Comparación de la Eficiencia en Hardware de los Cifradores de Flujo Grain, Mickey-128 y Trivium de Ecrypt

Blanca Nydia Pérez Camacho
Dr. René Cumplido
Departamento de Ciencias computacionales
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla. México
e-mail: brissadyn@ccc.inaoep.mx

Resumen

Debido a la necesidad de mantener segura la información que se envía a través de los canales de comunicación es que ha nacido el proyecto eStream de la organización Ecrypt[1], con el propósito de proponer nuevos estándares en cifradores de flujo y que se puedan implementar en hardware. Y que además, sean seguros, ocupen poca área y sencillos de implementar. Grain, Mickey-128 y Trivium son tres de los algoritmos de cifrado de flujo con implementación hardware que han pasado a la cuarta etapa del proyecto, los cuales son comparados en este trabajo, cuyas arquitecturas fueron diseñadas y sintetizadas bajo las mismas restricciones.

I. Introducción

Actualmente, el envió de la información se esta haciendo por medio de dispositivos móviles. Así, para mantener segura la información es que han surgido estándares en criptografía. Los algoritmos criptográficos se dividen en tres familias, las cuales son: algoritmos Hash, algoritmos asimétrico y algoritmos simétricos. Entre los algoritmos más importantes de cada familia encontramos MD5 (Message-Digest Algorithm 5) y SHAI (Secure Hash Algorithm I) son algoritmos Hash; RSA (iniciales de los apellidos de sus inventores) es un algoritmo asimétrico ; y DES (Data Encryption Standard), IDEA (International Data Encription Algorithm) y AES (Advanced Encryption Standard) son algoritmos simétricos. AES es el algoritmo más usado en la actualidad para hacer el envío de la información por medios móviles debido a que ha demostrado ser el más seguro. Con el propósito de implementar nuevos estándares es que la organización Ecrypt diseño el proyecto eStream, el cual esta enfocado cifradores de flujo con implementación en software y / o hardware. De los algoritmos que lleguen a la final es que se propondrán como posibles estándares. De entre los algoritmos de implementación hardware que han pasado a la cuarta fase se encuentran Grain, Mickey-128 y Trivium. Los cuales son implementados y comparados bajo las mismas condiciones, también se definirán bajo que restricciones es factible el uso de cada uno.

II. Especificación de los cifradores de Flujo

a. Grain

Grain fue desarrollado por Martín Hell, Thomas Johansson y Milli Meier [2], cuyos objetivos son el de ser veloz, seguro y simple; además, de cumplir con las características de un bajo consumo de recursos y facilidad de implementación. Grain es un cifrador de flujo síncrono y capaz de generar el flujo de la llave (*keystream*) de manera independiente al texto plano.

Este algoritmo hace uso de dos parámetros de entrada que son: la llave (*key*) que es de 80 bits y el vector de inicialización (*IV*) de 64 bits. También, esta definido por dos registros de 80 bits cada uno, los cuales son LFSR (*linear shift regsiter*) y NFSR (*non-linear shift register*), y tres funciones de las cuales dos son de retroalimentación y una que es la que genera el o los bits del keystream (puede tener hasta 2^{64} bits). El esquema de Grain es el que se presenta en la Figura 1.

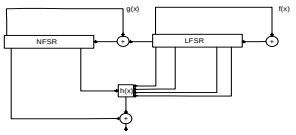

Figura 1. Esquema de Grain

El parámetro *key* sirve para inicializar el registro NFSR e IV para el registro LFSR, donde los bits que van de la posición 64 a la 79 son 1's. Para estabilizar los registros la

ISBN-13 978-968-7938-03-5

función h(x) hace un xor con los bits de las posiciones 3, 25, 46 y 64 del registro LFSR y el bit 63 del registro NFSR los cuales haciendo xor con las posiciones 1, 2, 4, 10, 31, 43 y 56 del registro NFSR , cuyo bit de resultado entra a las funciones g(x) y f(x) durante 160 ciclos ver Figura 2.

