ANALES DE LA
Academia de Ciencias de Cuba

CIENCIAS TÉCNICAS

Artículo original de investigación

Contribuciones a la implementación de sistemas electrónicos digitales embebidos sobre *hardware* reconfigurable

Alejandro José Cabrera Sarmiento [1*] https://orcid.org/0000-0001-6710-163X
Luís Manuel Garcés Socarrás [1] https://orcid.org/0000-0002-8158-6423
Alejandro Cabrera Aldaya [1] https://orcid.org/0000-0002-1544-6772
Raudel Cuiman Márquez [1] https://orcid.org/0000-0002-5145-7612
Santiago Sánchez Solano [2] https://orcid.org/0000-0002-0700-0447
Piedad Brox Jiménez [2] https://orcid.org/0000-0003-1059-5338
Egidio Ieno Junior [3] https://orcid.org/0000-0001-8502-4201
Tales Cleber Pimenta [3] https://orcid.org/0000-0002-2791-7332
Billy Bob Brumley [4] https://orcid.org/0000-0001-9160-0463

[1] *Universidad Tecnológica de La Habana. La Habana, Cuba*
[2] *Instituto de Microelectrónica de Sevilla. Sevilla, España*
[3] *Universidad Federal de Itajubá. Itajubá, Brasil*
[4] *Universidad de Tampere. Tampere, Finlandia*

**Autor de correspondencia:* alex@automática.cujae.edu.cu

RESUMEN

Introducción: Los sistemas electrónicos embebidos sobre dispositivos de ***hardware*** reconfigurable posibilitan la realización de implementaciones híbridas ***hardware*/*software*** que permiten acelerar mediante ***hardware*** las funciones de mayor latencia, por lo que constituye un campo de investigación actual el desarrollo de arquitecturas ***hardware*** parametrizables que posibiliten su utilización en diversas aplicaciones. El objetivo de este trabajo es presentar diferentes contribuciones en este campo relacionadas con la aceleración de funciones de procesamiento de imágenes y funciones criptográficas, así como en la detección de posibles vulnerabilidades ante diferentes tipos de ataques. **Métodos:** Se presentan diferentes metodologías de desarrollo dada la diversidad de las contribuciones que se exponen. Para la implementación de las funciones de procesamiento de imágenes se utilizó una metodología basada en modelos haciendo uso de la herramienta Xilinx System Generator junto con Matlab/Simulink así como un novedoso procedimiento de configuración automática de los componentes ***hardware*** de la biblioteca desarrollada; mientras que las implementaciones de las funciones criptográficas fueron desarrolladas en lenguaje de descripción de ***hardware***. Para la detección de las vulnerabilidades se utilizaron ataques de manipulación de contenidos y de canal colateral por consumo de potencia del tipo SPA. **Resultados:** Se expone la utilización de componentes ***hardware*** de la biblioteca desarrollada en una aplicación de identificación de matrículas de vehículos y de componentes ***hardware*** de funciones criptográficas, debidamente protegidas contra diferentes ataques, en el sistema criptográfico ARCANO, así como la validación de que el algoritmo binario de Euclides es vulnerable ante el conocimiento parcial de su flujo de ejecución. En Conclusiones se enfatizan las contribuciones fundamentales de las investigaciones realizadas, así como su utilidad práctica en aplicaciones reales.

Revisores [a]

Juan Valentín Lorenzo
Universidad Central Marta Abreu de Las Villas. Santa Clara, Cuba

Editor

Lisset González Navarro
Academia de Ciencias de Cuba. La Habana, Cuba

Traductor

Darwin A. Arduengo García
Academia de Ciencias de Cuba. La Habana, Cuba

a N. del E: En este apartado figuran los nombres de los árbitros que accedieron a revelar su identidad, como expresión de apertura progresiva del proceso de revisión por pares. No aparecen aquellos que optaron por el anonimato.

Palabras clave: sistemas embebidos; FPGA; SoC-FPGA; procesador de imágenes; funciones crip-tográficas; ataques de canal colateral

Contributions to the implementation of embedded digital electronic systems on reconfigurable *hardware*

ABSTRACT

Introduction: Embedded electronic systems on reconfigurable hardware devices make it possible to carry out hybrid hardware/software implementations that allow higher latency functions to be speeded up by hardware, which is why the development of parameterizable hardware architectures that allow their use in various applications is a current field of research. The objective of this work is to expose different contributions in this field related with the acceleration of image processing functions and cryptographic functions, as well as on the detection of possible vulnerabilities against different types of attacks. **Methods:** They are presented different development methodologies due to the diversity of the contributions that are staged. It was used for the implementation of the image processing functions, a model-based methodology, making use of the Xilinx System Generator tool together with Matlab/Simulink, as well as a new automatic configuration procedure for the hardware components of the developed library, while they were developed the implementations of the cryptographic functions in hardware description language. For the detection of vulnerabilities, they were used content manipulation and collateral channel attacks for power consumption of the SPA type. **Results**: The use of hardware components of the library developed in a vehicle license plate identification application and hardware components of cryptographic functions, duly protected against different attacks, in the ARCANO cryptographic system, as well as the validation that the Binary Euclidean Algorithm is vulnerable to partial knowledge of its execution flow. Conclusions, they are emphasized the fundamental contributions of the research results, as well as their practical utility in real applications.

Key words: embedded systems; FPGA; SoC-FPGA; image processing; cryptographic functions; side-channel attacks

INTRODUCCIÓN

Para el desarrollo de soluciones con limitaciones de volumen (peso), consumo de potencia o de costo, se requiere la utilización de sistemas electrónicos embebidos, como los basados en microcontroladores que, además de disponer de un sistema de procesamiento, también incluyen múltiples componentes para interactuar con el entorno.

Los sistemas de procesamiento presentes en los sistemas electrónicos embebidos, aunque con prestaciones muy inferiores a los disponibles en computadoras personales (PC), son capaces de ejecutar diferentes funciones *software* de un programa para una determinada aplicación. Sin embargo, existen aplicaciones (o funciones asociadas a estas) en las que estos sistemas de procesamiento no son capaces de satisfacer los requisitos de velocidad que demandan, entre las que se encuentran las relacionadas con la ejecución de algoritmos criptográficos que requieren procesar una gran cantidad de *bits*, estando limitadas por el ancho del *bus* de datos del procesador y de las operaciones disponibles en la unidad aritmética y lógica (ALU, por sus siglas en inglés); o las de procesamiento de imágenes en tiempo real, con gran independencia de datos (los *pixels* de la imagen) o de operaciones que no pueden ser aprovechadas, requiriendo desarrollar soluciones que logren acelerar la ejecución de aquellas funciones de mayor latencia.

La principal solución en estos casos radica en el desarrollo de arquitecturas que permitan alcanzar tiempos de respuesta que satisfagan los requerimientos de la aplicación; aprovechando el paralelismo inherente del *hardware* (conexiones eléctricas que se propagan simultáneamente), conjuntamente con la posibilidad de establecer rutas de datos de mayor tamaño (no limitadas al ancho del bus de datos del

sistema de procesamiento), de implementar operaciones específicas (no limitadas a las disponibles en la ALU) y de hacer modificaciones en el control de flujo de los datos (no limitadas a las instrucciones disponibles en el procesador).

El soporte ideal para estos desarrollos son los dispositivos de *hardware* reconfigurable como los FPGA (del inglés *Field Programmable Gate Array*) y los más recientes SoC-FPGA (del inglés *System on Chip-Field Programmable Gate Array*) dada su característica de que pueden realizarse cambios en sus conexiones internas para implementar una nueva funcionalidad, además de la diversidad de recursos *hardware* que incorporan. Este tipo de solución constituye la base de la denominada computación reconfigurable, paradigma establecido hace más de 6 décadas y que persigue acelerar la ejecución de determinadas funciones sobre *hardware*. [1,2]

Adicionalmente la amplia disponibilidad como módulos de propiedad intelectual (IP, por sus siglas en inglés) de los diversos componentes *hardware* de un sistema de procesamiento permite su inclusión, conjuntamente con las funcionalidades de *hardware* desarrolladas, en el mismo FPGA, facilitando el desarrollo de realizaciones híbridas *hardware/software* (HW/SW), en las que se implementan en *hardware* aquellas funciones que son críticas en tiempo, mientras las restantes se ejecutan por *software* en el sistema de procesamiento. En el caso de los SoC-FPGA ya incorporan, como parte del *hardware* del dispositivo, un potente sistema de procesamiento lo cual potencia el desarrollo de realizaciones híbridas HW/SW.

