetes múltiples MPEG-2 TS sin que el criptoanálisis sea sustancialmente más fácil. No es necesario mantener en secreto el vector de inicialización fijado a largo plazo IV_0. Se conocerá cuando los métodos de aleatorización y desaleatorización de unidades de datos aquí resumidos en la presente se usen en comunicaciones P2P. En la situación ilustrada en la fig. 1 donde un único proveedor controla los envíos y las recepciones, el vector de inicialización fijado a largo plazo IV_0, puede ser guardado en secreto. Puede ser provisto de un ECM o un EMM, por ejemplo. En una forma de realización alternativa, un vector del que derivan los vectores de inicialización independiente de los bloques de datos a aleatorizar se obtiene aplicando un algoritmo predeterminado en los datos incluidos en el membrete 28.

Debido a que el vector de inicialización IV_3 usado para cifrar los bloques P_1-P_3 en la primera segunda secuencia asociada 31 de los bloques se genera dependiendo de al menos un bloque de datos precedente al último bloque P_3 en la primera secuencia inicial 29 a partir de la cual se obtuvo, se consiguen más variantes de los vectores de inicialización.

El resultado de la codificación de la primera segunda secuencia 31 es una primera secuencia inicial 32 de los bloques aleatorizados C_3-C_1. En la forma de realización ilustrada, el orden de los bloques aleatorizados C_3-C_1 se invierte para formar una primera segunda secuencia 33 de bloques aleatorizados C_1-C_3. La primera segunda secuencia 33 de bloques aleatorizados C_1-C_3 se inserta en un paquete aleatorizado MPEG-2 TS 34, que comprende un membrete no codificado 35 y una carga útil aleatorizada 36.

Así, para la primera secuencia inicial 29 de los bloques básicos P_i, i = 1...m, los bloques básicos C_i codificados se obtienen de la manera siguiente:

$$C_M = E_k[P_k \hat{} IV_M]$$

$$C_i = E_k[P_i \hat{} C_{i+1}],\ i = M-1 \ldots 1.$$

En general, los bloques básicos codificados se determinan de la manera siguiente:

$$C_{j*M} = E_k[P_{j*M} \hat{} IV_{j*M}],$$

$$C_{(j-1)*M+i} = E_k[P_{(j-1)*M+i} \hat{} C_{(j-1)*M+i+1}],\ i = M-1 \ldots 1,\ j = 1 \ldots N\backslash M.$$

El símbolo “\” se refiere al cociente o a parte de un número entero de una proporción.

En la forma de realización ilustrada, el vector de inicialización IV_{j*M} para cifrar los bloques en la segunda secuencia j-th se obtiene de la siguiente manera:

$$IV_{j*M} = IV_0 \hat{} P_1 \hat{} \ldots P_i \hat{} P_{j-M+1}.$$

Así, los respectivos vectores de inicialización para codificar los bloques en cada segunda secuencia de bloques formada a partir de una primera secuencia de bloques mediante la inversión del orden de los bloques se genera dependiendo de al menos un bloque de datos precedente a un último bloque en esa misma primera secuencia. En esta forma de realización, éstos se generan dependiendo de todos los bloques de datos precedentes a un último bloque en esa misma primera secuencia.

No solo cada vector de inicialización para codificar una segunda secuencia de bloques formada por una primera secuencia de bloques en el flujo de paquete 26 se genera dependiendo de al menos un bloque, sino que en este caso todos los bloques, en cada uno de cualquiera de las primeras secuencias que preceden bloque básicos o super bloques.

Para codificar los bloques P_{N-2}-P_N en una última segunda secuencia 37, obtenida por inversión del orden de los bloques básicos en la última primera secuencia 30 de bloques básicos, la cifra de encriptación E_k es de nuevo operada en un modo de encadenamiento de bloque de cifra. Un vector de inicialización IV_N asociado con la última segunda secuencia 37 de los bloques básicos se genera realizando la operación XOR del vector de inicialización fijado a largo plazo IV_0 y cada uno de los bloques básicos que preceden el último bloque básico P_N. El resultado es una última secuencia 38 de bloques de datos aleatorizados. El orden de los bloques se invierte para obtener una última segunda secuencia 39 de los bloques de datos aleatorizados.

La fig. 3 ilustra cómo el procedimiento del método de desaleatorización, cuando el paquete de carga MPEG-2 TS 26 se constituye de un número entero de primera secuencia N\M de bloques básicos y una cantidad sucesiva de datos de tamaño inferior igual al de los bloques básicos M. En las formas de realización ilustradas, la cantidad sucesiva de datos puede dividirse en dos bloques básicos completos P_{N-2}, P_{N-1} y un bloque parcial P_N. El último bloque P_N se rellena con ceros del mismo tamaño que un bloque básico completo. Así, una primera secuencia final se forma por los bloques P_{N-2}, P_{N-2} y el bloque rellenado P_N N-th. Las posiciones de los últimos dos bloques de la primera secuencia final de bloques se intercambia para formar una segunda secuencia final 40 de bloques (mostrada en la fig. 3). Los bloques de la segunda secuencia final 40 de bloques son codificados usando la cifra de cifrado E_k en el modo de encadenamiento de bloque. Como el vector de inicialización IV_N, un vector de inicialización se genera aplicando una operación XOR que tiene como operando el vector de inicialización fijado a largo plazo IV_0, y cada uno de los bloques básicos completos que preceden el último bloque completo P_{N-1} precedente al bloque parcial P_N en el paquete de carga MPEG-2 TS 26. El resultado es una primera secuencia final 41 de bloques de datos aleatorizados. Una cantidad de datos aleatorizados C' que corresponde en tamaño y posición a los datos añadidos por el relleno se extrae del primer bloque en la primera secuencia 41 de bloques, y el orden de los bloques es invertida después para obtener una segunda secuencia final de bloques aleatorizados (no mostrada), que se inserta en el flujo codificado 36. Por supuesto, los datos aleatorizados C' pueden ser extraídos después para invertir el orden de los bloques.

