Algoritmos Finalistas AES Round 2 - Advanced Encryption Standard 1999

Jean-Pierre Villacura

Segundo Semestre 2023, Universidad Técnica Federico Santa María

1 Resumen

El siguiente informe analiza pruebas de rendimiento de los distintos algoritmos Round 2 en la convocatoria de la AES en el año 1999 para la selección de un nuevo algoritmo basandose en comparaciones realizadas en la literatura científica. El trabajo está enfocado en determinar ventajas del algoritmo Serpent por sobre los otros algoritmos, tomando en consideración contextos de implementacion mediante Software(Arquitectura iterada v/s Pipeline) y uso de dispositivos FPGA. Los resultados sugieren que Serpent alcanza el mayor Throughput entre los algoritmos en el caso de arquitecturas Pipelined mediante procesador a la vez que mantiene una eficiencia similar a Rijndael. Por otra parte el uso de dispositivos FPGA tiene una función protectora adicional y para estos casos Serpent alcanza una eficiencia similar a la de Rijndael. Según los resultados obtenidos por la comunidad científica se plantea a Serpent como una alternativa altamente competitiva versus Rijndael por las razones anteriores.

Contents

2 Introducción

2.1 Descripción General

Este informe se basa principalmente en los algoritmos que pasaron a la segunda etapa, encontrando las ventajas que podría presentar la utilización de Serpent en vez de Rijndael, dentro del marco de laconvocatoria organizada por el NIST en la selección de un nuevo algoritmo estándar AES [1], debido principalmente a que los algoritmos de ese

entonces; DES y Triple DES no resultaban convenientes en términos de seguridad, tiempo de ejecución y recursos, lo que hacía necesario la adopción de un nuevo algoritmo de encriptación más rápido y seguro.
Se busca describir el funcionamiento general de estos algoritmos, mencionar sus ventajas y desventajas junto con discutir el benchmarking de estos algoritmos en diferentes plataformas.
Por otra parte, resulta útil e interesante el funcionamiento de estos algoritmos sobre módulos FPGA, los cuales son dispositivos de Hardware que permiten una implementación eficiente de estos algoritmos debido a su capacidad de ser programados y configurados para tareas específicas [4], por lo que una parte de este trabajo tambien se enfoca en explicar los rendimientos de estos algoritmos sobre estos módulos.

2.2 Objetivos Planteados

- Comparar eficiencia de los algoritmos que avanzaron al segundo round en la competencia de Advanced Encryption Standard 2000 para la elección de un nuevo algoritmo estándar [1], desde un punto de vista de tiempo de ejecución y recursos utilizados.
- Plantear ventajas desde el punto de vista técnico de la utilización de Serpent como alternativa a Rijndael en base al benchmarking y conocimiento teórico desarrollado en el informe.
- Explicar el funcionamiento general de los algoritmos del segundo round en la competencia, considerando un enfoque orientado al análisis de resultados obtenidas por las pruebas en la literatura académica.
- Comparativas de rendimiento de estos algoritmos utilizando módulos FPGA.

2.3 Trabajo Realizado

El trabajo realizado consiste principalmente en investigación sobre la literatura académica disponible respecto a los Algoritmos que llegaron al segundo Round de la competencia AES en el año 2000, concretamente las fuentes primarias son:

- Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms, NSA [10].
- A comparative study of performance of AES final candidates using FPGAs [4]
- Announcing the Advanced Encryption Standard AES, FIPS197 [1].
- Papers con las propuestas de los algoritmos Round 2 [5] [3] [8] [2] [7].

Obtenidas con los siguientes keywords en Google Scholar: **Cryptography**; **Rijndael**; **Serpent**; **MARS**; **RC6**; **Twofish**; **FPGA**; **Benchmarking**. **Round 2 AES**; Las fuentes anteriores contienen pruebas de rendimiento comparativas realizadas con los algoritmos estudiados en este trabajo y se busca explicar de forma técnica los resultados obtenidos en los papers anteriores, con énfasis en encontrar posibles ventajas del algoritmo Serpent frente a Rijndael.

