Criptoanálisis del Generador Auto-Shrinking: Una Propuesta Práctica*

María Eugenia Pazo-Robles [1], Amparo Fúster-Sabater[2]

[1] ITBA, Instituto Tecnológico de Buenos Aires, Av E. Madero 399, Buenos Aires, Argentina
eugepazorobles@gmail.com
[2] Instituto de Física Aplicada, CSIC, Serrano 144, 28006 Madrid, España
amparo@iec.csic.es

Resumen. En este trabajo se presenta un ataque criptoanalítico sobre el generador auto-shrinking, un generador de secuencia cifrante bien conocido por su fácil implementación y sus buenas propiedades criptográficas. Se propone una variación práctica de la técnica criptoanalítica conocida como Guess-and-Determine, con supuestos claramente definidos y elaborados a lo largo del propio proceso. Se presentan resultados numéricos que mejoran el criptoanálisis planteado a este generador mediante la aplicación de la técnica conocida como Time Memory Trade Off (TMTO).Concretamente, se han logrado complejidades del orden de $O(2^{0.33L})$ en cuanto a cantidad de secuencia interceptada (siendo L la longitud del registro del generador), $O(L)$ en referencia a memoria consumida y una complejidad pre-computacional sumamente baja. Los programas de implementación del criptoanálisis y de resolución del sistema de ecuaciones han sido realizados en Matlab.

Palabras Clave: Generador auto-shrinking, criptoanálisis, cifrado en flujo, ecuaciones lineales.

1 Introducción

La criptografía de clave secreta se divide en dos grandes apartados: procedimientos de cifrado en flujo y procedimientos de cifrado en bloque. Los sistemas de cifrado en flujo son los más rápidos dentro de los métodos de cifrado, de ahí que en la actualidad se utilicen en numerosas aplicaciones prácticas. Prueba de ello son, por ejemplo, los algoritmos A5 (en su doble versión A5/1 y A5/2) que se utilizan en telefonía móvil GSM [1], el algoritmo E0 usado en especificaciones de Bluetooth [2] ó el algoritmo RC4 utilizado en Microsoft Word y Excel [3].
Un sistema de cifrado en flujo está compuesto por un algoritmo o generador de secuencia cifrante (conocido públicamente) y una clave de cifrado (conocida

* Trabajo financiado por CDTI (Centro para el Desarrollo Tecnológico Industrial, España) y las empresas INDRA, Unión Fenosa, Tecnobit, Visual Tools, Brainstorm, SAC y Technosafe en el marco del Proyecto Cenit-HESPERIA. También ha sido financiado por el Ministerio de Ciencia e Innovación (España) y el fondo europeo FEDER en el marco del Proyecto TIN2008-02236/TSI.

únicamente por los dos comunicantes). La clave corresponde normalmente al estado inicial del generador de secuencia. En el momento de su inicialización el generador toma la clave como semilla, posteriormente se ejecuta el algoritmo mediante un clock a la velocidad que necesite la aplicación y así se genera la secuencia cifrante.
Para cifrar, el emisor realiza una operación XOR, bit a bit, entre la secuencia cifrante y el texto claro, dando origen al texto cifrado que es el que se va a enviar por el canal de información. Para descifrar, el receptor genera la misma secuencia cifrante que suma bit a bit con el texto cifrado recibido y recupera así el texto claro original.
Muchos algoritmos de cifrado en flujo están basados en Registros de Desplazamiento con Realimentación Lineal [4], en inglés Linear Feedback Shift Registers (LFSRs), cuyas secuencias de salida, las *PN*-secuencias, se combinan entre sí mediante algún procedimiento o función no lineal. Entre los generadores de secuencias pseudoaleatorias más conocidos y mejor estudiados con aplicaciones criptográficas podemos señalar: generadores combinacionales, filtros no lineales, generadores controlados por uno o varios relojes, generadores multi-velocidad, generadores con decimación irregular, etc. Todas estas estructuras producen a su salida secuencias cifrantes con una alta complejidad lineal, períodos muy largos y buenas propiedades estadísticas.
Los procedimientos de cifrado de flujo se utilizan para dar seguridad criptográfica a sistemas de comunicaciones con requerimientos de velocidad y sincronismo. Uno de estos procedimientos de cifrado de amplia difusión es el generador auto-shrinking. La Unión Europea, a través del Proyecto Stork [5], propuso a la comunidad científica internacional romper este generador mediante alguna técnica criptoanalítica que mejorase el ataque TMTO.
Siguiendo esta premisa, se ha emprendido el criptoanálisis de este generador mediante variaciones a la técnica del Guess and Determine [6]. Se ha llevado a cabo una personalización de dicha técnica, mediante lo que se consideran suposiciones bien definidas para romper este generador de secuencia cifrante.
Este trabajo está dividido en secciones y organizado de la siguiente manera. En la sección 2, se describe el generador auto-shrinking con sus correspondientes características y propiedades de pseudoaleatoriedad. La sección 3 está dedicada tanto a la técnica de criptoanálisis propuesta como a los correspondientes resultados de implementación. En la sección 4, el ataque desarrollado se compara con otras técnicas criptoanalíticas en uso, mientras que en la sección 5 se presentan las conclusiones y futuros trabajos a realizar.

