Un Cifrado Feistel Integrando Autómatas Celulares

A. Tomeu[1]

Resumen

De todos es conocida la importancia de los cifrados tipo **Feistel** en aplicaciones comerciales. De hecho, y hasta hace apenas unos meses, el estándar mundial **DES** funcionaba de modo masivo, si bien previamente había sido desestimado por la NSA para cuestiones de alta seguridad. En Noviembre de 2000, **DES** fue finalmente sustituido en EE.UU. (y pronto en el resto del globo) por un nuevo cifrado conocido como **Rindjael**. En este trabajo hemos querido mostrar los resultados obtenidos de la fusión de dos paradigmas: el de cifrados en bloque tipo **Feistel**, y el modelo de cálculo teórico de los autómatas celulares. Los primeros resultados muestran que es posible conseguir métodos de cifrado en bloques aunando ambos modelos, con una seguridad razonable.

1 Autómatas Celulares

Un autómata celular unidimensional se compone de una hilera infinita de casillas, también llamadas células, que pueden tomar valores sobre un anillo conmutativo finito $\mathsf{Z}_k = \{0, 1, ..., k-1\}$. Notaremos a las células mediante la expresión a_i, para indicar a la célula situada en la posición i de la hilera. Como es lógico, el estado del autómata queda especificado por el conjunto de los valores situados en las casillas en un instante dado del tiempo t, y es llamado configuración:

$$..., a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)}, ...$$

El estado de la célula a_i es actualizado en pasos de tiempo discretos de acuerdo con una función de transición determinística, dependiente de las células vecinas de a_i, de acuerdo con la ecuación general siguiente:

$$a_i^{(t+1)} = \phi \overset{3}{\left(a_{i-r}^{(t)}, a_{i-r+1}^{(t)}, ..., a_{i+r}^{(t)} \right)} \qquad (1)$$

La vecindad se considera en general formada por la propia célula, y por las r células vecinas inmediatamente adyacentes a derecha e izquierda[1]. La actualización de los estados de las células de acuerdo con la ecuación 1 es realizada síncronamente para todas las células de la hilera, y la aritmética es computada módulo k. Incluso con valores paramétricos tan simples como $k = 2$ y $r = 1$ el comportamiento global ofrecido por un autómata celular puede llegar a ser extremadamente complejo, si definimos una función de transición ϕ adecuada para ello, y comenzamos a estudiar su evolución desde un estado inicial $..., a_{i-1}^{(0)}, a_i^{(0)}, a_{i+1}^{(0)}, ...$ concreto[2].

Una vez establecidos los parámetros k y r, podemos en principio determinar cualquier función de transición ϕ, que establecerá el mecanismo evolutivo del autómata. Si entonces consideramos el patrón generado a partir de una configuración inicial establecida aleatoriamente podemos observar distintos comportamientos según sea la función ϕ elegida [Wol94]. Por comodidad, podemos identificar el array con el conjunto de los números enteros Z, estableciendo la biyección dada por $f(i) = a_i$.

Una configuración de autómata celular será un vector $C^{(t)}$ de $\mathsf{Z}^k \times \mathsf{Z}^k \times ...$ En el caso de tratar con un autómata celular finito dispondremos de N células $a_0, a_1, ..., a_{N-1}$. El autómata celular va pasando de una configuración $C^{(t)}$ a la siguiente $C^{(t+1)}$ mediante la aplicación de la función de transición sobre cada una de las células $a_i^{(t)} \in C^{(t)}$. El parámetro r en la ecuación determina el ámbito de la regla, ya que el valor de $a_i^{(t+1)}$ vendrá dado por $2r + 1$ células: el suyo propio, y el de sus r células vecinas a derecha e izquierda. Es posible realizar la

[1] Dpto. de Lenguajes y Sistemas Informáticos. Universidad de Cádiz. E-mail: antonio.tomeu@uca.es

[1] Naturalmente, el rango de la vecindad puede ser más amplio y también más complicado. No obstante, suele respetarse la simetría respecto de la célula central a_i, si bien esto no es estrictamente necesario.

