Hardware Implementation of Authenticated Ciphers for Embedded Systems

Macarena C. Martínez-Rodríguez, Sergio Sauro, Piedad Brox, Santiago Sánchez-Solano

***Abstract*—The demand for embedded systems in applications that handle critical or private information has strongly focused designers' attention on the security aspects of this kind of system. Using the C programs and HDL descriptions available in the repositories of the CAESAR Competition and the ATHENa Project, this work presents a design flow that eases the development and evaluation of different solutions for the hardware implementation of authenticated ciphers and their incorporation as accelerating peripherals in embedded systems for different application cases. Three ciphers, finalists in the different categories established in the contest, have been analyzed, although the described approaches can be applied to any of the proposals submitted to the CAESAR Competition. A Zybo-Z7 development board that incorporates a Zynq-7000 device from Xilinx, which combines programmable logic from the FPGAs of the 7-Series with a dual-core Cortex-A9 ARM processing system, has been used as hardware platform in all the designs. The Vivado environment has been employed to perform the different stages of synthesis and verification necessary to carry out the implementation of the cipher, its conversion into an IP module, and its integration in an embedded system using different interconnection schemes that allow establishing cost/performance tradeoffs for different applications.**

***Index Terms*— Authentication, Ciphers, Cryptography Embedded Systems Security, FPGAs**

I. INTRODUCTION

EN una Sociedad cada vez más "digitalizada", en la que las Tecnologías de la Información y las Comunicaciones facilitan el intercambio de datos necesario para el desarrollo de muchas actividades cotidianas de carácter laboral, económico o social, los distintos aspectos relacionados con la seguridad en el almacenamiento y la transmisión de dichos datos cobran cada día mayor importancia para garantizar las bases de confianza requeridas por muchas aplicaciones y servicios [1].

Esta circunstancia, junto a otras motivaciones de naturaleza estratégica o defensiva, ha provocado un interés creciente por parte de estados, empresas y organizaciones internacionales en el desarrollo y estandarización de protocolos criptográficos seguros que permitan salvaguardar sus intereses y/o potenciar el desarrollo de sus áreas de actividad.

Frente a un desarrollo tradicionalmente "secreto" de los procedimientos y dispositivos criptográficos, en los últimos años se ha convertido en práctica habitual la celebración de competiciones abiertas, enfocadas a la selección de las propuestas que mejor se adapten a las necesidades y condiciones establecidas en el certamen. La Competición CAESAR (*Competition for Authenticated Encryption: Security, Applicability, and Robustness*) [2], fue iniciada en 2013 con el objetivo concreto de buscar algoritmos que permitieran cifrar y autenticar mensajes de forma simultánea, que resultaran robustos desde la perspectiva de la seguridad que proporcionan, y eficientes en cuanto al rendimiento de sus implementaciones hardware y software para distintos casos de aplicación. Además de proporcionar confidencialidad y asegurar la integridad de los mensajes, las propuestas presentadas debían ofrecer alguna ventaja frente al uso del AES-GCM, un cifrador autenticador basado en el AES (*Advanced Encryption Standard*) [3], ampliamente utilizado en muchos protocolos criptográficos actuales por sus buenas características en términos de rendimiento y eficiencia.

Entre las categorías establecidas en la competición CAESAR, la denominada "*Lightweight*" contempla el desarrollo de cifradores autenticadores que proporcionen una buena relación coste/rendimiento para hacer frente a los nuevos retos que plantea la proliferación de dispositivos con bajo coste, tamaño y consumo de potencia reducidos, y elevadas capacidades de cómputo y comunicación, surgidos en el contexto de "Internet de las cosas" (IoT), "Industria 4.0" y otros paradigmas similares que demandan el desarrollo de sistemas empotrados con estas características [4]-[6].

Los dispositivos programables disponibles actualmente constituyen una plataforma ideal de desarrollo y test, tanto para las propuestas asociadas a este caso de uso, como para las enfocadas a otras aplicaciones que requieran alto rendimiento (*High-performance*) o mecanismos de defensa más robustos (*Defense in depth*). En particular, los dispositivos de la familia Zynq-7000, que combinan un sistema de procesado basado en un ARM Cortex-A9 de doble núcleo con lógica programable de las FPGAs de la Serie 7 de Xilinx, resultan especialmente atractivos para el diseño de "Sistemas en un Chip" (SoC, *System-on-Chip*) que faciliten el desarrollo de sistemas empotrados que incorporen implementaciones hardware y software de las diferentes soluciones propuestas.

Este trabajo ha sido parcialmente financiado por los proyectos TEC2017-83557-R del Ministerio español de Ciencia, Innovación y Universidades y US-1265146 de la Junta de Andalucía, ambos con soporte FEDER.

M. C. Martínez-Rodríguez, Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC, Universidad de Sevilla), (e-mail: macarena@imse-cnm.csic.es).

S. Sauro, Escuela Politécnica Superior (Universidad de Sevilla), (e-mail: sersaudel@alum.us.es).

P. Brox, Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC, Universidad de Sevilla), (e-mail: brox@imse-cnm.csic.es).

S. Sánchez-Solano, Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC, Universidad de Sevilla), (e-mail: santiago@imse-cnm.csic.es).

Las características de estos dispositivos han sido explotadas en trabajos previos reportados en la literatura que describen la implementación, tanto hardware [7]-[10] como software [11], de sistemas criptográficos convencionales de clave simétrica (AES o DES-*Data Encryption Standard* [12]) y pública (RSA-*Rivest & Shamir & Adleman* [13]).

Las mejoras en las técnicas de ataque a implementaciones de estos criptosistemas, junto a la amenaza que supone el avance de la computación cuántica, han motivado la propuesta de nuevos algoritmos criptográficos capaces de proporcionar los niveles de seguridad requeridos por diversas aplicaciones [14].

De forma paralela, se hace necesario persistir en el desarrollo de nuevas técnicas de diseño que faciliten la implementación de las soluciones propuestas en sistemas empotrados que resulten robustos desde el punto de vista de la seguridad y eficientes en cuanto a tamaño y consumo energético [15].

Este trabajo describe la implementación de algunos de los algoritmos finalistas en la Competición CAESAR, mediante dispositivos SoC de la familia Zynq-7000 de Xilinx, a partir del código HDL suministrado por los participantes en la competición [16]. El objetivo del trabajo es doble. Por una parte, comparar distintas realizaciones hardware y software de estos algoritmos sobre la misma plataforma de desarrollo. Por otra, analizar el uso de diferentes esquemas de interconexión que permitan obtener implementaciones eficientes de los mismos para su uso en sistemas empotrados con recursos de cómputo y tamaños de memoria limitados.

En la Sección II se describen las características principales de las propuestas vencedoras en las diferentes categorías de la Competición CAESAR que han sido empleadas para validar la metodología propuesta. Las distintas etapas del flujo de desarrollo seguido, que contempla la implementación del cifrador, su conversión en un módulo IP y su incorporación en un sistema empotrado utilizando diferentes esquemas de interconexión, se introducen en la Sección III, donde también se detallan las herramientas aportadas por el Proyecto ATHENa [17] para facilitar el desarrollo y la comparación de las implementaciones hardware de las diferentes opciones. En la Sección IV se agrupan los resultados más significativos, en términos de coste y rendimiento, obtenidos en las etapas de implementación y verificación de las diferentes soluciones. Por último, la Sección V resume las principales conclusiones del trabajo.