2.4 Principales resultados

Los principales hallazgos en las pruebas comparativas realizadas son los siguientes:

- **Implementaciones a nivel de Software**: Rijndael y Serpent son los algoritmos que más sobresalen en rendimiento respecto a métricas como Throughput y tiempo de cifrado/descifrado por bloque, comportamiento similar en ambas arquitecturas medidas(Iterada y Pipeline).
- **Uso Módulos FPGA**: Los resultados sugieren que la implementación de los algoritmos Rijndael y Serpent en estos dispositivos resulta favorable debido a que las características de estos algoritmos escalan adecuadamente con el Hardware [4]. Un ejemplo de esta característica está relacionado con la arquitectura 'en-paralelo' en los FPGA, donde pueden 'paralelizarse' todas las operaciones del algoritmo Serpent y de esta manera lograr una mayor eficiencia.

2.5 Organización del Informe

El informe se estructura de la siguiente manera:

- Introducción: Son descritos los objetivos y el trabajo a realizar en este informe.
- Desarrollo: Se detallan en profundidad el funcionamiento de los algoritmos Round 2 en la competencia AES, adjuntando pruebas de Benchmarking realizadas durante este periodo las cuales se acompaña una posible explicación técnica a los resultados obtenidos en el rendimiento de estos algoritmos. Tambien se incluyen pruebas de rendimiento utilizando módulos FPGAs en esta sección.
- Conclusión: se detallan los principalez hallazgos obtenidos en esta investigación junto con posibles explicaciones desde un punto de vista técnico al comportamiento de estos algoritmos respecto a su rendimiento.

3 Desarrollo

3.1 Algoritmos Round 2 AES

En 1997, NIST anunció una convocatoria para desarrollar y escoger un nuevo algoritmo Advanced Encryption Standard para remplazar a el algoritmo Data Encryption Standard [9].
Fueron subidos cerca de 15 algoritmos en el primer Round y para el segundo round NIST escogió cinco de estos, los que son mencionados a continuación [9]. Los criterios de elección para este nuevo algoritmo se basan en tres perspectivas [9]:

- Security.
- Performance.
- Flexibility.

A continuación se detallan brevemente los algoritmos correspondientes a la segunda ronda de esta convocatoria, notar que los alcances de este trabajo están enfocados en el análisis de las pruebas de rendimiento de los algoritmos, por lo que mayor información del funcionamiento de estos algoritmos se sugiere leer su documentación en la literatura académica:

3.1.1 Rijndael

Rijndael es un algoritmo de cifrado que utiliza bloques y claves de 128, 192 y hasta 256 bits [6]. El número de rounds depende de la longitud de clave utilizada y esta compuesto por las siguientes operatorias: AddroundKey, y por cada ronda de cifrado SubBytes, ShiftRows, MixColumns, AddRoundKey. Para el caso de descifrado, las operatorias se invierten [5].
Muchas de las operaciones mencionadas anteriormente son definidas al nivel de un byte, con bytes representando elementos en el campo finito GF(G2^8) [5].
Es importante mencionar que la mayoría de las operaciones en Rijndael son paralelizables, esta particularidad tambien es presentada en el siguiente algoritmo a continuación, Serpent.