2 Generador Auto-Shrinking

El generador auto-shrinking fue diseñado por Meier y Staffelbach [7] para su uso en aplicaciones criptográficas. Este generador es de fácil implementación y consiste en un único LFSR de L etapas y polinomio de realimentación primitivo de grado L [4]. El esquema general de un LFSR aparece representado en la Fig. 1.

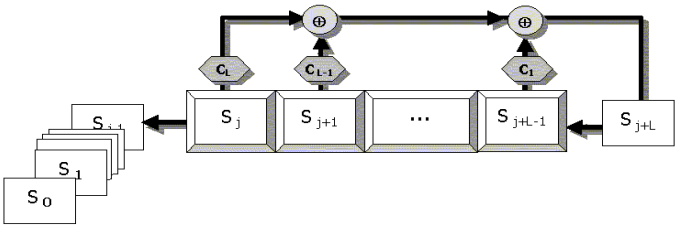

Fig.1 Registro de Desplazamiento con Realimentación Lineal

Este registro, que se nota, R_1, genera una única secuencia pseudoaleatoria, $\{s_n\}$, la cual se decima de forma irregular dando origen a la secuencia *auto-shrunken*, notada $\{z_n\}$, o secuencia de salida del generador. La regla de decimación es extremadamente simple,
se consideran pares (s_{2i}, s_{2i+1}) $(i = 0, 1, 2, \ldots)$ de bits consecutivos de $\{s_n\}$ tales que:

- Si $s_{2i} = 1$, entonces $z_j = s_{2i+1}$.
- Si $s_{2i} = 0$, entonces s_{2i+1} se descarta.

Es decir, si el primer bit del par considerado es un 1, entonces el segundo bit se inserta en la secuencia de salida. Por el contrario, si el primer bit del par considerado es un 0, entonces el segundo bit se rechaza. De esta manera, se van eliminando determinados bits de la secuencia $\{s_n\}$ y los que quedan constituyen la secuencia $\{z_n\}$ o secuencia auto-shrunken. La clave de este generador es el estado inicial del LFSR, aunque los autores recomiendan que el polinomio de realimentación forme también parte de la misma. De acuerdo con [7], el período, la complejidad lineal y las propiedades estadísticas de la secuencia $\{z_n\}$ son muy adecuados para su aplicación en criptografía. Asimismo y según [7], el polinomio característico $P(x)$ de la secuencia auto-shrunken $\{z_n\}$ viene dado por:

$$P(x) = (x + 1)^p, \tag{1}$$

donde el valor de p está acotado por:

$$2^{L-2} < p < 2^{L-1}. \tag{2}$$

Esto implica una relación de recurrencia lineal de la forma:

$$(E + 1)^p z_n = 0, \tag{3}$$

donde E es el operador desplazamiento que actúa sobre los términos de la secuencia $\{z_n\}$, es decir:

$$E\, z_n = z_{n+1} . \tag{4}$$

La ecuación (3) representa una ecuación en diferencias lineal con coeficientes binarios y constantes cuyo polinomio característico $P(x)$ dado por la ecuación (1) tiene una única raíz $\lambda = 1$ con multiplicidad p.
El período de esta secuencia auto-shrunken $\{z_n\}$ se denota por T y viene dado por:

$$T = 2^{L-1} . \tag{5}$$

La secuencia $\{s_n\}$, generada por el LFSR, puede considerarse formada por dos secuencias diferentes $\{c_n\}$ y $\{b_n\}$ que responden a los bits de subíndices pares e impares, respectivamente, de la secuencia $\{s_n\}$. Es decir:

