Recopilación de Ataques a Grain v1 y Grain 128

Silvestri, Valeria Ayelén
Cátedra: Criptografía (5° año)
Departamento de Ingeniería e Investigaciones Tecnológicas
Universidad Nacional de La Matanza
San Justo, La Matanza B1754
valeria.ayelen@gmail.com

Resumen

En este ensayo se presenta la familia de cifradores de flujo asincrónico Grain. Se explica brevemente el funcionamiento interno de Grain v1 y Grain 128 para luego explorar varios ataques, vulnerabilidades y otros problemas de seguridad que se encontraron desde su publicación, y cómo afectaron la evolución de dichos cifradores.

1. Introducción

Grain es una familia de cifradores de flujo sincrónicos, eficientes en hardware, pequeños y fáciles de implementar. Cuentan con la posibilidad de incrementar su velocidad paralelizando parte del procesamiento, característica que no se suele encontrar en otros cifradores. Se trata de uno de los ganadores del proyecto eSTREAM.

El proyecto eSTREAM se creó con el objetivo de identificar nuevos cifradores de flujo apropiados para el uso masivo, luego de que fallaran los seis cifradores seleccionados por el proyecto NESSIE. La convocatoria a presentar las primitivas comenzó en noviembre de 2004. El proyecto implicó varias fases de evaluación que se concluyeron en abril de 2008, anunciando siete ganadores.

Los participantes se clasificaron bajo dos perfiles. El perfil 1 fue orientado a los cifradores de flujo con aplicaciones en software, exigiendo una clave de 128 bits y un vector de inicialización (IV) de 64 o 128 bits. El perfil 2 se orientó hacia aplicaciones en hardware y recursos limitados, exigiendo una clave de 80 bits y un IV de 32 o 64 bits. A su vez se crearon los perfiles 1A y 2A para los cifradores con mecanismos de autenticación.

La primer versión de Grain, hoy conocida como versión 0, se postuló para el perfil 2, pero fue comprometida en las primeras etapas del proyecto eSTREAM (véase [1] para más detalles). Los autores corrigieron la vulnerabilidad agregando algunas variables a la función de filtro, de forma que el ataque propuesto ya no funcionaba. A su vez reconocieron la necesidad de utilizar una clave de 128 bits. Así, la versión 1 quedó definida por dos cifradores: Grain v1, que utiliza una clave de 80 bits cumpliendo con los requisitos del perfil 2, y Grain 128, que utiliza una clave de 128 bits.

Hoy en día se recomiendan las especificaciones de Grain v1 para la versión de 80 bits. Sin embargo, una serie de ataques y criptoanálisis llevó a la propuesta de una nueva versión de 128 bits, Grain 128a, que además incluye un mecanismo de autenticación (véase [2] para más detalles).

Este documento se centra en las versiones de Grain v1 y Grain 128. En la sección 2 se analiza el funcionamiento interno de ambas versiones y en la sección 3 se resumen los diferentes ataques y criptoanálisis que han realizado otros investigadores sobre dichas versiones.

2. Descripción de Grain

Como ya se mencionó, Grain es una familia de cifradores de flujo sincrónico. Es decir, su funcionamiento se basa en generar un *keystream* (flujo de clave) pseudoaleatorio que luego se utilizará para cifrar el mensaje original, aplicando un XOR bit a bit. Dicho *keystream* se genera a partir de una clave y un vector de inicialización (IV) que deben conocer tanto el emisor como el receptor.

Para transformar la clave y el IV en *keystream*, los cifradores Grain cuentan con dos registros de desplazamiento, uno lineal (LFSR) y uno no lineal (NFSR), y una función de filtro que se alimenta de ambos registros y devuelve un bit de keystream por

cada ciclo de ejecución. El LFSR garantiza un período mínimo en el keystream, mientras que el NFSR y la alinealidad de la función de filtro dificultan considerablemente el criptoanálisis.

La longitud de la clave, el IV y los registros varían entre las distintas versiones de Grain. En este documento se analizará la versión 1 y 128, aunque un análisis más exhaustivo se puede encontrar en [3] y [4] respectivamente.

