VISIÓN ELECTRÓNICA

IMPLEMENTACIONES CRIPTOGRÁFICAS EN FPGA

CRYPTOGRAPHIC IMPLEMENTATIONS FOR FPGA

Fabián Velásquez[1]
Javier F. Castaño[2]

Fecha de envío: Noviembre de 2010
Fecha de recepción: Diciembre de 2010
Fecha de aceptación: Marzo de 2011

Resumen:

Las técnicas que se requieren para proteger datos, en la actualidad, corresponden al campo de la criptografía basada en modelación matemática, que asegura confidencialidad en la transmisión y almacenamiento de la información. En este artículo, se presentan los resultados parciales del desarrollo de una infraestructura de clave pública, utilizando una plataforma reconfigurable basada en dispositivos FPGA. La arquitectura se soporta en el criptosistema de curvas elípticas (ECC Elliptic curve cryptosystem), además integrada por el algoritmo Rijndael (AES) para cifrado simétrico y SHA como algoritmo de integridad de la información. Se especifican los resultados alcanzados en el desarrollo del algoritmo AES y la implementación de criptografía de curvas elípticas en FPGA.

Palabras clave:
Infraestructura de clave pública, criptografía de curvas elípticas, AES, hardware reconfigurable.

Abstract:
At present, the techniques required to protect data correspond to the field of cryptography based on mathematical modeling, which ensures confidentiality when transmitting and storing information. In this article, we present the partial results of public-key infrastructure development using a reconfigurable platform based on FPGA devices. The architecture is based on the elliptic curve cryptosystem (ECC), including the Rijndael algorithm (AES) for symmetric encryption and the SHA as the data integrity algorithm. The results achieved when implementing both the AES algorithm and the elliptic curve cryptography with FPGA are presented.

Keywords:
Public-key infrastructure PKI, elliptic curve cryptography, AES, reconfigurable hardware.

1 Ingeniero Electrónico. M. Sc. (c) en Matemática Aplicada. Grupo de Investigación Macrypt, Universidad de los Llanos. Correo: fvelasquez@unillanos.edu.co

2 Ingeniero Electrónico. Esp (c) en Soluciones Telemáticas. Grupo de Investigación Macrypt, Universidad de los Llanos. Correo: jfcastano@unillanos.edu.co

1. Introducción

La seguridad informática ha adquirido gran importancia en las áreas de la industria, los negocios y en la sociedad actual. Las técnicas básicas que se requieren para proteger la información corresponden al campo de la criptografía, debido a que esta se aplica a la seguridad y es una herramienta primordial para asegurar confidencialidad en la transmisión y almacenamiento de la información; igualmente es necesario garantizar la autenticidad de la información, para validar las partes integrantes y del proceso, así como comprobar la integridad de la información, permitiendo detectar un cambio en la información originalmente transmitida o almacenada. Estas necesidades se resuelven con algoritmos de cifrado en el caso de la confidencialidad, esquemas de firma digital en el caso de la autenticidad y funciones hash en el caso de la integridad. La aplicación de estas técnicas se da en diversos escenarios, basados en una arquitectura general denominada PKI (*public-key infrastructure*). Estas aplicaciones incluyen toda clase de servicios telemáticos, tales como transacciones electrónicas, voto telemático, gobierno en línea, acceso y comunicación con servidores, entre otras.

El uso de técnicas criptográficas se ha masificado, principalmente en aplicaciones independientes de un PC, lo que implica el desarrollo de sistemas de bajo consumo dedicados, que realicen las funciones criptográficas en diferentes entornos. Estas implementaciones se soportan en sistemas embebidos, soportados en microcontroladores comerciales o en implementaciones totalmente soportadas en hardware reconfigurable, especialmente en FPGA (*field programmable gate array*). La tendencia actual es a la implementación de criptografía de curvas elípticas, en reemplazo de las implementaciones tradicionales basadas en RSA [1].