$$c_i = s_{2i} \qquad \forall\ i \geq 0 \tag{6}$$

$$b_i = s_{2i+1} \qquad \forall\ i \geq 0 \tag{7}$$

A su vez, estas dos secuencias corresponden a la misma *PN*-secuencia $\{s_n\}$ generada por el LFSR pero desplazadas entre sí una distancia de 2^{L-1} bits. La existencia de dicho desplazamiento permite expresar una secuencia en función de la otra [6]. Así vemos que:

$$b_i = \sum_{j=0}^{L-1} h(x)\, c_j \qquad (0 \leq i \leq L\text{-}1) \tag{8}$$

Donde el polinomio $h(x)$ [6] sirve para expresar unos coeficientes en función de los otros. A partir de esta relación, se puede armar una matriz de coeficientes H, que es la que usaremos para resolver el sistema de ecuaciones.
Esta matriz de coeficientes se construye para las sucesivos valores de b_i expresados en función de los coeficientes c_i específicos.

$$H = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ . \\ . \\ . \\ b_{L-1} \end{pmatrix} \tag{9}$$

3. Ataque Propuesto

El ataque consiste en generar o crear una serie de suposiciones para los $L/2+1$ bits iniciales de c_i y, a partir de ellos, determinar los restantes bits de c_i a la vez que los L bits de b_i. Juntos forman la condición inicial de la secuencia $\{s_n\}$. Romper el generador es equivalente a encontrar la condición inicial de la secuencia $\{s_n\}$ o, en su defecto, un estado inicial que continúe la secuencia lógica de bits $\{s_n\}$.
Para ello, se arman bloques de B bits de secuencia interceptada (suposiciones) y, para cada bloque, se comprueba si se rompe o no el criptosistema. En efecto, nos vamos desplazando dentro de estos B bits haciendo las comprobaciones pertinentes hasta encontrar la solución correcta.
Dado que es posible expresar los coeficientes de acuerdo con las ecuaciones (8) y (9), tenemos la posibilidad de plantear un sistema de ecuaciones lineales. Entonces y a

partir de esta matriz H y del polinomio característico del LFSR, se plantea el sistema de ecuaciones lineales con coeficientes binarios que habrá que resolver. No todas las suposiciones son aptas para la resolución del sistema, por lo que hay que seleccionarlas adecuadamente. Esta selección de suposiciones es lo que hace de nuestro criptoanálisis, un ataque efectivo y más eficaz que el que utiliza la técnica de Guess and Determine.
En la sección 3.3 se explica el proceso de selección de dichas suposiciones. En la sección 3.4 se describen los programas desarrollados para este criptoanálisis y la secuencia lógica que sigue el ataque y el diagrama de flujo detallado en la Fig. 2.

Definimos a continuación los parámetros que caracterizan el ataque criptoanalítico:

- T_s: Tiempo que se tarda en obtener todas las suposiciones posibles o las que se usaran en el ataque. Este parámetro no influye en el tiempo de procesado del criptoanálisis y depende del generador auto-shrinking a criptoanalizar (longitud y polinomio característico del LFSR).
- *Cantidad de Suposiciones*: Número de suposiciones. Para cada suposición, se conocen $L/2+1$ bits de secuencia $\{c_n\}$, de un total de L bits. El ataque consiste en probar cada una de estas suposiciones hasta romper el sistema. Es decir, hasta calcular el resto de los L-$L/2+1$ bits de $\{c_n\}$ y los L bits de $\{b_n\}$ con el fin de poder seguir generando bits de la secuencia $\{z_n\}$.
- N: Número de bloques en los que dividimos el período de secuencia de salida $\{z_n\}$ o secuencia auto-shrunken.
- B: Longitud del bloque en el que dividimos la secuencia $\{z_n\}$, es decir la cantidad de bits de secuencia interceptada que necesitamos para romper el generador.
- *Porcentaje de aciertos*: Número total de aciertos sobre el total de bloques estudiados de longitud B bits por bloque.
- *ECU*: Cantidad de bits supuestamente conocidos de c_i. Este valor es igual a $L/2+1$ para todo polinomio de L etapas a considerar.

