BSD: Algoritmos eficientes para la descomposición binomial de secuencias binarias

Jose Luis Martín Navarro
Instituto de Tecnologías Físicas
y de la Información (CSIC)
Serrano 144, 28006 Madrid, España
Email jomarna6@inf.upv.es

Amparo Fúster Sabater
Instituto de Tecnologías Físicas
y de la Información (CSIC)
Serrano 144, 28006 Madrid, España
Email amparo@iec.csic.es

***Resumen*—La revolución del Internet de las cosas deja paso a un amplio rango de servicios críticos que se fundamentan en dispositivos IoT. Estos dispositivos de bajo coste suelen carecer de la seguridad suficiente, convirtiéndose en la puerta de entrada para ataques sobre la totalidad del servicio. Los protocolos de seguridad de IoT a menudo se basan en cifradores en flujo, donde los generadores de números aleatorios (PRNG) son una parte esencial de estos sistemas. En este trabajo se analiza una técnica de descomposición de secuencias binarias (descomposición binomial) que permite analizar sus características y propiedades. Basados en dicha descomposición, se presentan dos algoritmos que evaluan la fortaleza de dichas secuencias, contribuyendo así al dominio de seguridad IoT.**

***Index Terms*—PRNG, secuencias binomiales, complejidad, IoT**

I. Introducción

La irrupción de Internet de las cosas en el campo de la electrónica y las comunicaciones hacen de IoT una realidad con un número creciente de sensores y dispositivos en nuestro entorno personal, profesional y social. A día de hoy, numerosos servicios críticos como tarjetas inteligentes, comunicaciones móviles, comercio electrónico, servicios sanitarios y automoción dependen de infraestructuras IoT. Al mismo tiempo, igual que crecen los servicios también crecen los riesgos a la hora de aplicarlos [1], [2].

Los dispositivos IoT de bajo coste, normalmente caracterizados por sus limitaciones en cuanto a potencia computacional, memoria, tamaño o consumo energético, también exhiben una mínima o no-existente seguridad. Combinando esta falta de seguridad con su dependencia a la red, estos dispositivos constituyen el blanco perfecto para un ataque sobre todo el sistema [3]. Esta es la razón por la que se trabaja activamente ya sea en el campo de investigación general [4], de investigación aplicada a 5G [5] o a nivel de convocatorias como la del NIST para el diseño de primitivas criptográficas [6]. La criptografía ligera en general y los PRNG en particular constituyen la piedra angular de IoT y deben velar por la seguridad tanto de la red como de los diferentes dispositivos insertados en ella.

En este trabajo analizamos un tipo de descomposición (descomposición binomial) de secuencias binarias para estudiar las características y propiedades de ciertos tipos de PRNG con aplicación criptográfica. A partir de la descomposición binomial y de las propiedades de las secuencias binomiales, proponemos dos algoritmos distintos, el algoritmo b-BSD y el algoritmo f-BSD, que nos permiten analizar la complejidad lineal de una clase amplia de secuencias criptográficas. En líneas generales el parámetro complejidad lineal es una métrica ampliamente extendida [7] para evaluar la robustez de una secuencia binaria con vistas a su utilización en criptografía ligera. Los algoritmos aquí desarrollados obtienen mejores prestaciones que las del algoritmo de Berlekamp-Massey [8], utilizado habitualmente para el cálculo de este parámetro. Se comparan la cantidad de secuencia requerida y la complejidad computacional de los algoritmos propuestos con relación al mencionado algoritmo de Berlekamp-Massey. La aplicación de estos algoritmos a una familia particular de secuencias (*generalized sequences*) definidas en [9] y patentadas en [10] mejora las prestaciones de dichos algoritmos. El estudio de la fortaleza de las secuencias criptográficas mediante un análisis de su complejidad lineal previene posibles debilidades en los dispositivos de criptografía ligera y en los servicios construidos a partir de ella.