2.1. Especificaciones de Grain v1

Grain v1 utiliza una clave de 80 bits, un IV de 64 bits y dos registros de desplazamiento de 80 bits cada uno. Para su análisis llamaremos f(x) a la función de retroalimentación del LFSR, g(x) a la del NFSR y h(x) a la función de filtro. El estado actual del cifrador queda definido por los valores de ambos registros, lo que permite 2^{160} estados posibles.

En la figura 1 se puede observar un diagrama del funcionamiento del algoritmo y la relación entre los registros y funciones.

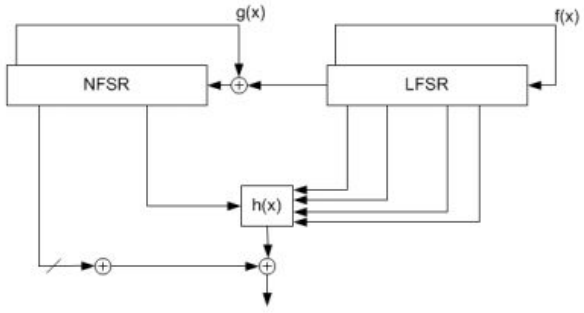

Figura 1. Cifrador Grain.

La función f(x) es una función lineal que toma como entrada algunos bits del LFSR y lo retroalimenta con un nuevo bit. Es decir, por cada ciclo del cifrador f(x) tomará su entrada, luego el LFSR desplazará todos los bits una posición a la izquierda y rellenará la última posición con el bit de salida de f(x).

Sean los bits del LFSR denotados como s_i, con i de 0 a 79, f(x) queda definida como:

$$f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}$$

Para evitar cualquier ambigüedad, también se puede definir como:

$$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i$$

Por su parte, la función g(x) es una función no lineal que toma como entrada un bit del LFSR y algunos bits del NFSR para retroalimentar a este último con un nuevo bit mientras realiza su desplazamiento.

Sean los bits del NFSR denotados como bi, con i de 0 a 79, g(x) queda definida como:

$$g(x) = 1 + x^{18} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{66} + x^{71} + x^{80} + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}$$

Nuevamente, puede ser expresada también como:

$$b_{i+80} = s_i + b_{i+62} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + b_{i+14} + b_{i+9} + b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + b_{i+15}b_{i+9} + b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} + b_{i+63}b_{i+45}b_{i+28}b_{i+9} + b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} + b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} + b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}$$

La función h(x) es una función no lineal que toma 5 bits del nuevo estado, es decir, una vez desplazados los registros, y agrega alinealidad a la salida. h(x) queda definida como:

$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

Donde x_0, x_1, x_2, x_3 y x_4 corresponden a los bits s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64} y b_{i+63}.

Por último, a dicha función se le suman otros bits del NFSR, quedando la salida definida como:

$$\text{Bit de keystream} = b_{i+1} + b_{i+2} + b_{i+4} + b_{i+10} + b_{i+31} + b_{i+43} + b_{i+56} + h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63})$$

2.1.1. Inicialización de la clave

Previamente se explicó cómo funciona Grain v1 en un ciclo normal de ejecución. Pero para que esto sea posible, primero es necesario preparar el entorno mediante un proceso de inicialización. El objetivo es dejar al cifrador en un estado con suficiente aleatoriedad y así dificultar el criptoanálisis desde el primer bit del keystream.

El NFSR se inicializa con la clave. El LFSR se inicializa con el IV y, como el IV es más corto, se completan los últimos bits con 1. Esto a su vez permite utilizar un IV con todos los bits en 0 sin anular el funcionamiento del cifrador.

Luego, el cifrador ejecuta 160 ciclos utilizando el bit de salida como retroalimentación del siguiente ciclo. En la figura 2 se puede observar el funcionamiento. Las 160 vueltas aseguran que la clave y el IV fueron completamente desplazados y ya no forman parte del estado actual. La retroalimentación añade confusión al proceso, logrando mayor aleatoriedad en el estado con el que comenzará la generación del *keystream*.

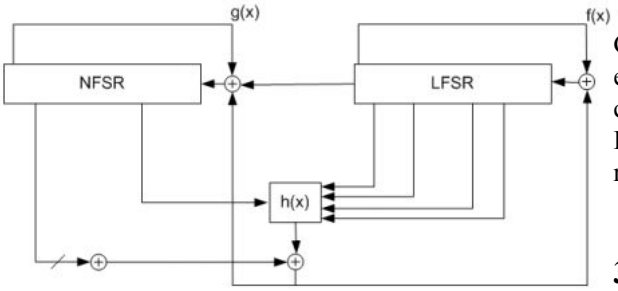

Figura 2. Inicialización de la clave.