La idea básica consiste en analizar, para valores pequeños de L, el número de aciertos sobre todo un período de la secuencia de salida, extrapolando luego estos resultados a generadores auto-shrinking en un rango de aplicación criptográfica. Este análisis intensivo puede llevarse a efecto hasta valores de $L \approx 20$, con longitudes mayores el período de la secuencia generada se hace relativamente grande.
Las pruebas hasta el momento se han realizado en Matlab, simulando diferentes generadores auto-shrinking con registros de desplazamiento de longitud L, donde L varía en el intervalo 10 - 20 etapas. A su vez, para cada una de estas longitudes se han simulado LFSRs con diferentes polinomios de realimentación.

3.1. Método propuesto

Para realizar este ataque habrá que plantear, a partir de una suposición y una cantidad de secuencia interceptada, un sistema booleano de ecuaciones lineales y, posteriormente, resolverlo. La resolución se lleva a cabo aplicando sencillamente el método de Gauss. Claramente, el sistema planteado no siempre va a tener solución. A

su vez hay que hacer notar que, en cada suposición considerada, sólo los bits c_i que toman el valor 1 aportan una ecuación al sistema. Por el contrario, los bits c_i con valor 0 no añaden nuevas ecuaciones.
A partir del programa desarrollado también es posible expresar los coeficientes c_i en función de los b_i y viceversa. Esta doble vía nos permite calcular una mayor cantidad de incógnitas y no sólo obtenerlas a partir de la resolución del sistema.

3.2. Tiempo, Memoria y Serie Cifrante

El tiempo de procesamiento máximo del ataque criptoanalítico es el que tarda el programa en procesar todas las suposiciones sobre todos los bits de cada bloque *B*. Pasamos ahora a exponer algunos resultados numéricos.
Para generadores con $L = 20$, bloques de secuencia interceptada de valor $B = 120$ bits y *Cantidad de Suposiciones* = 34, se ha alcanzado un éxito de rotura del criptosistema del 82%. Por tanto, la cantidad de bits necesarios para lograr este porcentaje de éxitos será del orden de $O(2^{0.33*L})$. Dicho orden podría disminuir si se agregara alguna suposición adicional. A partir de los ejemplos programados puede también apreciarse que, con una cantidad de secuencia interceptada entre 100 - 200 bits, se obtiene un éxito de rotura del 93 %. Luego, para $L = 20$, la cantidad promedio de bits necesarios para romper el generador estaría en torno a los 150 bits.
Es importante destacar también que, para bloques de $B = 120$ bits, aunque haya un 18% de casos que no puedan resolverse sólo el 2% de los mismos son consecutivos. Luego, si bien hay ya garantizado un 82% de éxitos, tendremos un 98% de éxitos al analizar el bloque siguiente.
Cabe la misma disquisición para bloques de $B = 100$ bits, donde sólo el 6% de los bloques no resueltos son consecutivos, con lo cual si bien existe ya un 75% de éxitos, tendremos un 94% de éxitos de rotura con el bloque siguiente.
El tiempo de procesamiento hay que computarlo considerando el número de suposiciones que se utilizan en el ataque y la cantidad de bits que componen el bloque de secuencia interceptada. Se tendría así el número total de intentos para romper el criptosistema.
Hay que tener presente que solo son necesarios *L* bits de secuencia interceptada para plantear el sistema de ecuaciones y romper el generador. El problema es que no sería realista considerar que el ataque comienza precisamente en el bit de secuencia interceptada a partir del cual el sistema es capaz de romper el generador. Por ello, hay que tomar *B* bits de secuencia interceptada y desplazarse dentro del bloque hasta encontrar la solución correcta o estado inicial del LFSR.
Los tiempos de procesamiento deben ser aún evaluados convenientemente. De todas maneras, la forma en que están desarrollados los programas permite un procesamiento en paralelo, lo cual también disminuirá drásticamente los tiempos de procesamiento.
El ataque en paralelo supondría, entre otras cosas, tener una CPU por cada suposición. De este modo el tiempo de procesamiento disminuiría en un orden que sería directamente proporcional al número de suposiciones. Otro ejemplo de paralelismo estaría dado si se consideraran tantas CPU´s como bits hay en el bloque de secuencia interceptada. En este caso, el tiempo de procesamiento estaría dado por el tiempo que

tarda el programa en procesar los L bits de secuencia interceptada que necesita para resolver el sistema de ecuaciones.
La cantidad de memoria en uso es muy baja, y responde a la cantidad de secuencia interceptada necesaria para plantear el sistema de ecuaciones y a la tabla de suposiciones que tenemos que tener pre-computada.