II. CIFRADORES AUTENTICADORES

La Competición CAESAR estableció el conjunto de requisitos que debían cumplir las posibles propuestas. En primer lugar, desde el punto de vista funcional, un cifrador autenticador corresponde a una función con cinco entradas y dos salidas consistentes en cadenas de bits. Las cinco entradas son: texto a cifrar o "texto plano", de longitud variable; datos asociados, de longitud variable; número de mensaje privado, de longitud fija; número de mensaje público, de longitud fija; y clave, de longitud fija. La longitud de los números de mensaje público y privado puede ser de cero bits. Las salidas son un texto cifrado de longitud variable y una etiqueta de autenticación de longitud fija. Asimismo, debe ser posible recuperar el texto original y el número de mensaje privado a partir del texto cifrado, los datos asociados, el número de mensaje público y la clave, así como verificar si la etiqueta de autenticación corresponde al mensaje cifrado.

Además de una serie de datos relativos a cada cifrador o familia de cifradores sometidos a evaluación, los autores debían también aportar una implementación software de referencia de su propuesta, con idea de facilitar la comprensión de la misma, así como el criptoanálisis y la verificación de implementaciones posteriores. Las propuestas seleccionada para pasar a la segunda ronda (la competición se desarrolló a lo largo de tres rondas sucesivas) debían incluir asimismo un diseño hardware de referencia codificado en VHDL o Verilog.

El resultado del certamen se publicó el 20 de febrero de 2019, dándose a conocer la lista de propuestas vencedoras en las distintas categorías. Para entornos con recursos limitados se establecieron dos propuestas finalistas: 1ª) ASCON [18]; 2ª) ACORN [19]. Para aplicaciones de alto rendimiento se seleccionaron dos propuestas: AEGIS-128 [20] y OCB [21], sin prevalencia de una sobre la otra. Por último, para la categoría de defensa en profundidad hubo también dos propuestas finalistas: 1ª) DEOXYS-II [22]; 2ª) COLM [23]. Los ejemplos recogidos en este artículo se centran en las primeras propuestas de cada categoría, aunque la metodología y el flujo de diseño propuestos pueden ser aplicados a la totalidad de los cifradores participantes en la Competición CAESAR.

A. ASCON

ASCON fue seleccionado por el Comité de la Competición CAESAR como primera opción para la categoría de aplicaciones en entornos con recursos limitados. Se trata de una familia de algoritmos de cifrado autenticado y generación de resúmenes (*hashing*) que emplean la misma función de permutación. ASCON utiliza un modo de operación basado en esponja dúplex para el cifrado autenticado. El valor recomendado para las longitudes de clave, etiqueta y *nonce* es de 128 bits. La esponja opera sobre un estado de 320 bits, con bloques de mensajes de 64 o 128 bits. El proceso de cifrado/descifrado se completa en cuatro fases: inicialización, procesado de datos asociados, procesado de texto plano/texto cifrado y finalización [18] (Fig. 1).

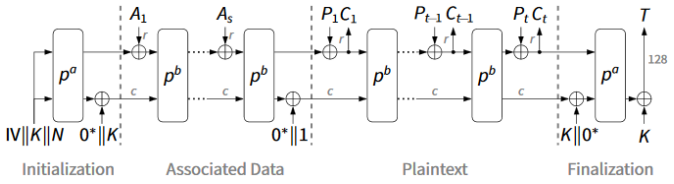


Fig. 1. Modo de esponja duplex en que está basado el esquema de cifrado autenticado usado por ASCON (según ilustración en [18]).

Las diferentes variantes del algoritmo pueden ser implementadas eficientemente sobre distintas plataformas, permitiendo el establecimiento de compromisos adecuados

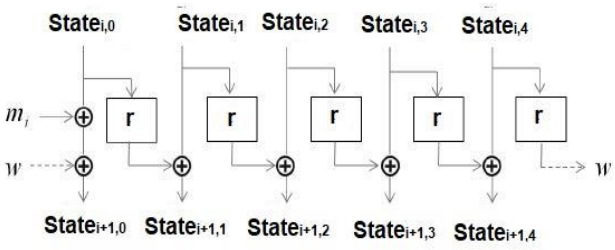


Fig. 2. Función de actualización del estado en AEGIS [20].

entre velocidad de operación, latencia, cantidad de recursos lógicos y consumo de potencia.

La variante implementada en este trabajo de la versión software de referencia "ascon128v12", a partir del código VHDL proporcionado por los autores, usa una arquitectura iterativa de ronda única con salida registrada y emplea bloques de mensajes de 64 bits (v1).

B. AEGIS-128

El cifrador autenticador AEGIS fue uno de los finalistas para aplicaciones de alto rendimiento. Se trata de un cifrador de bloque basado en funciones de rondas del AES. El coste computacional de AEGIS está en torno a la mitad del coste computacional del propio AES y proporciona el mismo nivel de seguridad que este cuando se selecciona un tamaño de clave, etiqueta y *nonce* de 128 bits.

La función básica de este algoritmo es la función de actualización del estado de 80 bytes tomando como entrada 128 bits, que realiza 5 funciones de rondas de AES en paralelo, como se muestra en la Fig.2. Primero se realiza una inicialización del estado, ejecutando 10 rondas de la función de actualización del estado. Después, se cargan los datos asociados, realizando tantas rondas como necesite para procesar los datos asociados divididos en 128 bits. De la misma manera se procesa el mensaje plano. En cada actualización del estado también se genera el cifrado del bloque procesado. La etiqueta de autenticación se extrae tras realizar 6 nuevas rondas de actualización del estado, lo cual implica que el coste en recursos para autenticar el mensaje es prácticamente nulo [20].

La descripción VHDL enviada a CAESAR e implementada en este trabajo, correspondiente al software de referencia "aegis128l", utiliza una arquitectura iterativa de ronda única con 8 rondas de AES operando en paralelo.

C. DEOXYS-II

DEOXYS es un esquema de cifrado autenticado basado en el cifrador de bloques ajustable DEOXYS-BC, una red iterativa de sustitución-permutación o SPN (*Substitution-Permutation Network*) que también utiliza funciones de rondas del AES. Además del texto plano y la clave, este tipo de cifradores utilizan una tercera entrada, denominada *tweak* [22], como muestra la Fig. 3 que ilustra la operación *AddRoundTweakey*.

La propuesta proporciona seguridad de 128 bits, tanto para privacidad como para autenticidad, y permite dos modos de operación: uno para aplicaciones en las que el *nonce* no debe reutilizarse; y otro capaz de proporcionar seguridad incluso cuando se reutiliza el *nonce*. Esta última opción, DEOXYS-II, fue la finalista preferente de la Competición CAESAR para aplicaciones con requisitos de defensa en profundidad.

La variante implementada en el trabajo corresponde a la opción DEOXYS-I, con versión software de referencia "deoxysi128v141" y arquitectura de ronda única con pre-cálculo especulativo.

III. METODOLOGÍA DE DESARROLLO

La plataforma usada como soporte de las implementaciones software y hardware comparadas en este trabajo ha sido una placa Digilent Zybo-Z7 [24]. Además de un dispositivo Zynq-7000 XC7Z020-1CLG400C, esta placa de desarrollo de FPGA incluye 1 GB de memoria DDR3 y 16 MB de memoria Flash, conexiones Gigabit Ethernet y USB 2.0, puertos HDMI de entrada y salida y distintos pulsadores, conmutadores y LEDs.

Como punto de partida para la implementación de los cifradores seleccionados se han utilizado las descripciones HDL enviadas por los autores a partir de la segunda ronda de la Competición CAESAR [16].

De especial ayuda para la realización del trabajo ha resultado el material proporcionado por el proyecto ATHENa, puesto en marcha en la universidad George Mason, para dar soporte a CAESAR y otras competiciones similares, con el objetivo de desarrollar una herramienta para la evaluación automatizada de primitivas criptográficas, tanto software como hardware, dirigidas a FPGAs, SoCs y ASICs [17].