2.2 Especificaciones de Grain 128

Grain 128 utiliza una clave de 128 bits, un IV de 80 bits y dos registros de desplazamientos de 128 bits cada uno. El funcionamiento es similar al de Grain v1.

Nuevamente, llamaremos f(x) a la función de retroalimentación del LFSR, g(x) a la del NFSR, h(x) a la función de filtro, s_i a los bits del LFSR y b_i a los bits del NFSR, con i de 0 a 127 en ambos casos.

La función f(x) queda definida como:

$$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}$$

O lo que es lo mismo:

$$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}$$

La función g(x) queda definida como la suma de una función lineal y una *bent function* (una función booleana difícil de aproximar por su naturaleza):

$$g(x) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}$$

O lo que es lo mismo:

$$b_{i+128} = s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84}$$

La función h(x) es de grado 3 y muy simple. Toma como entrada 9 bits del estado actual, conformado por los 256 bits que suman ambos registros. Queda definida como:

$$h(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$$

Donde las variables x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7 y x_8 corresponden a los bits b_{i+12} , s_{i+8} , s_{i+13} , s_{i+20} , b_{i+95} , s_{i+42}, s_{i+60} , s_{i+79} y s_{i+95} respectivamente.

La salida queda definida como:

$$z_i = b_{i+2} + b_{i+15} + b_{i+36} + b_{i+45} + b_{i+64} + b_{i+73} + b_{i+89} + h(x) + s_{i+93}$$

La inicialización también es muy similar a la de Grain v1. Primero se inicializa el NFSR con la clave y el LFSR con el IV, rellenando las últimas posiciones con 1 dado que el IV es más corto que el registro. Luego se ejecutan 256 ciclos, utilizando la salida como retroalimentación del siguiente ciclo.

3. Ataques y vulnerabilidades

En esta sección describiremos brevemente los distintos ataques, criptoanálisis y vulnerabilidades descubiertas para Grain v1 y Grain 128.

Recordemos que en 2005 cerró la convocatoria de eSTREAM y comenzaron las evaluaciones. Los participantes pasaron por tres etapas de evaluación hasta mayo de 2008, cuando se anunciaron los ganadores.

3.1. Distinguishing Attacks

En enero de 2006, a menos de un año del cierre de la convocatoria de eSTREAM, se publican los primeros ataques contra Grain v1, presentados en [5]. Se trata de los *distinguishing attacks*, donde el objetivo es distinguir la salida del cifrador de una secuencia aleatoria, con baja probabilidad de error y más rápido que un ataque por fuerza bruta.

Para evaluar la resistencia de Grain ante este tipo de ataques se utiliza el método de *linear sequential circuit approximation* presentado por Golic en [6]. Dicho método se basa en que los generadores de keystream se pueden plantear como autómatas finitos cuyo estado inicial y posiblemente su estructura dependen de la clave secreta. Golic demuestra que para un generador de keystream con M bits de memoria existe una función lineal de a lo sumo M+1 bits de salida consecutivos, que es una función del estado inicial. A dicha función se le puede aplicar la prueba chi-cuadrado para armar un *distinguishing attack*.

En [5] se demuestra que se pueden aplicar *distinguishing attacks* sobre Grain utilizando 2^{58} bits del keystream con un preprocesamiento de complejidad $O(2^{40})$, indicando que las funciones de retroalimentación del NFSR, la función de filtro y tal vez la del LFSR fueron pobremente elegidas.

3.2. Slide resynchronization attack

Grain v1 recibió otro ataque en 2006. En [7] se presenta el *Slide Resynchronization Attack*, que si bien no permite recuperar una clave secreta, demuestra una vulnerabilidad en el proceso de inicialización que será explotada en otros ataques.

El ataque es la aplicación de un *slide attack*, normalmente utilizado para cifradores en bloque, y se basa en dos observaciones sobre Grain: los estados del

proceso de inicialización son similares entre sí, y tanto el proceso de inicialización como el proceso de generación de keystream también son similares entre sí. Con estas condiciones se demuestra que para todo par (K, IV) existe otro par (K', IV') relacionado con ¼ de probabilidad de producir el mismo keystream desplazado un bit. Esto se puede generalizar a desplazamientos de n bits con 2^{-2n} de probabilidad.