Si el paquete de carga MPEG-2 TS 26 se constituye por un número entero de primeras secuencias N\M de bloques básicos y una cantidad sucesiva de datos cuyo tamaño es inferior al de un bloque básico, la operación descrita en la fig. 4 se realiza. Un bloque parcial P_N se rellena con ceros para un bloque final de tamaño completo 42. Un vector de inicialización IV_N se genera en función del vector de inicialización fijado a largo plazo IV_0 y de al menos un bloque en cualquier primera secuencia de bloque que precede el paquete de carga 26. Podría ocurrir que no haya primeras secuencias precedentes. Por lo que, para no utilizar el vector de inicialización fijado a largo plazo IV_0 directamente como operando en una operación XOR, el vector de inicialización IV_N asociado con el bloque final de tamaño completo 39 se descodifica primero, utilizando una cifra de descifrado que es inversa a la cifra de cifrado E_k y al CW como clave. El OR exclusivo del resultado y el bloque final de tamaño completo 42 se obtienen, y son codificados aplicando el bloque de cifra E_k. El resultado es un bloque aleatorizado de tamaño completo 43, que está truncado por eliminación de una parte correspondiente al tamaño y posición a los datos añadidos rellenando el bloque parcial P_N. La parte restante C_N se inserta en una carga útil aleatorizada 36.

Las cargas útiles de una sucesión de paquetes MPEG-2 TS 27 se codifican de esta manera, formando así un flujo aleatorizado de datos. El robo de texto cifrado y el método para manipular bloques parciales únicos aseguran que la carga aleatorizada 36 es igual en tamaño a la carga 26 del paquete de texto común original MPEG-2 TS 27. Así, el membrete 35 del paquete aleatorizado MPEG-2 TS 34 no necesita ser alterado substancialmente en relación con el texto común del paquete MPEG-2 TS 27, excepto para indicar que se ha aleatorizado y, opcionalmente, de un CW par e impar que se han utilizado para la cifra de bloques E_k.

La fig. 5 ilustra la operación de desaleatorización que corresponde a la operación de aleatorización ilustrada en la fig. 2. Como en la fig. 2, la fig. 5 está basada en la suposición de que existen exactamente N\M super bloques en la carga útil aleatorizada 36. La carga útil aleatorizada 36 forma una sucesión de secuencias de bloques C_i de datos aleatorizados, cada secuencia corresponde a un super bloque. Se muestra una primera secuencia 44 de bloques C_1-C_3 de datos aleatorizados y una última secuencia 45 de bloques C_{N-2}-C_N.

Cada secuencia de bloques de datos aleatorizados se desaleatoriza por separado para formar una secuencia asociada de bloques de datos desaleatorizados. Así, la primera secuencia 44 de los bloques C_1-C_3 se desaleatoriza para formar una primera secuencia 46 de bloques de texto P_1-P_3. La última secuencia 45 de bloques C_{N-2}-C_N se desaleatoriza para formar una última secuencia 47 de bloques de texto P_{N-2}-P_N. Las secuencias de bloque de texto se usan para formar un texto de paquete de carga MPEG-2 TS 48, precedido por un membrete 49, y formar así un paquete de texto común reconstituido MPEG-2 TS 50.

Un primer bloque P_1 de datos desaleatorizados en la primera secuencia 46 de los bloques de texto se obtiene aplicando una cifra de descripción D_k que es la inversa de la cifra de cifrado E_k a un primer bloque C_1 de datos aleatorizados y aplicando un operador XOR que tiene como operando el resultado de la cifra de descifrado D_k y un bloque aleatorizado siguiente C_2 en la secuencia 44 de bloques de datos aleatorizados. Un segundo bloque P_2 se obtiene de la misma manera. El CW obtenido de una ECM se usa como clave para la cifra de descifrado D_k.

Un bloque final P_3 de datos desaleatorizados en la primera secuencia 46 de bloques de texto se obtiene aplicando la cifra de descifrado D_k a un bloque final C_3 en la primera secuencia 44 de bloques C_1-C_3 de datos aleatorizados. El resultado es XOR-ed con un vector de inicialización IV_3 asociado con el primer super bloque. Este vector de inicialización IV_3 se genera en función de al menos un bloque de datos obtenidos aplicando la cifra de descifrado

a un bloque en la primera secuencia 44 de bloques de datos aleatorizados que precede al bloque final C_3 de datos aleatorizados en la misma secuencia aplicando un operador, el operados XOR, que tiene de operando al menos el resultado de la cifra de descifrado D_k y bloque de datos desaleatorizados en una posición siguiente en la secuencia de bloques de datos aleatorizados. En la forma de realización ilustrada, el vector de inicialización IV_3 es el XOR del vector de inicialización fijado a largo plazo IV_0 y todos los bloques de la primera secuencia 46 de bloques de texto que precede el bloque de texto final P_3. Dado que estos bloques de texto precedentes se obtienen antes de que se tenga que obtener el bloque de texto final P_3, el método de desaleatorización es relativamente eficiente.

En general, en una forma de realización que usa un número fijo M de bloques de datos aleatorizados para cada una de las sucesiones de secuencias de bloques de datos aleatorizados incluidas en la carga útil aleatorizada 35, los bloques de texto se determinan de la siguiente manera:

$$P_{j*M} = D_k[P_{j*M} \hat{} IV_{j*M}],$$

$$P_{(j-1)*M+i} = C_{(j-1)*M+i+1} \hat{} D_k[C_{(j-1)*M+i}], \quad i = M-1 \ldots 1, \quad j = 1 \ldots N\backslash M.$$

En la forma de realización ilustrada, el vector de inicialización IV_{j*M} para descodificar los bloques en la secuencia j-th se obtiene de la siguiente manera:

$$IV_{j*M} = IV_0 \hat{} P_1 \hat{} \ldots P_i \hat{} P_{j*M-1}.$$

De esta manera, el vector de inicialización utilizado para descodificar la segunda y otras secuencias de bloques de datos aleatorizados se generan en función de al menos un bloque de datos desaleatorizados en una secuencia de bloques de datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente de bloques de datos desaleatorizados en la carga útil aleatorizada 36.