3.1.2 Serpent

Este algoritmo pertenece a la categoría de sustición-permutación con 32 rounds operando sobre cuatro palabras de 32 bits, con un tamaño de bloque de 128 bits y usa 33 128 bit SubKeys obtenidas desde una llave de 256 bits [6].
El cifrado consiste en una permutación inicial, seguido 31 rounds con las operaciones de Key Mixing, S-box y transformación lineal, continuando con el round final que contiene una operación de Key Mixing y otra adicional, terminando con una permutación final [2]. La ventaja de este algoritmo es que todas sus operaciones son completamente paralelizables, lo que permite aprovechar al máximo el rendimiento cuando corre en arquitecturas paralelas. En este contexto, **una ventaja adicional que posee Serpent sobre Rijndael es que maneja una mayor cantidad de operaciones paralelizables**.
El autor [6] menciona la que las operaciones de este algoritmo pueden ser implementadas en FPGA muy facilmente, por lo que **para el uso de FPGAs, es sugerible pensar en primera instancia el uso del algoritmo Serpent** frente a los otros del Round 2.

3.1.3 MARS

El algoritmo MARS soporta bloques de 128 bits y un tamaño variable de Key desde 128 a 1248 bits [6]. La primera etapa consiste en Key Addition, seguido de ocho rondas de Unkeyed Forward Mixing, ocho de Keyed Forward Transformation, ocho de Keyed Backwards Transformation y otros ocho finales de Unkeyed Backwards Mixing, con una etapa final de Key Substraction [3]. El desencriptado es el inverso del proceso anterior.

3.1.4 RC6

El algoritmo RC6 utiliza plaintexts de 128 bits, divididos en bloques de tamaño fijo de 32 bits con un número de rounds variable dependiente de la longitud de la clave, sea 128; 192 ó 256 bits. [6]
El autor [6] nota que todas las operaciones en las rondas no pueden ser realizadas en paralelo lo que podría afectar su rendimiento en dispositivos FPGA.

3.1.5 Twofish

Este algoritmo utiliza 128 bits, puede trabajar con 128, 192 o 256 bits de longitud de Keys [6] y consiste en 16 rondas basadas en una estructura Feistel modificada.
Sus etapas son las siguientes: S-Boxes, MDS Matrices, transformaciones pseudo-Hadamard, Whitening y Key-Schedule [7].

3.2 Implementaciones mediante Software

A continuación se presentan y se discuten los resultados obtenidos por la NSA respecto al Hardware Performance de los algoritmos Round 2 [10], los que abordan dos arquitecturas para la comparación: Velocidad Media - Small Area Iterated y Velocidad Alta - Area larga Pipelined. Estos consideran modelos de hardware utilizando VHDL y CMOS.
La idea de incluir esta sección es tener una idea del comportamiento de estos algoritmos Round 2 respecto a su velocidad y coste de recursos, los que en la siguiente sección tambien se realiza una comparación pero utilizando módulos FPGA.
Es importante mencionar que en los alcances de esta comparativa no se realizan optimizaciones específicas para cada algoritmo y que existen dos arquitecturas que serán medidas:

- Iterative Architecture: Implementa el algoritmo en una velocidad baja-media con énfasis en limitar el tamaño físico del hardware [10]. La característica principal es que existe una reutilización de los comonentes durante los ciclos reloj de ejecución.
- Pipeline Architecture: Busca tener el mayor Throughput(Ratio de encriptación) teniendo como desventaja un mayor consumo de recursos(Area) para conseguir el nivel de rendimiento requerido [10].

Los siguientes parámetros serán medidos, los que acompañan una descripción:

- Area: El que consiste en estimados de area de prelayout de los algoritmos.
- Thoughput: Que podemos considerar como el ratio de encriptación, una magnitud más alta implica mayor procesamiento por unidad de tiempo.
- N° de Transistores: Medida que puede ser considerada como recurso y resulta sumamente relevante a la arquitectura de computadores al poder estimar 'costes' de implementación.
- Inputs/Outputs requeridos y Time To Decrypt One Block: Cuyos resultados son excluidos de este trabajo ya que los alcances sólo consideran el uso de recursos y tiempo.
- Key Setup Time: Referido a la cantidda de tiempo requerido para que la etapa de SubKey Expansion esté lista para ejecutarse.