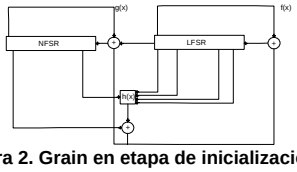

Figura 2. Grain en etapa de inicialización.

Las funciones g(x), f(x) y h(x) se encuentran definidas en [2].
El bit resultante de las funciones g(x) y f(x) son colocados en la posición 79 de cada uno de sus respectivos registros.
Para generar el keystream se usa el resultado de la función z_i, tal como se muestra en la Figura 1.

b.Mickey-128

Mickey-128 fue diseñado por Steve Babbage y Matthew Dodd, cuyos objetivos son: una baja complejidad en hardware y ofrecer un alto grado de seguridad [3]. Mickey-128 usa ciclos de reloj irregulares para sus registros; requiere de dos parámetros de entrada la llave de 128 bits y el IV que es de entre 0 y 128 bits, el keystream puede ser de hasta 2^{64} bits. El esquema de este cifrador de muestra en la Figura 3.

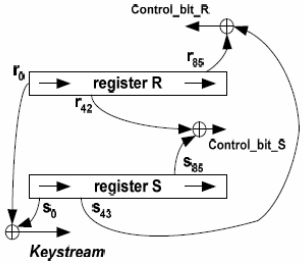

Figura 3. Esquema general de Mickey-128

Este cifrador de flujo se componen de dos registros R (*registro lineal*) y S (*registro no-lineal*) de 160 bits cada uno. Tiene dos señales de control principales Mixing e Input_bit., las cuales van a modificar junto con los bits de las posiciones 42 y 85 del registro R, y 43 y 85 del registro S a las señales de control internas que modificarán a los registros en su totalidad.

En la etapa de inicialización los registros son llenados con ceros, en esta etapa la señal Mixing siempre va a tomar el valor de true, el valor de Input_bit va a ir tomando el valor del bit de la posición que corresponda de cada uno de los parámetros de entrada. Primero del IV y luego de la llave, esto es descrito mediante ciclos for, donde el valor de i es el indicador de la posición que se debe de tomar.
Después viene una etapa de estabilización de registros mediante 160 ciclos, en la cual el INPUT_BIT es definido como cero.Para generar el keystream se hace un xor conlos bits de la posición cero del registro R y S, la señal mixing se pone en bajo. Para mayor detalle consultar [3].

c.Trivium

Trivium es un algoritmo síncrono de cifrado de flujo diseñado por Christoph De Cannière y Bart Preneel, este algoritmo esta diseñado para ser flexible sin dejar de ser compacto en ambientes restringidos, con eficiencia en potencia y de gran velocidad en la encriptación [4].
El puede ser de hasta 2^{64} bits, usando como entradas una llave y un vector de inicialización de 80 bits cada uno.
Esta compuesto por un registro de 288 bits los cuales van de la posición 0 a la 287, la Figura 4 muestra un esquema de Trivium.

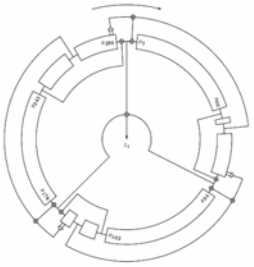

Figura 4 Esquema de Trivium

Una vez que se ha modificado un bit en el registro este no vuelve ha ser usado hasta después de 64 iteraciones, por lo que se puede tener una paralelización de hasta 64 bits.
En la etapa de inicialización el algoritmo hace uso de la llave y del IV, los cuales son colocados en el registro, de la posición 1 a la 80 la llave y de la 94 a la 173 el vector de inicialización, poniendo a cero el resto de las posiciones ha excepción de las tres ultimas que se ponen a uno.