2. Criptografía de curvas elípticas

Una de las mejores técnicas en criptografía es el denominado criptosistema de curvas elípticas (conocido formalmente como ECC, por sus siglas en inglés), planteado independientemente por Koblitz [2] y Miller [3]. Se han presentado numerosas investigaciones teóricas como prácticas de esta técnica y se han demostrado sus capacidades para cifrar información y su posicionamiento como el criptosistema que ofrece mejores condiciones de seguridad, eficiencia y uso de memoria. La investigación actual en ECC se centra en mejorar los métodos aritméticos necesarios para operar sobre las curvas elípticas. Esto implica el cálculo de los puntos sobre las curvas en un campo finito que sirven para propósitos criptográficos. Desde el planteamiento del ECC, se han implementado numerosas arquitecturas en ambientes como software de alto nivel en diferentes sistemas operativos y en dispositivos de lógica programable. Cada vez se destaca más la importancia de la investigación de implementaciones eficientes de ECC en *embedded systems*, tales como *smart cards*, cajeros, teléfonos, entre otros.

2.1. Aritmética en el campo GF(2^m) con base polinomial

2.1.1. Conceptos de bases polinomiales

Si se considera una extensión finita $F=F_{q^m}$ del campo finito $F=F_q$ como un espacio vectorial sobre K, entonces F tiene dimensión m sobre K, y si $(\alpha_1, \alpha_2, ... \alpha_m)$ es una base de F sobre K, entonces cada elemento $\alpha \in F$ se puede representar de forma única como: $\alpha = c_1 \alpha_1 + ...+ c_m \alpha_m$, donde $c_j \in K$, $1 \leq j \leq m$.

La aplicación de estas técnicas se da en diversos escenarios, basados en una arquitectura general denominada PKI (*public-key infrastructure*).

Se puede establecer la siguiente correspondencia lineal de F en K mediante la función traza $Tr_{F/K}(\alpha)$: sea K un campo finito y sea $Tr_{F/K}(\alpha)$ una extensión finita de K, la base polinomial $(1, \alpha, \alpha^2, ... \alpha^{m-1})$ se construye con las potencias de un elemento definitorio α de F sobre K, donde el elemento α es un elemento primitivo de F. Con esta base se representan los elementos de un campo finito mediante polinomios con coeficientes a_i, que pertenecen al campo K y las potencias de los elementos de la base. Cada polinomio que representa un término es una clase residual módulo el polinomio irreductible, es decir $F_{q^m} = \{a_0 + a_1 x + a_2 x^2 + ... + a_m x^m / a_i \in F_q\}$.

2.1.2. Multiplicación en base polinomial

Dos elementos $A, B \in GF(2^m)$ con polinomio irreducible $p(z)$ se representan en la forma de polinomios:

$$A = a_0 + a_1 x + a_2 x^2 + ... + a_m x^{m-1}$$
$$B = b_0 + b_1 x + b_2 x^2 + ... + b_m x^{m-1}$$

El producto $C = A*B$ se representa como:

$$C = c_0 + c_1 x + c_2 x^2 + ... + c_m x^{m-1}$$

De esta expresión puede notarse que el producto C tiene el mismo tamaño que los operandos A y B, lo cual se debe a que son elementos de un campo finito.

Para realizar la multiplicación entre dos elementos de un campo finito con representación en base polinomial, se realiza la multiplicación normal de polinomios, lo que origina un polinomio de grado máximo $2m-2$. Como puede observarse, este polinomio no pertenece al campo finito, por lo cual se implementa una reducción módulo el polinomio irreductible. Dado que la multiplicación de polinomios se puede realizar paso a paso, este tipo de multiplicadores se denominan seriales [4].

El otro enfoque toma como base el polinomio irreductible, al calcular las expresiones que generan cada uno de los coeficientes del resultado de la multiplicación, la cual, por tanto, se puede implementar en paralelo. Por esto, se denomina multiplicadores paralelos a este tipo de enfoques.

2.1.3. Adición en base polinomial

Dados dos elementos:

$$A = a_0 + a_1 x + a_2 x^2 + ... + a_m x^{m-1}$$
$$B = b_0 + b_1 x + b_2 x^2 + ... + b_m x^{m-1}$$

en el campo finito GF(2^m), entonces $A + B = C = c_0 + c_1 x + c_2 x^2 + ... + c_m x^{m-1}$, donde $c_i = (a_i + b_i) \bmod 2$, lo que equivale a la operación XOR bit a bit.

2.2. Aritmética de curvas elípticas

Las curvas elípticas son una importante familia de funciones que se originan al solucionar cierto tipo de ecuaciones en dos variables y otras funciones especiales. Particularmente, las curvas elípticas se estudian desde el enfoque de la teoría de curvas modulares y con base en ellas se han dado importantes desarrollos en la matemática, como el método de factorización de Lenstra y la demostración del último teorema de Fermat. Desde el punto de vista de la criptografía, son de interés las curvas elípticas definidas sobre un campo finito.