Desde el año 2003 el Grupo de Investigación de Sistemas Digitales Empotrados (GISDE) de la CUJAE ha sido promotor de la introducción en Cuba de las tecnologías avanzadas de diseño de sistemas electrónicos digitales embebidos en dispositivos de *hardware* reconfigurable, trabajando en su divulgación, capacitación y desarrollo de resultados de I+D+i. [3-5]

En esta propuesta se presentan 3 conjuntos de contribuciones que forman parte de las investigaciones desarrolladas por el GISDE en los últimos años en la temática de implementación de sistemas digitales embebidos sobre *hardware* reconfigurable. Estas contribuciones están relacionadas con el desarrollo y validación de arquitecturas *hardware* parametrizables que permitan acelerar funciones de procesamiento de imágenes y videos, así como funciones criptográficas debidamente protegidas ante diferentes tipos de ataques.

MÉTODOS

Dado que se presentan 3 conjuntos de contribuciones en las cuales se han utilizado diferentes metodologías de desarrollo, a continuación, se describen de forma independiente.

Metodología utilizada para la obtención de una biblioteca de componentes *hardware* para procesamiento de imágenes y video

Con anterioridad al desarrollo de esta contribución existían diseños hardware para implementar diferentes funciones de procesamiento de imágenes y videos. Sin embargo, todos presentaban diversas limitaciones entre las que resaltan la muy poca disponibilidad de componentes; carecer de opciones de configuración que permitan adaptar el diseño a los requisitos de la aplicación; o utilizar un flujo de diseño que requiere de un profundo conocimiento del diseño electrónico y esfuerzo por parte del desarrollador de la aplicación.

En base a lo anterior se adoptó seguir la metodología de implementación basada en modelos de MATLAB/Simulink (herramienta de amplio dominio por los desarrolladores de aplicaciones software de procesamiento de imágenes y video) en combinación con la herramienta Xilinx System Generator (XSG), encargada de trasladar a descripciones hardware los modelos de MATLAB/Simulink para ser implementados sobre un FPGA.

Con esto se alcanza un nivel de abstracción muy superior, liberando al desarrollador de la aplicación de los detalles específicos de las implementaciones hardware de cada uno de los componentes y reduciendo así el tiempo de desarrollo. La posibilidad de diseñar y simular el diseño, mediante la ejecución de MATLAB/Simulink en un PC, y su posterior implementación y verificación sobre un dispositivo de *hardware* reconfigurable, son algunas de las ventajas de esta metodología. Para ello XSG dispone de una biblioteca de modelos de componentes hardware básicos con los cuales se implementan componentes hardware de mayor complejidad.

Utilizando los componentes básicos de XSG se desarrolló una biblioteca de componentes hardware, denominada XIL XSGImgLib, la cual dispone de 54 componentes, altamente configurables, que facilitan la implementación hardware de algoritmos de procesamiento de imágenes o videos. [6-9] Estos componentes incluyen, entre otros, bloques de conversión de información de pixels a paralelo, bloques de control de sincronismo, bloques para implementar muy diversos filtros para procesamiento espacial, bloques de detección de movimiento, bloques de cálculo de histogramas, etc.

Dado que en el diseño hardware basado en modelos con XSG se trabaja con una representación gráfica o esquema de bloques de la arquitectura, es necesario "redibujar" este esquema para realizar modificaciones en la arquitectura *hardware*. El diseño de un bloque de procesamiento empleando *Simulink* se efectúa mediante la inserción y la unión de forma gráfica, por el diseñador, de módulos disponibles en las bibliotecas de la herramienta, generando un modelo almacenado en un

archivo (.mdl/slx). Dado que MATLAB incluye un conjunto de funciones que permiten obtener y establecer los parámetros de los componentes del bloque, así como desconectar, eliminar, interconectar y agregar diferentes módulos disponibles, se desarrollaron rutinas de código MATLAB para adquirir la posición y la orientación de los módulos que componen un bloque, así como su posterior reelaboración. Esto equivale a elaborar un nuevo esquema del bloque.

Como parte de las contribuciones se desarrolló un novedoso procedimiento para lograr la modificación automática de los bloques de la biblioteca XIL XSGImgLib, detallado en Garcés-Socarrás. [10-11] La metodología utilizada, ilustrada en la figura 1, consiste en desarrollar arquitecturas *hardware* generales para cada bloque (obteniendo así su correspondiente archivo .mdl/slx) y, en función de la configuración específica establecida para el bloque mediante sus máscaras de configuración y utilizando las rutinas de MATLAB desarrolladas (script de configuración), modificar el archivo .mdl/slx general obteniendo así un nuevo modelo (archivo MOD.mdl/slx) con su correspondiente representación gráfica. [10,11]

Debe destacarse que la velocidad de respuesta de cada uno de los componentes de la biblioteca *XIL XSGImgLib* por separado y de un diseño en su conjunto, al explotar el paralelismo del *hardware* y de la implementación específica, es muy superior al de las funciones *software* equivalentes. A ello contribuye que, en las arquitecturas de cada bloque, es posible establecer estructuras de segmentación (*pipeline*) que permiten almacenar los resultados intermedios de una etapa para ser procesados por la siguiente, lo cual incrementa la simultaneidad de la ejecución de las operaciones.

Metodología utilizada para la implementación de funciones criptográficas sobre FPGA y SoC-FPGA

La segunda contribución abarca el desarrollo de un conjunto de nuevas arquitecturas *hardware* para la implementación de diferentes funciones de criptografía simétrica y asimétrica, también altamente parametrizables para poder ser ajustadas a diversos escenarios de aplicación mediante realizaciones híbridas *hardware/software*; que van desde coprocesadores para la generación de números aleatorios o multiplicaciones modulares hasta el desarrollo de un criptoprocesador para acelerar las complejas operaciones asociadas al cálculo de emparejamientos bilineales sobre curvas elípticas. Buena parte de estas implementaciones *hardware* aprovecha los resultados del tercer conjunto de contribuciones (relacionadas con analizar y detectar vulnerabilidades de implementaciones *hardware* de funciones criptográficas ante diferentes tipos de ataques, así como las contramedidas propuestas) para robustecerlas.

La metodología de diseño utilizada en estos desarrollos está basada en la utilización del lenguaje de descripción de *hardware* VHDL. Esto no supone limitante alguna pues el campo de la implementación hardware de funciones criptográficas se circunscribe a especialistas de esta área del conocimiento.

Muchas de las implementaciones hardware de algoritmos criptográficos precedentes se basaban en realizaciones de la totalidad de un algoritmo considerando una aplicación específica con lo cual, si bien podían alcanzar una elevada velocidad de respuesta, se caracterizan por consumir muchos recursos del dispositivo programable y, sobre todo, no permitir su fácil adaptación a diversas aplicaciones.