La fig, 6 ilustra el procedimiento del método de aleatorización cuando la carga útil aleatorizada 36 está constituida por un número entero de las primeras secuencias N\M de bloques básicos y la cantidad sucesiva de datos del mismo tamaño que un múltiple entero del tamaño del bloque básico y una fracción del tamaño del bloque básico. En la forma de realización ilustrada, la cantidad sucesiva de datos puede ser dividida en dos bloques codificados completos C_{N-2}, C_{N-1} y en un bloque parcial C_N. Así, se forma una secuencia final con los bloques C_{N-2}. C_{N-1} y el último bloque básico completo. Un bloque de texto final P_N se forma aplicando la cifra de cifrado D_k al bloque C_{N-1} que precede inmediatamente al bloque final del final de la secuencia de bloques de datos aleatorizados, aplicando un operador XOR al resultado de la cifra de cifrado y el último bloque básico completo 51 y extrayendo una parte C' del resultado del operador XOR que corresponde en tamaño y posición al de los datos añadidos rellenando el último bloque codificado C_N. Cada uno de cualquiera de los bloques que preceden los últimos dos bloques de texto P_{N-1}, P_N - en este caso, el primer bloque P_{N-2} de la secuencia es solo dicho bloque- formado aplicando la cifra de cifrado D_k a un bloque en una posición correspondiente en el final de la secuencia de bloques de datos aleatorizados y aplicando un operador XOR al resultado de la cifra de cifrado D_k y un bloque de datos aleatorizados lindando en la secuencia final de bloques de datos aleatorizados.

El bloque P_{N-1} precede el último bloque en la secuencia final de bloques de texto que se obtiene aplicando la cifra de descifrado a un bloque 52 formado por concatenación del último bloque aleatorizado C_N y la parte extraída C' que corresponde en tamaño y posición a los ceros usados para el relleno, y luego por aplicación de un operador XOR que tiene como operando el resultado de la cifra de cifrado D_k y un vector de inicialización IV_N. Como vector de inicialización IV_N, un vector de inicialización se genera por aplicación de una operación XOR que tienen como operando un vector de inicialización fijado a largo plazo IV_0 y cada uno de los últimos dos bloques P_{N-1}, P_N en el paquete de carga útil 48 de texto común MPEG-2 TS.

Si la carga útil aleatorizada 36 se constituye por una secuencia de número entero de N\M bloques aleatorizados y la cantidad sucesiva de datos de tamaño inferior al de un bloque básico, la operación descrita en la fig. 7 se realiza. Un bloque parcial aleatorizado C_N se rellena con ceros hasta obtener un bloque aleatorizado final 53 de tamaño completo. Un vector de inicialización IV_N se genera en función del vector de inicialización fijado a largo plazo IV_0 y al menos un bloque en cualquier secuencia que lo precede de los bloques aleatorizados de la carga útil aleatorizada 36. Puede ocurrir que no haya secuencias precedentes de bloques aleatorizados. Para no usar el vector de inicialización fijado a largo plazo IV_0 directamente como operando en una operación XOR, el vector de inicialización asociado con el bloque codificado final a tamaño completo 53 se descifra primero, utilizando la misma cifra de descifrado D_k utilizada en el modo de encadenamiento de bloque, con el CW usado como clave. El OR exclusivo o el resultado y el bloque descodificador final a tamaño completo 50 es obtenido. El resultado es un bloque de tamaño completo 54, que está truncado por eliminación de una parte C' cuyo tamaño y posición corresponden a los datos añadidos por relleno del bloque aleatorizado parcial C_N. La parte restante P_N se inserta en la carga útil de paquete del texto común MPEG-2 TS.

Así, un método en el que los super bloques tienen un tamaño predeterminado, pero que pueden asociarse con cargas útiles de paquetes de diferentes tamaños incluyendo las que no son un múltiple entero del tamaño seleccionado del super bloque se han descrito en detalle. El vector de inicialización para cada super bloque no se usa nunca directamente

como máscara para un operador XOR. Éste depende al máximo de bloques de texto común precedentes en la carga útil del paquete, para que se pueda obtener toda la variedad posible en el vector de inicialización. Los requisitos de memoria para almacenar los resultados de las operaciones son sustancialmente los mismos para el aleatorizador y desaleatorizador y se pueden mantener tan bajos como se considere aceptable por motivos de seguridad escogiendo un tamaño inferior de super bloque.

La invención no se limita a las formas de realización descritas, las cuales pueden variar en el campo de las reivindicaciones anexas. En particular, se puede tener una forma de realización en la que la primera secuencia de los bloques aleatorizados no se convierte en una segunda secuencia de datos aleatorizados mediante la inversión del orden de los bloques en el aleatorizador, sino donde la inversión se realiza en el desaleatorizador antes de realizar el método de desaleatorización de un flujo de datos aleatorizados ilustrado en la presente.

REIVINDICACIONES

1. Método de aleatorización de un flujo de datos, que incluye obtener a partir de la sucesión de las primeras secuencias (29, 30) de bloques (P_i) de datos,

invertir el orden de los bloques (P_i) en cada una de las primeras secuencias (29, 30) de los bloques para formar segundas secuencias respectivas de bloques de datos, y

y codificar los bloques en cada segunda secuencia (31, 37) de bloques usando una cifra (E_k) en el bloque a modo de encadenamiento, inicializado con un vector de inicialización respectivo (IV_3, IV_N) para cada segunda secuencia (31, 37) de bloques,

caracterizado por el hecho de que,

para una sucesión de primeras secuencias (29, 30) de bloques incluida en una unidad (26) de datos en el flujo, al menos un vector de inicialización (IV_N) usado para codificar una segunda secuencia (37) de bloques formados a partir de una primera secuencia (30) de bloques en la unidad de datos es generado en función de al menos un bloque en una primera secuencia precedente (29) de los bloques de la unidad.

2. Método según la reivindicación 1, donde vectores de inicialización respectivos (IV_3, IV_N) para codificar los bloques en cada segunda secuencia (31, 37) de bloques formado por una primera secuencia (29, 30) de bloques generado en función de al menos un bloque de datos que precede el último bloque de la misma secuencia (29, 30).