II. Complejidad lineal y Secuencias binomiales

En términos generales la complejidad lineal de una secuencia da una medida de la cantidad de secuencia que es necesaria para poder reconstruir el resto de la misma. En términos criptográficos, la complejidad lineal, notada LC, debe ser tan grande como sea posible, es decir lo más próxima a su longitud l (período). El algoritmo de Berlekamp-Massey [8] es el procedimiento general para calcular la complejidad lineal de cualquier secuencia, independientemente del valor de l. Este algoritmo necesita procesar un número de bits igual a dos veces el valor de la complejidad lineal de la secuencia analizada. Para secuencias cuya LC esté próxima a su longitud l, por ejemplo las secuencias generalized, este algoritmo necesitaría conocer y procesar aproximadamente $2{\cdot}l$ bits de la secuencia con una complejidad computacional de $O(l^2)$ [11]. Los nuevos algoritmos que se desarrollan en este trabajo se basan en las secuencias binomiales que se definen a continuación.

El número combinatorio $\binom{n}{k}$ es el coeficiente de la potencia x^k en el desarrollo del polinomio $(1+x)^n$, donde k es un entero fijo y $n \geq 0$. Sabemos que $\binom{n}{0} = 1$ y $\binom{n}{k} = 0$ para todo $n < k$.

Se define una secuencia binomial $\{\binom{n}{k}\}_{n\geq 0}$ como una secuencia binaria cuyos términos son números combinatorios $\binom{n}{k}$ reducidos módulo 2, es decir

$$\left\{\binom{n}{k}\right\}_{n\geq 0} = \left(\binom{0}{k}, \binom{1}{k}, \binom{2}{k}, \dots\right)_{mod\ 2}, \quad (1)$$

donde k es el *índice* de la secuencia binomial. Los k primeros términos de la secuencia serán ceros y el término $\binom{k}{k}$ corresponde al primer 1. En este trabajo y con el fin de simplificar la notación, denotaremos la secuencia binomial $\{\binom{n}{k}\}_{n\geq 0}$ simplemente por $\binom{n}{k}$.

Asimismo, denotamos por $\binom{n}{k}_{i,j}$ la subsecuencia de $\binom{n}{k}$ cuyos bits están comprendidos entre los índices i y $j-1$ (la numeración de términos empieza en cero), mientras que $\binom{n}{k}_j$ corresponde a la subsecuencia con índice $i=0$.

En la *Tabla I* se muestran las 8 primeras secuencias binomiales $\binom{n}{k}$ $(0 \leq k < 8)$ con sus correspondientes longitudes l_k y complejidades lineales LC_k.

A continuación, se enumeran diferentes propiedades de las secuencias binomiales:

1. Dada una secuencia $\binom{n}{k}$ con $k = 2^m + i$ donde m es un número entero y el índice i toma valores en el intervalo $0 \leq i < 2^m$, entonces la ley de formación de la secuencia binomial $\binom{n}{2^m+i}$ es la siguiente [12]:

$$\binom{n}{2^m+i} = \begin{cases} 0 & \text{si } 0 \leq n < 2^m + i, \\ \binom{n}{i}_{mod\,2} & \text{si } 2^m + i \leq n < 2^{m+1}, \end{cases}$$

   Es decir una secuencia binomial se construye recursivamente a partir de otras binomiales de índice inferior.
2. La secuencia binomial $\binom{n}{2^m+i}$ tiene longitud $l = 2^{m+1}$ [12, Proposición 1.b].
3. La complejidad lineal de la secuencia binomial $\binom{n}{2^m+i}$ es $LC = (2^m + i) + 1$ [12, Teorema 3].

La importancia de las secuencias binomiales radica en que toda secuencia binaria cuya longitud l sea una potencia de 2 puede escribirse como una combinación lineal de un número finito de secuencias binomiales, véase [12]. Esta combinación lineal es la *descomposición binomial* de dicha secuencia. La descomposición binomial nos permite analizar propiedades fundamentales de la secuencia:

a) Dada una secuencia $\{a_n\}$ con descomposición binomial $\sum_{i=1}^{t}\binom{n}{k_i}$, donde $k_1 < k_2 < \cdots < k_t$ son índices enteros, entonces su longitud l es la longitud de la secuencia binomial $\binom{n}{k_t}$, véase [13].
b) Dada una secuencia $\{a_n\}$ con descomposición binomial $\sum_{i=1}^{t}\binom{n}{k_i}$, donde $k_1 < k_2 < \cdots < k_t$ son índices enteros, entonces su complejidad lineal LC es $LC = k_t + 1$, véase [13].