El repositorio disponible a través de la página web del proyecto [25] incluye los códigos C de referencia de las distintas propuestas, descripciones VHDL o Verilog que permiten llevar a cabo la implementación hardware de las mismas utilizando una interfaz de comunicación común (CAESAR *Hardware API* [26]), así como una serie de

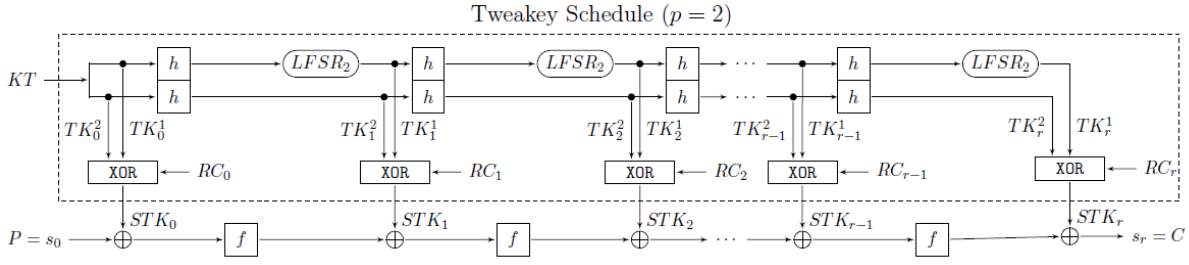


Fig. 3. Instanciación de la estructura *Tweakey* usada en el cifrador de bloques DEOXYS-BC (según ilustración en [22]).

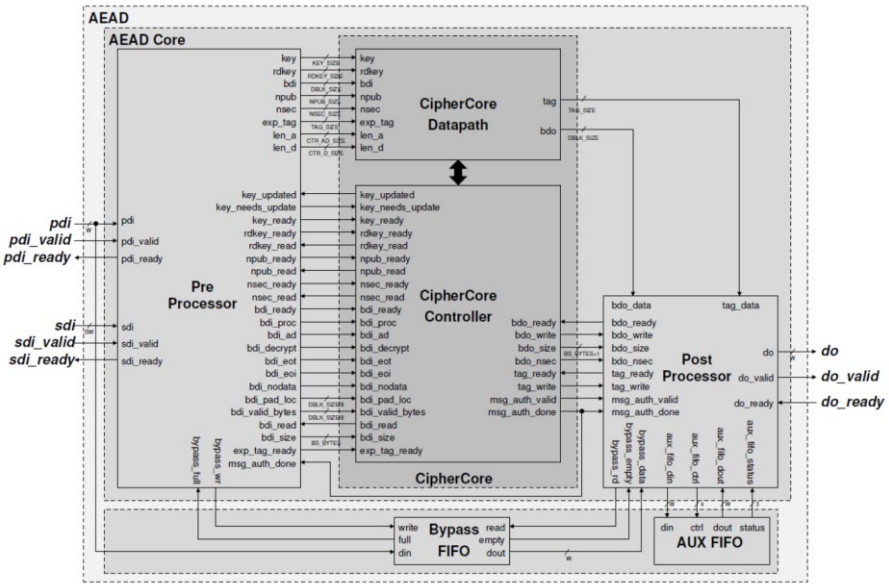


Fig. 4. Interfaz de comunicación para módulos de cifradores autenticadores suministrada por el proyecto ATHENa [26].

utilidades para unificar el proceso de generación de vectores de test. Para facilitar el diseño de la interfaz por parte de los autores de las propuestas, se suministró asimismo un paquete de desarrollo [27] que incluía la descripción HDL de los módulos que la implementan, junto a las correspondientes instrucciones de uso [28].

Las herramientas proporcionadas por el proyecto ATHENa facilitan en gran medida la comparación de implementaciones hardware de los cifradores autenticadores presentados a la Competición CAESAR. Por este motivo han sido utilizadas repetidamente, tanto para obtener los datos de rendimiento, ocupación de recursos y consumo de potencia que permiten comparar las distintas propuestas presentadas al certamen [29] [30], como para llevar a cabo estudios más específicos para, por ejemplo, analizar la incidencia del tipo de interfaz utilizada por un cifrador [31] o la resistencia de determinadas implementaciones frente a ataques de canal lateral [32].

Todos los trabajos anteriores se limitan, sin embargo, a evaluar las características de las implementaciones hardware de los cifradores sin considerar su posible participación como bloque funcional en el contexto de un System-on-Chip. Esta perspectiva, que coincide con el planteamiento seguido en el presente artículo, es igualmente la base de la propuesta descrita en [33], donde también se utiliza una placa de desarrollo basada en un dispositivo Zynq-7000 de Xilinx. El soporte de aplicaciones software basadas en Python limita, en ese caso, el esquema utilizado para interconectar el módulo del cifrador y el sistema de procesado del dispositivo, mientras que la aproximación seguida en el presente trabajo permite utilizar otros esquemas de interconexión alternativos para alcanzar distintos compromisos coste/rendimiento.

Como se muestra en la Fig. 4, la interfaz de comunicación proporcionada en la Competición CAESAR incluye una etapa de pre-procesado, encargada del análisis de las cabeceras de la información que llega a través de los buses de datos de entrada públicos, PDI, y secretos, SDI, la carga y activación de las claves, la conversión serie-paralelo de los bloques de datos de entrada, el relleno de los bloques de entrada y el control del número de bytes de datos que quedan por procesar. También incluye una etapa de post-procesado encargada de proporcionar la salida del cifrador, a través del bus de datos de salida, DO, llevando a cabo las tareas de eliminación de los datos no correspondientes al texto cifrado o al texto plano, conversión paralelo-serie de los bloques de salida en palabras de datos, transformación de dichas palabras en segmentos de datos, almacenamiento de los bloques descifrados en una FIFO auxiliar hasta que el resultado de la autenticación sea conocido y, por último, generación del bloque de estado con el resultado de la autenticación.

El hecho de que todos los candidatos de la Competición CAESAR presenten la misma interfaz externa y admitan un formato de entrada unificado hace posible que la metodología de desarrollo descrita en los siguientes apartados pueda ser aplicada para facilitar la incorporación de cualquiera de los diseños en sistemas empotrados construidos sobre dispositivos SoC de Xilinx. Todas las etapas de síntesis y verificación se han llevado a cabo con las herramientas de la versión 2018.2 del entorno de diseño Vivado [34].

A. Descripción HDL y Verificación del Cifrador

El primer paso del flujo de desarrollo seguido consiste en incluir la descripción HDL del cifrador en el entorno de desarrollo Vivado y verificar su comportamiento funcional mediante simulación. Para ello se emplean las fuentes suministradas por los autores, que incluyen las descripciones de los bloques pre- y post-procesador proporcionados por el Proyecto ATHENa, junto a un fichero de testbench también disponible en la hardware API (*Application Programming Interface*) proporcionada a los participantes en la competición. El testbench es auto-verificable; toma los datos públicos y

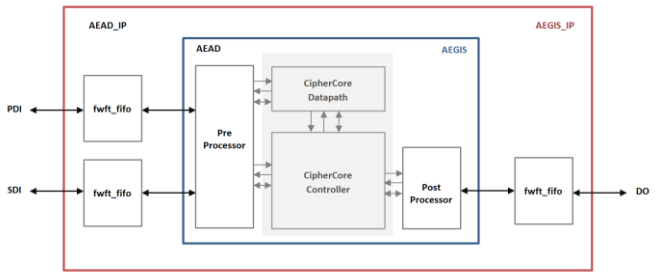


Fig. 5. Inserción de FIFOs en los puertos de entrada y de salida del cifrador para convertirlo en módulo IP.

privados correspondientes a un determinado test de los ficheros "pdi.txt" y "sdi.txt" y comprueba que la salida del diseño coincide con los datos esperados contenidos en "do.txt". Los ficheros ".txt" son generados por la herramienta aeadtvgen (*Authenticated Encryption Test Vector Generation* script), que genera un conjunto completo de vectores de test para cada candidato de CAESAR a partir del código C de referencia de este candidato y los parámetros proporcionados por el usuario. Con idea de comparar bajo condiciones de uso similares las distintas implementaciones llevadas a cabo en este trabajo, tanto las descripciones VHDL como los scripts de generación de vectores de test han sido configurados para que proporcionaran datos de 32 bits.