Una de las contribuciones del GISDE consistió en implementar en *hardware* sobre un FPGA, con el mayor grado de parametrización posible, solamente aquellas funciones de mayor latencia de un determinado algoritmo criptográfico, ejecutando mediante *software* en un sistema de procesamiento (también embebido en el mismo FPGA) las funciones no críticas en tiempo del algoritmo. Para determinar las funciones de mayor latencia se utilizaron técnicas y herramien-

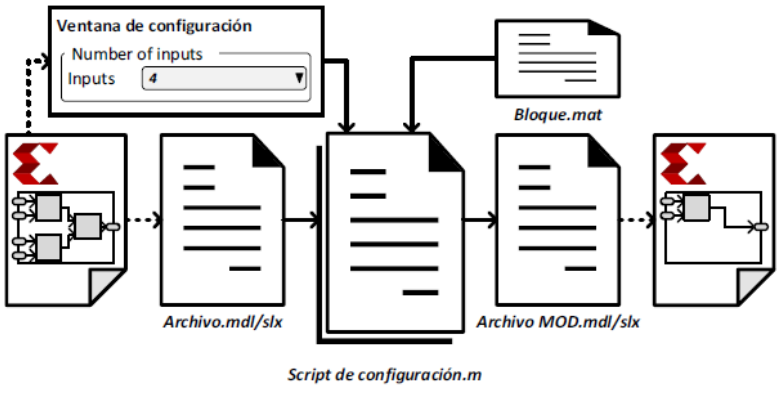

Fig. 1. Proceso de reconfiguración automática de los bloques de procesamiento. Fuente: Imagen tomada de Garcés-Socarrás [11]

tas de análisis de rendimiento (*profiling*) de la implementación completamente *software* del algoritmo criptográfico sobre el sistema de procesamiento embebido. En base a los resultados y analizando el grado de paralelismo de las funciones criptográficas de mayor latencia, se realizó el particionado *hardware*/*software* y se procedió a su diseño e implementación hardware, desarrollando arquitecturas que permitiesen su adaptabilidad a diferentes escenarios. [12-16] Para el intercambio de información entre las funciones ejecutadas en *software* en el sistema de procesamiento y las funciones *hardware* desarrolladas, estas fueron convertidas en coprocesadores para ser acoplados a los buses del sistema de procesamiento embebido basado en módulos IP *softcore* del procesador MicroBlaze y de sus componentes periféricos.

Así, mediante estas realizaciones híbridas HW/SW, se obtienen excelentes resultados que combinan la flexibilidad de las realizaciones *software* con la alta velocidad de respuesta de las implementaciones hardware, también flexibles, con un consumo racional de recursos del dispositivo programable.

Con la evolución de la tecnología microelectrónica y el surgimiento de los dispositivos SoC-FPGA que, con independencia del fabricante, se caracterizan por incorporar, en el mismo circuito integrado, una parte de hardware reconfigurable (con los mismos recursos de un FPGA) así como un potente sistema de procesamiento ya implementado en *hardware* (*hardcore*), compuesto por 1 o más procesadores ARM con sus buses AMBA/AXI, memorias cache de primer y segundo nivel, coprocesador de instrucciones SIMD (del inglés Single Instruction - Multiple Data), etc. Dado que estos recursos ya están implementados en *hardware* en el SoC-FPGA este sistema de procesamiento puede trabajar a frecuencias que pueden alcanzar el orden de GHz, muy superiores a las frecuencias de operación de los sistemas de procesamiento softcore utilizados previamente.

Este avance tecnológico conlleva nuevos análisis en el desarrollo de implementaciones híbridas HW/SW, sobre todo en lo referente al particionado de las tareas. Esto implica que determinadas funciones que anteriormente se implementaban en hardware puedan ser ahora implementadas en software sin detrimento en el rendimiento, lo cual equivale a desplazar la frontera de decisión para el particionado en las realizaciones híbridas HW/SW.

Sobre una plataforma de dispositivos SoC-FPGA de la familia Zynq-7000 es que se desarrollaron otras contribuciones relacionadas con la aceleración de implementaciones de funciones para el cálculo de emparejamientos bilineales sobre curvas elípticas. Si bien la teoría alrededor de estos emparejamientos es conocida desde hace mucho tiempo, su utilización en criptografía data de poco más de 20 años, después que se evidenciasen sus potencialidades para construir diferentes protocolos criptográficos, siendo de amplia utilización en la actualidad. [17-21]

Sin embargo, el proceso de cálculo de un emparejamiento bilineal constituye una tarea compleja y con un tiempo de ejecución muy superior al de las operaciones involucradas en criptosistemas de clave públicas convencionales. Por tal motivo, para extender la utilidad práctica de los emparejamientos bilineales en criptografía es determinante desarrollar formulaciones algorítmicas y evaluar estrategias de implementación que permitan obtener soluciones eficientes.

En la investigación desarrollada para la aceleración de implementaciones de funciones criptográficas basadas en el emparejamiento Ate óptimo sobre curvas de Barreto-Naehrig (uno de los más utilizados) utilizando un SoC-FPGA de la familia Zynq-7000, con un sistema de procesamiento *hardcore* doble núcleo trabajando a una frecuencia de 667 MHz, se han explorado 2 estrategias diferentes.

Una se basa en implementaciones *software* para procesadores ARM que analizando a profundidad las diferentes funciones a implementar, permitieron determinar cuáles pueden ser paralelizables utilizando los recursos del sistema de procesamiento. En base a estos resultados se realizaron 3 variantes de soluciones *software* basadas en: a) optimizar las operaciones en lenguaje ensamblador sobre 1 de los núcleos del sistema de procesamiento; b) utilizar instrucciones SIMD de la unidad Neón; c) paralelizar las operaciones en ensamblador mediante la utilización de ambos núcleos de procesamiento con su correspondiente intercambio de información. [22,23]

La segunda estrategia seguida a partir de los resultados obtenidos con las diferentes soluciones *software* desarrolladas previamente, se basa en implementaciones híbridas HW/SW con el objetivo de explotar en mayor medida el alto grado de paralelismo que exhiben las formulaciones algorítmicas involucradas en el cálculo de un emparejamiento. [24] El desarrollo de estas variantes se ha enfocado además en la obtención de soluciones flexibles que permitan calcular varios tipos de emparejamientos sin que ello implique modificaciones a nivel de *hardware*. Así, la primera variante se basa en acelerar la operación crítica en tiempo que mayor incidencia tiene en el cálculo de un emparejamiento mediante su implementación en un coprocesador *hardware*, mientras que el resto de la pirámide de procesamiento se ejecuta en *software*. Con esta variante se implementó en *hardware* la operación de multiplicación en el campo finito Fp^2 por ser la de mayor latencia e implicar operaciones internas que pueden ser paralelizadas. De esta forma, la implementación *software* sobre uno de los núcleos ARM del sistema de procesamiento interactúa con el

coprocesador *hardware* implementado en la zona de lógica programable del Zynq cada vez que requiera la ejecución de esta función.

Por su parte, la segunda variante extiende el aprovechamiento del *hardware* mediante el diseño de un criptoprocesador aritmético con un repertorio de instrucciones propio, que incorpora un conjunto de coprocesadores que aceleran el cálculo de varias operaciones básicas. Estos coprocesadores se integran en un bloque funcional encargado de establecer diferentes secuencias de operación codificadas mediante las diferentes instrucciones. De esta manera, la arquitectura *hardware* del criptoprocesador permite implementar cualquier funcionalidad que pueda ser descrita mediante su repertorio de instrucciones, el cual ha sido especialmente diseñado para calcular emparejamientos bilineales.

El criptoprocesador aritmético para el cálculo de emparejamientos bilineales (CAPEB) desarrollado posee 4 elementos que lo distinguen: a) la disponibilidad de un repertorio de instrucciones, muchas de ellas de tipo SIMD para explotar el paralelismo del *hardware*, específicas para el cálculo de diferentes tipos de emparejamientos; b) la simultaneidad en la ejecución de varias operaciones aritméticas; c) la reducción de las transferencias de datos con el sistema de procesamiento al implementarse en el criptoprocesador la mayor parte de las operaciones y realizar las transferencias mediante acceso directo a memoria (DMA, por sus siglas en inglés); d) su flexibilidad, la cual permite implementar diferentes tipos de emparejamientos, sobre diferentes tipos de curvas elípticas y para diferentes niveles de seguridad. La figura 2 muestra la estructura general de CAPEB, observándose sus 2 unidades principales: la unidad de control y la unidad aritmética.