Las secuencias binomiales corresponden a las diagonales desplazadas del triángulo de Sierpinski y también aparecen en la representación gráfica de determinados autómatas celulares (ley 102 y 60) [14].

Tabla I
PRIMERAS 8 SECUENCIAS BINOMIALES CON SUS LONGITUDES l_k Y COMPLEJIDADES LC_k.

k	Sec. binomiales	l_k	LC_k
0	$(1,1,1,1,1,1,1,1,\ldots)$	$l_0 = 1$	$LC_0 = 1$
1	$(0,1,0,1,0,1,0,1,\ldots)$	$l_1 = 2$	$LC_1 = 2$
2	$(0,0,1,1,0,0,1,1,\ldots)$	$l_2 = 4$	$LC_2 = 3$
3	$(0,0,0,1,0,0,0,1,\ldots)$	$l_3 = 4$	$LC_3 = 4$
4	$(0,0,0,0,1,1,1,1,\ldots)$	$l_4 = 8$	$LC_4 = 5$
5	$(0,0,0,0,0,1,0,1,\ldots)$	$l_5 = 8$	$LC_5 = 6$
6	$(0,0,0,0,0,0,1,1,\ldots)$	$l_6 = 8$	$LC_6 = 7$
7	$(0,0,0,0,0,0,0,1,\ldots)$	$l_7 = 8$	$LC_7 = 8$

A continuación desarrollamos y analizamos dos algoritmos para calcular la LC de secuencias $\{a_n\}$ cuya longitud l sea de la forma $l = 2^r$, siendo r un número entero.

III. ALGORITMO B-BSD (BASIC-BINOMIAL SEQUENCE DECOMPOSITION)

Este algoritmo se aplica exclusivamente a secuencias cuya longitud l sea una potencia de 2. Hemos visto en la sección anterior que la descomposición binomial de estas secuencias y, en particular, la binomial $\binom{n}{k_t}$ nos aporta el valor de la LC de la secuencia bajo estudio. Por tanto, el algoritmo b-BSD calcula la descomposición binomial de una secuencia y a partir de ella su LC.

Algorithm 1 : Algoritmo b-BSD

Input: sec = secuencia a analizar
  $binom = [\emptyset]$
  for $i = 0;\ i < length(sec);\ i$++ **do**
    if $sec_i \neq 0$ **then**
      $sec+ = \binom{n}{i}$
      $binom.add(i)$
    end if
  end for
  Output: $binom$ = descomposición binomial de sec.

El algoritmo b-BSD toma como entrada la secuencia sec cuya complejidad lineal va a calcularse y busca el primer bit igual a 1. Si $bit_i == 1$, entonces suma bit a bit la secuencia original con la correspondiente binomial $\binom{n}{i}$ es decir $(sec = sec + \binom{n}{i})$ dando lugar a la secuencia suma, almacenada en sec, de la misma longitud que las secuencias sumadas. A continuación, se repite el mismo procedimiento sobre la secuencia suma y así se procede sucesivamente. El algoritmo termina cuando, en el paso final, la última secuencia suma sec es la secuencia idénticamente nula.

El algoritmo b-BSD aparece descrito en *Algorithm 1* mientras que la *Tabla II* describe paso a paso el algoritmo aplicado a la secuencia $sec_{16} = (0,0,1,0,0,1,0,0,1,0,1,1,1,1,0,1)$,

Tabla II
ALGORITMO B-BSD APLICADO A LA sec_{16}

Paso	Op.	Sec.	Posición de los bits 0	4	8	12
1		sec.	0 0 1 0	0 1 0 0	1 0 1 1	1 1 0 1
	+	$\binom{n}{2}$	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
2	=	sec.	0 0 0 1	0 1 1 1	1 0 0 0	1 1 1 0
	+	$\binom{n}{3}$	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
3	=	sec.	0 0 0 0	0 1 1 0	1 0 0 1	1 1 1 1
	+	$\binom{n}{5}$	0 0 0 0	0 1 0 1	0 0 0 0	0 1 0 1
4	=	sec.	0 0 0 0	0 0 1 1	1 0 0 1	1 0 1 0
	+	$\binom{n}{6}$	0 0 0 0	0 0 1 1	0 0 0 0	0 0 1 1
5	=	sec.	0 0 0 0	0 0 0 0	1 0 0 1	1 0 0 1
	+	$\binom{n}{8}$	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
6	=	sec.	0 0 0 0	0 0 0 0	0 1 1 0	0 1 1 0
	+	$\binom{n}{9}$	0 0 0 0	0 0 0 0	0 1 0 1	0 1 0 1
7	=	sec.	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
	+	$\binom{n}{10}$	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
final	=	sec.	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
$sec_{16} = \binom{n}{2} + \binom{n}{3} + \binom{n}{5} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10}$						