Las pruebas mostradas a continuación consideran como equipo el uso de modelamiento VDHL[10], el cual es un lenguaje de hardware que permite describir los recursos necesarios(Por ejemplo cantidad de transistores) para la implementación de cada algoritmo. De este modelamiento es posible obtener distintas parámetros de performance mencionados anteriormente, tales como area y Throughput [10].
La arquitectura general utilizada en las pruebas es la siguiente:

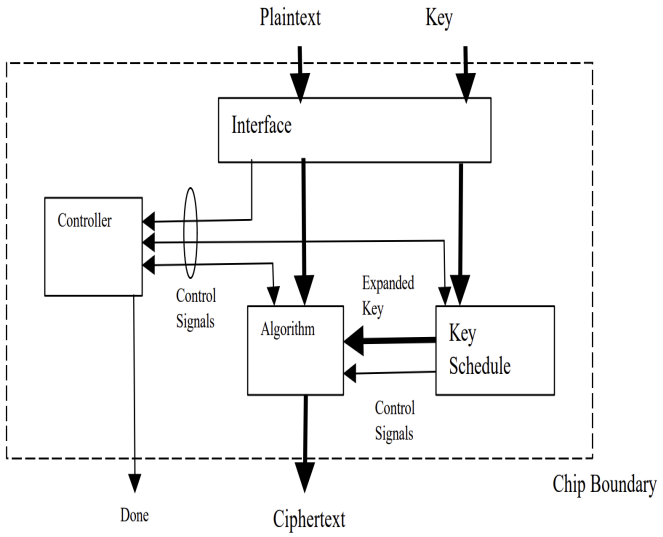

Figure 1: Top Level Architecture(Extraído de [10])

Notar que la arquitectura en donde se realizan las pruebas consiste en la interface que recibe todas las entradas de datos(Plaintext y Key). Existe un componente de Key Schedule que se encarga de la generación de SubKeys las que son utilizadas en los algoritmos de encriptación y este proceso es apoyado por el componente llamado Controller, que entrega las señales de control necesarias para mantener una correcta sincronización entre los distintos bloques generados [10].
Los resultados de los parámetros anteriores se muestran a continuación en la figura 2 y 3:

Parameter	Algorithm				
	MARS	RIJNDAEL	RC6	SERPENT	TWOFISH
Area (um2)	126,827,662	33,851,050	19,248,830	23,274,086	16,110,756
Transistor Count	1,941,371	641,681	307,247	345,483	264,058
Input/Outputs Required	520	520	520	520	520
Throughput (Mbps)	56.71	605.77	102.83	202.33	105.14
Key Setup Time Encrypt (ns)	9553	0	5742	19.77	60.87
Key Setup Time Decrypt (ns)	9553	211.3	5742	672.18	105.2
Algorithm Setup Time (ns)	0	0	0	0	0
Time to Encrypt One Block (ns)	2256.92	211.3	1244.8	632.64	1217.4
Time to Decrypt One Block(ns)	2256.92	211.3	1244.8	632.64	1217.4

Table 15 Iterated Summary

Figure 2: Resumen Parámetros obtenidos en la implementación de Algoritmos - Versión 'Iterated' (Extraído de [10])

En la figura 2, para la arquitectura Iterada es posible notar las siguientes observaciones [10]:

- El algoritmo que más área ocupa(less is better) corresponde a Mars, el cual supera hasta 4 veces en magnitud al segundo algoritmo que le sigue en cifras, Rijndael. Los otros algoritmos mantienen un valor similar a este último, en concreto Rijndael y Serpent tienen valores similares.
- En cuanto al número de transistores requerido, Mars requiere hasta 3 veces más que el segundo en la lista, Rijndael. Serpent requiere ligeramente menos transistores que Rijndael.
- En cuanto a la métrica de Throughput, **Rijndael** encabeza el ranking con una magnitud hasta 3 veces mayor que el segundo en la lista, Serpent.
- El menor tiempo de Key-Setup Time en cifrado y descifrado lo lideran Rijndael, Serpent y Twofish, con un valor muy por debajo de los otros algoritmos.
- El tiempo de cifrado y/o descifrado de un bloque es considerablemente más pequeño para Rijndael y Serpent.