2.2.1. Curvas elípticas sobre GF(2^m)

Una curva elíptica es una ecuación en donde x y y son las indeterminadas, y los $a_{1,\ldots,}$ a_5 son elementos constantes de un campo. Para el caso de GF(2^m), es de interés la curva elíptica

$$E(F_{2^7}): y^2 + xy = x^3 + ax^2 + b \quad (1)$$

donde $a, b \in GF(2^m)$, $b \neq 0$.

Los puntos que satisfacen (1) se denominan *puntos racionales* y forman una estructura de grupo abeliano con respecto a la *suma de puntos,* con el punto O, llamado punto al infinito como elemento neutro.
La suma de puntos sobre la curva $E_{a,b}$ se define de la siguiente manera:

a. $O + O = O$.

b. $P + O = O + P = P$, para todo $P \in E(F2^m)$.

c. Si $P = (x, y) \in E(F2^m)$, entonces $(x, y) + (x, x + y) = O$. El punto $(x, x + y)$ se denota $-P$ y se llama el negativo de P. $-P$ es un punto de la curva.

d. Sean $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$ dos puntos de la curva $E(Z_p)$, donde $x_1 \neq x_2$, se tiene que $P_1 + P_2 = (x_s, y_s)$ es:

$x_s = \lambda^2 + \lambda + x^1 + x2 + a$

$y_s = \lambda(x_1 - x_s) + x_s + y_1$,

donde $\lambda = \dfrac{y_2 + y_1}{x_2 + x_1}$

e. Sea $P_1 = (x_1, y_1)$ un punto de la curva, con $y_1 \neq 0$. Entonces $P_1 + P_1 = (x_s, y_s)$,

donde $x_s = \lambda^2 + \lambda + a$, $y_s = x_1^2 + (\lambda + 1)x_s$

donde $\lambda = x_1 + \dfrac{y_1}{x_1}$

La suma y duplicación de puntos se implementan en términos de la aritmética de campos finitos.

En algunas aplicaciones es conveniente representar los puntos racionales en coordenadas proyectivas, lo cual provee una disminución en el número de inversiones requeridas, lo que se traduce en mayor facilidad de implementación. el National Institute of Standards and Technology (NIST), estandarizó las condiciones de las curvas elípticas para que sean eficientes en aplicaciones comerciales, en cuanto a la seguridad.

2.2.2. Problema del logaritmo discreto en curvas elípticas (ECDLP)

Sea E una curva elíptica definida sobre un campo finito $GF(2^m)$ y sea $G \in E(F_q)$ un punto sobre E de orden n (n un número primo y grande). El ECDLP, se plantea así: dados E y un múltiplo escalar Q de G, determinar un entero *a* tal que $Q = aG$.

2.2.3. Multiplicación de puntos

La fundamental y más costosa operación en el criptosistema de curvas elípticas es la *multiplicación de puntos* o *multiplicación escalar kP*, donde *k* es un entero y *P* es un punto de la curva elíptica. La multiplicación escalar se define en términos de la suma:

$$kP = \underbrace{P + P + \ldots + P}_{\text{k veces}} \quad (2)$$

Para realizar esta operación se usa el método de López-Dahab [5], el cual realiza el mismo tipo de operaciones (multiplicaciones y sumas) en cada ciclo, lo que permite que sea resistente a ataques por análisis de potencia. En términos de las operaciones en el campo finito, el costo aproximado del método es de 6 m + 20 s para calcular *kP*.