3. Método según la reivindicación 1 o 2, donde cada vector de inicialización (IV_N) se genera en función de al menos un bloque en cada una de cualquiera de las primeras secuencias precedentes (29) de bloques de la unidad (26).

4. Método según cualquiera de las reivindicaciones 1 a 3, que incluye la recepción de un paquete de datos (27) que comprende un membrete (28) y una carga útil (26), donde la unidad es formada por el la carga útil.

5. Método según cualquiera de las reivindicaciones 1 a 4, donde la cifra (E_k) es una cifra de bloque configurado para funcionar en bloques básicos de tamaño predeterminado, donde los bloques (P_i) en al menos las segundas secuencias (31, 37) de datos corresponden en tamaño al tamaño de bloque básico.

6. Método según la reivindicación 5, donde, si la unidad (26) se constituye de la sucesión de primeras secuencias (29, 30) de bloques y la cantidad sucesiva de datos cuyo tamaño es inferior a un múltiple del tamaño del bloque básico,

la cantidad de datos se completa en un tamaño igual al múltiple del tamaño de un bloque básico para formar una primera secuencia final de al menos dos bloques,

los dos últimos bloques de la primera secuencia final de bloques se intercambian y el orden de los bloques en la primera secuencia final de bloques se invierte para formar una segunda secuencia final (40) de bloque de datos, los bloques de la segunda secuencia final (40) de bloques se codifican utilizando la cifra (E_k) a modo de encadenamiento de bloques, inicializado por un vector de inicialización (IV_N) generado en función de al menos un bloque en una primera secuencia precedente (29, 30) de bloques de la unidad.

7. Método según la reivindicación 5, donde, si la unidad está constituida por cero o más primeras secuencias de un número predeterminado de bloques y por una cantidad de datos (P_N) cuyo tamaño es igual a menos del tamaño del bloque básico,

la cantidad de datos se completa hasta un tamaño igual al tamaño de un bloque básico para formar un bloque final (42), la cantidad de datos se completa hasta alcanzar un tamaño igual al tamaño de un bloque básico para formar un bloque final (42), el bloque final (42) se codifica usando la cifra (E_k) a modo de encadenamiento de bloques, inicializado por un vector de inicialización generado en función de al menos un bloque en al menos una de cualquiera de las secuencias precedentes de bloques de la unidad.

8. Método según la reivindicación 7, donde el vector de inicialización se genera aplicando una operación criptográfica (D_k), preferiblemente una descodificación que es la inversa de la cifra, en un vector (IV_N) basado en al menos un vector (IV_0) que es independiente de cualquier bloque en cualquier primera secuencia precedente de bloques de la unidad.

9. Sistema de aleatorización de un flujo de datos, que incluye

una entrada para recibir el flujo a modo de sucesión de primeras secuencias (29, 30) de bloques (P_i) de datos, una pluralidad de registros y al menos una unidad lógica para invertir el orden de los bloques en cada una de las primeras secuencias (29, 39) de bloques (P_i) para formar segundas secuencias respectivas (31, 37) de bloques de datos, y

una disposición de procesamiento para codificar los bloques en cada segunda secuencia (31, 37) de bloques usando una cifra (E_k) a modo de encadenamiento de bloques, inicializado con un vector de inicialización respectivo (IV_3, IV_N) para cada segunda secuencia (31, 37) de bloques,

caracterizado por el hecho de que,

el sistema se dispone, para una sucesión de primeras secuencias (29, 30) de bloques incluidos en una unidad (26) de datos en el flujo, para generar al menos un vector de inicialización (IV_N) usado para codificar una segunda secuencia (37) de bloques formada a partir de una primera secuencia (30) de bloques en la unidad de datos en función de al menos un bloque en una primera secuencia precedente (29) de bloques de la unidad (26).

10. Sistema según la reivindicación 9, configurado para ejecutar un método según cualquiera de las reivindicaciones 1 a 8.

11. Método de desaleatorización de un flujo de datos aleatorizados para formar un flujo de datos, que incluye el hecho de:

obtener del flujo de datos aleatorizados una sucesión de secuencias (44, 45) de bloques (C_i) de datos aleatorizados, y

desaleatorizar cada secuencia (44, 45) de bloques de datos aleatorizados para formar una secuencia asociada (46, 47) de bloques (P_i) de datos desaleatorizados, usando una cifra de descifrado (D_k) en modo de encadenamiento inverso, donde, para descifrar una secuencia (46, 47) de bloques de datos aleatorizados,

un bloque final (P_3, P_N) en la secuencia (46, 47) de bloques de datos desaleatorizados se obtiene aplicando la cifra de descifrado (D_k) a un bloque final (C_3, C_N) en la secuencia asociada (44, 45) de bloques de datos aleatorizados y aplicando un operador que tenga como operando al menos el resultado de la cifra de descifrado (D_k) y un vector de inicialización (IV_3, IV_N), y donde cada bloque que precede el bloque final (P_3, P_N) en la secuencia (46, 47) de bloques de datos desaleatorizados se obtiene aplicando la cifra de descifrado (D_k) en un bloque en la secuencia de un bloque (44, 45) de datos aleatorizados en una posición correspondiente y aplicando un operador que tiene como operando al menos el resultado de la cifra de descifrado (D_k) y un bloque de datos codificados en una posición sucesiva en la secuencia (44, 45) de bloques de datos aleatorizados,

caracterizado por el hecho de que,

para una sucesión de secuencia (44, 45) de bloques de datos aleatorizados incluida en una unidad (36) de datos en el flujo de datos aleatorizados, al menos un vector de inicialización (IV_N) usado para desaleatorizar una secuencia (45) de bloques de datos aleatorizados se genera en función de al menos un bloque de datos desaleatorizados en una secuencia (46) de bloques de datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente (44) de bloques de datos aleatorizados de la unidad (36).