En la unidad de control sobresalen los registros de configuración que determinan su flexibilidad; la interfaz de control de DMA para las transferencias de los datos hacia y desde la memoria principal sin la intervención del sistema de procesamiento; la memoria de programa, en la cual se almacenan las diferentes instrucciones a ejecutar para el cálculo de un emparejamiento y que también contribuyen a la flexibilidad de CAPEB; así como el bloque de ejecución, encargado del control de la ejecución de las instrucciones en estrecha interacción con la unidad aritmética.

En la unidad aritmética se ejecutan las tareas asociadas a cada instrucción, acciones gobernadas por el bloque de control en interacción con el acelerador aritmético compuesto por 2 núcleos de arreglos de coprocesadores encargados de diversas operaciones. Esta unidad contiene su propia

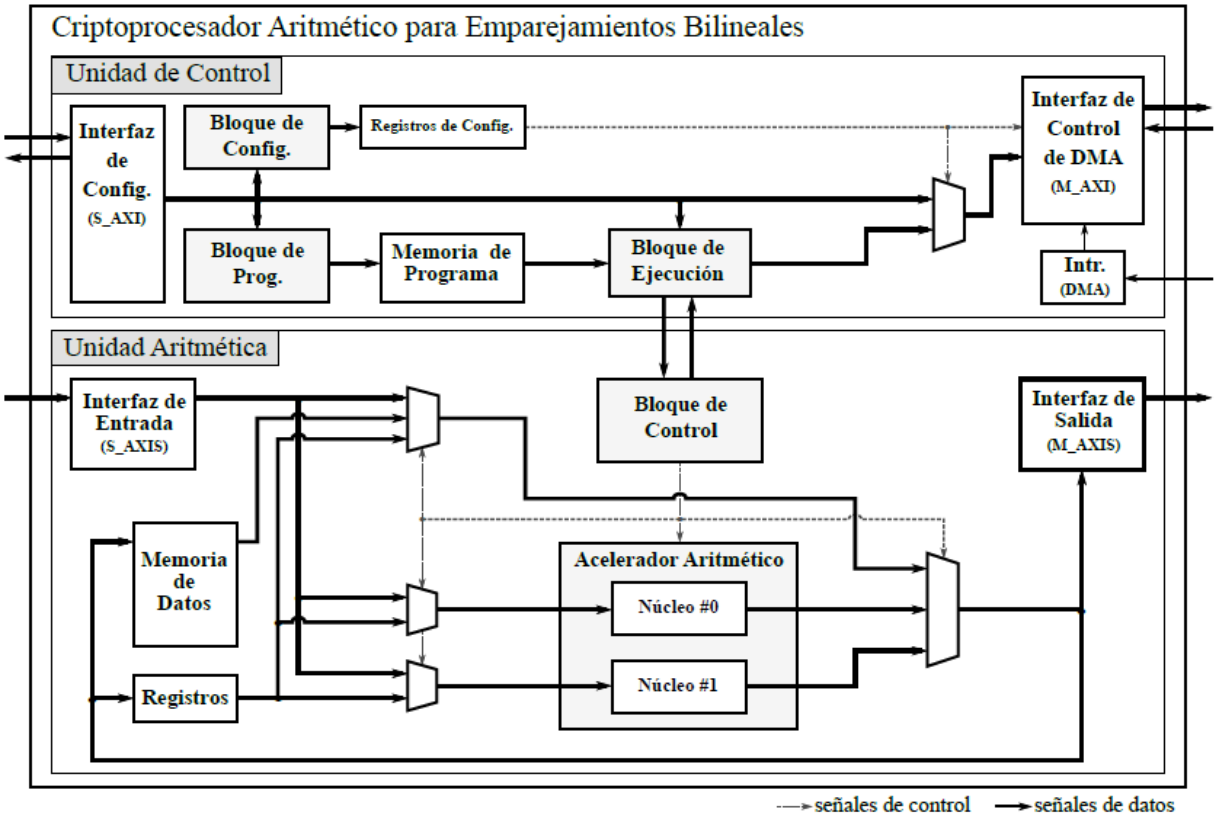

Fig. 2. Estructura del criptoprocesador CAPEB. Fuente: Imagen tomada de Cuiman-Márquez [24]

memoria para los datos que se transferirán por DMA, así como un conjunto de registros de propósito general utilizados para la ejecución de diferentes instrucciones. Los detalles de la implementación de CAPEB aparecen expuestos en Cuiman-Márquez. [24]

Metodología utilizada para la detección de vulnerabilidades en implementaciones *hardware* de funciones criptográficas y contramedidas

El tercer conjunto de contribuciones presentadas en esta propuesta trata acerca del criptoanálisis realizado a diferentes implementaciones hardware de funciones criptográficas para la detección de nuevas vulnerabilidades ante diferentes tipos de ataques por parte de un adversario que quiera obtener la información secreta que procesan, así como las contramedidas correspondientes.

La metodología utilizada se ha basado en la realización de diferentes tipos de ataques, sobre todo de manipulación (tampering) de contenidos, así como de canal colateral de consumo de potencia, Partiendo de que el atacante posea un profundo conocimiento de las características de las implementaciones y pleno acceso.

La primera de las contribuciones, expuesta en detalle en Cabrera-Aldaya, [25] consiste en la detección de la vulnerabilidad de las implementaciones del algoritmo AES (del inglés Advanced Encryption Standard) basadas en la utilización de los bloques de memoria RAM (BRAM) de las FPGA.

El algoritmo AES consiste en aplicar 4 transformaciones a un bloque de entrada de 128 *bits*, denominadas AddRoundkey, SubBytes, ShiftRows, MixColumns. Si bien la implementación de las transformaciones ShiftRows y AddRoundKey es sencilla con los recursos lógicos de un FPGA, las transformaciones SubBytes y MixColumns son más complejas de implementar. Esta complejidad puede simplificarse si se implementan mediante la utilización de tablas en memoria, conocidas como T-Box, las cuales poseen la característica de permitir altas velocidades de procesamiento a expensas de la capacidad de almacenamiento requerida.

Así, las implementaciones del algoritmo AES en FPGA se basan en aprovechar la disponibilidad de bloques de memoria RAM para implementar las T-Box, considerándose seguras (antes de esta contribución) debido a la complejidad relacionada con el fichero de configuración (*bitstream*) del FPGA. Sin embargo, utilizando herramientas proporcionadas por el propio fabricante del dispositivo, es posible identificar cuáles BRAM del FPGA son utilizadas para almacenar las T-Box y, mediante un ataque de manipulación (*tampering*), modificar sus contenidos para facilitar la obtención de la clave secreta.

De esta forma el procedimiento general para recuperar la clave secreta utilizada por un algoritmo AES, implementado en hardware utilizando BRAM para almacenar las T-Box, mediante este tipo de ataque consiste en: a) extraer el contenido de las BRAM del bitstream e identificar cuáles son utilizadas para almacenar las T-Box. b) modificar las BRAM que contienen las T-Box de forma tal que permitan la recuperación de la clave y c) volver a configurar el FPGA con el bitstream modificado y observar la salida del algoritmo.

Otra de las contribuciones de esta investigación es que puede ser utilizado para ataques a diferentes implementaciones de AES de 128 *bits*, 192 *bits* y 256 *bits* con 3 formas de implementar la ronda final, en las cuales el atacante solo puede observar el valor de la salida del algoritmo. En el caso de la implementación de AES-128 solo es necesario realizar una vez el procedimiento expuesto y reprogramar una vez el FPGA, siendo necesario una cantidad mayor en caso de AES-192 y AES-256 que varían entre 2 y 1280, en función de la forma de implementar la etapa final.