- Recordando que el enfoque de este informe es buscar ventajas del algoritmo Serpent frente a Rijndael, se señala que para el enfoque Iterado, en cuanto a valores significativos, Rijndael alcanza mejores valores que Serpent en todos los casos excepto área. Por lo que **para esta arquitectura(Iterada) no se aprecia observaciones considerables en este aspecto**.

Parameter	Algorithm				
	MARS	RIJNDAEL	RC6	SERPENT	TWOFISH
Area (um2)	1,332,658,283	419,886,025	453,248,223	438,561,500	225,298,323
Transistor Count	20,661,116	4,292,749	7,536,366	5,741,469	3,783,973
Input/Outputs Required	520	520	520	520	520
Throughput (Mbps)	2189.16	5745.06	2196.67	8030.11	2273.53
Key Setup Time Encrypt (ns)	3712	0	3728.5	18.98	0
Key Setup Time Decrypt (ns)	3712	226.87	3728.5	212.55	0
Algorithm Setup Time (ns)	0	0	0	0	0
Time to Encrypt One Block (ns)	1987.98	222.8	1165.4	510	1126
Time to Decrypt One Block(ns)	1987.98	247	1165.4	510	1126

Table 16 Pipelined Summary

Figure 3: Resumen Parámetros obtenidos en la implementación de Algoritmos - Versión 'Iterated'(Extraído de [10])

En cuanto a la arquitectura 'Pipeline', es posible notar en la figura 3:

- El área es considerablemente mayor en todos los casos para esta arquitectura frente a la iterada, esto no es dificil de ver si pensamos que la arquitectura iterada recurre constantemente a la reutilización de componentes. Se repite el patrón en que MARS es el algoritmo que más espacio ocupa. Serpent y Rijndael ocupan un área similar.
- Respecto al número de transistores, el patrón se vuelve a reptir, MARS ocupa hasta 3 veces más que cualquiera de los otros. No hay diferencias considerables entre Serpent y Rijndael.
- Respecto a la métrica Throughput, Serpent alcanza la mayo cifra superando considerablemente a a los otros. Rijndael ocupa el segundo valor en el ratio de encriptación.
- El menor tiempo de Key-Setup time para el cifrado y descifrado lo vuelve a ocupar Rijndael, Serpent y Twofish.
- En cuanto al tiempo de cifrado y/o descifrado de un bloque, Rijndael y Serpent superan con creces a los otros algoritmos, con un factor de al menos 3, siento en este caso Rijndael más rápido.
- Respecto a observaciones generales que podemos destacar de Rijndael y Serpent en esta tabla comparativa para la versión de Pipeline, el comportamiento de ambos algoritmos es similar nombrado anteriormente para la versión Iterativa.

Se concluye para esta sección con los resultados obtenidos en la medición de los distintos parámetros según arquitectura Iterada v/s Pipeline que **Serpent puede plantearse como alternativa si es que interesan los valores de Throughput y tiempo de cifrado/descifrado de un bloque**, al tener valores cercanos a los mejores obtenidos por Rijndael.
Notar que en la mayoría de casos el comportamiento de los parámetros fue similar respecto al tipo de arquitectura utilizada(Pipiline v/s Iterada).