Vemos en las siguientes Tablas algunos resultados concretos. En la Tabla 1, se representa el porcentaje de aciertos, o número de veces en que se rompió el generador, para distintas longitudes L del LFSR, R_1. También en esta tabla aparece la cantidad de suposiciones consideradas para distintos valores de L.
Así por ejemplo, para un auto-shrinking con L = 20 etapas, 16 suposiciones y 300 bits de secuencia interceptada, se obtiene un 84 % de aciertos considerando un número de bloques igual a 200. Es decir se ha corroborado que suponiendo ECU valores de c_i, en este caso 11, se obtendrían los 9 coeficientes c_i restantes y los 20 valores de b_i .

Tabla 1. Diferentes parámetros para generadores auto-shrinking con diferentes longitudes L

L	*Cantidad Suposiciones*	*Bloques: N*	*Longitud del Bloque: B*	*Período: T*	*Porcentaje de aciertos*	*ECU: L*/2+1
20	13-16	200	300	2^19	84%	11
16	8	405	80	2^15	83%	9
14	8	171	48	2^13	87%	8
10	4	332	16	2^9	86%	6

Para seguir con las comparaciones, presentamos la Tabla 2. Se puede distinguir en la misma, como el éxito en la rotura del generador depende de la cantidad de secuencia interceptada o tamaño del bloque.
Así puede observarse que siguiendo con polinomios de grado 20 y aumentando el número de suposiciones, se aumenta la capacidad de éxito del ataque. De todos modos, hay que señalar que el éxito del ataque depende prioritariamente de la cantidad de bits que componen el bloque. Sin embargo, hay que tener presente que en la práctica este número va a ser pequeño.Son conocidas las dificultades para obtener secuencia interceptada para el criptoanálisis, y además que cuanto mayor sea este número mayor será el tiempo de procesamiento, pues el ataque se basa en ir desplazándose dentro del bloque de secuencia interceptada .

Tabla 2. Diferentes parámetros para generadores auto-shrinking con la misma longitud L

L	*Cantidad Suposiciones*	*Bloques: N*	*Longitud del Bloque: B*	*Período: T*	*Porcentaje de aciertos*	*ECU: L*/2+1
20	34	301	100	2^19	75%	11
20	34	301	120	2^19	82%	11
20	13-16	200	300	2^19	84%	11

Como se aprecia en la Tabla 2,a medida que disminuye el número de bits en el bloque, disminuye asimismo el porcentaje de éxito en la rotura. Sin embargo, realizando un

estudio más exhaustivo, sí se observa que si no se logra romper el generador con los *B* bits definidos, la probabilidad de encontrar la solución para este polinomio de grado 20 en los siguientes *B* bits aumenta considerablemente. Este porcentaje de éxitos se refleja en la Tabla 3.

Tabla 3. Comparación para Segundos intentos entre polinomios de igual grado *L*, pero distinta longitud de Bloque

L	*Cantidad Suposiciones*	*Bloques: N*	*Longitud del Bloque: B*	*Período T*	*Porcentaje de aciertos*	*ECU: L*/2+1
20	34	85	100	2^19	94%	11
20	34	85	120	2^19	98%	11
20	34	85	150	2^19	99%	11

3.3. Pre-Procesamiento

No es necesario para nuestro criptoanálisis, realizar un pre-procesamiento con tiempos elevados. Los datos a pre-procesar son las suposiciones que vamos a necesitar tener calculadas antes de correr el programa.
De hecho es relativamente fácil encontrar estos supuestos que luego se utilizan para la determinación de los coeficientes c_i y b_i $(0 \leq i \leq L\text{-}1)$.
Calcular las suposiciones, en realidad, implica recorrer el universo de posibles soluciones al sistema de ecuaciones. Como se ha dicho anteriormente, existen suposiciones con un número *X* de 0´s en los bits supuestos de c_i, lo que supone (*ECU-X*) ecuaciones válidas para resolver. A su vez, de estas ecuaciones no todas nos permitirán obtener una solución. Por otro lado, el lugar donde se encuentren los 0´s dentro de la suposición también será un factor a considerar en la resolución del sistema.
En resumen, hallar las suposiciones correctas implica hallar la mayor cantidad posible de ecuaciones que resuelvan el sistema.
El método práctico utilizado en este criptoanálisis, para hallar las suposiciones, consiste en tomar en principio aquellas que tienen solo 1 cero y corroborar que se resuelve el sistema de ecuaciones. Luego en lugar de seleccionar esas suposiciones , se consideran las que se encuentran a continuación, en general tienen más de un cero, aumentándose la cantidad de ceros, en función del grado L, del polinomio en cuestión.
Debe ser tenida en cuenta la resolución del sistema de ecuaciones. En muchos casos un cero en determinado bit de la suposición, genera que el sistema no pueda ser resuelto. Dichos bits deben ser analizados, para el posterior descarte de la suposición, de ser necesario, en cada generador Auto-Shrinking.
Para el caso de L=10 , ya considerado , Tabla 1, las 4 suposiciones elegidas y con 2 ceros cada una, son :