B. Generación del Módulo IP

Una vez verificado su comportamiento, para encapsular el diseño como un módulo IP que pueda ser incorporado en un sistema empotrado sobre dispositivos programables de Xilinx, es necesario dotar a los puertos de entrada y salida del cifrador de una serie de interfaces que faciliten la conexión del módulo IP a los buses estándar soportados por el procesador ARM disponible en los dispositivos Zynq-7000 o el procesador MicroBlaze que puede ser incluido como soft-core en las FPGAs de este fabricante.

Como ilustra la Fig. 5, las entradas y salidas del cifrador se conectan a través de memorias FIFO, empleando para ello los bloques utilizados previamente en el testbench del hardware API proporcionado por ATHENa. Como estándar de conexión asociado a estas FIFOs se usará el bus AXI4-Stream. Tras emplear la herramienta para crear y empaquetar módulos IP de Vivado, el nuevo diseño debe ser validado. La verificación se lleva a cabo modificando el testbench auto-verificable para que instancie el nuevo componente.

C. Conexión del Módulo IP al Procesador

Una vez generado el módulo IP e incluido en el catálogo de IPs de Vivado, el cifrador puede ser incluido en el diseño de un sistema empotrado, conectándolo como periférico del procesador utilizado (ARM o MicroBlaze). El intercambio de información entre el módulo-IP y el procesador puede llevarse a cabo a través de las interfaces AXI4-Stream utilizando diferentes mecanismos. Uno de ellos consiste en enviar y recibir los datos directamente, a través del bus AXI4 o AXI4-Lite, utilizando para ello el módulo IP denominado "AXI-Stream FIFO". Cuando se usa la otra alternativa, el procesador controla la operación de un IP denominado "AXI DMA (*Direct Memory Access*)" que permite el intercambio directo de información entre el cifrador y la memoria del sistema. Las dos opciones se detallan en los apartados siguientes.

1) Conexión mediante FIFOs:

El sistema empotrado que se muestra en la Fig. 6 utiliza dos bloques "AXI-Stream FIFO" para interconectar el módulo IP y el procesador ARM. Uno de ellos es el encargado de la transmisión de datos secretos, SDI, mientras que el otro se encarga de la transmisión de los datos públicos, PDI, y la recepción de los datos de salida, DO.

La interfaz de configuración del bloque "AXI-Stream FIFO" permite seleccionar qué tipo de bus será utilizado para conectarlo al procesador. Debido al reducido volumen de información que es necesario transmitir a través suya, para el bloque conectado al bus de datos de entrada secretos se ha utilizado en todos los casos una interfaz AXI4-Lite. Sin

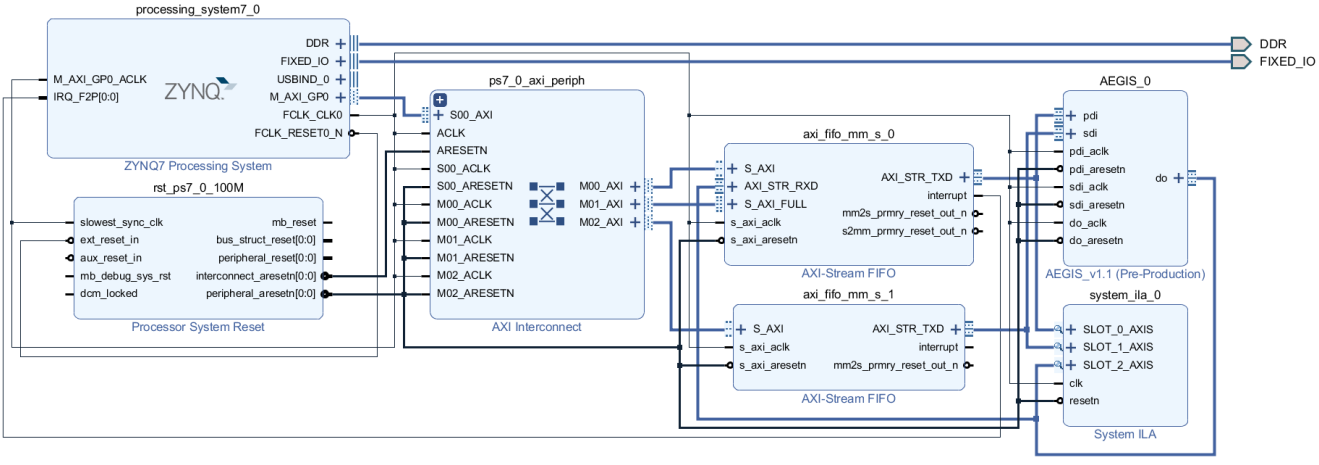


Fig. 6. Incorporación del módulo IP del cifrador AEGIS_v1.1 en el sistema empotrado utilizando bloques AXI-Stream FIFOs conectados al procesador a través de buses AXI4 (PDI y DO) y AXI4-Lite (SDI).

embargo, para las FIFOs asociadas a los datos públicos de entrada y los datos de salida puede emplearse un bus AXI4-Full con idea de conseguir mejores prestaciones. El diagrama de la Fig. 6 corresponde a esta última opción. Cuando se utiliza un bus AXI4-Lite el diagrama de bloques es similar, salvo que no aparece la entrada S_AXI_FULL en el bloque "AXI-Stream FIFO" superior y que el bloque "AXI Interconnect" presenta solo dos conexiones de salida.

El diagrama de bloques del sistema empotrado se completa con un módulo "System ILA (*Integrated Logic Analyzer*)" incluido para monitorizar los buses del módulo IP.

2) Conexión mediante DMA:

El bloque "AXI DMA" proporciona un acceso directo de ancho de banda elevado entre la memoria y los periféricos de destino a través de buses AXI4-Stream. Sus capacidades opcionales de recopilación de dispersión también descargan las tareas de movimiento de datos a los procesadores. Los registros de inicialización, estado y gestión son accedidos a través de una interfaz esclava AXI4-Lite. Este componente es usado en el sistema empotrado de la Fig. 7 para facilitar la transmisión directa, desde y hacia la memoria, de los datos públicos de entrada y los datos de salida intercambiados con el cifrador. El sistema empotrado incorpora en este caso otro componente, denominado "AXI Smart Connect", que se encarga de realizar la conexión maestro-esclavo entre el bloque "AXI DMA" y los buses de acceso del procesador y la memoria de la placa de desarrollo. Por último, el diagrama incluye un bloque "Concat" para conectar al procesador las dos señales de interrupción generadas por las etapas de transmisión y recepción del bloque "AXI DMA".

Para cualquiera de los sistemas empotrados anteriores, el flujo de desarrollo continúa con la validación del diseño y la generación de un envoltorio HDL, para llevar a cabo las etapas de síntesis, implementación y generación del bitstream que define la plataforma hardware que será exportada al entorno SDK (*Software Development Kit*) para el desarrollo de aplicaciones [35].