12. Método según la reivindicación 11, donde los respectivos vectores de inicialización (IV_3, IV_N) para desaleatorizar cada secuencia (44, 45) de bloques (C_i) de datos aleatorizados se generan en función de al menos un bloque de datos desaleatorizados obtenido aplicando la cifra de descifrado (D_k) a un bloque en la secuencia (44, 45) de bloques de datos aleatorizados que precede el bloque final (C_3, C_N) de datos aleatorizados en la misma secuencia (44, 45) y aplicando un operador que tiene como operando al menos el resultado de la cifra de descifrado y un bloque de datos aleatorizados en una posición siguiente en la misma secuencia (44, 45) de bloques de datos aleatorizados.

13. Método según la reivindicación 11 o 12, donde cada vector de inicialización (IV_N) para desaleatorizar una secuencia (45) de bloques de datos aleatorizados en la unidad (36) se genera en función de al menos un bloque de datos desaleatorizados de cada una de cualquiera de las secuencias (46) de bloques de datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente (44) de bloques de datos aleatorizados en la unidad (36).

14. Método según las reivindicaciones 11 a 13, que incluye la recepción de un paquete de datos (34) comprendiendo un membrete (35) y una carga útil (36), donde la unidad es formada por la carga útil.

15. Método según cualquiera de las reivindicaciones 11 a 14, donde la cifra de descifrado (D_k) es una cifra de bloque configurada para funcionar en bloques básicos de tamaño predeterminado, donde los bloques (C_i) en las secuencias (44, 45) de bloques de datos aleatorizados corresponden en tamaño al tamaño del bloque básico.

16. Método según la reivindicación 15, donde, si la unidad (36) se constituye de la sucesión de secuencias de bloques de datos aleatorizados y una cantidad sucesiva de datos del mismo tamaño que un múltiple entero del tamaño del bloque básico y una fracción (C_N) del tamaño del bloque básico,

la cantidad de datos se completa con datos predeterminados de tamaño igual a un múltiple del tamaño de bloques básicos para formar una secuencia final de bloques de datos aleatorizados,

una final (P_N) de una secuencia final de bloques de datos desaleatorizados se forma aplicando la cifra de descifrado a un bloque que precede inmediatamente el bloque final de la secuencia final de datos aleatorizados, aplicando un operador XOR que tiene como operando el resultado de la cifra de descifrado y el bloque final de la secuencia final de bloques de datos aleatorizados, y eliminando una parte (C') del resultado del operador XOR que corresponde en tamaño a los datos predeterminados,

cada uno de cualquier bloque que precede los dos bloques finales (P_{N-1}, P_N) de la secuencia final de bloques de datos desaleatorizados se forma aplicando la cifra de descifrado (D_k) para un bloque en una posición correspondiente en la primera secuencia final de bloques de datos aleatorizados y aplicando un operador XOR que tiene como operando el resultado de la cifra de descifrado y un bloque de datos aleatorizados en una posición siguiente en la secuencia final de bloques de datos aleatorizados, y

un bloque (P_{N-1}) que precede el bloque final (P_N) en la secuencia final de bloques de datos desaleatorizados se obtiene aplicando la cifra de descifrado a un bloque (52) formado por concatenación de la parte eliminada (C') que corresponde en tamaño al tamaño de los datos predeterminados y el bloque final de la secuencia final de bloques de datos aleatorizados, y aplicando un operador XOR que tiene como operando el resultado de la cifra de descifrado y un vector de inicialización (IV_N) generado en función de al menos un bloque en al menos una de cualquiera de las secuencias de bloques de los datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente de bloques de datos aleatorizados en la unidad (36).

17. Método según la reivindicación 15, donde, si una unidad sucesiva está constituida por cero o más secuencias de un número predeterminado de bloques y por una cantidad sucesiva (C_N) de datos del tamaño igual a menos del tamaño de un bloque básico, la cantidad de datos se completa hasta un tamaño igual al tamaño de un bloque básico para formar un bloque final (53),

el bloque final (53) se descodifica utilizando la cifra (D_k) a modo de encadenamiento de bloque, inicializado por un vector de inicialización generado en función de al menos un bloque en al menos una de cualquiera de las secuencias de bloques de los datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente de bloques aleatorizados en la unidad.

18. Método según la reivindicación 17, donde el vector de inicialización se genera aplicando una operación criptográfica (D_k), preferiblemente la cifra de descifrado, en un vector basado en al menos un vector independiente de cualquier bloque en cualquier bloque precedente de datos desaleatorizados obtenibles por desaleatorización de una secuencia precedente de bloques de datos cifrados en la unidad.

19. Sistema para desaleatorizar un flujo de datos aleatorizados para formar un flujo de datos, que incluye:

una entrada para recibir el flujo de datos aleatorizados en forma de sucesión de secuencias (44, 45) de bloques (C_i) de datos aleatorizados, y

una disposición del procesador para desaleatorizar cada secuencia (44, 45) de bloques (C_i) de datos aleatorizados para formar una secuencia asociada (46, 47) de bloques de datos desaleatorizados, usando una cifra de descifrado (DK) en un modo de encadenamiento inverso, donde para desaleatorizar una secuencia (44, 45) de bloques de datos aleatorizados,

se obtiene un bloque final (P_3, P_N) de datos desaleatorizados en la secuencia (46, 47) aplicando la cifra de descifrado (D_k) a un bloque final (C_3, C_N) en la secuencia asociada (44, 45) de bloques de datos aleatorizados y aplicando un operador que tiene como operando al menos el resultado de la cifra de descifrado (D_k) y un vector de inicialización (IV_3, IV_N), y donde cada bloque precedente de datos desaleatorizados en la secuencia (46, 47) se obtiene aplicando la cifra de descifrado a un bloque en la secuencia de bloques de datos aleatorizados en una posición correspondiente y aplicando un operador que tiene como operando al menos el resultado de la cifra de descifrado y un bloque de datos aleatorizados en la posición siguiente en la secuencia (44, 45) de bloques de datos aleatorizados,

caracterizado por el hecho de que,

el sistema se configura, para una sucesión de secuencias (44, 45) de bloques de datos aleatorizados incluidas en una unidad (36) de datos del flujo de datos aleatorizados, para generar al menos un vector de inicialización (IV_N) usado para descodificar una secuencia (45) de bloques de datos aleatorizados en función de al menos un bloque de datos desaleatorizados en una secuencia de bloques de datos desaleatorizados obtenidos por desaleatorización de una secuencia precedente (44) de bloques de datos aleatorizados de la unidad (36).