Como parte de la investigación se analizaron y comprobaron 3 contramedidas para proteger la implementación hardware de AES contra este tipo de ataque, consistentes en: a) verificación *hardware* del contenido de las T-Box, con un incremento despreciable de consumo de recursos adicionales, aunque incrementa la latencia en el primer acceso; b) generar por *hardware* el contenido de las T-Box antes de la operación de AES, de forma que no pueda ser modificado por el atacante. Esta contramedida consume algo más de recursos y disminuye entre 8 la latencia del primer acceso; c) implementar las T-Box en memoria distribuida, opción que hace imposible identificar cuales recursos del FPGA son utilizados para almacenar las T-Box, aunque incrementa la latencia. En las implementaciones *hardware* de los coprocesadores AES del apartado anterior fue utilizada la segunda contramedida.

Otra de las contribuciones obtenidas acerca de la detección de vulnerabilidades de implementaciones de algoritmos criptográficos está relacionada con ataques de canal colateral de consumo de potencia del dispositivo soporte de la implementación del algoritmo. Desde la concepción de la criptografía moderna, la seguridad de los algoritmos criptográficos se ha establecido en términos formales utilizando principalmente argumentos matemáticos y de costo computacional. Sin embargo, en 1996 un conjunto de investigadores identificó que la implementación de estos algoritmos en un sistema de cómputo puede producir fugas de información a través de variables físicas, tales como el tiempo de ejecución del algoritmo, el consumo de potencia y la emisión de radiaciones electromagnéticas del dispositivo en el cual se ejecuten. [26,27] Estas magnitudes físicas son ejemplos típicos de canales colaterales en dispositivos electrónicos. Cuando los valores de alguna de estas magnitudes están relacionados con un dato

secreto, entonces se está en presencia de una fuga de información a través de un canal colateral de dicha magnitud.

Aunque en esta investigación se han desarrollado varios tipos de ataques, los resultados asociados a las contribuciones que se exponen están basados en ataques de análisis simples de potencia (SPA, por sus siglas en inglés). El término *simple* en su definición no significa que sea sencillo de realizar, sino que la distinción de diferentes patrones en una traza de consumo de potencia puede realizarse *visualmente*, es decir, estos patrones pueden ser claramente distinguibles para el atacante. Este tipo de ataque puede necesitar en algunos casos solo una traza de consumo de potencia para obtener la información deseada. Estas características convierten a los ataques SPA en una gran amenaza para la seguridad de muchos criptosistemas, sobre todo en implementaciones (*hardware* o *software*) en sistemas embebidos en los cuales es más factible asociar las trazas de consumo de potencia con la ejecución de un determinado algoritmo criptográfico, resaltando la importancia del análisis de vulnerabilidades ante ataques SPA de estas implementaciones.

Es importante recalcar 2 aspectos relacionados con el criptoanálisis a implementaciones basado en ataques SPA por consumo de potencia. En primer lugar, la necesidad de que el atacante domine las posibles alternativas de implementación para poder identificar los patrones en las trazas de consumo de potencia. En segundo lugar, no siempre es necesario obtener todos los *bits* de la información secreta pues, con algunos *bits* y utilizando procedimientos matemáticos (como la solución de una instancia del problema del número oculto -HNP, por sus siglas en inglés-) es posible recuperar el resto de los *bits*. No obstante, en cualquier caso, la obtención de algunos *bits* de la información secreta disminuye la seguridad del criptosistema.

Dentro de los ataques SPA realizados a implementaciones de algoritmos criptográficos, la contribución principal está relacionada con el criptoanálisis a diferentes implementaciones en sistemas embebidos del algoritmo binario de Euclides (BEA, por sus siglas en inglés) y a su variante extendida (BEEA), ambos muy utilizados en criptografía. Estas contribuciones aparecen detalladas en Cabrera-Aldaya [28-30] por lo que se resume a continuación la metodología utilizada.

La figura 3A muestra el seudocódigo del BEA para el cálculo de máximo común divisor de 2 números enteros (a y b) donde 1 de ellos es un valor secreto. El ciclo principal de este algoritmo es la instrucción 8, la cual incluye 2 iteraciones internas, denominadas *u-loop* y *v-loop* así como una operación condicional conocida como *sub-step*, mientras que la etapa *sub-step* está compuesta por las operaciones de comparación y sustracción que se ejecutan entre los pasos 13 y 16.

Dada la dependencia del flujo de ejecución del BEA con sus entradas, en el desarrollo de la investigación se introdujo una notación para caracterizarlo, centrándose en el ciclo principal del algoritmo (paso 8), cuyo diagrama de flujo se muestra en la figura 3B. De esta forma, el flujo de ejecución de este algoritmo se puede caracterizar por 2 variables (Zi y Xi) que toman valores para cada iteración *i* del ciclo principal del algoritmo:

- Zi: Cantidad de iteraciones ejecutadas por el ***u-loop*** o el ***v-loop*** en la iteración ***i***, es decir la cantidad de divisiones entre 2 de las variables ***u*** o ***v*** respectivamente.
- Xi: Valor binario (***'u'*** ó ***'v'***) que representa el resultado de la condición del ***sub-step*** en la iteración ***i*** (Xi =***'u'***, si ***u*** es mayor o igual que ***v***; y Xi=***'v'*** en caso contrario).

El estado del arte previamente existente acerca de la vulnerabilidad de este algoritmo, establecía que era posible recuperar los valores de las entradas (es decir, el valor secreto) en tiempo polinomial si y solo si se dispone del conocimiento de los valores de Zi y de Xi para todas las iteraciones *i* del ciclo principal de este algoritmo (deshaciendo las diferentes operaciones, comenzando por los valores finales hasta llegar a los valores de entrada a y b), aspecto con extremadamente baja probabilidad de obtención mediante ataques, por lo que era considerado seguro ante el conocimiento *parcial* (es decir, solo algunos valores de Zi y Xi) de su flujo de ejecución. A este modelo se le denominó *todo-o-nada*.

En Cabrera-Aldaya [28-30] los autores demuestran la vulnerabilidad del algoritmo binario de Euclides y sus variantes ante el conocimiento *parcial* de su flujo de ejecución, por lo que la relevancia de la contribución estriba en varios aspectos: a) la amplia utilización de estos algoritmos como parte de múltiples funciones criptográficas empleadas en muy diversos criptosistemas; b) el hecho de que estos algoritmos involucran información secreta, por lo que una fuga de información puede comprometer su seguridad; c) la creencia, antes de esta contribución, de que las implementaciones del BEA y sus variantes son seguras mientras no se disponga de "toda" la información relativa al flujo de ejecución del algoritmo, considerándose seguras ante su conocimiento "parcial"; d) la posibilidad de obtener esta información parcial en diferentes implementaciones en sistemas embebidos de criptosistemas que utilicen este algoritmo mediante ataques SPA.

Como parte de la investigación también se obtuvieron las características de las trazas de consumo de potencia de implementaciones de este algoritmo y su relación con los valores Zi, Xi que permiten realizar ataques SPA, así como la relación entre la cantidad de pares Zi,Xi conocidos con la cantidad de *bits* que pueden ser recuperados. De esta forma se obtuvieron 3 modelos con sus respectivos procedimientos los cuales se