$$Sup = [111110;111010;110110;101110]$$

Vemos en la Tabla 4. , la distribución de estas suposiciones a lo largo del período para L=10 como ejemplo. Es importante que la selección se distribuya convenientemente a lo largo del bloque de 16 bits, en este caso, de secuencia interceptada. El objetivo es hallar la condición inicial del generador Auto-Shrinking, con solo esos 16 bits, y por ende las suposiciones deberían estar distribuidas para encontrar esa solución. Como ejemplo práctico vemos en la Tabla 4, la distribución de estas soluciones propuestas a lo largo del período.

Tabla 4. Ubicación de las correspondientes suposiciones dentro de la secuencia de Auto-Shrinking. Suposición vs Puntero a la secuencia para un L=10 con período T=512

Supuestos	*Punteros de la secuencia Auto-Shrinking*															
111110	14	46	136	152	191	230	239	248	255	317	344	379	430	471	491	505
111010	110	181	193	211	390	432	482	497								
110110	17	36	120	127	294	300	405	421								
101110	34	60	90	95	101	104	114	175	210	217	243	287	310	366	389	437

Vemos en la Tabla 4., que con estas suposiciones, las soluciones halladas se distribuyen a lo largo del período, arribando a un éxito de un 86 % tomando solo 16 bits de secuencia para criptoanalizar, de un período $T= 2^{L-1}$.
El cálculo de suposiciones, no es en tiempo real de procesamiento, por ende su cálculo no perjudica al sistema para nada. La cantidad de suposiciones que integran el sistema criptoanalítico , es la variable que debe ser maximizada para un criptoanálisis efectivo en tiempo real. En el caso analizado de 2^{Ecu} posibles suposiciones, el sistema solo debe procesar 4, y ahí el tiempo de procesamiento es óptimo.

3.4. Programas

Todos los programas están realizados en Matlab. Consisten en un programa o función principal que requiere de otros programas o funciones secundarias. Dichas funciones son llamadas para realizar tareas específicas tales como:

- Separar en bloques de longitud estipulada los bits que componen la serie cifrante y que se consideran los datos de entrada.
- Introducir como datos todos los supuestos. Éstos son valores del estado inicial de la secuencia $\{c_n\}$. Dicha secuencia se genera a partir del LFSR del generador auto-shrinking.
- Plantear el sistema de ecuaciones necesario a partir de la secuencia cifrante correspondiente. El sistema de ecuaciones se plantea ya que es conocida la expresión de los coeficientes $b_i = F(c_i)$.
- Resolver el sistema de ecuaciones mediante método de Gauss. Es posible que con los datos propuestos no se pueda resolver, en ese caso habrá que considerar otra suposición. Si se logra resolver el sistema, entonces como salida produce la condición inicial de $\{c_n\}$ de L bits.

- Verificar si cada uno de los datos c_i y b_i hallados son correctos. Esta comprobación se lleva a cabo generando a partir de los resultados obtenidos más secuencia de salida y comparándola con la secuencia interceptada.

Vemos en el siguiente diagrama de flujo, la secuencia de tareas que están programadas. Todos los programas corren en Matlab.