3.3 Uso de Módulos FPGA

Resulta interesante el comportamiento de estos algoritmos finalistas AES llevandolos a una implementación a nivel de Hardware usando Field Programmable Gate Arrays(FPGAs).
La atracción particular de estos dispositivos es que están en un punto intermedio entre procesadores y ASICs(Circuito Integrado de Aplicación Específica). Concretamente son más rápidos en la ejecución de operaciones que un procesador y al mismo tiempo contienen mayor flexibilidad que el uso de ASICs(Debido a que estos últimos tienen un costo de diseño elevado) [4].
Estos dispositivos llevan a cabo extremadamente bien operaciones comunes presentes en estos algoritmos de cifrados

tales como: Sustituciones de Bits, Permutaciones, lectura de tablas y funcoiens booleanas.
La característica principal del uso de FPGAs para los algoritmos de cifrado es que explotan eficientemente el uso paralelismo en operaciones y para el caso de los algoritmos de cifrado, su implementación en estos dispositivos mejora el rendimiento del cifrado que si se hubiese usado una implementación por Software [4].
El ejemplo más característico de esta situación es el algoritmo de cifrado Serpent, que destaca por el uso de operaciones completamente paralelizables, el cual consigue los primeros lugares en términos de eficiencia debido al uso de arquitectura 'en-paralelo' con el uso de dispositivos FPGA.
Los resultados obtenidos en la implementación de estos algoritmos finalistas sobre módulos FPGA, utilizando como equipo **Virtex FPGA Series**, con módulos de memoria Block SelectRAM [4], son los siguientes:

3.3.1 Benchmarking de algoritmos usando FPGA

Comparando el **Throughput**('Ratio de Encripción' - More is better), de las implementaciones en los módulos FPGA se obtiene la siguiente gráfica mostrada en la Figura 4 [4].

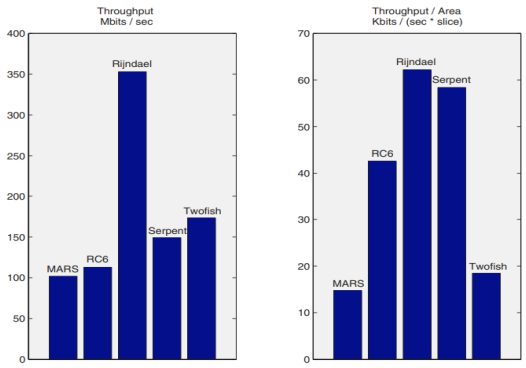

Figure 4: Throughput Comparisons of Our FPGA Implementations(Extraído de [4]) [4])

En la Figura 4 es posible notar que la métrica Throughput(Ratio de encriptación) es considerablemente mayor en el algoritmo RIJNDAEL, teniendo con un factor mayor a 2 que el segundo algoritmo más alto, Twofish, seguido de Serpent.
El comportamiento de Rijndael puede deberse principalmente a a que las características del algoritmo son altamente compatibles con el módulo FPGA utilizado [4]. Resulta curioso que en la métrica de Throughput, el algoritmo de Serpent no se encuentre cercano al valor alcanzado por Rijndael, ya que una de las características principales de Serpent es que alcanza una mayor cantidad de operaciones 'paralelizables' que Rijndael.
Respecto a la métrica Throughput/Area, que es posible verlo como un parámetro de eficiencia(Velocidad de encriptación v/s uso de recursos), podemos notar que Rijndael y Serpent se encuentran casi 'igualados' en términos de métrica, ocupando ambos los 2 primeros lugares. Este comportamiento calza con la idea de que al ser ambos algoritmos 'altamente paralelizables', según al configuración FPGA adoptada el hardware compatibiliza correctamente con el uso de estos algoritmos teniendo como consecuencia una elevada métrica de Throughput/Area. Los otros algoritmos no tienen algún comportamiento destacable en las gráficas, a excepción de RC6 que ocupa el tercer lugar en la métrica Throughput/Area y su cifra es considerablemente mayor a la de MARS y Twofish.
De esta figura podemos rescatar la idea que la eficiencia Throughput/Area de Rijndael y Serpent es similar utilizando FPGAs y desde esta perspectiva Serpent podría ser una alternativa a Rijndael, considerando la ventaja que tiene Serpent al tener un diseño 'conservador' lo que podría hacerlo más seguro que Rijndael. Siguiendo con la idea anterior, en la figura 5 se muestra una comparativa entre el Throughput y Key-Setup Latency de la implementación de estos algoritmos basada en Software ó usando FPGA(Columna **Our** en la tabla).