[2] Stephen Wolfram define un mecanismo numérico de especificación de las funciones de transición ϕ que asumimos. Veáse [Wol94] para más detalles.

extensión de la función de transición ϕ a una configuración mediante la función de transición global $\phi_g : C \rightarrow C$ siendo C el conjunto de todas las configuraciones posibles, y dada por

$$C^{(t+1)} = \phi_g \left(C^{(t)} \right) = \phi \left(a_i^{(t)} \right)_{i \in Z} \quad (2)$$

Si $C^{(i)}$ y $C^{(j)}$ son dos configuraciones distintas de un autómata celular, se dice que $C^{(j)}$ es sucesora de $C^{(i)}$ si y solo si, para cada $a_q^{(j)} \in C^{(j)}$ existen r y $a_{q-r}^{(i)}, ..., a_{q+r}^{(i)}$ tales que $\phi(a_{q-r}^{(i)}, ..., a_{q+r}^{(i)}) = a_q^{(j)}$. Si $C^{(j)}$ es sucesora de $C^{(i)}$, lo notamos por $C^{(i)} \circ C^{(j)}$. Si $C^{(i)}$ y $C^{(j)}$ son dos configuraciones distintas de un autómata celular, se dice que $C^{(j)}$ deriva de $C^{(i)}$ si y solo si, existen configuraciones distintas $C^{(1)}, ..., C^{(n)}$ tales que $C^{(1)} \circ C^{(2)} \circ \cdots C^{(n-1)} \circ C^{(n)}$ y además $C^{(i)} = C^{(1)}$ y $C^{(j)} = C^{(n)}$. Si $C^{(j)}$ deriva de $C^{(i)}$, lo notamos por $C^{(i)} \overset{*}{\circ} C^{(j)}$.

2 Diseño del Cifrado

2.1 Redes de Feistel

La mayoría de los cifrados de bloques tienen un funcionamiento muy similar: dividen los bloques de datos a ser cifrados en dos mitades L y R, y realizan un cifrado iterativo en el que la salida de cada paso se usa como entrada para el próximo. Las siguientes ecuaciones definen un tipo de cifrado en bloque conocido como red de **Feistel** estándar.

Una red de **Feistel** estándar [Men96] está dada por

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i) \end{aligned} \quad (3)$$

siendo $\oplus$ la operación de suma módulo 2 o XOR bit a bit, muy utilizada en este tipo de cifrados, y K_i las claves internas del sistema, calculadas a partir de la clave inicial K. Este tipo de estructura cuenta con la interesante propiedad de ser reversible y comportarse involutivamente, independientemente de cómo sea la función f descrita en las ecuaciones generales que definen a la red. Para descifrar, basta con aplicar de nuevo el mismo algoritmo al texto cifrado invirtiendo el orden de las K_i.

El cifrado de bloque que vamos a presentar poseerá estructura tipo **Feistel**, pero con la particularidad de que construiremos la función f involucrada en base a autómatas celulares. Llamaremos al cifrado **CIBAC-1** (**Cifrado de Información en Bloque con Autómatas Celulares 1**). Para ello, reformulamos la ecuación 3 para obtener la que define la función principal de cifrado de **CIBAC-1**:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_i \oplus AC(R_{i-1}, K_i) \end{aligned} \quad (4)$$

donde

$$AC(R_{i-1}, K_i) = R_{i-1}(j) \oplus a_l^{(j)} \quad (5)$$

para $j = 1, 2, ...m$ siendo m la longitud de sub-bloque, siendo $a_l^{(j)}$ la evolución temporal de la $l-$ésima celula de un autómata celular concreto, cuya configuración inicial es $C^{(0)} = K_i$ y siendo n el número de iteraciones del cifrado.

Sin embargo, para que la ecuación anterior tenga cabida en una cifra tipo **Feistel**, es necesario que la función principal de cifrado que describe se comporte como una involución. Veamos que es así.