D. Desarrollo del Software del Sistema Empotrado

Para comprobar la funcionalidad de los diseños y comparar sus prestaciones se han seguido tres estrategias distintas: la primera consiste en ejecutar el código C correspondiente al software de referencia del cifrador en el procesador ARM del sistema empotrado. Las otras dos estrategias están dirigidas a verificar el funcionamiento de las implementaciones hardware del cifrador, utilizando para ello técnicas basadas en consultas sucesivas (*polling*) y en interrupciones del flujo de ejecución de los programas de test.

Tras exportar la plataforma hardware correspondiente al sistema empotrado a SDK, las herramientas de este entorno de programación basado en Eclipse facilitan la creación de aplicaciones integradas en los procesadores de Xilinx. La primera fase de esta tarea consiste en la generación de un paquete básico de soporte (BSP, *Board Support Package*) que proporciona una serie de rutinas básicas para manejar los distintos elementos del sistema empotrado, e incluye los drivers generados por el asistente utilizado para convertir el cifrador en un módulo IP.

La base de todos los programas ejecutados para verificar la funcionalidad y comparar las distintas opciones de diseño es común y reproduce el esquema utilizado por los testbench empleados parar verificar las implementaciones hardware. No obstante, se han tenido que realizar algunos cambios para adaptarlos a las características de los sistemas empotrados. En primer lugar, al carecer de soporte para utilizar un sistema de ficheros, fue necesario sustituir los ficheros ".txt" que contienen las entradas y salidas esperadas del cifrador por un fichero de cabecera, ".h", que proporciona estos datos al programa en la fase de compilación de la aplicación. Otra modificación obligada consistió en incluir las funciones necesarias para transmitir y recibir los datos del cifrador a

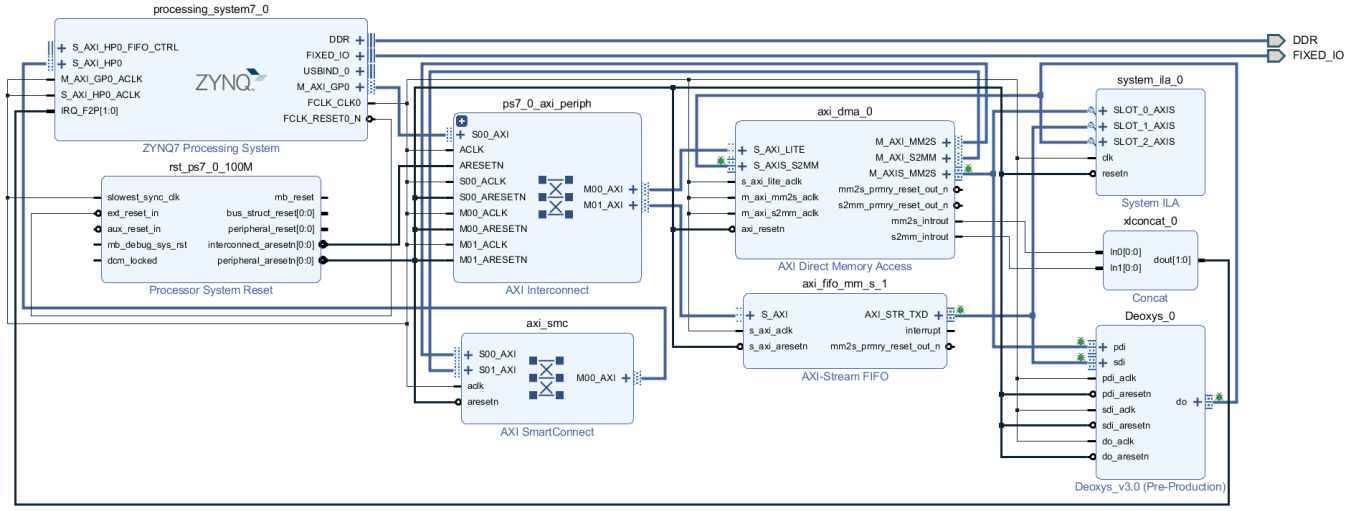


Fig. 7. Incorporación del módulo IP del cifrador DEOXYS_v3.0 en el sistema empotrado utilizando un bloque AXI-Stream FIFO conectado al procesador a través del bus AXI4-Lite (SDI) y acceso directo a memoria mediante un bloque AXI DMA (PDI y DO).

través de los elementos de conexión utilizados en cada caso (FIFOs y/o DMA) y de acuerdo con la estrategia utilizada (interrupciones o *polling*). Por último, para facilitar la comparación de las distintas opciones bajo diferentes condiciones de operación, se incorporaron rutinas de medida de tiempos de ejecución y se incluyó código para permitir la selección del número de veces que se ejecuta la secuencia de test y la información generada durante la ejecución de los programas.

La Fig. 8 ilustra la secuencia de tareas correspondiente a la aplicación software empleada para verificar los sistemas empotrados que utilizan bloques "AXI FIFO" para conectar la implementación hardware del cifrador al procesador ARM. Para los sistemas que utilizan el bloque "AXI DMA" la estructura del programa es similar, variando solo los pasos 1 y 5, que en este caso son:

1. Initialize FIFO for SDI and DMA for PDI and DO
5. Transmit secret data; Start transmission of public data and reception of output data

```
1.  Initialize FIFOs for SDI, PDI and DO
2.  For round=1 to number of rounds
3.    Get public, secret and data streams
4.    For each single test in the test vector
5.      Transmit secret data; Transmit public data
6.        Wait output data ( waiting loop / interruption from receiver)
7.      Receive output data
8.      Compare output data
9.    End test
10  End round
11. Print results
```

Fig. 8. Secuencia de tareas del programa utilizado para verificar los sistemas empotrados que utilizan FIFOs para conectar el módulo IP del cifrador.

IV. RESULTADOS

Las herramientas de síntesis e implementación del entorno Vivado pueden ser configuradas de acuerdo con distintas estrategias que permiten obtener soluciones especialmente adaptadas para cumplir un determinado objetivo de diseño: consumo de recursos, velocidad de operación, tiempo de desarrollo, etc., así como establecer compromisos adecuados entre estos objetivos. Los resultados post-implementación hardware de los tres cifradores analizados en el trabajo cuando se utilizan las estrategias de síntesis e implementación por defecto se muestran en la Tabla I, mientras que la Tabla II recoge los recursos consumidos por cada uno de los diseño tras ser convertido en módulo IP. Como puede observarse en la Tabla I, el núcleo básico de la variante implementada del cifrador ASCON (*Lightweight*) utiliza el 43% de las LUTs y el 56% de los Flip-Flops empleados por DEOXYS (*Defense in depth*) y el 19% de las LUTs y el 42% de los Flip-Flops requeridos por AEGIS (*High-performance*).

La incorporación de los módulos pre- y post-procesador proporcionados por la API de ATHENa, junto a las memorias FIFO necesarias para convertir el cifrador en un módulo-IP, suponen un incremento neto de algo más de 1.000 LUTs y 100 Flip-Flops, como puede apreciarse en la Tabla II.

TABLA I
RECURSOS DE LA FPGA CONSUMIDOS POR LOS DISTINTOS CIFRADORES.

	ASCON	AEGIS	DEOXYS
LUT	1.298	7.246	2.994
LUTRAM	22	22	298
FF	897	2.118	1.589
IO	5	5	5
BUFG	1	1	1

TABLA II
RECURSOS CONSUMIDOS POR LOS MÓDULOS IP DE LOS CIFRADORES.