AES Algorithm	Throughput			Key-Setup Latency		
	MBits/sec		Speed-up	µs		Speed-up
	Software	Our		Software	Our	
MARS	[3] 188.00	101.88	1/1.84	[4] 8.22	1.96	4.19
RC6	[3] 258.00	112.87	1/2.28	[4] 3.79	0.17	22.29
Rijndael	[3] 243.00	353.00	1.45	[4] 2.15	0.07	30.71
Serpent	[4] 60.90	148.95	2.44	[4] 11.57	0.08	144.62
Twofish	[3] 204.00	173.06	1/1.17	[4] 15.44	0.18	85.78

Figure 5: Performance Comparisons with FPGA Implementations v/s Software(Extraído de [4])

Utilizando bloques de datos de 128 Bits con llaves de 128 Bits, es posible notar en la figura 5 que las implementaciones basadas en FPGA respecto a la métrica de Throughput son mejores en el caso de Rijndael y Serpent al alcanzar mayor magnitud. **Este comportamiento es esperable debido a que los algoritmos mencionados poseen operaciones altamente paralelizables**. Los otros algoritmos alcanzan un menor tráfico en su implementación por softwarem principalmente debido a los operaciones de multiplicación, las cuales no pueden ser aprovechadas al máximo utilizando FPGAs [4].
Respecto a la métrica de Key-Setup Latency, en todos los algoritmos este tiempo de preparación de las llaves (Menos es mejor) es considerablemente menor en los FPGA que en la implementación por Software, alcanzando un factor de hasta 144 en el caso de Serpent. Este comportamiento puede explicarse ya que en las implementaciones por software el proceso de cifrado no puede comenzar hasta que todas las rondas de Key-Setup esten completas. [4].

4 Evaluación de Resultados y Conclusiones

Existe una comparación de rendimiento en la literatura científica en estos algoritmos Round 2 que resulta útil de mencionar para esta investigación ya que nos permite tener una idea del rendimiento de distintas paramétricas de estos algoritmos que pasaron al segundo Round de la AES, las que nos sirve para extraer conclusiones respecto a Rijndael y Serpent. Concretamente las pruebas de rendimiento realizadas por la NSA [10] para los 5 algoritmos Round 2 comparando una arquitectura iterativa v/s Pipeline miden varios parámetros: Area, Throughput, número de transistores, tiempo de cifrado/descrifrado por bloque y Key Setup Time para el cifrado y descifrado. Los resultados sugieren un comportamiento relativo entre los algoritmos similar entre ambas arquitecturas, acompañado con un menor uso de recursos(Área) en el caso de la arquitectura iterativa. Los resultados sugieren que Rijndael y Serpent son los algoritmos que mejores parámetros obtienen en las pruebas, lo que apoya la idea que Serpent en caso de haber sido escogido podría ser tan competitivo como Rijndael. En el parámetro de Throughput, para arquitectura 'Pipelined' Serpent supera a Rijndael significativamente, teniendo un factor magnitud de hasta 3 con los otros algoritmos. Los resultados obtenidos en este benchmarking sostienen una competitividad 'casi a la par' entre Serpent y Rijndael.