dando como resultado su descomposición binomial:

$$sec_{16} = \binom{n}{2} + \binom{n}{3} + \binom{n}{5} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10}.$$

Como la secuencia binomial de mayor índice k_t es $\binom{n}{10}$, entonces $LC(sec_{16}) = 10 + 1 = 11$.

Vemos que el algoritmo b-BSD calcula la complejidad lineal de una secuencia al igual que lo hace el algoritmo de Berlekamp-Massey. La diferencia entre ambos estriba en que el algoritmo de Berlekamp-Massey necesita procesar $2 \cdot l$ bits de secuencia con $O(l^2)$ mientras que el algoritmo b-BSD realiza simplemente la suma de 2 secuencias de l bits (l sumas de bits) por cada secuencia binomial empleada, es decir $O(t \cdot l)$, siendo t el número de secuencias binomiales en las que se ha descompuesto la secuencia original con $t \ll l$.

III-A. Mejora del Algoritmo b-BSD

La eficiencia del algoritmo anterior puede mejorarse si evitamos la suma de subsecuencias que sean idénticamente nulas. Por un lado, sabemos que los números combinatorios $\binom{n}{k} = 0 \ \forall n < k$. Por otro lado, sabemos que, en el paso i-ésimo del algoritmo b-BSD, los k_i primeros términos de la secuencia también son ceros.

Combinando ambos hechos, podemos reducir el número de operaciones (sumas) a nivel algorítmico. Cuando se encuentra el primer 1 de sec en la posición i-ésima, en lugar de sumar dos secuencias de l bits, $(sec + \binom{n}{i})$ basta con sumar ambas secuencias entre el i-ésimo y el $(l-1)$-ésimo bit $(sec_{i,l} + \binom{n}{i}_{i,l})$, ya que los bits $([0, i-1])$ en ambas secuencias son cero.

Comparado con $t \cdot l$, ahora el número de operaciones se reduce a:

$$\sum_{i=1}^{t} (l - k_i) < t \cdot l. \tag{2}$$

En el caso de considerar secuencias cuya complejidad lineal LC esté acotada superiormente, por ejemplo las secuencias *generalized* [9], el número de operaciones en el algoritmo b-BSD puede reducirse todavía más. Esto es debido a que en las secuencias generalized el índice máximo $k_{max} = l - \log l$, luego una vez alcanzado este índice no hace falta llevar a cabo más operaciones. Por tanto, el número final de sumas entre bits será:

$$\sum_{k_i=k_1}^{k_{max}} (k_{max} - k_i) < \sum_{i=1}^{t} (l - k_i) < t \cdot l. \tag{3}$$

Estas modificaciones pueden introducirse en el código de *Algorithm 1*, donde sólo afectarían a la suma de ambas secuencias $sec = sec_{i,max} + \binom{n}{i}_{i,max}$, con $max = l - \log l$. Al mismo tiempo y para estas secuencias el número de bits requeridos para calcular la LC sería $l - \log l$.

IV. Algoritmo f-BSD (folding-Binomial Sequence Decomposition)

En esta sección desarrollamos un nuevo algoritmo, el algoritmo de plegado f-BSD, para calcular la LC de secuencias cuya longitud l sea una potencia de 2. El nuevo algoritmo mejora las prestaciones del anterior (algoritmo b-BSD) ya que hace uso de la simetría que exhiben las propias secuencias binomiales.

En la siguiente subsección damos una representación matricial de la descomposición binomial de una secuencia $\{a_n\}$ de longitud l.