20. Sistema según la reivindicación 19, configurado para realizar un método según cualquiera de las reivindicaciones 11 a 18.

21. Aparato para enviar y recibir datos, incluyendo un dispositivo (14) dispuesto para aplicar un método según cualquiera de las reivindicaciones 1 a 9 y un método según cualquiera de las reivindicaciones 11 a 18.

22. Programa informático comprendiendo un conjunto de instrucciones capaces, cuando se incorporan en una medio legible por máquina, de dirigir un sistema para procesar información para realizar un método según cualquiera de las reivindicaciones 1 a 9 o cualquiera de las reivindicaciones 11 a 18.

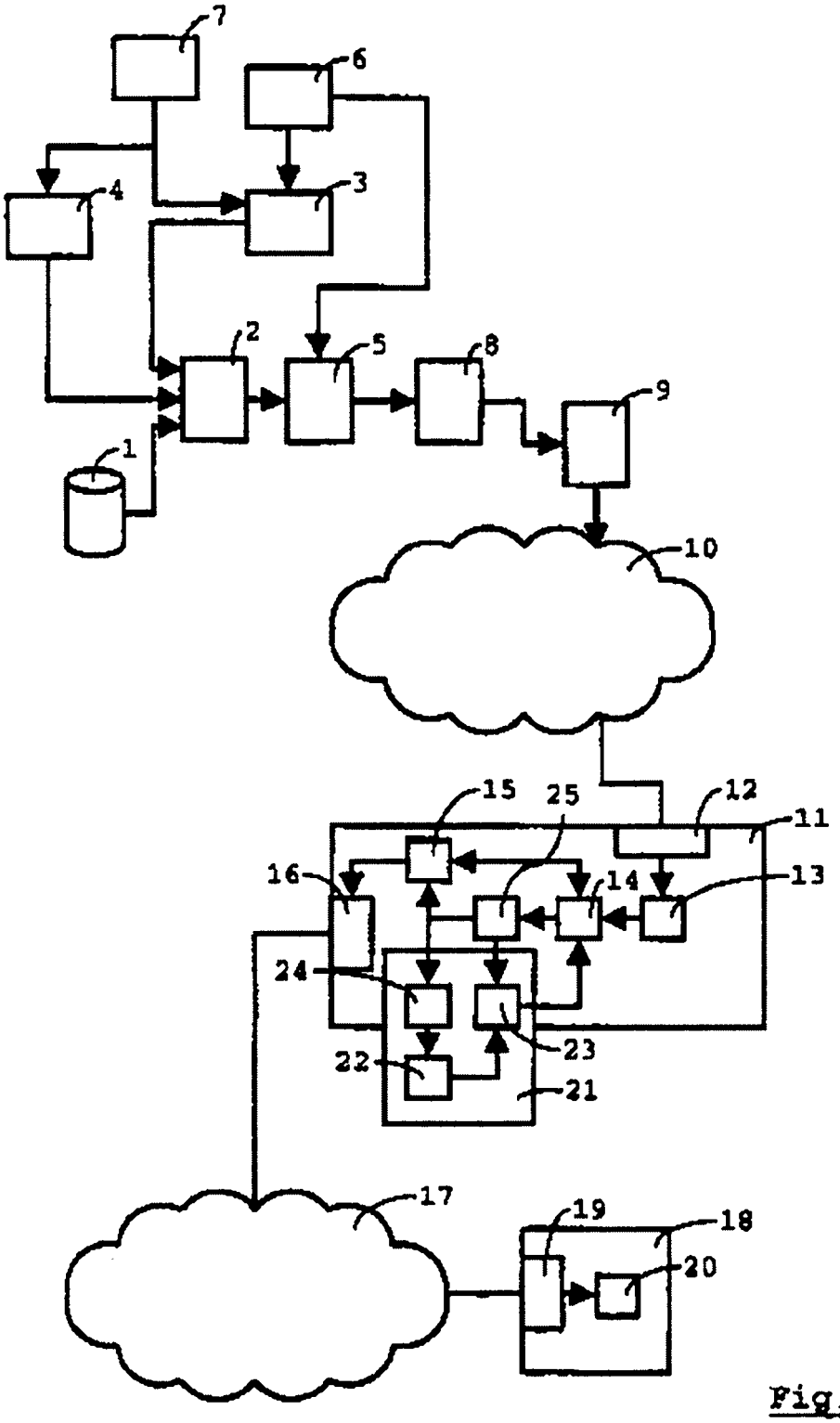

Fig. 1

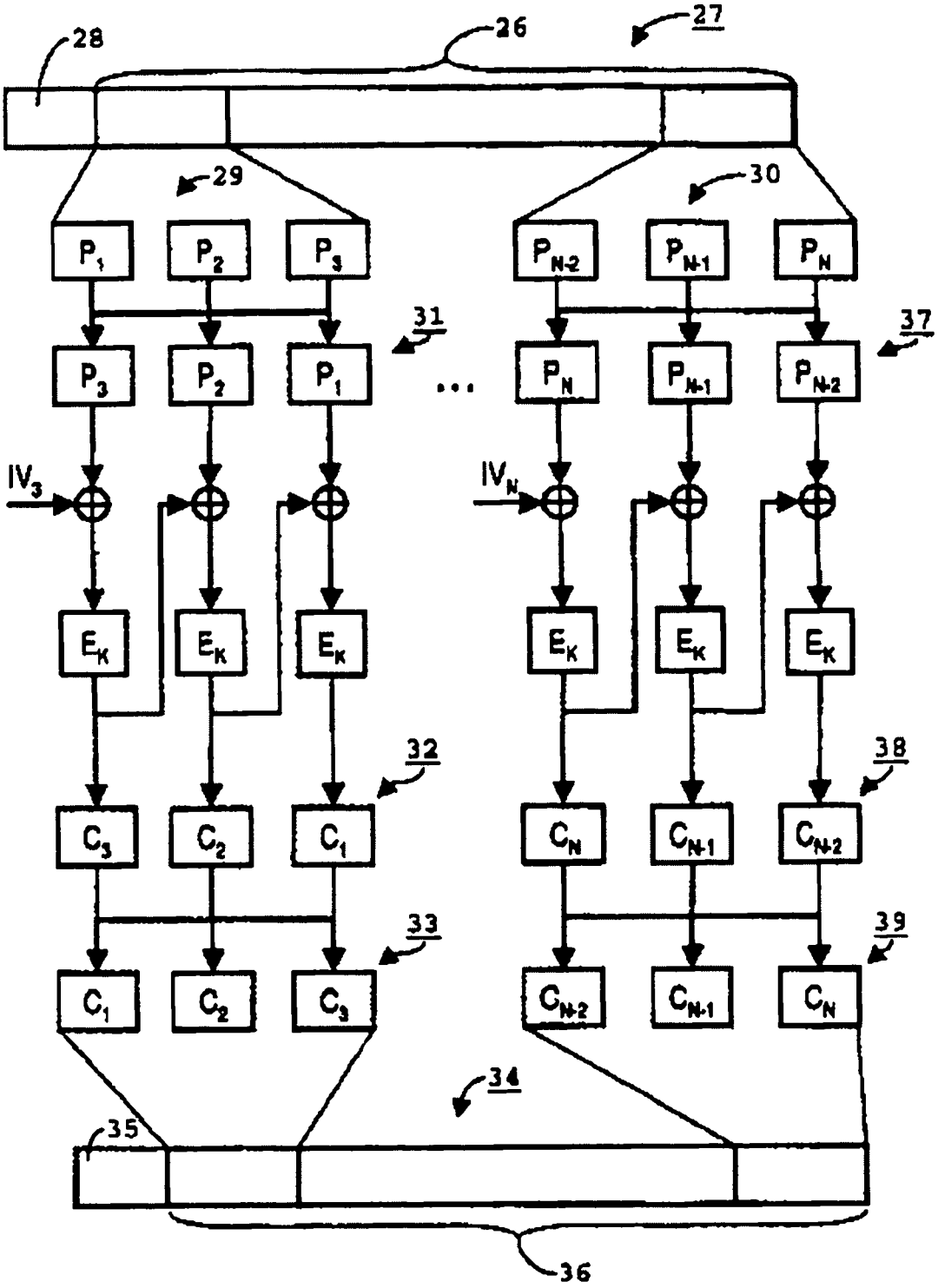
28
26
27
29
30
P1
P2
P3
PN-2
PN-1
PN
31
37
P3
P2
P1
...
PN
PN-1
PN-2
IV3
IVN
EK
EK
EK
EK
EK
EK
32
38
C3
C2
C1
CN
CN-1
CN-2
33
39
C1
C2
C3
CN-2
CN-1
CN
34
35
36

Fig. 2

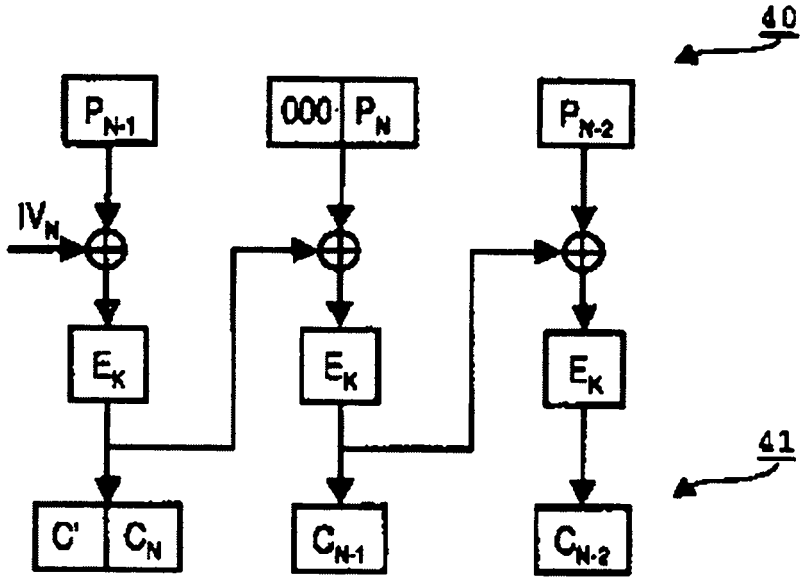
40
P_{N-1}
000 P_N
P_{N-2}
IV_N
E_K
E_K
E_K
41
C' C_N
C_{N-1}
C_{N-2}