ISBN-13 978-968-7938-03-5

En el pseudo-código[4] hay un ciclo de 4*288 para terminar de inicializar al registro. Se generan nuevos valores t, los cuales son usados en las posiciones 1, 94 y 178; y se realizan corrimientos a la derecha sin retroalimentación.

En la etapa de generación del keystream se realiza un proceso iterativo en el cual se usan los bits de las posiciones 66, 69, 91, 92, 93, 162, 171, 175, 176, 177, 243, 264, 286, 287y 288 para poder generar los 3 nuevos bits para el registro. Los bits de las posiciones 66, 93, 162, 177, 243 y 288 son para generar z_i usando la función xor entre ellos, se continua el proceso $N \leq 2^{64}$ de veces.

III. Arquitecturas Hardware

Las Arquitecturas son implementaciones directas de los cifradores de flujo, con el propósito de realizar comparaciones bajo las mismas restricciones; las arquitecturas fueron diseñadas usando el lenguaje VHDL en la herramienta de simulación ModelSim usando un diseño estructural.

a. Grain

La arquitectura está compuesta por componentes, los cuales son: controlgenerico, regNFSR, regLFSR, funcionfx, funciongx, funcionhx, funcionxor y reg_es. Al componente principal entran las siguientes señales: *iv*, *llave*, *m*, *clk* e *ini*; y salen las siguientes señales: *zi*, *dato_valido*, *ocupado* y *termino*.

En la etapa de generación del keystream, a diferencia de la etapa de inicialización, el componente funcionhx ya no funciona como parámetro para calcular los nuevos bits que se guardaran en los registros sino que funcionara como salida valida para *zi*. En esta etapa de generación se detendrá hasta calcular los m bits que definen en la señal de entrada *m*.

Una vez calculados los m bits del keystream, las señales de monitoreo pasaran a estar activadas en bajo con excepción de la señal *termino* que estará en alto.

b. Mickey-128

Resulta imposible paralelizar operaciones debido a la manera en la que se comporta el algoritmo. Ya que hace al ejecutar una operación y en dependencia de valores específicos de control se modifica totalmente los valores de los registros y al no poder predecir cual va a ser el nuevo valor de cierta posición especifica no se puede paralelizar a mas de un bit la arquitectura. En la Figura 6 se muestra la arquitectura a bloques de Mickey-128, y en las Figura 7, Figura 8, Figura 9 los componentes internos que describen la arquitectura de Mickey-128.

Esta compuesta por cuatro componentes principales: compo_sr, compo_clockkg, bloque_inputbit y control. Dentro del componente compo_clockkg se encuentra los componentes: compo_inputbit, compo_clockr y compo_clocks, compo_clockr tiene a su vez a compo_rtaps y compo_rprim; y compo_clocks tiene a compo_sg y compo_sp. Internamente para el registro S que esta definido en el bloque compo_clocks, se encuentran definidas cuatro secuencias de 128 bits cada una, estas secuencias son COMP0 y COMP1 en el componente compo_sg , FB0 y FB1 en el componente compo_sp.

Las señales que se requieren de entrada a la arquitectura son: *iv* longitud determinada por el usuario, k de 128 bits, n el número de bits que se esperan de salida e *ini* para iniciar la arquitectura. Y las señales de salida son: *zi* que son los bits del keystream, *dato_valido*, *ocupado* y *termino* que son señales de monitoreo.

c. Trivium

La arquitectura de Trivium, Figura 10, se diseño dividiendo el registro principal en tres partes, ya que actúan de manera independiente. En lo único que se requiere a las tres partes es para obtener el bit del keystream y para determinar los nuevos bits que se colocaran en el registro.

La arquitectura que se propone para el cifrador de flujo Trivium fue en base a un diseño ascendente, se dividió el registro en tres partes. Esta arquitectura esta compuesta por componentes, y son: genericontrol_Trivium, registro_es, regs1_193, regs94_s177, regs178_s288, t1_1, t2_2, t3_3 y funzi. Al componente principal entran las siguientes señales: iv, llave, m, clk e ini; y salen las siguientes señales: zi, dato_valido, ocupado y finalizo.