IV-A. Simetría de las secuencias binomiales

Dada una secuencia $\{a_n\}$ de longitud l cuya descomposición binomial es $\sum_{i=1}^{t} \binom{n}{k_i}$, analizamos la simetría de las binomiales $\binom{n}{k_i}$ $(1 \le i \le t)$ que intervienen en dicha descomposición. Expresamos cada una de esas secuencias binomiales como secuencias de l bits divididas en dos subsecuencias de longitud $\frac{l}{2}$. Escritas de esta manera, todas las secuencias $\binom{n}{k_i}$ presentan la siguiente estructura:

- Si $0 \le k_i < \frac{l}{2}$, entonces la secuencia $\binom{n}{k_i}$ puede escribirse como:

$$\binom{n}{k_i}_l = \left(\binom{n}{k_i}_{\frac{l}{2}}, \binom{n}{k_i}_{\frac{l}{2}} \right),$$

  donde la primera y segunda subsecuencia son idénticas.
- Si $\frac{l}{2} \le k_i < l$, entonces la secuencia $\binom{n}{k_i}$ puede escribirse como:

$$\binom{n}{k_i}_l = (ceros_{\frac{l}{2}}, \binom{n}{k_i - \frac{l}{2}}_{\frac{l}{2}}),$$

  donde la primera subsecuencia es idénticamente nula mientras que la segunda subsecuencia corresponde a los $\frac{l}{2}$ primeros bits de la secuencia binomial $\binom{n}{k_i - \frac{l}{2}}$.

Ambos resultados son consecuencia directa de la ley de formación de las secuencia binomiales definida en el item 1 de la sección II.

Tabla III
Descomposición de secuencias binomiales para sec_{16}

sec_{16}	0 0 1 0	0 1 0 0	1 0 1 1	1 1 0 1
$\binom{n}{2} = (\binom{n}{2}_{\frac{l}{2}}, \binom{n}{2}_{\frac{l}{2}})$	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
$\binom{n}{3} = (\binom{n}{3}_{\frac{l}{2}}, \binom{n}{3}_{\frac{l}{2}})$	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
$\binom{n}{5} = (\binom{n}{5}_{\frac{l}{2}}, \binom{n}{5}_{\frac{l}{2}})$	0 0 0 0	0 1 0 1	0 0 0 0	0 1 0 1
$\binom{n}{6} = (\binom{n}{6}_{\frac{l}{2}}, \binom{n}{6}_{\frac{l}{2}})$	0 0 0 0	0 0 1 1	0 0 0 0	0 0 1 1
$\binom{n}{8} = (ceros_{\frac{l}{2}}, \binom{n}{0}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
$\binom{n}{9} = (ceros_{\frac{l}{2}}, \binom{n}{1}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	0 1 0 1	0 1 0 1
$\binom{n}{10} = (ceros_{\frac{l}{2}}, \binom{n}{2}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
$sec_{16} = \binom{n}{2} + \binom{n}{3} + \binom{n}{5} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10}$				

En la *Tabla III* aparecen las secuencias binomiales de la anterior secuencia sec_{16} de longitud $l = 16$ escritas como dos subsecuencias de 8 bits. En las binomiales $\binom{n}{2}$, $\binom{n}{3}$, $\binom{n}{5}$ y $\binom{n}{6}$, se observa que los 8 primeros bits se repiten en ambas subsecuencias. Por el contrario, las binomiales $\binom{n}{8}$, $\binom{n}{9}$ y $\binom{n}{10}$ empiezan con 8 ceros y luego aparecen los 8 primeros bits de las secuencias $\binom{n}{k_i - 8}$ con $k_i \in [8, 9, 10]$.

En *Algorithm 2* aparece la forma matricial de las secuencias binomiales $\binom{n}{k_i}_l$ con $(1 \le i \le t)$.