Fig. 3

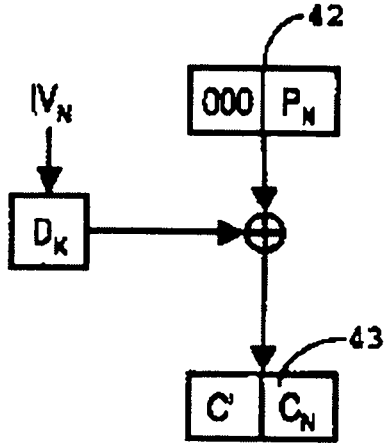
42
000 P_N
IV_N
D_K
43
C' C_N

Fig. 4

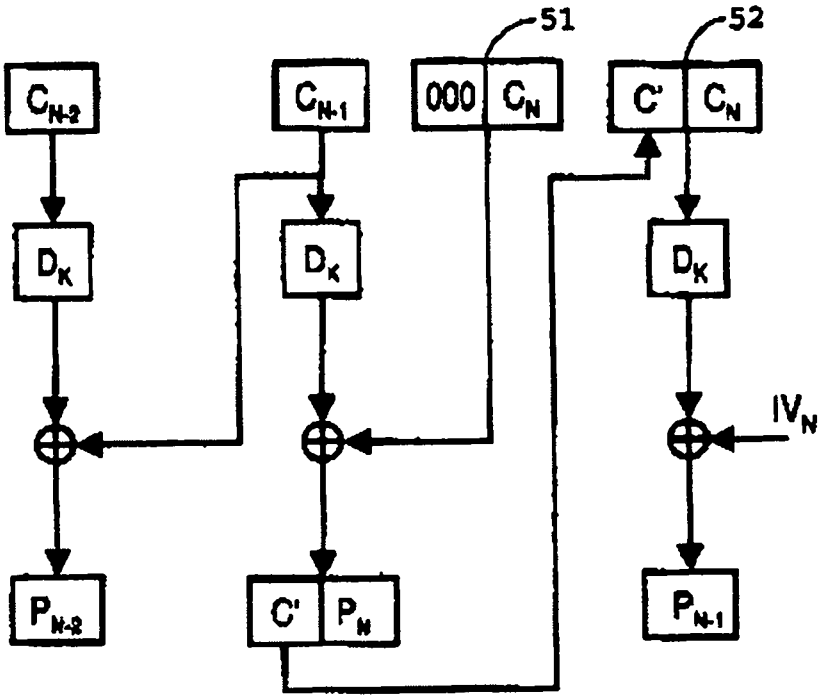
51
52
C_{N-2}
C_{N-1}
000 C_N
C' C_N
D_K
D_K
D_K
IV_N
P_{N-2}
C' P_N
P_{N-1}

Fig. 6

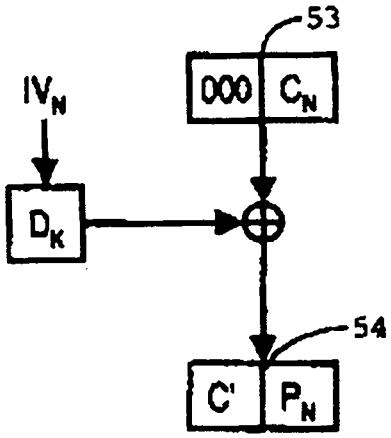
53
IV_N
000 C_N
D_K
54
C' P_N

Fig. 7

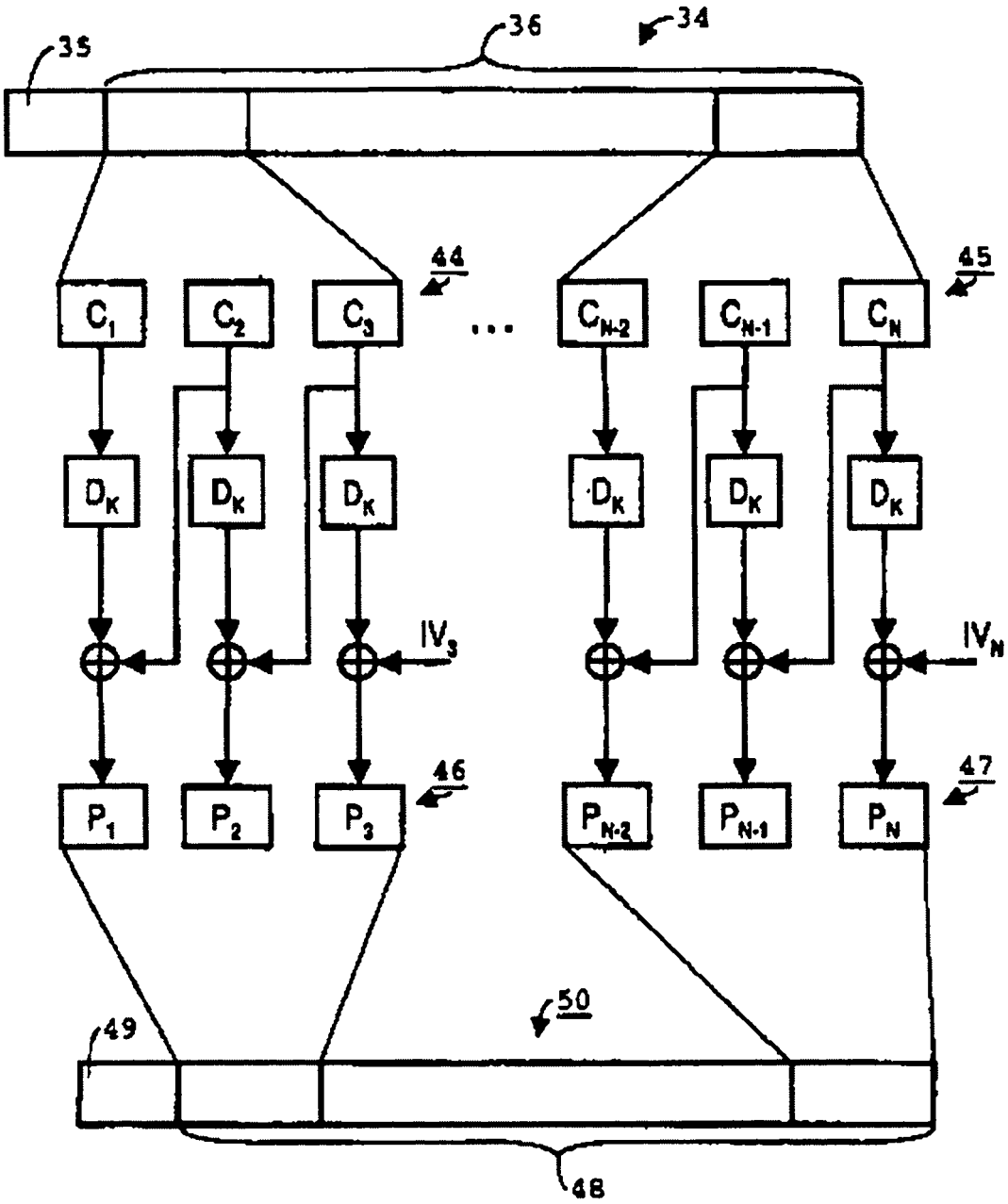
35
36
34
44
45
C1
C2
C3
...
CN-2
CN-1
CN
DK
DK
DK
DK
DK
DK
IV3
IVN
46
47
P1
P2
P3
PN-2
PN-1
PN
50
49
48

Fig. 5