	ASCON	AEGIS	DEOXYS
LUT	2.450	8.373	4.080
LUTRAM	790	790	1.046
FF	971	2.188	1.664
IO	104	104	104
BUFG	1	1	1

La Tabla III muestra los recursos de la FPGA consumidos por los sistemas empotrados que incorporan el módulo IP del cifrador ASCON con diferentes esquemas de conexión (FIFOs con AXI4-Lite, FIFOs con AXI4-Full y DMA). Como puede observarse, tanto el número de LUTs como el de elementos de memoria ha sufrido un incremento considerable, si bien es necesario resaltar que gran parte de este incremento (1.200 LUTs y 2.000 Flip-Flops, junto a la mitad de los bloques BRAM que aparecen en la tabla, son utilizados por el bloque "System-ILA" introducido para facilitar la depuración hardware de los diferentes diseños. El incremento en el consumo de recursos es similar, en términos absolutos, para los otros cifradores considerados en nuestro estudio. Al comparar el consumo de recursos de cada uno de los esquemas de conexión puede observarse que las dos opciones que emplean FIFOs tienen un coste muy similar (solo un 0,7% superior cuando la FIFO que realiza la transferencia de datos públicos hacia y desde el cifrador se conecta al procesador a través de un bus AXI4 con total funcionalidad), mientras que la opción que emplea DMA requiere un 4% más de LUTs y Flip-Flops (referidos al número total de recursos de cada tipo disponibles en el dispositivo FPGA).

Los resultados obtenidos son en todos los casos inferiores a los correspondientes al esquema de interconexión empleado en [33], que usa módulos DMA para transmitir tanto los datos públicos como los secretos, ocupando el 17,14% de los LUTs, el 10,25% de las LUTRAMs y el 9,84% de los FFs disponibles en el dispositivo.

TABLA III
RECURSOS DE LA FPGA CONSUMIDOS POR SISTEMAS EMPOTRADOS QUE UTILIZAN EL CIFRADOR ASCON CON DIFERENTES ESQUEMAS DE CONEXIÓN.

	FIFO AXI4-LITE		FIFO AXI4-FULL		DMA	
LUT	5.494	10,3 %	5.872	11,0 %	8.349	15,7 %
LUTRAM	1.154	6,6 %	1.163	6,7 %	1.630	9,4 %
FF	5.386	5,1 %	5.524	5,2 %	9.498	9,0 %
BRAM	6,50	4,6 %	6,50	4,6 %	6,50	4,6 %
BUFG	2	6,2 %	2	6,2 %	2	6,2 %

Las herramientas del entorno Vivado permiten evaluar el consumo energético asociado a los diferentes componentes de un diseño. La Tabla IV recoge los datos correspondientes a los tres cifradores considerados en este trabajo. En todos los casos las estimaciones fueron obtenidas para el dispositivo incluido en la placa de desarrollo considerando una temperatura ambiente de 25ºC y medidas de actividad de nodos recogidas en un fichero SAIF (*Switching Activity Interchange format*), generado mediante simulación funcional post-implementación de los diferentes diseños, usando los mismos patrones de test empleados para verificar su funcionalidad.

La primera fila de la tabla muestra el consumo total de cada diseño, incluyendo tanto la contribución estática como la dinámica. El valor absoluto de esta última, junto al porcentaje que supone del consumo energético total, se muestran en la segunda fila. Estas cantidades incluyen la implementación de los cifradores (que se detallan en la tercera fila de la tabla) más la de los circuitos de interfaz.

TABLA IV
CONSUMO ENERGÉTICO DE LOS DISTINTOS CIFRADORES (en vatios).

	ASCON	AEGIS	DEOXYS
Total on-chip	0,132	0,191	0,176
C. Dinámica	0,027 (21%)	0,085 (45%)	0,070 (40%)
➢ Cifrador	0,023 (17%)	0,079 (42%)	0,063 (36%)

La Tabla V muestra los datos correspondientes a sistemas empotrados que incorporan módulos IP de los distintos cifradores usando un esquema de conexión basado en bloques de acceso directo a memoria (los resultados para los otros esquemas de conexión son muy parecidos). La tabla muestra una nueva fila que corresponde al consumo energético del sistema de procesado incluido en el dispositivo programable, idéntico en todos los diseños.

TABLA V
CONSUMO ENERGÉTICO DE SISTEMAS EMPOTRADOS CON DISTINTOS CIFRADORES Y ESQUEMAS DE CONEXIÓN MEDIANTE DMA (en vatios).

	ASCON	AEGIS	DEOXYS
Total on-chip	1,622	1,657	1,671
C. Dinámica	1,482 (91%)	1,516 (91%)	1,523 (91%)
➢ Procesador	1,405 (87%)	1,405 (85%)	1,405 (84%)
➢ Cifrador	0,024 (1%)	0,060 (4%)	0,054 (3%)

Los valores de consumo asociados a los cifradores son en este caso menos fiables porque corresponden a una estimación probabilística en lugar de emplear vectores de test reales. No obstante, los datos que aparecen en ambas tablas permiten categorizar y comparar las distintas propuestas. De acuerdo con la Tabla IV, el consumo energético del cifrador ASCON (*Lightweight*) es el 36% del de DEOXYS (*Defense in depth*) y el 30% del de AEGIS (*High-performance*). Por otra parte, como se pone de manifiesto en la Tabla V, la contribución de los bloques de cifrado a la factura energética del sistema empotrado es, en todos los casos, inferior al 3%.

La disponibilidad de un procesador de elevadas prestaciones en los dispositivos de la serie Zynq-7000 de Xilinx permite, no solo utilizar este elemento para controlar la operación de los cifradores implementados en hardware sobre la lógica programable, sino también ejecutar sobre el propio procesador el software de referencia suministrado por los diseñadores de los algoritmos de cifrado y autenticación presentados a la competición CAESAR.

Los datos que aparecen en las cuatro primeras filas de la Tabla VI corresponden a los tiempos de ejecución sobre el procesador ARM de las versiones software de referencia de las variantes implementadas de los cifradores ASCON, AEGIS y DEOXYS, utilizando un vector de test que incluye las operaciones de cifrado y descifrado para un caso con longitud de mensaje 750 bytes y longitud de datos asociados de 100 bytes. Al analizar las distintas columnas de la tabla se observa la tremenda incidencia (tiempos entre 16 y 20 veces menores) que supone la utilización de las caches de instrucciones y datos de que dispone el procesador ARM. Por otra parte, al examinar las distintas filas puede apreciarse la diferencia de rendimiento de los distintos cifradores. En este caso, AEGIS ejecuta el test 5 veces más rápido que ASCON y 17 veces más rápido que DEOXYS. Esta relación, sin embargo, no se mantiene en los datos que aparecen en la quinta fila de la tabla, que muestra los tiempos de ejecución de los mismos vectores de test sobre sistemas empotrados que utilizan las variantes implementadas de cada uno de los cifradores, empleando el esquema de conexión basado en FIFOs y buses AXI4-Full y usando una estrategia de interrupciones en el programa de verificación. Los resultados obtenidos ponen de manifiesto que, para la secuencia de test utilizada, los cifradores ASCON y DEOXYS presentan un comportamiento temporal similar, mientras que AEGIS invierte casi el doble de tiempo en concluir las operaciones de cifrado y descifrado.

La última fila de la tabla muestra la aceleración obtenida por las implementaciones hardware de los distintos cifradores cuando se las compara con las más rápidas de las ejecuciones en software. Mientras que los tiempos de respuesta de las alternativas hardware y software son muy parecidos en el caso del cifrador AEGIS (implementado con buses de datos de 32 bits), los factores de aceleración de las versiones hardware de DEOXYS Y ASCON toman valores de 10 y 31, respectivamente.