Dos años después esta vulnerabilidad es explotada tanto en Grain v1 como Grain 128. En [8] se aprovecha esta vulnerabilidad para acelerar la búsqueda exhaustiva de claves a la mitad del tiempo. Asumiendo que conocemos el IV y los primeros t bits del keystream, normalmente probaríamos 2^{80} claves, procesando los 160 ciclos de inicialización y los primeros t bits del keystream por cada una para comparar con el keystream conocido. Pero con esta vulnerabilidad podemos encontrar claves relacionadas a la que estamos probando en cierta ronda, que sabemos que producirán el mismo keystream desplazado n bits, y así testear varias a la vez sin volver a procesar la inicialización por cada una.

Esta vulnerabilidad se podría haber evitado de varias maneras, por ejemplo: incorporando un contador en cada paso, o al menos bajado su probabilidad inicializando los últimos 16 bits del LFSR con (0, …, 0, 1) en vez de (1, …, 1).

3.3. Time / Memory / Data Tradeoffs

Los atacantes dejaron a Grain en paz durante todo el 2007, pero en febrero del 2008 volvieron a la carga. En [9] se presentan los ataques por *time / memory / data tradeoffs*, que si bien no son prácticos por requerir un preprocesamiento con una complejidad mayor a la de un ataque por fuerza bruta, demuestran otra vulnerabilidad de Grain v1.

Time / memory (TM) tradeoffs es una manera genérica de atacar cifradores por bloque, pero puede generalizarse al problema de invertir una función de un solo sentido. Se pueden mejorar usando varios puntos de datos, conocido como *time / memory / data (TMD) tradeoffs*.

En [9] se presenta el concepto de *BSW-sampling* y se demuestra que Grain v1 tiene una resistencia al *BSW-sampling* de solo 2^{-18}, que resulta relativamente pequeña y contradictoria con las especificaciones del cifrador, donde se asegura que la resistencia es "grande".

Pero como el estado interno de Grain v1 es el doble de la longitud de la clave, el ataque no se puede aplicar directamente. Acá hacemos notar ciertos aspectos de Grain v1. Por un lado, cada bit que ingresa a los registros no se utiliza hasta 16 ciclos después, lo que permite paralelizar hasta 16 ciclos. Por otro lado, la función que actualiza los estados es invertible tanto durante la generación del keystream como durante la inicialización, por lo que, si recuperamos un estado interno del cifrador en un instante t, podemos ciclar hacia atrás hasta recuperar la clave.

En [3] se basan en estas características para optimizar el ataque y conseguir una complejidad online acotada por $O(2^{71})$ y utilizando $2^{53.5}$ bits de keystream. Esto genera una segunda contradicción con las especificaciones, donde se asegura que la complejidad de este tipo de ataques no puede ser menor que 2^{80}.

3.4. Differential Attacks

Los ataques continuaron a lo largo del 2008. En [8] se presentan los *differential attacks*, donde el objetivo es aplicar diferencias en la entrada de una función criptográfica y encontrar propiedades en la salida relacionadas a dichas diferencias. Para los cifradores en flujo, las entradas y salidas que se asumen accesibles para el adversario son el IV y el keystream.

En [8] se estudian las características diferenciales truncadas de Grain v1, se particiona el espacio de claves e IVs y se logra un ataque que recupera una clave de entre 2^9, utilizando dos claves relacionadas y 2^{55} IVs. Para Grain 128 se halla la clave entre 2^{41} utilizando 2^{73} IVs.

Si bien la practicidad de este ataque es debatible, demuestra otra vulnerabilidad del cifrador.

3.5. Related-Key Chosen IV Attacks

En julio de 2008, es decir, a penas después del cierre de eSTREAM y el anuncio de Grain como uno de los ganadores, aparece el primer ataque efectivo contra Grain v1 y Grain 128. En [10] se presentan los *Related-Key Chosen IV Attacks*. Estos ataques aprovechan la vulnerabilidad presentada para el *Slide Resynchronization Attack* y logran recuperar la clave.