Cuando las secuencias binomiales $\binom{n}{k_i}_l$ se representan como indica *Algorithm 2*, entonces la representación binomial

Algorithm 2 : Forma matricial de las secuencias binomiales

Para una secuencia binomial $\binom{n}{k_i}_l$:
if $k_i < \frac{l}{2}$ **then**
$\binom{n}{k_i}_l := (\binom{n}{k_i}_{\frac{l}{2}}, \binom{n}{k_i}_{\frac{l}{2}})$
else
$\binom{n}{k_i}_l := (ceros_{\frac{l}{2}}, \binom{n}{k_i - \frac{l}{2}}_{\frac{l}{2}})$
end if

escrita en forma matricial se expresa como:

$$\begin{pmatrix} \binom{n}{k_0} \\ \vdots \\ \binom{n}{k_t} \end{pmatrix} = \begin{pmatrix} \binom{n}{k_0} \\ \vdots \\ \frac{\binom{n}{k_{i-1}}}{\binom{n}{k_i}} \\ \vdots \\ \binom{n}{k_t} \end{pmatrix} = \left(\begin{array}{c|c} \binom{n}{k_0}_{\frac{l}{2}} & \binom{n}{k_0}_{\frac{l}{2}} \\ \vdots & \vdots \\ \binom{n}{k_{i-1}}_{\frac{l}{2}} & \binom{n}{k_{i-1}}_{\frac{l}{2}} \\ \hline 0 & \binom{n}{k_i - \frac{l}{2}}_{\frac{l}{2}} \\ \vdots & \vdots \\ 0 & \binom{n}{k_t - \frac{l}{2}}_{\frac{l}{2}} \end{array}\right), \quad (4)$$

donde $k_{i-1} < \frac{l}{2} \leq k_i$.

Por tanto, la representación binomial puede escribirse como una matriz dividida, a su vez, en 4 submatrices como:

$$\begin{pmatrix} \binom{n}{k_0} \\ \vdots \\ \binom{n}{k_t} \end{pmatrix} = \left(\begin{array}{c|c} M_0 & M_0 \\ \hline 0 & M_1 \end{array}\right). \quad (5)$$

El algoritmo f-BSD está basado en tres características de la representación matricial de la *Ec. (4)*:

- Las dos submatrices superiores son iguales.
- La submatriz inferior izquierda es cero.
- Como la longitud de las secuencias es potencia de 2, entonces la representación matricial puede desarrollarse de forma recursiva a partir de M_1

$$\left(\begin{array}{c|c} M_0 & M_0 \\ \hline 0 & M_1 \end{array}\right) = \left(\begin{array}{c|cc} M_0 & \multicolumn{2}{c}{M_0} \\ \hline 0 & M_2 & M_2 \\ & 0 & M_3 \end{array}\right) = \left(\begin{array}{c|cc} M_0 & \multicolumn{2}{c}{M_0} \\ \hline 0 & M_2 & \multicolumn{1}{c}{M_2} \\ & 0 & \multicolumn{1}{c}{\begin{array}{c|c} M_4 & M_4 \\ \hline 0 & M_5 \end{array}} \end{array}\right) = \ldots \quad (6)$$

hasta llegar a una submatriz de dimensión 1, es decir un único dígito binario.

La *Ec. (7)* es un ejemplo de la representación matricial de la secuencia sec_{16} descrita en la *Tabla III*. Para calcular el valor de LC, el algoritmo f-BSD hace uso de la simetría de las secuencias binomiales reduciendo recursivamente la longitud de la secuencia analizada.

$$\begin{pmatrix} \binom{n}{2} \\ \binom{n}{3} \\ \binom{n}{5} \\ \binom{n}{6} \\ \binom{n}{8} \\ \binom{n}{9} \\ \binom{n}{10} \end{pmatrix} = \left(\begin{array}{cc|cc} 0011 & 0011 & 0011 & 0011 \\ 0001 & 0001 & 0001 & 0001 \\ 0000 & 0101 & 0000 & 0101 \\ 0000 & 0011 & 0000 & 0011 \\ \hline 0000 & 0000 & 1111 & 1111 \\ 0000 & 0000 & 0101 & 0101 \\ 0000 & 0000 & 0011 & 0011 \end{array}\right). \quad (7)$$

IV-B. Algoritmo f-BSD

La subsección anterior nos proporciona los elementos necesarios para el desarrollo del algoritmo f-BSD. Este determina la secuencia binomial $\binom{n}{k_t}$ con el índice máximo para, a partir de él, determinar la LC de la secuencia considerada. A cada paso del algoritmo, se suma la primera mitad de la secuencia sec con la segunda. Si el resultado es distinto de la secuencia nula, entonces se toma como secuencia de partida la secuencia suma. En cada paso del algoritmo de plegado, la longitud de la secuencia considerada se reduce a la mitad con un total de $\log l$ pasos. En cada plegado se eliminan las secuencias binomiales correspondientes a (M_0, M_0), (M_2, M_2), (M_4, M_4), ..., respectivamente.