TABLA VI
TIEMPOS DE EJECUCIÓN EN EL SISTEMA EMPOTRADO DE LAS VERSIONES SOFTWARE DE REFERENCIA DE LOS DISTINTOS CIFRADORES Y ACELERACIÓN OBTENIDA CON LA IMPLEMENTACIÓN HARDWARE (en microsegundos).

	Cache Inst.	Cache Datos	ASCON	AEGIS	DEOXYS
SW	NO	NO	43.308,33	5.748,67	111.102,82
SW	SI	NO	34.266,40	4.881,04	105.251,72
SW	NO	SI	20.022,52	2503,87	76.447,00
SW	SI	SI	2.043,45	352,97	6.031,37
HW (FIFO AXI4)			197,34	355,44	195,64
Aceleración (HW/SW)			10	1	31

Vectores de test: cifrado y descifrado de un mensaje de 750 bytes de longitud y 100 bytes de datos asociados.

Para comprobar si los resultados anteriores están condicionados por los vectores de test empleados, se ha llevado a cabo, por último, una serie de pruebas con vectores de test, generados mediante la utilidad aeadtvgen, consistentes en 25 operaciones sucesivas de cifrado y descifrado con diferentes longitudes de mensajes y datos asociados.

Los resultados recogidos en la Tabla VII ponen de manifiesto que, con independencia de la estrategia seguida para realizar el test, los sistemas basados en FIFOs mantienen una tendencia similar a la anterior, mientras que el comportamiento de todos los diseños es similar cuando se emplea DMA.

TABLA VII
TIEMPOS DE EJECUCIÓN DE PROGRAMAS DE TEST BASADOS EN ESTRATEGIAS DE POLLING E INTERRUPCIONES EN SISTEMAS EMPOTRADOS QUE UTILIZAN MÓDULOS IP DE LOS DISTINTOS CIFRADORES (en microsegundos)

Polling	ASCON	AEGIS	DEOXYS
FIFO AXI4-Lite	449,52	723,55	467,52
FIFO AXI4-Full	399,50	637,24	416,38
DMA	156,07	158,94	159,62
Interrupciones	ASCON	AEGIS	DEOXYS
FIFO AXI4-Lite	503,40	775,58	519,55
FIFO AXI4-Full	450,60	683,95	463,89
DMA	164,86	158,44	151,36

Vectores de test: 25 operaciones sucesivas de cifrado y descifrado con diferentes longitudes de mensajes y datos asociados.

V. CONCLUSIONES

El uso cada día más importante de sistemas empotrados para aplicaciones que requieren elevados niveles de privacidad y autenticidad de la información que manejan ha motivado un creciente interés por los aspectos de seguridad en este tipo de sistemas.

Este trabajo describe un flujo de diseño que facilita el desarrollo y evaluación de distintas soluciones para la implementación hardware de cifradores autenticadores y su incorporación como periféricos aceleradores en sistemas empotrados para distintos casos de aplicación.

Las implementaciones de los cifradores como periféricos de los procesadores ARM o MicroBlaze, disponibles en los dispositivos FPGA y SoCs de Xilinx, utilizan buses AXI4-Stream, compatibles con las líneas de datos y señales de protocolos definidos en la interfaz proporcionada por los organizadores de la Competición CAESAR para los cifradores presentados a la misma. Esta característica asegura que las soluciones aportadas en este trabajo son directamente aplicables para la implementación de cualquiera de los cifradores participantes en el certamen.

Los datos de ocupación de recursos, consumo energético y velocidad de operación de los tres cifradores analizados responden a las expectativas de las distintas categorías a las que fueron presentadas dentro de la competición (*Lightweight, High-performance* y *Defense in depth*). No obstante, cuando los cifradores son convertidos a módulos IP e incorporados en un sistema empotrado es preciso considerar otros aspectos del diseño, como la infraestructura de conexión empleada.

Para llevar a cabo la interconexión de los módulos IP de los cifradores con el resto de los componentes del sistema empotrado se han explorado dos soluciones: una basada en el uso del módulo "AXI-Stream FIFO" y otra en el del bloque "AXI DMA". Las FIFOs empleadas en el primer caso pueden, a su vez, conectarse a la infraestructura de comunicaciones del sistema empotrado utilizando buses AXI4 o AXI4-Lite.

Los resultados de ocupación de recursos de la FPGA de las distintas alternativas, en combinación con los tiempos de ejecución de las diferentes opciones obtenidos por aplicaciones de verificación que usan técnicas de *polling* o de interrupciones, proporcionan las medidas necesarias para establecer compromisos coste/rendimiento adecuados para diferentes tipos de aplicaciones. En particular, las soluciones propuestas en el trabajo facilitarán el diseño de sistemas empotrados con módulos hardware específicos de cifrado y autenticación para implementar protocolos criptográficos de intercambio de información entre los dispositivos usados en IoT e Industria 4.0.

Finalmente, como líneas de progresión del trabajo merece la pena señalar la extensión del flujo de diseño descrito a los cifradores propuestos en otros certámenes actualmente en curso, así como el desarrollo de librerías software que faciliten el uso de las versiones hardware de los cifradores en aplicaciones codificadas en lenguajes de programación de alto nivel como C o Python.

REFERENCIAS

[1] J. R. Almeida, J. B. Camargo, P. S. Cugnasca, "Safety and Security in Critical Applications and in Information Systems - a Comparative Study," *IEEE Latin America Transactions*, vol. 11, no. 4, pp. 1127-1133, June 2013, 10.1109/TLA.2013.6601759.

[2] CAESAR Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2012. [Online]. Disponible: http://competitions.cr.yp.to/caesar.html

[3] J. Daemen, V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, Springer Science & Business Media, 2013.

[4] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen, S. Shieh, "IoT Security: Ongoing Challenges and Research Opportunities," in *Proc. IEEE 7th International Conference on Service-Oriented Computing and Applications*, Matsue, 2014, 10.1109/SOCA.2014.58.

[5] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," in *Proc. 2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, Cluj-Napoca, 2014, 10.1109/AQTR.2014.6857843.

[6] L. Thames, D. Schaefer, *Cybersecurity for Industry 4.0*. Springer Series in Advanced Manufacturing. Springer, 2017. 10.1007/978-3-319-50660-9.

[7] B. Zhou, M. Egele, A. Joshi, "High-performance low-energy implementation of cryptographic algorithms on a programmable SoC for IoT devices," in *Proc. 2017 IEEE High Performance Extreme Computing Conference,* Waltham, 2017, 10.1109/HPEC.2017.8091062.