Para ello, se realizan m pasos con m+1 claves, cada una con un desplazamiento de α bits respecto de la anterior. Al inicio se generan $2^{2\alpha}$ IVs candidatos. En cada paso se obtiene un único IV válido para K_i y K_{i+1}. A partir de dicho IV se plantean α ecuaciones lineales, obteniendo α bits de la clave. Al finalizar todas las iteraciones, se realiza una búsqueda exhaustiva para encontrar los bits restantes de la clave.

Se han probado varios ataques con distintos valores de m y α. Para Grain v1 se logró recuperar la clave generando $2^{22.59}$ IVs y $2^{26.29}$ bits de keystream, con una complejidad computacional de $2^{22.90}$ (mucho menor que la complejidad 2^{80} de un ataque por fuerza bruta). Sobre una computadora Pentium-4, CPU 2.4GHz, 2.0 Gb RAM, sistema operativo Windows XP Pro SP2 se recupera una clave de Grain v1 en un promedio de 145 segundos y una clave de Grain-128 en un promedio de 95 minutos.

3.6. Fault Attacks

En julio de 2009, en [11] se plantea un *fault attack* en Grain 128. Para ello se basa en un modelo de fallas realista, donde el adversario se asume capaz de invertir exactamente un bit del LFSR aunque sin ser capaz de elegir su posición, capaz de controlar perfectamente el momento de introducir dicha falla y de realizarla varias veces en la misma posición, y capaz de reiniciar el dispositivo y repetir el mismo u otro ataque.

Con este modelo se propone un ataque se recupera la clave secreta utilizando 24 fallas consecutivas y un par de minutos de procesamiento offline.

En Julio de 2011, en [12] se plantea un ataque similar que utiliza el NFSR.

3.7. Cube Attacks

2010 fue otro año de descanso para Grain, hasta que en febrero de 2011 se demostró un ataque efectivo contra Grain 128. En [13] se plantean una serie de ataques llamados *dynamic cube attacks*, donde se utilizan *cube testers*, que son una familia de distinciones que pueden ser aplicados a la representación como caja negra de cualquier criptosistema. Estos ataques explotan distinciones obtenidas de dichos *cube testers*, a diferencia de los *cube attacks* estándares que plantean un sistema de ecuaciones lineales. Así pueden crear representaciones del cifrador de menor grado, reduciendo su complejidad.

El primer ataque planteado recupera la clave completa en tiempos de ejecución prácticos cuando las rondas de inicialización se reducen a 207. El segundo funciona para 250 rondas de inicialización y ejecuta 2^{28} veces más rápido que una búsqueda exhaustiva. El tercero funciona para la versión original de 256 rondas de inicialización, pero solo recupera la clave si pertenece a un subconjunto de 2^{118} claves posibles, y es 2^{15} veces más rápido que una búsqueda exhaustiva sobre el mismo subconjunto.

Varios meses después, en diciembre de 2011, los mismos autores en [14] plantean el primer ataque que recupera una clave de Grain 128 sin utilizar claves relacionadas ni depender de claves débiles ni versiones reducidas del cifrador, y es 2^{38} veces más rápido que una búsqueda exhaustiva para el 7.5% de las claves. Sin embargo, este ataque no amenaza la seguridad de la versión de Grain con 80 bits.

Como este ataque es muy difícil analizar matemáticamente y su complejidad puede ser muy alta para clusters de computadoras convencionales, para comprobar su funcionamiento se implementó un *cube tester* de 50 dimensiones sobre RIVYERA, un hardware basado en FPGA y que es masivamente paralelo y reconfigurable.

4. Conclusión

En este documento se explicó el funcionamiento interno de los cifradores Grain v1 y Grain 128 y se resumieron los ataques que sufrieron entre el año 2006 y 2011. Es interesante notar que los ataques realmente efectivos surgieron después de los años de evaluación y el anuncio de los ganadores de eSTREAM.

Se puede observar que la mayoría de los ataques no tienen una aplicación práctica, pero demuestran vulnerabilidades importantes en ambos cifradores que luego permitieron el desarrollo de ataques efectivos. Sin embargo, ambos cifradores aún son utilizados al día de hoy.

Se desaconseja utilizar Grain v1, ya que el ataque propuesto en [10] recupera la clave en tiempos relativamente cortos y un equipo poco potente y barato en comparación con lo que se consigue en el mercado hoy en día.