El componente funzi recibe los bits correspondientes de t1_1, t2_2 y t3_3, y mediante una función xor se determina los bits del keystream.

Una vez que se han generado el número de bits determinados por la señal de entrada m, es que las señales de monitoreo se activan en bajo y la señal finalizo se activa en alto.

ISBN-13 978-968-7938-03-5

IV. Resultados

El código VHDL fue sintetizado con XST de Xilinx ISE 8.2i usando un recurso FPGA. La tarjeta usada fue la Spartan3 development kit con el dispositivo xc3s400-5fg320 para la cual se sintetizó cada una de las arquitecturas.

Los resultados que se muestran para cada una de las arquitecturas son optimizaciones por velocidad. Para la arquitectura de Grain con sus diferentes grados de paralelización posibles se muestran en la Tabla 1. En la Tabla 2 son los resultados para la arquitectura Mickey-128 variando el tamaño del iv. Y en la Tabla 3 son los resultados para Trivium con todos sus posibles grados de paralelización.

# de bits en paralelo	Frecuencia máx. de reloj MHz	Tiempo ns	Velocidad de procesamiento Mbit/s	Area Slices	Vel. De proc / área Mbit/s/slices
1	140.531	7.116	140	156	0.897
2	140.531	7.116	280	220	1.272
4	140.531	7.116	560	252	2.222
8	140.531	7.116	1120	305	3.672
16	140.531	7.179	2240	425	5.270

Tabla 1 Eficiencia de la arquitectura Grain con diferentes grados de paralelización optimizados por velocidad

# de bits del iv	Frecuencia máx. de reloj MHz	Tiempo ns	Velocidad de procesamiento Mbit/s	Area Slices	Vel. De proc/ área Mbit/s/slices
1	96.584	10.354	96	315	0.0104
2	96.584	10.354	192	316	0.608
4	95.938	10.423	380	317	1.199
8	95.344	10.488	760	321	2.368
16	80.217	12.466	1280	374	3.422
32	84.217	11.874	2688	353	7.615
64	83.471	11.980	5312	368	14.435
128	89.165	11.215	11392	385	29.590

Tabla 2 Eficiencia de la arquitectura Mickey-128 con diferentes tamaños de iv optimizada por velocidad

# de bits del iv	Frecuencia máx. de reloj MHz	Tiempo ns	Velocidad de procesamiento Mbit/s	Area Slices	Vel. De proc/ área Mbit/s/slices
1	96.584	10.354	96	315	0.0104
2	96.584	10.354	192	316	0.608
4	95.938	10.423	380	317	1.199
8	95.344	10.488	760	321	2.368
16	80.217	12.466	1280	374	3.422
32	84.217	11.874	2688	353	7.615
64	83.471	11.980	5312	368	14.435
128	89.165	11.215	11392	385	29.590

Tabla 3 Eficiencia de la arquitectura Trivium optimizada por velocidad

En la Tabla 4 se comparan los resultados de las arquitecturas optimizadas por el mínimo de área en sus casos base.

Cifrador de Flujo	Frecuencia máx. de reloj MHz	Tiempo ns	Velocidad de procesamiento Mbit/s	Area Slices	Vel de proc/ área Mbit/s/slices
Grain (1 bit)	116.220	8.604	116	156	0.744
Mickey-128 (1 bit)	48.307	20.701	48	398	0.121
Trivium (1 bit)	126.844	7.884	126	216	0.583

Grain presenta un bajo consumo de recursos y una frecuencia alta comparada a las otras dos arquitecturas. El tipo de aplicaciones para las cuales es factible el uso de esta arquitectura es en aquellas en las cuales se requiera del envió de la información de manera rápida, haciendo uso de un bajo consumo de recursos.