Proposición 1 **La función de cifrado AC de CIBAC-1 es una involución actuando en una red de Feistel de n etapas.**

Demostración: Mediante inducción en el número de etapas, consideremos $n = 1$. En este caso tenemos, de acuerdo a 4 que $L_1 = L_0 \oplus AC(R_0, K_1)$ y que $R_1 = R_0$ donde el par (L_0, R_0) conforman el texto llano de entrada a la red. Aplicando las mismas ecuaciones (esto es, aplicando la involución) sobre el par (L_1, R_1), vemos que $L_1 = L_0 \oplus AC(R_0, K_1) \oplus AC(R_1, K_1)$ y como $R_1 = R_0$ se termina obteniendo $L_1 = L_0 \oplus AC(R_0, K_1) \oplus AC(R_0, K_1) = L_0$. Para el sub-bloque R_0 el resultado es directo. Sea ahora una hipótesis de inducción según la cual una cifra Feistel de n etapas con función de cifrado descrita por la ecuació 4 y es una involución. La extensión de la prueba a $n+1$ etapas con las premisas establecidas es directa. ■

La definición dada en 4 y en 5, si bien establece la función principal de cifrado de **CIBAC-1**, no pone en claro algunas cuestiones fundamentales de cara al diseño del cifrado en la práctica como pueden ser:

- Longitud del bloque y del sub-bloque.
- Longitud de la clave.

- Número de rondas del cifrado.
- Elección del autómata celular cifrante.

A continuación resolveremos todas estas cuestiones.

2.2 Longitud de Bloque

Un análisis de la realidad actual en lo que al cifrado de bloque se refiere, nos ha mostrado que de los ocho algoritmos de cifrado en bloque considerados más seguros en la práctica y cuyas características se muestran en la tabla 6, todos ellos excepto uno utilizan una longitud de bloque igual a **64** bits. Por ello, y en base a la experiencia acumulada por sus constructores, fijaremos el parámetro m de la ecuación 4 en un valor de **32** bits, conformando por tanto ambos bloques el total de **64** bits. Esta elección no es gratuita, ya que garantiza un compromiso entre la seguridad del cifrado y la rapidez con que el mismo puede realizarse, tanto mediante implantación **software** como, fundamentalmente, **hardware**. Como vemos, **LUFICER** trabaja con bloques de **128** bits, pero fue diseñado para ser implementado específicamente mediante **hardware** de alta velocidad.

Cifrado	(1)	(2)	(3)	(4)
LUCIFER	128	128	16	$\surd$
DES	64	56 + 8	16	$\surd$
LOKI	64	64	16	$\surd$
RC2	64	*variable*	–	–
CAST	64	64	8	$\surd$
BLOWFISH	64	*variable*	16	$\surd$
IDEA	64	128	8	–
SKIPJACK	64	80	32	*

(6)

La leyenda de la tabla es la siguiente: (1) indica el tamaño de bloque en bits; (2) indica el tamaño de clave en bits; (3) indica el número de iteraciones y (4) indica si el cifrado es de tipo Feistel ($\surd$) o no. En el caso marcado con $*$ la información no fue publicada.

2.3 Longitud de Clave

En este caso la evolución histórica de los sistemas de cifrado en bloque ha dado a los diseñadores de criptosistemas una lección clara: longitudes de clave cortas llevan a poner en riesgo la seguridad del criptosistema, ya que definen espacios de claves lo suficientemente pequeños como para poder ser atacados por métodos computacionales de búsqueda exhaustiva o parcialmente exhaustiva guiada por heurísticas. Este fué el problema de **DES** en su día, para una longitud de clave de **56** bits efectivos[3] (más **8** de paridad).

Sobre el particular, en [Bla96] se propone una interesantísima discusión sobre el problema de la longitud de clave, y los autores concluyen que de acuerdo con la curva esperada de incremento de potencia computacional para ataques basados en fuerza bruta, claves de al menos **90** bits garantizan cifrados simétricos en bloque seguros durante los próximos **20** años. El cifrado de bloque actualmente más seguro, **IDEA**, utiliza claves de **128** bits, y lo hace invulnerable a cualquier ataque de fuerza bruta o diferencial. Nosotros nos vemos obligados a determinar el parámetro de lontigud de clave en función de dos condicionantes:

a) Una longitud suficientemente larga, entendiendo por tal a aquella igual o mayor a **128** bits.

b) El hecho de que la clave actuará como origen de las claves intermedias que darán lugar a la configuración inicial $C^{(0)}$ de uno o varios autómatas celulares cifrantes, y que debe tender a ser larga, en términos de garantizarnos aleatoriedad, entropías altas, y períodos de longitud razonable.

El siguiente resultado trata de establecer el necesario compromiso.

Proposición 2 Una longitud de clave de 200 bits posibilita un cifrado seguro.