Por otro lado, el ataque contra Grain 128 propuesto en [14] requiere un equipo muy costoso y demora un tiempo considerable. Sin embargo, el avance de la tecnología logrará que los tiempos y costos disminuyan. Por esta razón surgió, y se aconseja utilizar, la nueva versión Grain 128a.

Esta investigación quedó acotada a los cifradores Grain v1 y 128, y a ataques hasta el año 2011. Como continuación de este trabajo se pueden investigar ataques y vulnerabilidades más recientes de Grain v1 y 128. Otro trabajo futuro puede ser la recopilación de ataques y vulnerabilidades para Grain 128a.

5. Agradecimientos

Este trabajo no podría haber sido posible sin el apoyo de la cátedra de Criptografía de la Universidad Nacional de la Matanza. Se agradece la paciencia del jefe de cátedra, Jorge Eterovic, quien insistió en que se presente este trabajo. También se agradece a Alexis Aranda Ocampo y Lara Cavicchioli, quienes ayudaron en la implementación de Grain v1.

6. Referencias

[1] Côme Berbain, Henri Gilbert y Alexander Maximov. “Cryptanalysis of Grain”. https://www.ecrypt.eu.org/stream/papersdir/2006/019.pdf

[2] Martin Agren, Martin Hell ,Thomas Johansson and Willi Meier. “A New Version of Grain-128 with

Authentication”. http://lup.lub.lu.se/search/ws/files/6156802/1981684.pdf
[3] Martin Hell, Thomas Johansson y Willi Meier. “Grain - A Stream Cipher for Constrained Environments”. https://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf
[4] Martin Hell, Thomas Johansson, Alexander Maximov y Willi Meier. “A Stream Cipher Proposal: Grain-128”. https://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain128_p3.pdf
[5] Shahram Khazaei, Mahdi M. Hasanzadeh, y Mohammad S. Kiaei. “Linear sequential circuit approximation of grain and trivium stream ciphers”. IACR Cryptology ePrint Archive, 2006:141, 2006
[6] Jovan Dj. Golic. “Intrinsic statistical weakness of keystream generators”. En ASIACRYPT, páginas 91–103, 1994
[7] Ozgul Kucuk. “Slide resynchronization attack on the initialization of grain 1.0”. http://www.ecrypt.eu.org/stream/papersdir/2006/044.ps
[8] CC. De Cannière, Küçük y B. Preneel. "Analysis of Grain's Initialization Algorithm," en Progress in Cryptology - AFRICACRYPT 2008, Lecture Notes in Computer Science 5023, S. Vaudenay (ed.), Springer-Verlag, pp. 276-289, 2008.8
[9] T.E. Bjørstad. “Cryptanalysis of grain using time / memory /data tradeoffs”. http://www.ecrypt.eu.org/stream/papersdir/2008/012.pdf
[10] Yuseop Lee, Kitae Jeong, Jaechul Sung y Seokhie Hong. “Related-key chosen iv attacks on grainv1 and grain-128”. En Yi Mu, Willy Susilo, and Jennifer Seberry, editors, Information Security and Privacy, volumen 5107 de Lecture Notes in Computer Science, páginas 321–335. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-70500-0 24.
[11] Guilhem Castagnos, Alexandre Berzati, C´ecile Canovas, Blandine Debraize, Louis Goubin, Aline Gouget, Pascal Paillier, y Stephanie Salgado. “Fault analysis of grain-128”. En HOST, páginas 7–14, 2009
[12] Sandip Karmakar y Dipanwita Roy Chowdhury. “Fault analysis of grain-128 by targeting nfsr”. En Proceedings of the 4th international conference on Progress in cryptology in Africa, AFRICACRYPT’11, páginas 298–315, Berlin, Heidelberg, 2011. Springer-Verlag.
[13] Itai Dinur y Adi Shamir. “Breaking grain-128 with dynamic cube attacks”. En FSE, páginas 167–187, 2011
[14] Itai Dinur, Tim G¨uneysu, Christof Paar, Adi Shamir, y Ralf Zimmermann. “An experimentally verified attack on full grain-128 using dedicated reconfigurable hardware”. En ASIACRYPT, páginas 327–343, 2011