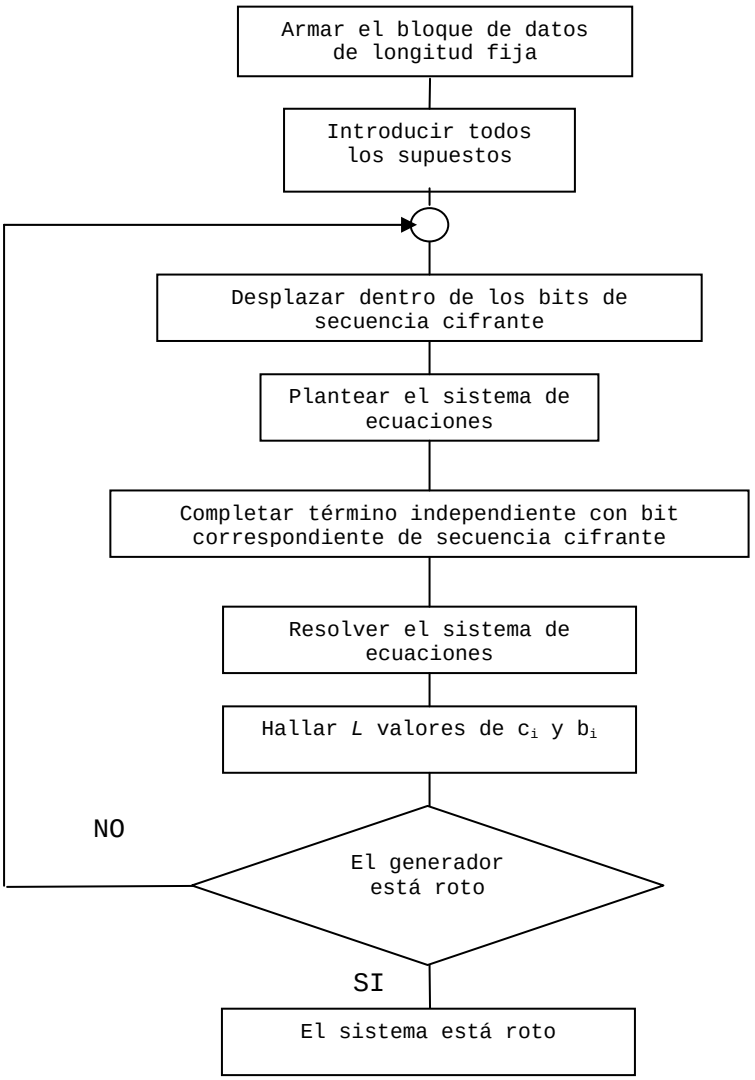

Fig. 2. Diagrama de flujo del proceso para romper el generador auto-shrinking

Existen otras funciones o tareas llevadas a cabo por programas secundarios, que no se corren en tiempo real como ocurre con los anteriores, pero que son necesarios para el criptoanálisis. Las tareas que realizan se pueden agrupar en:

- Determinar F, función que define $b_i = F(c_i)$.
- Armar y guardar la matriz de coeficientes H.
- Generar secuencia cifrante para determinar suposiciones válidas.

Los programas son de índole general, pudiendo ser aplicados en la resolución de un generador auto-shrinking cualquiera sea el polinomio generador del mismo. De hecho, estos programas han evaluado satisfactoriamente polinomios generadores que van desde grado 10 al grado 20. A partir del análisis realizado con registros o polinomios de grados relativamente bajos, [10 - 20], es posible extrapolar los resultados a LFSRs de mayor número de etapas.

4 Otros Criptoanálisis

En esta sección se muestran algunos criptoanálisis efectuados sobre este generador. En el marco del proyecto STORK, Strategic Road Map for Crypto [5], anteriormente referenciado se plantean una serie de problemas abiertos en criptografía. Entre ellos se encuentra el criptoanálisis del generador auto-shrinking logrando órdenes de complejidad menores a los obtenidos aplicando la Técnica de Time-Memory Trade-Off.
A continuación veremos algunas propiedades tanto de la técnica de TMTO, como de otra que ha logrado mejores resultados y que se conoce como Técnica de Guess-and-Determine.

4.1 Time Memory Trade Off

Existen trabajos publicados sobre técnicas criptoanalíticas aplicadas al generador auto-shrinking. En general, los parámetros que se tienen en cuenta son la complejidad de la fase pre-computacional, la complejidad de memoria, la complejidad en tiempo y la cantidad de serie cifrante o secuencia interceptada que se requiere para su rotura.
Aplicando esta técnica de TMTO [5][6] se logran órdenes de $O(2^{0.5*L})$ para la complejidad en tiempo, $O(2^{0.5*L})$ para la complejidad de memoria, $O(2^{0.25*L})$ para la cantidad de secuencia interceptada necesaria con una pre-computación de $O(2^{0.75*L})$.
Estos órdenes son elevados y un criptoanálisis en tiempo real para un generador auto-shrinking en un rango de aplicación criptográfica sería sumamente complejo por los recursos que requeriría.
Esta Técnica de TMTO es ampliamente conocida [8] y es aplicada actualmente para el criptoanálisis del algoritmo A5/1 utilizado en comunicaciones cifradas de telefonía celular, en la banda de GSM, [9].