Demostración: La estimación actual en cuanto a qué tamaño debe tener el espacio de claves para lograr inmunidad frente a ataques computacionales sitúa a éste en números de un orden de magnitud de 2^{128} = **340 282 366 920 938 463 463 374 607 431**

[3] De hecho, hace años que **DES** fue desestimado como estándar de seguridad gubernamental por la NSA norteamericana, al demostrarse que con los recursos informáticos disponibles por las grandes agencias gubernamentales era inseguro. Por ello, a finales del año 2000, fue finalmente desestimado en EE.UU. a todos los niveles y sustituido por una nuevo algoritmo de cifrado en bloque, conocido como **Rindjael (AES)**.

768 211 456. Vemos que nuestra elección de longitud de clave supera holgadamente la cifra; $2^{200} >> 2^{128}$. ■

2.4 Rondas del cifrado

Nuevamente acudiremos a la experiencia previa dada por los cifrados de bloque ya aceptados por la comunidad de criptógrafos. En este caso, todos los cifrados tipo **Feistel** utilizan **16** rondas. **IDEA** utiliza sólo **8**, pero no está considerado un tipo **Feistel**. Es justo indicar que este parámetro fue fijado en las décadas de los setenta y ochenta, cuando la potencia de cálculo disponible era mucho menor que hoy día. Nosotros optaremos por una longitud de **16** rondas, para lograr el grado de seguridad requerido y al mismo tiempo garantizar tiempos de cifrado razonables en máquinas de propósito general. No obstante, en el momento de la implementación práctica del cifrado, parece razonable dejar al usuario la posibilidad de seleccionar el número de rondas de cifrado de entre un rango disponible (a partir de **16** iteraciones), con objeto de que pueda adaptar el grado de seguridad a sus necesidades. El número de rondas del cifrado será por tanto de **16** iteraciones.

2.5 Autómata Cifrante

En este caso la elección es clara, y utilizaremos un mismo autómata celular cifrante en cada ronda del algoritmo, lo cual garantiza su seguridad y proporciona una mejor difusión de la información original. Por tanto, será necesario disponer de múltiples autómatas celulares binarios cifrantes, para escoger uno de ellos, de acuerdo a las siguientes especificaciones:

- serán binarios, esto es con $k = 2$ y con **200** células;
- su función de transición será no lineal;
- tendrán un alto grado de aleatoriedad; y
- tendrán períodos altos.

Serán utilizados los principios de selección definidos en [Tom99] para autómatas que actúan como buenos generadores aleatorios. No obstante, puesto que aquí deseamos una difusión máxima de la información, utilizaremos $r = 99$. En esta discusión, nos limitamos a indicar sus características de definición en términos de código numérico de la función de transición, y valores medios de los parámetros de complejidad relevantes tales como distancia media de Hamming interconfiguración ($\overline{H}$), entropía de configuración media ($\overline{S}$) y entropía temporal media para una célula ($\overline{S_t}$). Los datos relativos al autómata celular en cuestión, junto con muchos otros, fueron computados vía el **Analizador-CA**, **software** de propósito específico escrito **ad hoc** por nosotros, y se aprecian en la tabla 7.

Función	$\overline{H}$	$\overline{S}$	$\overline{S_t}$
683	100.0440	0.9991	0.9996

(7)

Naturalmente, todos ellos han sido elegidos en términos de los datos recogidos en la tabla anterior, y de una característica adicional de importancia capital en la seguridad del cifrado: la no linealidad de la función de cifrado f de la ecuación 4. Esto es de suma importancia, ya que como la experiencia ha mostrado, la seguridad de los cifrados de bloque radica precisamente en esa no linealidad. El ejemplo clásico de función no lineal a este nivel de diseño lo encontramos en las S-cajas de DES.

3 La Cifra CIBAC-1

Proposición 3 Con las especificaciones de diseño anteriores, existe un cifrado de bloques con autómatas celulares binarios tipo Feistel.