El uso de FPGAs en la implementación de estos algoritmos de cifrado puede mejorar considerablemente el rendimiento de estos algoritmos comparandolo con una implementación por software. En [4] las comparativas miden las métricas de Throughput MBs/sec y Key-Setup Latency μsec. Los resultados arrojan a Rijndael y Serpent como los algoritmos que experimentan un mejor ratio Throughput(Con Rijndael teniendo un factor mayor a 2 respecto al segundo lugar, Serpent), este comportamiento se puede explicar debido a que ambos algoritmos poseen operaciones altamente paralelizables. En concreto los otros algoritmos poseen operaciones de multiplicación las cuales la implementación por software no alcanza a explotar completamente el potencial [4] teniendo como resultado que para estos algoritmos la métrica de Throughput se aproveche más mediante implementación de Software.
Respecto a la métrica de Key-Setup Latency, todos los algoritmos en la implementación por FPGA alcanzan un menor tiempo de latencia en la preparación de la llave, lo que resulta esperable ya que en el caso de la implementación por software el proceso de crifrado no puede comenzar hasta que todas las rondas de Key-Setup sean completadas [4].
En base a lo anterior se afirma que **una implementación mediante uso de módulos FPGA puede ser conveniente en el caso de los algoritmos Rijndael y Serpent**, al experimentar mejoras en las métricas de

Througput y Key-Setup Latency. **Esta implementación frente a procesadores comunes tambien trae consigo una ventaja en el aspecto de la seguridad**, ya que estos módulos pueden incluir una capa de protección que comunmente los procesadores normales no cuentan [4].

Se concluye en base a las comparaciones realizadas en [10], [4] que Serpent es competitivo frente a Rijndael en aspectos de rendimiento de Throughput y Eficiencia Throughput/Area bajo ciertas condiciones.

- En el caso de implementaciones mediante Software, la arquitectura 'Pipelined' mencionada en la sección 3 muestra que Serpent es el algoritmo con mayor Throughput(More is better), superando hasta en un 30 por cierto a Rijndael mientras que el área(recursos) utilizados por ambos es similar.
- Para el caso de usos de dispositivos FPGA, Serpent alcanza una eficiencia Throughput/Area muy similar al de Rijndael, sin embargo Rijndael alcaza hasta dos veces la magnitud de Throughput obtenida por Serpent.

Se plantea el uso de Serpent como una alternativa altamente competitiva a Rijndael cuando se utilizan módulos FPGAs, que pueden contar con una protección adicional que el que tendría un procesador común. En concreto Serpent alcanza eficiencias Throughput/Area similares a Rijndael. Respecto a procesadores comunes, refiriendonos a implementacion por software, cuando se utiliza la arquitectura 'Pipelined' mencionada en [10] Serpent tiene ventajas adicionales en términos de Throughput que Rijndael, por lo que sugiere considerar este tipo de arquitectura para aprovechar al máximo esta ventaja.

References

[1] 197, F. I. P. S. P. Announcing the advanced encryption standard aes. In *Announcing the Advanced Encryption Standard AES* (2001).

[2] Anderson, R., Biham, E., and Knudsen, L. Serpent: A proposal for the advanced encryption standard.

[3] Burwick, C. G., Coppersmith, D., D'Avignon, E., Gennaro, R., Halevi, S., Jutla, C. S., Matyas, S. M., O'Connor, L. J., Peyravian, M., Safford, D. R., and Zunic, N. Mars - a candidate cipher for aes.

[4] Dandalis, A., Prasanna, V., and Rolim, J. A comparative study of performance of aes final candidates using fpgas. vol. 1965, pp. 133–153.

[5] Daor, J., Daemen, J., and Rijmen, V. Aes proposal: rijndael.

[6] Fischer, V. Realization of the round 2 aes candidates using altera fpga. In *submitted for 3rd Advanced Encryption Standard (AES) Candidate Conference, New York* (2000), Citeseer.

[7] Kelsey, B., Whiting, D., Wagner, D., Hall, C., and Ferguson, N. Twofish: A 128bit block cipher.

[8] Rivest, R., Robshaw, M., Sidney, R., and Yin, Y. L. The rc6 block cipher.

[9] Schneier, B., and Whiting, D. A performance comparison of the five aes finalists. pp. 123–135.

[10] Weeks, B., Bean, M. O., Rozylowicz, T., and Ficke, C. Hardware performance simulations of round 2 advanced encryption standard algorithms. In *AES Candidate Conference* (2000).