De esta manera dada una secuencia sec de longitud l, el número de operaciones que el algoritmo tiene que calcular es el siguiente:

$$\frac{l}{2} + \frac{\frac{l}{2}}{2} + \cdots = \frac{l}{2} + \frac{l}{4} + \cdots = \sum_{i=0}^{\log l} \frac{l}{2^i} \approx l$$

Algorithm 3 : Algoritmo de plegado

Input: sec = bits interceptados
$k_t = 0$
while $length(sec) > 1$ **do**
$l = length(sec)$
$sum = sec_{0,\frac{l}{2}}$ + $sec_{\frac{l}{2},l}$
if $sum \neq 0_{\frac{l}{2}}$ **then**
$sec = sum$
$k_t += \frac{l}{2}$
else
$sec = sec_{0,\frac{l}{2}}$
end if
end while
Output: k_t = binomial de índice máximo.

El código de este algoritmo aplicado a una secuencia sec de longitud l aparece descrito en *Algorithm 3*. Su complejidad computacional es $O(l)$.

Para calcular la LC de sec tanto el algoritmo b-BSD como el f-SFD buscan la binomial de índice máximo. La diferencia está en que el algoritmo b-BSD lleva a cabo en cada uno de sus pasos sucesivas comparaciones para determinar la correspondiente binomial y despues sumarla, mientras que el algoritmo de plegado tiene que realizar en total $\sum_{i=0}^{\log l} \frac{l}{2^i}$ operaciones suma.

Vemos un ejemplo de *Algorithm 3* aplicado a la secuencia anterior $sec_{16} = 00100100\ 10111101$.

- Step 1:
$$\begin{array}{rcc} & 0010 & 0100 \\ + & 1011 & 1101 \\ \hline & 1001 & 1001 \end{array}$$
Como sum = $1001\,1001 \neq 0_8$, entonces sec = sum = $1001\,1001$ and k_t = 8.

- Step 2:
$$\begin{array}{rcc} & 10 & 01 \\ + & 10 & 01 \\ \hline & 00 & 00 \end{array}$$
Como $sum = 0_4$, entonces $sec = 1001$.

- Step 3:
$$\begin{array}{cccc} & & 1 & 0 \\ + & & 0 & 1 \\ \hline & & 1 & 1 \end{array}$$

  Como $sum \neq 0_2$, $sec = sum = 1\ 1$ y $k_t = 8 + 2$.

- Step 4:
$$\begin{array}{cc} & 1 \\ + & 1 \\ \hline & 0 \end{array}$$

  Como $sum = 0, sec = 0$.

- Final: índice máximo $k_t = 10 \rightarrow LC = k_t + 1 = 11$.

IV-C. Comparación entre algoritmos

Considerando los tres algoritmos que pueden calcular la LC, la *Tabla IV* compara la cantidad de secuencia a procesar y la complejidad computacional de cada uno de ellos.

Aunque el algoritmo de Berlekamp-Massey es capaz de calcular la complejidad lineal de cualquier secuencia, cuando LC y longitud l estén próximas, este algoritmo no es la mejor opción para el cálculo de LC ya que su complejidad computacional es $O(l^2)$. Es en estos casos cuando los algoritmos b-BSD y f-BSD resultan mucho más convenientes.

Una diferencia entre b-BSD y f-BSD es que la complejidad de este último algoritmo no depende del número de secuencias binomiales que tenga su descomposición binomial, luego sus prestaciones computacionales serán mejores.

Se puede añadir que el algoritmo f-BSD es paralelizable mientras que el algoritmo b-BSD calcula en forma secuencial.