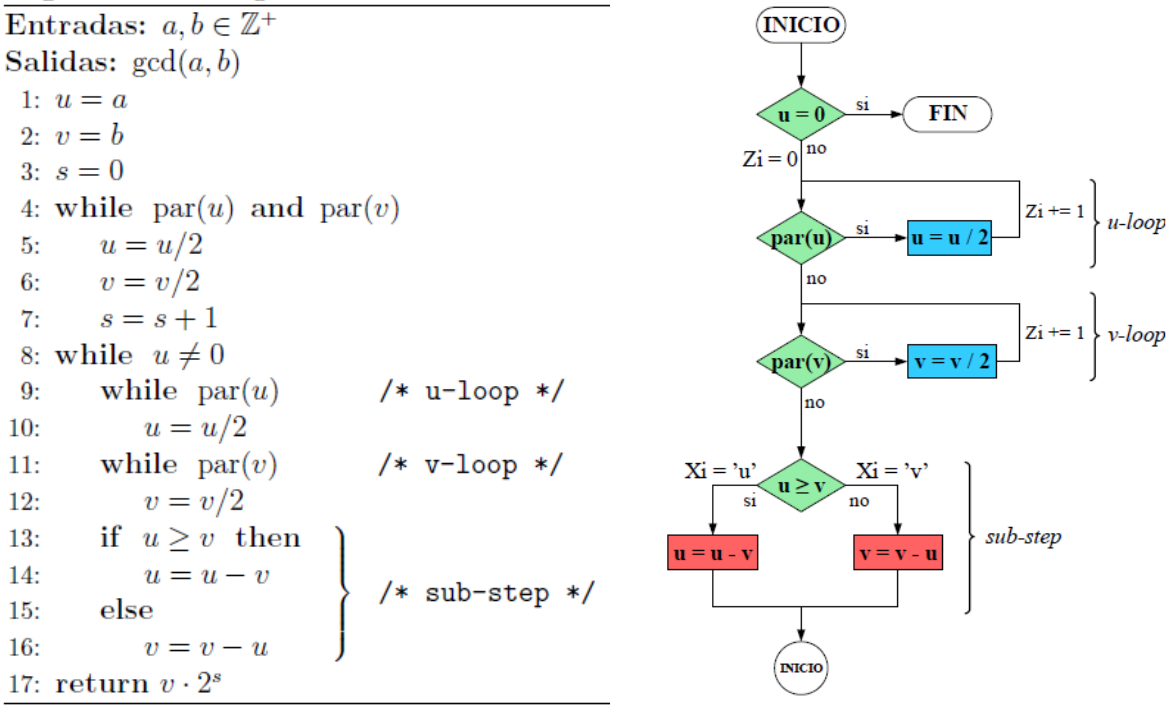

Fig. 3. Algoritmo binario de Euclides. A) Seudocódigo; B) Diagrama de flujo del ciclo principal. Fuente: Imagen tomada de Cabrera-Aldaya [30]

resumen en la tabla 1, junto con el modelo *todo-o-nada* original que requiere el conocimiento de los valores de los pares Zi, Xi para todas las T iteraciones del algoritmo, indicándose la cantidad de *bits* que pueden ser recuperados con cada uno. De forma general, mientras más pares Zi,Xi sean conocidos por el atacante, más información (mayor cantidad de *bits*) se puede obtener sobre las entradas

Estos 3 modelos de información parcial aparecen denominados como Zi,Xi; Z_1; y Zi = > Xi. El más simple es el modelo Z_1 asociado con la primera iteración del ciclo principal del *u-loop* o el *v-loop,* es decir la cantidad de divisiones entre 2 de las variables *u* o *v* respectivamente en la primera iteración, mediante el cual se pueden recuperar Z_1 +1 *bits* del dato secreto. Si bien esta cantidad de *bits* puede ser pequeña, en dependencia del criptosistema puede ser suficiente para comprometer su seguridad.

El método más general (Zi,Xi) está basado en la cantidad de pares Zi,Xi que pueden ser obtenidos, mientras que para el tercero (Zi=>Xi), se obtuvieron las formulaciones matemáticas correspondientes que permiten determinar valores de Xi a partir de valores de Zi. Tanto para este método como para el anterior, en dependencia del criptosistema, puede ser necesario conocer o no, 1 de los valores de entrada.

Los modelos anteriores establecen cuánta información puede ser obtenida a partir del conocimiento parcial del flujo de ejecución, desarrollándose los procedimientos para obtener dicha información a partir del análisis de las trazas de consumo de potencia resultado del ataque SPA. Para esto se realizaron los correspondientes análisis de las implementaciones de: números grandes; de las operaciones condicionales; del *u-loop* y *v-loop*; del *sub-step*; y del ciclo principal del algoritmo binario de Euclides (o sus variantes) que sustentan la hipótesis de que son vulnerables ante el conocimiento parcial de su flujo de ejecución, todas las cuales se detallan en Cabrera-Aldaya. [30]

RESULTADOS

Varios de los componentes de la biblioteca *XIL XSGImgLib* han sido introducidos en una aplicación híbrida HW/SW de detección e identificación de matrículas de vehículos en tiempo real (tiempo máximo de 20 ms para el procesamiento de una imagen), como parte del proyecto Sistema inteligente de transporte (SIT) que se desarrolla para el MININT.

Uno de los componentes de este sistema es un controlador maestro (resultado preliminar obtenido por el GISDE) basado en un FPGA Spartan3E-1600 (o Spartan6-LX45), el cual incluye un sistema de procesamiento embebido altamente configurable basado en módulos IP *softcore* del procesador MicroBlaze y múltiples periféricos de entrada/salida, de temporización, interrupción, controladores de memoria y de interfaces de comunicación. [12] Así las imágenes de una cámara son adquiridas por el sistema de procesamiento embebido a

Tabla 1. Comparación entre modelos/procedimientos.

Modelo/ Procedimiento	Información necesaria	Número de bits recuperados	¿Requiere conocer una entrada?
todo-o-nada	**total** Z_i, X_i $1 \leq i \leq T$	todos	no
Z_i, X_i	**parcial** $Z_i : 1 \leq i \leq t \leq T$ $X_i : 1 \leq i \leq t-2$	$n \geq \sum_{i=2}^{t} Z_i + 1$	si/no
Z_1	**parcial** Z_1	$Z_1 + 1$	no
$Z_i \Rightarrow X_i$ Z_i, X_i	**parcial** $Z_i\ ,\ a < f(b)$ $1 \leq i \leq t \leq T$	$\min : 3$ $\max : \sum_{i=1}^{t} Z_i + 1$	si/no

Fuente: Imagen tomada de Cabrera-Aldaya [30]

través de una interfaz Ethernet y pasadas a los bloques *hardware* de la biblioteca *XIL XSGImgLib* encargados de detectar la zona de la imagen donde se encuentra la matrícula, segmentar los caracteres e identificarlos. Estas mismas operaciones, implementadas en *software* (lenguaje C) sobre el procesador MicroBlaze con un reloj de 125 MHz, no satisfacen el requisito de operación en tiempo real.

De forma similar implementaciones *hardware* de las funciones criptográficas de AES-256, funciones resumen SHA-256 y para la generación de números aleatorios (protegidas contra diferentes ataques) han sido incorporadas a la biblioteca OpenSSL utilizada por el controlador maestro señalado previamente para acelerar su ejecución y robustecer las comunicaciones con el centro de control del SIT. [12-15]

Gran parte de las funciones criptográficas implementadas en *hardware*, incorporando las medidas de seguridad contra diferentes tipos de ataques, han sido utilizadas en el desarrollo del sistema criptográfico ARCANO, cuyo componente fundamental es un módulo de seguridad *hardware* basado en un FPGA y diseñado para la protección de información digital. Este módulo constituye un contenedor seguro de identidades digitales con las cuales se pueden realizar operaciones de cifrado y firma digital en el propio dispositivo evitando en todo momento la exposición de información sensible. Entre sus aplicaciones se encuentran el almacenamiento seguro de claves criptográficas que se pueden emplear como credenciales de acceso; la protección de datos mediante cifrado simétrico (AES), la verificación de la integridad y autenticidad de la información mediante firmas digitales (RSA y ECDSA), así como su integración con clientes de correo electrónico, editores de documentos y navegadores web mediante el estándar PKCS#11.

El sistema criptográfico ARCANO está compuesto por una plataforma *hardware* y un soporte de aplicaciones *software*, tanto de explotación del sistema como de configuración de la plataforma *hardware,* que ejecutadas desde una computadora personal permite la interacción con el *hardware*.