Demostración: Más adelante damos una descripción en seudocódigo del mismo (Ver Algoritmo CIBAC-1). Como vemos, el texto llano es sometido a una permutación inicial (no descrita en este trabajo, aunque ya ha sido establecida) y dividido en los subloques L_0 y R_0. En la siguiente etapa, se obtiene L_1 a partir de R_0 mientras que R_1 es obtenido a partir de la función de cifrado $R_1 = L_0 \oplus AC(R_0, K_1)$ donde $AC(R_0, K_1)$ es la función de cifrado cuyos detalles se dan en la ecuación 4. Esto continúa así durante las **15** iteraciones siguientes, aplicándose el último ciclo tal y como se detalla. En cada etapa se utiliza una subclave particular K_i, deducida a partir de la clave inicial de acuerdo al Algoritmo de Cálculo de Subclaves que presentaremos en su momento. ■

Proposición 4 El cifrado CIBAC-1 descrito por las ecuación 4 es de tipo Feistel y simétrico.

Demostración: Tal y como se aprecia en el algoritmo, vemos que la salida del mismo está formada por los subloques (L_{16}, R_{16}) que conforman el criptograma. Si ahora damos como entrada al mismo algoritmo este criptograma, inviertiendo el orden de utilización de las claves, vemos que el primer paso es hacer $PI(L_{16}, R_{16})$ lo cual nos proporciona que $(L_0, R_0) = (R_{16}, L_{16})$. La primera etapa del algoritmo nos dice ahora que $L_1 = L_{16} = R_{15}$ y que $R_1 = L_0 \oplus AC(R_0, K_{16}) = R_{16} \oplus AC(L_{16}, K_{16}) = R_{16} \oplus AC(R_{15}, K_{16})$. Sustituyendo R_{16} por su valor, tenemos que $R_1 = L_{15} \oplus AC(R_{15}, K_{16}) \oplus AC(R_{15}, K_{16})$. Los dos últimos términos de la parte derecha de esta igualdad se cancelan y tenemos que $R_1 = L_{15}$. Por lo tanto, la función concreta de autómata celular escogido puede ser cualquiera. Continuando el procedimiento concluimos que el descifrado concluye con $R_{16} = R_1$ y $L_{16} = L_1$. Finalmente, aplicando $IF^{-1}(R_1, L_1)$ recuperamos el texto llano original (L_1, R_1). ■

A continuación se especifican ciertos detalles del la cifra CIBAC-1, en relación con aspectos como la función de cifrado, el algoritmo de cálculo de subclaves, etc.

3.1 La Función de Cifrado

El algoritmo que implementa la función de cifrado descrita en la ecuación 4

Algoritmo de la Función de Cifrado $AC(R_{i-1}, K_i)$
INPUT: Sub-bloque de bits $R_{i-1} = r_1 r_2 \cdots r_{32}$ y clave $K_i = k_0 k_1 \cdots k_{199}$.
OUTPUT: Sub-bloque de bits $L_i = l_1 l_2 \cdots l_{32}$.
1. Inicialización: $C^{(0)} \leftarrow K_i$, $j \leftarrow 0$.
2. While $j < 32$ do
 2.1 $L_i(j) = R_{i-1}(j) \oplus C^{(j)}(100)$.
 2.2 $j \leftarrow j + 1$

Como podemos ver, cada etapa de confusión-difusión es realizada utilizando el cifrado estándar orientado al flujo propuesto en [Tom99], y realizando una suma módulo 2 ($\oplus$) entre el sub-bloque de 32 bits R_{i-1} y la evolución de la célula central (número 100) del autómata celular no aditivo escogido como función de cifrado, cuya configuración inicial $C^{(0)}$ se obtiene a partir de la clave K_i. Vemos que la subclave $K_i = k_{i0} k_{i1} \ldots k_{i199}$ de la etapa, de 200 bits, establece la configuración inicial del autómata celular cifrante, ya que $C^{(0)}(j) = a_j = k_{ij}$ para $j = 0, 1, \ldots 199$. Posteriormente este autómata evoluciona durante 32 generaciones, y el valor de la célula central a_{100} es utilizado para cifrar los bits del sub-bloque R_{i-1} y obtener los del sub-bloque de salida L_i.