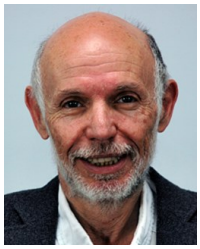
[8] R Cowart, D Coe, J Kulick,, A Milenković, "An Implementation and Experimental Evaluation of Hardware Accelerated Ciphers in All-Programmable SoCs," in *Proc. of the SouthEast Conference. ACM*, Kennesaw 2017, 10.1145/3077286.3077297.
[9] N. Jacob et al., "Securing FPGA SoC configurations independent of their manufacturers," In *Proc. 30th IEEE International System-on-Chip Conference*, Munich, 2017, 10.1109/SOCC.2017.8226019.
[10] A. Silitonga, Z. Jiang, N. Khan, J. Becker, "Reconfigurable Module of Multi-mode AES Cryptographic Algorithms for AP SoCs," in *Proc, 2019 IEEE Nordic Circuits and Systems Conference*, Helsinki, 2019, 10.1109/NORCHIP.2019.8906923.
[11] I. G. Agramont, H. Calderón, "Evaluation of Cryptographic Algorithms over an All Programmable SoC (AP SoC) Device," *Journal on Systemics, Cybernetics and Informatics*, vol. 15, no. 2, pp. 84-91, 2017.
[12] National Institute of Standards and Technology "FIPS-46:(1979): Data Encryption Standard (DES)," Revisión 46-3 (1999) disponible en: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.
[13] R. Rivest, A. Shamir, L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21 no. 2, pp.120–126. 1978, 10.1145/359340.359342.
[14] M. Grassl, B. Langenberg, M. Roetteler, R. Steinwandt, "Applying Grover's Algorithm to AES: Quantum Resource Estimates," in: Takagi T. (eds) Post-Quantum Cryptography 2016. *Lecture Notes in Computer Science*, vol 9606. Springer, Cham. 10.1007/978-3-319-29360-8_3.
[15] J. Xie, K. Basu, K. Gaj, U. Guin "Special Session: The Recent Advance in Hardware Implementation of Post-Quantum Cryptography," in *Proc. 2020 IEEE 38th VLSI Test Symposium*, San Diego, 2020, pp. 1-10, 10.1109/VTS48691.2020.9107585.
[16] GMU Source Code of Round 3 & Round 2 CAESAR Candidates, AES-GCM, AES, and Keccak Permutation F, [Online]. Disponible: https://cryptography.gmu.edu/athena/index.php?id=CAESAR_source_codes.
[17] K. Gaj, J.-P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, B. Y. Brewster, "ATHENa – automated tool for hardware evaluation: Toward fair and comprehensive benchmarking of cryptographic hardware using FPGAs," in *Proc. 20th International Conference on Field Programmable Logic and Applications*, 2010. pp. 414–421.
[18] C. Dobraunig, M. Eichlseder, F Mendel, M. Schläffer, "Ascon v1.2, Submission to the CAESAR Competition," Sep. 2016. [Online]. Disponible: http://competitions.cr.yp.to/round3/asconv12.pdf.
[19] H. Wu, "ACORN: A Lightweight Authenticated Cipher (v3)," Sep. 2016. [Online]. Disponible: https://competitions.cr.yp.to/round3/acornv3.pdf.
[20] H. Wu, B. Preneel, "AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)," Sep. 2016. [Online]. Disponible: http://competitions.cr.yp.to/round3/aegisv11.pdf.
[21] T. Krovetz, P. Rogaway, "OCB (v1.1)," Sep. 2016. [Online]. Disponible: http://competitions.cr.yp.to/round3/ocbv11.pdf.
[22] J. Jean, I. Nikolic, T. Peyrin, Y. Seurin, "Deoxys v1.41," Sep. 2016. [Online]. Disp.: http://competitions.cr.yp.to/round3/deoxysv141.pdf.
[23] E. Andreeva el al., "COLM v1," Sep. 2016, [Online]. Disponible: http://competitions.cr.yp.to/round3/colmv1.pdf.
[24] Digilent, "Zybo Z7: Zynq-7000 ARM/FPGA SoC Development Board," [Online]. Disponible: https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/start
[25] George Mason University, "ATHENA: Automated Tool for Hardware EvaluatioN," 2010. [Online]. Disponible: https://cryptography.gmu.edu/athena/index.php?id=ATHENa
[26] E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, P. Yalla, J. P. Kaps, K. Gaj, "CAESAR Hardware API," Cryptology ePrint Archive 2016/626, 2016. [Online]. Disponible: http://eprint.iacr.org/2016/626.pdf
[27] George Mason University: "Development Package for Hardware Implementations Compliant with the CAESAR Hardware API, v2.0," 2017. [Online]. Disponible: https://cryptography.gmu.edu/athena/index.php?id=CAESAR
[28] E. Homsirikamol, P. Yalla, F. Farahmand, W. Diehl, A. Ferozpuri, J.-P. Kaps, K. Gaj, "Implementer's Guide to Hardware Implementations Compliant with the CAESAR Hardware API," George Mason University, Fairfax, VA, GMU Report, 2016.
[29] Cryptographic Engineering Research Group at GMU, "GMU ATHENa Database of Results," Dec. 2017. [Online]. Available: https://cryptography.gmu.edu/athenadb/fpga_auth_cipher/rankings_view
[30] Cryptographic Engineering Research Group at GMU, "Hardware Benchmarking of CAESAR Candidates," 2019. [Online]. Available: https://cryptography.gmu.edu/athena/index.php?id=CAESAR,.
[31] F. Farahmand, W. Diehl, A. Abdulgadir, J.-P. Kaps, K. Gaj, "Improved Lightweight Implementations of CAESAR Authenticated Ciphers," in *Proc. 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines*, Boulder, CO, Apr. 2018.
[32] W. Diehl, F. Farahmand, A. Abdulgadir, J. P. Kaps, K. Gaj, "Face-off between the CAESAR Lightweight Finalists: ACORN vs. Ascon," In *Proc. International Conference on Field Programmable Technology* (ICFPT), Naha, Japan. Dec. 2018.
[33] M. Tempelmeier, F. De Santis, G. Sigl, J.-P. Kaps, "The CAESAR-API in the real world — Towards a fair evaluation of hardware CAESAR candidates," in *Proc. 2018 IEEE International Symposium on Hardware Oriented Security and Trust*, Washington, DC, Apr. 2018.
[34] Xilinx, "Vivado Design Suite User Guide v. 2018.2, " Jul. 2018. [Online]. Disponible: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug973-vivado-release-notes-install-license.pdf
[35] Xilinx, "Xilinx Software Development Kit (XSDK) 2018.2. " [Online]. Disponible: https://www.xilinx.com/products/design-tools/embedded-software/sdk.html

Macarena C. Martínez-Rodríguez, recibió el grado en Ingeniería Informática, el grado en Ingeniería Electrónica (con honores), el Máster en Microelectrónica y el doctorado en Ciencias y Tecnologías Físicas por la Universidad de Sevilla, España, en 2007, 2010, 2012 y 2018, respectivamente. Desde 2010, trabaja en el Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC, Universidad de Sevilla). Su principal interés de investigación se centra en el desarrollo de circuitos digitales eficientes para implementar funciones afines a tramos y el diseño microelectrónico de sistemas criptobiométricos, sensores y controladores virtuales confiables.

Sergio Sauro del Valle, recibió el grado en Ingeniería Electrónica Industrial por la Universidad de Sevilla, España, en 2019. Su interés de investigación se centra en la implementación y comparación de algoritmos de cifrado sobre sistemas empotrados. Desde 2019 trabaja en Babel Sistemas de la Información donde desempeña funciones de programación y homologación de dispositivos electrónicos

Piedad Brox Jiménez, Doctora en Física por la Universidad de Sevilla en 2009, Científica Titular del CSIC, adscrita al Instituto de Microelectrónica de Sevilla, IMSE-CNM, (CSIC, Universidad de Sevilla), Sevilla, España. Sus actividades de investigación incluyen el diseño e implementación de sistemas neurodifusos y su aplicación al procesado digital de imagen y vídeo, diseño de sistemas de visión empotrados, diseño de circuitos digitales de altas prestaciones incluyendo dispositivos FPGA y ASIC para aplicaciones de procesado de

imagen y vídeo, criptografía, biometría, controladores empotrados y sensado virtual.



Santiago Sánchez Solano, Doctor en Ciencias Físicas por la Universidad de Sevilla en 1990, Investigador Científico del CSIC adscrito al Instituto de Microelectrónica de Sevilla, IMSE-CNM, (CSIC, Universidad de Sevilla), Sevilla, España. Sus líneas de investigación se centran en el desarrollo de sistemas empotrados con componentes hardware/software sobre FPGA y la realización microelectrónica de sistemas neurodifusos, así como sus aplicaciones en robótica, procesado de imágenes, seguridad y redes de sensores inteligentes.