Para obtener una solución flexible que permita la implementación de muy diversos algoritmos y funciones criptográficas se ha diseñado una plataforma *hardware* genérica con capacidad de reconfiguración, la cual posee como componente fundamental un FPGA de la familia Spartan6, en el cual se implementa un sistema de procesamiento MicroBlaze, así como las diferentes funciones criptográficas requeridas por las aplicaciones. Una vez configurado el FPGA asume la carga principal de procesamiento criptográfico. El dispositivo seleccionado posee recursos de *hardware* suficientes para albergar los principales algoritmos criptográficos utilizados actualmente. Asimismo, pueden incluírsele nuevos algoritmos en el futuro (tales como los algoritmos de cifrado simétrico GOST 28147-89, *Serpent*, *Twofish* y RC6, desarrollados también como parte de la investigación) debido a la capacidad de reconfiguración de los FPGA que permite realizar modificaciones funcionales actualizando simplemente el fichero de configuración (*bitstream*) sin tener que realizar cambios en el diseño constructivo del dispositivo.

Como parte de la introducción del sistema criptográfico ARCANO se fabricó una serie cero de 100 dispositivos *hard-*

ware, como el mostrado en la figura 4 (en la cual se pueden apreciar los principales componentes) con dimensiones de 6,9 cm x 4,3 cm, los cuales están siendo utilizados por diferentes entidades. [24]

De las 3 alternativas de implementaciones *software* sobre el sistema de procesamiento *hardcore* del SoC-FPGA (trabajando a una frecuencia de 667 MHz) para el cálculo del emparejamiento Ate-Opt-BN, la de mejores resultados resultó ser la implementación basada en doble núcleo, con una latencia inferior a la de las implementaciones reportadas previamente, [23] mientras que con la implementación del criptoprocesador CAPEB, operando a 100 MHz, se logra acelerar el cálculo por 8, del emparejamiento con relación a la mejor de las implementaciones de *software*.

Buena parte de las implementaciones *hardware* expuestas previamente, sobre todo los coprocesadores incorporados en el sistema criptográfico ARCANO y en el criptoprocesador CAPEB, fueron robustecidas en base a los resultados derivados del criptoanálisis realizado.

Los métodos y procedimientos expuestos fueron aplicados a diferentes implementaciones del BEEA en sistemas embebidos utilizadas en criptosistemas de firmas digitales ECDSA (del inglés *Elliptic Curve Digital Signature Algorithm*) y RSA (de Rivest, Shamir, Adleman, los apellidos de sus creadores).

Para el criptosistema ECDSA se desarrolló una implementación del BEEA sobre un FPGA, realizándose ataques de canal colateral SPA. La figura 5 muestra una traza de consumo de potencia de una implementación del criptosistema ECDSA en la cual se pueden identificar diferentes patrones que en base a los análisis detallados en Cabrera-Aldaya [29,30] se corresponden con diferentes operaciones del algoritmo. A partir de estos resultados y aplicando los diferentes métodos desarrollados se obtienen un conjunto de pares Zi,Xi que permiten determinar un conjunto de *bits* de la clave secreta.

Conociendo que un criptosistema ECDSA puede ser comprometido si un atacante conoce algunos *bits* de los valores de la clave secreta empleados para generar un conjunto de firmas digitales, solucionando una instancia del problema del número oculto (HNP), se realizaron diferentes procesos de firmas.

Mientras mayor sea la cantidad de *bits* conocidos, menor será la cantidad de firmas necesarias a obtener. Por ejemplo, con el conocimiento de sólo 8 *bits* de la clave secreta, con apenas 42 firmas digitales se puede obtener una clave secreta de 256 *bits* (uno de los valores más utilizado en ECDSA), ratificando la hipótesis que el algoritmo binario de Euclides es vulnerable ante el conocimiento parcial de su flujo de ejecución.

Conclusiones

Se han presentado 3 conjuntos de contribuciones relacionadas con implementaciones *hardware* altamente parametrizables de funciones de procesamiento de imágenes y criptográficas, debidamente protegidas contra diferentes ataques, para acelerar su ejecución mediante implementaciones híbridas HW/SW embebidas en un dispositivo de *hardware* reconfigurable.

Una de ellas consiste en una biblioteca de 54 funciones hardware de procesamiento de imágenes y videos, basada en modelos de Matlab/Simulink, potenciada con un novedoso procedimiento de configuración automática que permite su mejor adaptación a las necesidades de la aplicación.

También se desarrollaron diferentes componentes hardware en VHDL altamente parametrizables para implementar funciones criptográficas y acelerar su ejecución. Con la introducción de los dispositivos SoC-FPGA es necesario reanalizar el particionado en las realizaciones híbridas HW/SW dada la potencialidad muy superior de su sistema de procesamiento.

Aplicando diferentes tipos de ataques se identificaron vulnerabilidades en implementaciones hardware del algoritmo AES y del algoritmo binario de Euclides. Una importante contribución es la verificación práctica de que el BEA y sus variantes son vulnerables a ataques de canal colateral SPA, así como su vulnerabilidad ante el conocimiento parcial de su flujo de ejecución.

Fig. 4. Estructura interna y externa de la plataforma ***hardware*** del sistema criptográfico ARCANO. Fuente: Imagen tomada de Cuiman-Márquez [24]

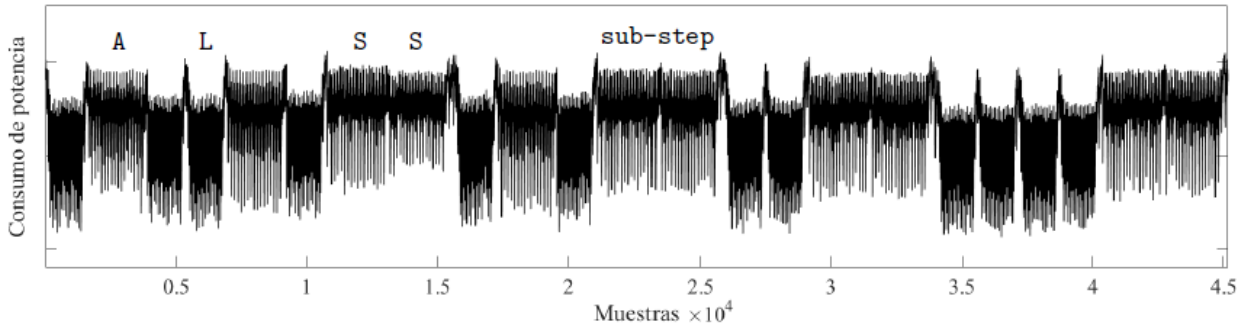

Fig. 5. Traza de consumo de potencia de una implementación del BEEA en FPGA. Fuente: Imagen tomada de Cabrera-Aldaya [30]

Algunos de los componentes de la biblioteca XIL XSGImgLib y de los coprocesadores hardware de funciones criptográficas, debidamente protegidos en base a los resultados obtenidos, han sido introducidos en aplicaciones reales.