V. Conclusiones

En este trabajo se han desarrollado y descrito dos algoritmos b-BSD y f-BSD para cálculo de la complejidad lineal de secuencias binarias cuya longitud sea una potencia de 2. Ambos presentan mejores prestaciones que el conocido algoritmo de Berlekamp-Massey utilizado tradicionalmente para este cálculo. Se trata de una contribución importante en el estudio de esta clase de secuencias y facilita la detección de algún tipo de vulnerabilidad como por ejemplo un valor reducido de su LC. La aparición de tales vulnerabilidades en un PRNG de aplicación práctica, por ejemplo en IoT, podría comprometer el correspondiente dispositivo y los servicios que éste conlleva.

Por otro lado, la descomposición binomial de secuencias como una manera de extraer información sobre ellas es una herramienta novedosa que abre diferentes posibilidades a la hora de aplicarla a otro tipo de secuencias criptográficas.

Por último, puede añadirse que la estructura fractal de las secuencias binomiales se puede aprovechar en la descomposición de una secuencia sin tener que conocer ni manipular la secuencia completa. Esta última consideración se deja como trabajo a desarrollar de cara al futuro.

Tabla IV
Comparación de algoritmos

Algoritmos	Longitud de secuencia	Complejidad
Berlekamp-Massey	$2 \cdot l$	$O(l^2)$
b-BSD	$l - \log l$	$O(t \cdot l)$
f-BSD	$l - \log l$	$O(l)$

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía, Industria y Competitividad (MINECO), la Agencia Estatal de Investigación (AEI) y el Fondo Europeo de Desarrollo Regional (FEDER, UE), a través del proyecto COPCIS, referencia TIN2017-84844-C2-1-R, y por la Comunidad de Madrid (España) a través del proyecto CYNAMON, referencia P2018/TCS-4566, también co-financiado con fondos FEDER de la Unión Europea. El primer autor ha sido financiado con una beca JAE-Introducción del Ministerio de Ciencia e Innovación.

Referencias

[1] Chin, W.L., Li, W., Chen, H.H.: "Energy big data security threats in IoT-based smart grid communications,". *IEEE Communications Magazine*, 55(10), 70–75, 2017.

[2] Meyer, D., Haase, J., Eckert, M., Klauer, B.: "New attack vectors for building automation and IoT,". In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 8126–8131, 2017.

[3] Gallegos-Segovia, P.L.: "Internet of things as an attack vector to critical infrastructures of cities,". In *2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS)*, pp. 117–120, 2019.

[4] Cynthia, J., Sultana, H.P., Saroja, M.N., Senthil, J: "Security protocols for IoT,". *Ubiquitous Computing and Computing Security of IoT*, pp. 1–28, 2019.

[5] Mavromoustakis, C.X., Mastorakis, G., Batalla, J.M.: "Internet of Things (IoT) in 5G mobile technologies,". Springer, 8, pp. 26–36, 2016.

[6] "NIST Lightweight Cryptography project,". https://csrc.nist.gov/Projects/Lightweight-Cryptography, Last accessed 30 March 2020.

[7] Golomb, S.W.: "Shift Register-Sequences". Aegean Park Press, Laguna Hill, California, 1982.

[8] Massey, J.L.: "Shift-register synthesis and BCH decoding,". *IEEE Transactions on Information Theory*, 15(1), 122–127, 1969.

[9] Hu, Y., Xiao, G.: "Generalized self-shrinking generator,". *IEEE Trans. Inf. Theory*, 50, 714–719, 2004.

[10] United States Patent. Inventors: Chang, K. Y. (Daejeon, KR) et al. Electronics and Telecommunications Research Institute (Daejeon, KR). Document Identification: US 20060098820 A1. Date: May 11, 2006.

[11] Cusick, T.W., Stanica, P.: "Cryptographic Boolean functions and applications". Academic Press, England, 2017.

[12] Cardell, S. D., Fúster-Sabater, A.: "Binomial representation of cryptographic binary sequences and its relation to cellular automata,". *Complexity*, 2019, Article ID 2108014, 1–13, 2019.

[13] Fúster-Sabater, A.: "Generation of Cryptographic Sequences by means of Difference Equations,". *Applied Mathematics & Information Sciences*, 8(2), 475–484, 2014.

[14] Cardell, S. D., Fúster-Sabater, A.: "Linear models for the self-shrinking generator based on CA,". *Journal of Cellular Automata*, 11(2-3), 195–211, 2016.