La arquitectura de Mickey-128 comparada con las otras dos arquitecturas (Grain y Trivium), presentó una frecuencia menor. El tipo de aplicaciones para las cuales es factibles usar la arquitectura de Mickey-128 es en aquellas en las que no se requiera de manejar bits en paralelo pero si de una manera segura, tal como el envío de un archivo que contenga un número confidencial o información restringida.

Trivium, se encontró que maneja una frecuencia relativamente menor a la que maneja la arquitectura de Grain. Pero el punto más importante para esta arquitectura es con relación al consumo del área, la cual no aumenta de manera proporcional a como se incrementa el grado de paralelización, sino más bien lo hace de manera gradual. Por lo que el tipo de aplicaciones para las cuales resulta ser útil esta arquitectura es en aquellas en las que se requiera de un envío rápido de la información y un bajo uso de los recursos, como podría ser en la transmisión móvil.

ISBN-13 978-968-7938-03-5

V. Conclusiones

En este trabajo se mencionan los resultados de las implementaciones de las arquitecturas de los cifradores de flujo Grain, Mickey-128 y Trivium. Que para efecto de compararlas es que se diseñaron en el lenguaje vhdl en la herramienta ModelSim, fueron sintetizadas con XST de Xilinx ISE 8.2i usando un recurso FPGA. La tarjeta usada fue la Spartan3 development kit con el dispositivo xc3s400-5fg320 para cual se sintetizo cada uno de las arquitecturas de los cifradores de flujo. Los tres cifradores de flujo son seguros, fáciles y sencillos de entender, por lo cual han pasado a la siguiente etapa del proyecto. Las tres arquitecturas satisfacen, cada una, distintas necesidades, las cuales pueden ser determinadas por el área, velocidad y número de bits en paralelo.

VI. Bibliografía

[1] www.ecrypt.eu.org/stream
[2] M. Hell, et al., "Grain- A Stream Cipher for Constrained Environments". Disponible en http://www.ecrypt.eu.org/stream/ciphers/grain/grain.pdf
[3] S. Babbage y Matthew Dodd, "The stream cipher Mickey-128 2.0", eSTREAM, ECRYPT Stream Cipher Project. Disponible en http://www.ecrypt.eu.org/stream/hw.html
[4] C. De Canniere y B. Preneel, "Trivium, A Stream Cipher Construction Inspired by Block Cipher Design Principles", eSTREAM, ECRYPT Stream Cipher Project.
[5] P. Kitsos "Hardware Implementations for the ISO/IEC 180033-4: 2005 Standard for Stream Ciphers", International Journal of Signal Processing V. 3 N. 1,2006
[6] T. Good, et al, "Review of stream cipher candidates from a low resource hardware perspective". Disponible en http://www.ecrypt.eu.org/stream/hw.html
[7] F. K. Gurkaynak, et al, "Hardware evaluation of eSTREAM Candidates: Achterbahn, Grain, MICKEY, MOSQUITO, SFINKS, Trivium, VEST, ZK-Crypt", eSTREAM, ECRYPT Stream Cipher Project. Disponible en http://www.ecrypt.eu.org/stream/hw.html
[8] K. Gaj, et al, "Comparison of hardware performance of selected Phase II eSTREAM candidates", eSTREAM, ECRYPT Stream Cipher Project.
[9] P. Bulens, et al., "FPGA Implementations of eSTREAM Phase-2 focus Candidates with Hardware Profile", eSTREAM, ECRYPT Stream Cipher Project.
[10] F. K. Gurkaynak, "Recommendations for Hardware Evaluation of Cryptographic Algorithms", eSTREAM, ECRYPT Stream Cipher Project. Disponible en http://www.ecrypt.eu.org/stream/hw.html
[11] C. J. Mitchell y A. W. Dent, "International standards for stream ciphers: A progress report", eSTREAM, ECRYPT Stream Cipher Project.

ISBN-13 978-968-7938-03-5