3.2 Computación de Subclaves

Nos resta por indicar el método de cálculo de claves intermedias (subclaves) a partir de la clave inicial K. Tal método utiliza la técnica estándar de las permutaciones sucesivas a partir de la clave inicial, pero introduciendo una dependencia de los datos que hace más seguro el sistema. Además, como operación no lineal añadida al cálculo de subclaves, incluiremos varias rotaciones, dependientes tanto del texto llano como de la propia clave K. Los detalles se muestran en el seudocódigo del Algoritmo CIBAC-1 de Cómputo de Subclaves. Como vemos, el algoritmo procede haciendo la suma módulo 2 ($\oplus$) del texto llano y la clave de entrada[4] para lograr subclaves dato-dependientes. Para lograr esto, y dado que la clave tiene una longitud de 200 bits y el mensaje de 64, es necesario realizar un expansión del mismo para conformar el total de 200 $(m_0 m_1 \ldots m_{199})$ bits de mensaje que se aprecian en el algoritmo. Tal expansión se realiza en una tabla de expansión diseñada a tal efecto que no mostramos aquí .

Posteriormente, se suma un vector A fijo, con una buena distribución estadística de unos y ceros, para evitar que la suma $K \oplus M$ sea nula. Se aplica una operación no lineal en forma de rotación, y se expande el resultado a 400 bits, de acuerdo a tablas de expansión diseñadas específicamente para ello. Posteriormente, 200 de estos bits conforman la i-ésima subclave K_i, y el resto son utilizados en una nueva iteración para calcular la subclave K_{i+1}. La ventaja de este método es que conocida una clave K_i es muy difícil conocer la clave previa K_{i-1} dificultándose seriamente el criptoanálisis.

[4] De esto modo, las posteriores operaciones y las dieciséis sublaves obtenidas como subproducto dependende tanto de los datos del texto llano como de la clave, dificultando el criptoanálisis.

Algoritmo CIBAC-1 de Cómputo de Subclaves
INPUT: Una clave $K = k_0k_1 \cdots k_{199}$ de **200** bits y texto llano $M = m_0m_1 \ldots m_{199}$.
OUTPUT: Dieciséis subclaves intermedias de **200** bits por dadas$K_1, K_2, \ldots, K_{16}$.
1. Definir $v_i(i = 1, 2, \ldots, 16)$ como sigue: $v_1 = 1$ para $i \in \{1, 2, 9, 16\}$; $v_2 = 2$ en otro caso.
2. $p \leftarrow K \oplus M$.
2.1. For $i = 1$ to **16** do
2.1.1. $p \leftarrow p \oplus A$.
2.1.2 If $i \in \{1, 2, 9, 16\}$ then p © v_1 else p © v_2.
2.1.3 Expandir p a **400** bits..
2.1.4 $K_i \leftarrow p_1p_3 \ldots p_{399}$.
2.1.5.$p \leftarrow p_2p_4 \ldots p_{400}$.

En este algoritmo, la operación p © v_i implica rotar a la izquierda los bits que conforman a p el número de posiciones que indique el subíndice i.

3.3 Implementación de CIBAC-1

Procedemos a continuación a mostrar los aspectos fundamentales referidos a la implementación de la misma, mostrando para ello a continuación el seudocódigo de la cifra **CIBAC-1**. Como podemos ver, el algoritmo teórico se ha diseñado para un total de **16** iteraciones, aunque en la práctica, se permitirán algunas más. El algoritmo procede tomando por entrada un texto llano de **64** bits dado por $m_1 ... m_{64}$ y una clave inicial de **200** bits dada por $k_0 ... k_{199}$, para producir un texto cifrado de **64** bits dado por $c_1 ... c_{64}$. Tras computar las claves intermedias mediante el algoritmo correspondiente, aplica la permutación inicial PI sobre el texto llano y lo divide en los subloques (L_0, R_0). Comienza ahora la ronda de iteraciones donde se aplican con carácter general las ecuaciones 4 a través del **Algoritmo de la Función de Cifrado.**

Algoritmo CIBAC-1

INPUT: Texto llano $m_1 ... m_{64}$; clave $K = k_0 ... k_1$ de **200** bits.
OUTPUT: Texto cifrado (criptograma) $C = c_1 ... c_{64}$
1. (claves intermedias) Computar $\{K_i\}_{i=1}^{16}$ desde K vía Alg. Cálculo de Subclaves.
2.$(L_0, R_0) \leftarrow PI(m_1 ... m_{64})$. (Usar la PI).
3. For $i = 1$ to **15** do
3.1. $L_i \leftarrow R_{i-1}$.
3.2 $C^{(0)} \leftarrow K_i$. (Conf. inicial aut. celular cifrante)
3.3. For $j = 1$ to **32**
3.3.1 $R_i(j) \leftarrow L_{i-1}(j) \oplus (AC(R_{i-1}, K_i))\,(j)$
4. $L_{16} \leftarrow R_{15}$.
5. $R_{16} \leftarrow L_{15} \oplus AC(R_{15}, K_{16})$.
6. $b_1 ... b_{64} \leftarrow (R_{16}, L_{16})$.
7. $c_1 ... c_{64} \leftarrow PI^{-1}(b_1 ... b_{64})$. (Usar la PI^{-1}).