REFERENCIAS BIBLIOGRÁFICAS

1. Estrin G. Organization of Computer Systems - The Fixed Plus Variable Structure Computer. Proceedings of Western Joint Computer Conference. New York: Editorial: Association for Computing Machinery May; 1960
2. Estrin G. Reconfigurable computer origins: the UCLA fixed-plus-variable (F+V) structure computer. IEEE Annals of the History of Computing. 2002;24(4):3-9
3. Cabrera-Sarmiento AJ, Plataforma para el desarrollo de controladores difusos híbridos como sistemas empotrados sobre un dispositivo programable [Tesis de doctorado]. Universidad Tecnológica de La Habana; 2004
4. Cabrera-Sarmiento A, Sánchez-Solano S, Brox P, Barriga A, Senhadji R. Hardware/software codesign of configurable fuzzy control systems, Applied Soft Computing. 2004;4(3):271-85
5. Sánchez-Solano S, Cabrera-Sarmiento A, Baturone I, Moreno F, Brox M, FPGA implementation of embedded fuzzy controllers for robotic applications. IEEE Transactions on Industrial Electronics. 2007;54(4):1937-45
6. Garcés-Socarrás LM, Cabrera-Sarmiento A, Sánchez-Solano S, Brox P, Ieno E, Cleber- Pimenta T. Self-modificable image processing library for model-based design on FPGAs. IEEE Latin America Transactions. 2019;17(2):742-50
7. Garcés-Socarrás LM, Sánchez-Solano S, Brox P, Cabrera-Sarmiento A. Library for model-based design of image processing algorithms on FPGAs. Revista Facultad de Ingeniería Universidad Antioquia. 2013; (68):36-47
8. Perdomo E, Garcés-Socarrás LM, Cabrera-Sarmiento A. Diseño de bloques para el procesado de imágenes en lenguaje de descripción de hardware. Revista Ingeniería Electrónica, Automática y Comunicaciones. 2013; 34(2):9-18
9. Garcés-Socarras LM. Aceleración de algoritmos mediante hardware reconfigurable: Biblioteca de procesamiento de imágenes para System Generator [Tesis de Maestría en Sistemas Digitales]. Universidad Tecnológica de La Habana; 2011
10. Garcés-Socarrás LM, Cabrera-Sarmiento A, Sánchez-Solano S, Brox P, Ieno E, Cleber- Pimenta T. Modificación automática de arquitecturas de módulos hardware de procesado de imágenes. Revista Ingeniería Electrónica, Automática y Comunicaciones. 2016;37(3):21-33
11. Garcés-Socarrás LM. Biblioteca basada en modelos de módulos ***hardware*** configurables para procesado de imágenes y vídeos [Tesis de doctorado]. Universidad Tecnológica de La Habana; 2016
12. Cabrera-Aldaya A, Cabrera-Sarmiento A. Controlador empotrado en FPGA para Sistema Inteligente de Transporte. Revista Ingeniería Electrónica, Automática y Comunicaciones. 2011;32(3):35-44
13. Cabrera-Aldaya A. Integración de algoritmos criptográficos en sistemas empotrados sobre FPGA [Tesis de Maestría en Sistemas Digitales]. Universidad Tecnológica de La Habana; 2012
14. Torres A, Martínez Y, Cuiman-Márquez R, Díaz H, Cabrera-Sarmiento A. Implementación eficiente de la multiplicación modular de Montgomery sobre hardware reconfigurable. Revista Ingeniería Electrónica, Automática y Comunicaciones. 2013;34(3)32-40
15. Cuiman-Márquez, R., Generación de números primos en FPGA utilizando diseño híbrido hardware-software [Tesis de Maestría en Sistemas Digitales]. Universidad Tecnológica de La Habana; 2014
16. López A, Aceleración de algoritmos de criptografía asimétrica mediante hardware reconfigurable y diseño híbrido [Tesis de Maestría en Sistemas Digitales]. Universidad Tecnológica de La Habana; 2014
17. Boneh D, Franklin M. Identity-Based Encryption from the Weil Pairing. Advances in Cryptology. Lecture Notes in Computer Science Book Series. Editorial Springer, Berlin. 2001;2139:213-29
18. Joux A, A One Round Protocol for Tripartite Diffie-Hellman. Journal of Cryptology. 2004;17(2):263-76
19. Sakai R, Ohgishi K, Kasahara M. Cryptosystems Based on Pairings. Ponencia presentada en: Symposium on Cryptography and Information Security - SCIS 2000. Jun 2000. Okinawa, Japón.
20. Verheul E. Self-Blindable Credential Certificates from the Weil Pairing. Advances in Cryptology. Lecture Notes in Computer Science Book Series. Editorial Springer, Berlin. 2001;2248:533-51
21. Herbert V, Biswas B, Fontaine C. Design and implementation of low-depth pairing-based homomorphic encryption scheme. Journal of Cryptographic Engineering, 2019;9(2):185-201
22. Cuiman-Márquez R, Cabrera-Sarmiento A, Sánchez-Solano S. Speeding up elliptic curve arithmetic on ARM processors using

NEON instructions, Revista Ingeniería Electrónica, Automática y Comunicaciones. 2020;41(3):1-20
23. Cuiman-Marquez R, Cabrera-Sarmiento A, Sánchez-Solano S. Implementing Cryptographic Pairings on ARM dual-core Processors. IEEE Latin America Transactions 2020;18(2):232-40
24. Cuiman-Márquez R. Estrategias de aceleración software y hardware para la implementación de emparejamientos bilineales en sistemas empotrados [Tesis de doctorado]. Universidad Tecnológica de La Habana; 2022.
25. Cabrera-Aldaya A, Cabrera-Sarmiento A, Sánchez-Solano S. AES T-Box Tampering Attack. Journal of Cryptographic Engineering. 2016;6(1):31-48
26. Kocher P. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. Advances in Cryptology. Lecture Notes in Computer Science Book Series. Editorial Springer, Berlin. 1996;1109:104-13
27. Kocher P, Jaffe J, Jun B. Differential power analysis. Advances in Cryptology. Lecture Notes in Computer Science Book Series. Editorial Springer, Berlin. 1999;1666:388-97
28. Cabrera-Aldaya A, Cuiman-Marquez R, Cabrera-Sarmiento A, Sánchez-Solano S. Side-Channel Analysis of the Modular Inversion Step in the RSA Key Generation Algorithm, International Journal of Circuit Theory and Applications. 2017;45(2):199-213
29. Cabrera-Aldaya A, Cabrera-Sarmiento A, Sánchez-Solano S. SPA vulnerabilities of the binary extended Euclidean algorithm. Journal of Cryptographic Engineering. 2017;7(2):273-85
30. Cabrera-Aldaya A. Criptoanálisis de canal colateral a implementaciones del Algoritmo Binario de Euclides en sistemas empotrados: vulnerabilidad ante el conocimiento parcial de su flujo de ejecución [Tesis de doctorado]. Universidad Tecnológica de La Habana; 2019

Recibido: 18/08/2023
Aprobado: 12/09/2023

Conflictos de interés

Los autores declaran que no existen conflictos de intereses entre ellos, ni con la investigación presentada.

Contribuciones de los autores

Conceptualización: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez
Curación de datos: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Egidio Ieno Junior
Análisis formal: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez, Billy Bob Brumley
Adquisición de fondos: Alejandro José Cabrera Sarmiento, Santiago Sánchez Solano, Piedad Brox Jiménez, Tales Cleber Pimenta, Billy Bob Brumley
Investigación: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez, Egidio Ieno Junior
Metodología: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez, Egidio Ieno Junior, Tales Cleber Pimenta, Billy Bob Brumley
Administración del proyecto: Alejandro José Cabrera Sarmiento, Santiago Sánchez Solano, Piedad Brox Jiménez, Tales Cleber Pimenta, Billy Bob Brumley
Recursos: Alejandro José Cabrera Sarmiento, Santiago Sánchez Solano, Piedad Brox Jiménez, Tales Cleber Pimenta, Billy Bob Brumley
Software: Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Egidio Ieno Junior
Supervisión: Alejandro José Cabrera Sarmiento, Santiago Sánchez Solano, Piedad Brox Jiménez, Tales Cleber Pimenta, Billy Bob Brumley
Validación: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez, Egidio Ieno Junior
Visualización: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez, Egidio Ieno Junior
Redacción-borrador original: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez
Redacción-revisión y edición: Alejandro José Cabrera Sarmiento, Luís Manuel Garcés Socarrás. Alejandro Cabrera Aldaya, Raudel Cuiman Márquez, Santiago Sánchez Solano, Piedad Brox Jiménez, Egidio Ieno Junior, Tales Cleber Pimenta, Billy Bob Brumley

Financiamientos

Las investigaciones desarrolladas fueron parcialmente financiadas por el Programa de Cooperación Internacional i-COOP+ del Consejo Superior de Investigaciones Científicas de España y por el programa CAPES/MES entre Cuba y Brasil.

Cómo citar este artículo

Cabrera Sarmiento AJ, Garcés Socarrás LM, Cabrera Aldaya A, Cuiman Márquez R, Sánchez Solano S, Brox Jiménez P, et al. Contribuciones a la implementación de sistemas electrónicos digitales embebidos sobre hardware reconfigurable. An Acad Cienc Cuba [internet] 2023 [citado en día, mes y año];13(4):e1482. Disponible en: http://www.revistaccuba.cu/index.php/revacc/article/view/1482