4.2 Guess and Determine

Es conocida en la Literatura una técnica de criptoanálisis, denominda Guess-and-Determine [6], y que ha sido aplicada sobre el generador auto-shrinking por B. Zhang and D. Feng (Científicos de la Academia China de Ciencias). Si bien el criptoanálisis que logran es aparentemente muy eficaz, su puesta en práctica resulta compleja. En efecto, se muestra un ejemplo de criptoanálisis de un generador auto-shrinking de longitud $L = 40$, aunque no resulta tan clara la necesidad que tienen en lo que se refiere a complejidad de memoria ni a la implementación de los supuestos. Más aún, no aparece comentado el porcentaje de aciertos que logran para distintos polinomios ni la cantidad exacta de secuencia interceptada que han utilizado. Logran resolver en forma confiable el generador auto-shrinking con una complejidad de tiempo de $O(2^{0.556*L})$, complejidad de memoria $O(L^2)$, y con sólo $O(2^{0.161*L})$ de bits de secuencia interceptada. En cualquier caso, su trabajo ha sido de relevancia para la elaboración del nuestro.

5 Conclusiones y Trabajo Futuro

El presente trabajo sienta las bases para lograr de cara al futuro una programación del criptoanálisis del generador auto-shrinking que optimice órdenes de complejidad, especialmente en lo que se refiere a tiempos de procesamiento.
Este criptoanálisis se presta a ser programado utilizando programación en paralelo, como ya hemos visto.
El ataque es efectivo y rompe el generador, siempre que se trabaje con las suposiciones correctas. La elección de estas suposiciones se realiza durante el tiempo de pre-procesamiento, por ende el tiempo que consume hallarlas no constituye un tiempo a añadir al ataque en tiempo real.
Romper un generador de aplicación criptográfica no es tarea fácil, de todos modos aquí se presenta una técnica criptoanalítica con parámetros muy claros y bien definidos para su resolución. Si bien todavía es prematuro hablar de tiempos de procesamiento, pues habría que optimizar la programación, si se ha logrado sintetizar una Técnica Criptoanalítica eficaz en la rotura de este generador. Se ha logrado asimismo encontrar un método de aplicación general que es independiente del polinomio generador o de su grado, y que consigue romper el Generador Auto-Shrinking. Para su implementación, sólo hay que variar en un programa el vector o array que representa el polinomio de realimentación de R_1 .
Este trabajo se refiere a una propuesta práctica para un criptoanálisis efectivo y pensando en un criptoanalista sin grandes recursos. Concretamente, hemos logrado órdenes de complejidad que están en $O(2^{0.33L})$ en lo que se refiere a secuencia interceptada, $O(L)$ en cuanto a cantidad de memoria consumida y cuya complejidad pre-computacional es sumamente baja, únicamente lo requerido para hallar los supuestos.

Referencias

1. GSM, Global Systems for Mobile Communications, available at http://cryptome.org/gsm-a512.htm)
2. Bluetooth, Specifications of the Bluetooth system, Version 1.1, 2001, available at http://www.bluetooth.com/
3. Rivest, L.: RSA Data Security, Inc., March 12, 1998. (1999)
4. Golomb, S.W.: Shift Register Sequences, Aegean Park Press, Laguna Hill, California (1982)
5. Stork Proyect, available at http://www.stork.eu.org/documents/RUB-D6-2_1.pdf
6. Zhang, B., Feng, D.: New Guess-and-Determine Attack on the Self-Shrinking Generator. In: Proc. Asiacrypt'06. LNCS, vol. 4284, pp. 54--68. Springer, Heidelberg (2006)
7. Meier, W., Staffelbach, O.: The Self-Shrinking Generator. In: Proc. Eurocrypt '94. LNCS, vol. 950, pp. 205--214. Springer, Heidelberg (1994)
8. Hellman, M.: A Cryptanalytic Time-Memory Trade-Off. IEEE Trans. Information Theory, 26 (4), 234--247 (1980)
9. Barkan, E., Biham, E., Keller, N.: Instant Cipher Text-Only Cryptanalysis of GSM Encrypted Communication. Technion, Computer Science Department. Technical Report CS2006-07, (2006)