4 Criptoanálisis

El cifrado obtenido es simétrico y presenta un espacio de claves (2^{200}) razonablemente amplio frente a ataques de fuerza bruta.

4.1 Claves Débiles

Como es habitual en esta clase de cifrados, el sistema presenta claves débiles. En concreto, aquellas que actúen como puntos fijos del autómata celular inserto en la función de cifrado, o que induzcan ciclos de muy corta longitud en él. Por tanto, la elección del autómata celular concreto que define la función $AC(R_{i-1}, K_i)$ debe ser analizada cuidadosamente en términos de la metodología descrita en [Tom99]. A partir de este análisis, las cláves débiles pueden ser derivadas en cada caso concreto.

4.2 Criptoanálisis Estadístico

Su realización es crucial para garantizar la seguridad del cifrado [Kel99], puesto que cualquier regularidad estadística en el criptograma puede ofertar al atacante una base de trabajo. El enfoque que le hemos dado a esta fase del criptoanálisis ha seguido múltiples líneas de estudio, que detallamos a continuación:

a) Test de Aleatoriedad; utilizando como autómata celular cifrante el indicado en la tabla 7

se han cifrado diversos textos llanos con una longitud de **1000** bits. Sobre la secuencia de bits del criptograma se aplicaron los seis test de aleatoriedad clásicos, a fin de comprobar que el algoritmo de cifrado no introducía regularidades estadísticas indeseables, obteniéndose la superación de todos los test para el **95%** de las muestras.

b) Distribución Estadística de la Salida para Entradas Monótonas; una entrada es monónotona cuando se compone únicamente de ceros o de unos. Un buen algoritmo de cifrado de bloques es capaz, sobre esta clase de entradas, de producir un bloque de salida con un grado de confusión-difusión de la información tal que supere los test de aleatoriedad. Se comprobó que la cifra **CIBAC-1** satisfacía este criterio.

c) Correlación Cruzada entre Texto Llano y Texto Cifrado: se encuentra siempre por debajo del **1%**, siendo por tanto adecuada.

5 Mejoras de Seguridad y Conclusiones

Adicionalmente al criptoanálisis ya comentada, es nuestra intención realizar el criptoanálisis diferencial. Como mejoras de seguridad adicionales, proponemos también la integración de múltiples autómatas celulares en una misma red de **Feistel**. En este sentido, hemos probado un resultado que muestra que para una red de n etapas, se pueden utilizar hasta $n/2$ autómatas diferentes, siempre que n sea un número par. Otras vías de trabajo actualmente objeto de análisis son la introducción de S-cajas en el modelo que hemos presentado aquí, o la construcción de redes de **Feistel** no balanceadas (con tamaños de los sub-bloques variables) que integren autómatas celulares. El objetivo final es lograr cifrados de bloque que, integrando autómatas celulares, sean razonablemente seguros, y que ofrezcan buenos tiempos de cifrado.

Referencias

[Bla96] Blaze, M; Diffie, W.; Rivest, R.; Schneier, B.; Shimomura, T.; Thompson, E. & Weiner, M. **Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security**. En www.counterpane.com/keylenght.html.

[Kel99] Kelsey, J; Schneier, B; Wagner, D. & Hall, C. **Cryptoanalytic Attack On Pseudoramdom Number Generators**. En www.counterpane.com/pseudoramdom.html

[Men96] Menezes, A; Van Oorschort, P & Vanstone, S. **Handbook of Applied Cryptography**. (Disponible gratuitamente por capítulos en www.cacr.math.uwaterloo.ca/hac)

[Tom99] Tomeu, A. & Silva, E. **Pseudo-ramdom Sequences with Cellular Automata**. Proceedings of International Conference on Modeling and Simulation, 1999.

[Wol94] Wolfram, S. **Celullar Automata as Models of Complexity**. Nature, 311. (419-424), 1984.