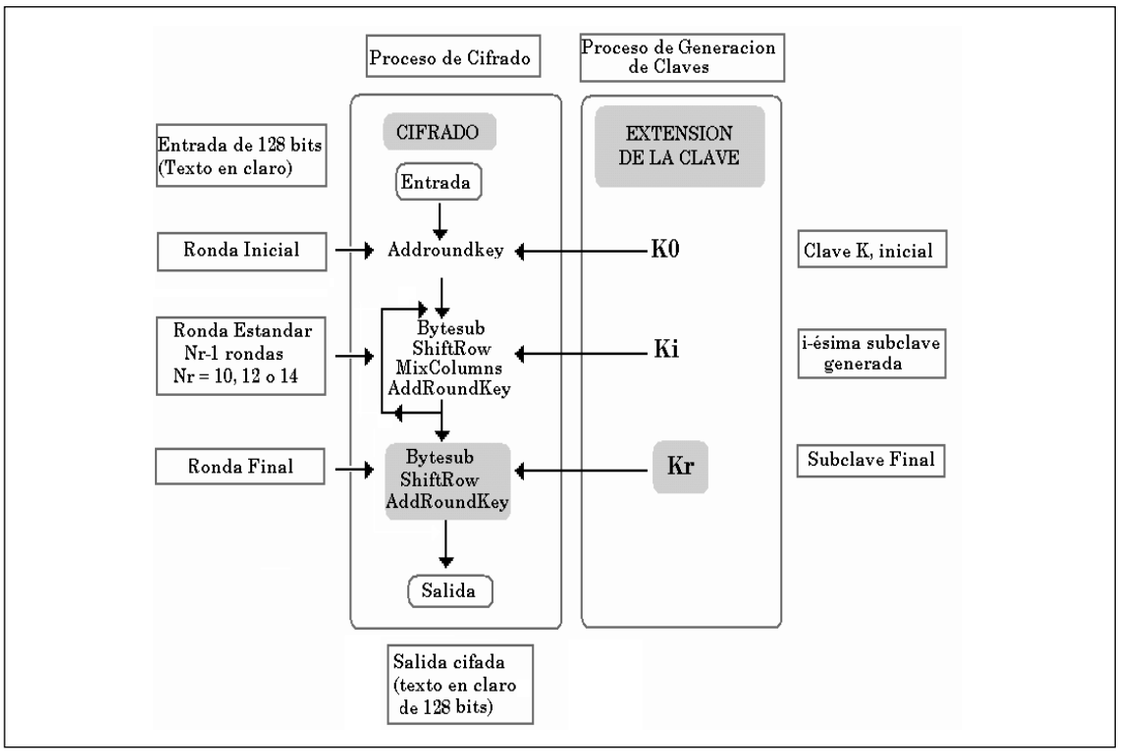
3. AES (Advanced Encryption Standard)

La estructura del algoritmo Rijndael está formada por un conjunto de rondas (conjunto de iteraciones de cuatro funciones matemáticas diferentes e invertibles). El algoritmo se basa en la aplicación de un número de rondas a una información en claro para producir una información cifrada.

Cada ronda está compuesta por cuatro operaciones basadas en transformaciones uniformes e invertibles llamadas “capas” –que

Las curvas elípticas son una importante familia de funciones que se originan al solucionar cierto tipo de ecuaciones en dos variables y otras funciones especiales.

Figura 1. Estructura del algoritmo Rijndael – AES (cifrado).



han sido diseñadas para resistir a los criptoanálisis lineal y diferencial–, las cuales son:

- *Capa de mezcla lineal:* Garantiza un alto nivel de difusión a lo largo de las múltiples rondas.
- *Capa no lineal:* Consiste en la aplicación de S-Cajas en paralelo que tiene propiedades óptimas de no linealidad.
- *Capa de adición de clave:* Se trata de una operación OR exclusivo entre el estado.

3.1. Bytesub: Sustitución de Byte

Consiste en una sustitución no lineal que se aplica a cada byte de la matriz de estado de forma independiente, generando un nuevo byte. Esta transformación consiste en la sustitución de cada byte por el resultado de aplicarle la tabla de sustitución S-Box. Esta tabla lógicamente es invertible.

3.2. *Shiftrow:* Desplazamiento de filas

En esta transformación se rotan cíclicamente las filas del estado intermedio varios lugares a la izquierda, dependiendo de la posición de la fila y el tamaño de bloque cifrado. Cuando son bloques de 128 o 196 bits, la primera fila (fila 0) no se desplaza, la fila 1 se desplaza cíclicamente un lugar, la fila 2 dos lugares y la fila 3 tres lugares. Cuando los bloques son de 256 bits, estos desplazamientos son algo diferentes: 0, 1, 3 y 4 lugares, respectivamente.

3.3. *Mixcolumn:* Mezcla de columnas

Esta transformación actúa sobre los bytes de una misma columna de la matriz de estado que tiene a la entrada. En realidad, esta función permite una mezcla de los bytes de las columnas. La transformación considera las columnas de bytes como polinomios cuyos coeficientes pertenecen a $GF(2^8)$, es decir, también son polinomios. La función MixColumn consiste en multiplicar las columnas de bytes módulo x^4+1 por el polinomio c(x). Matemáticamente, C(x) está representado por:

$$C(x)=03_{16}x^3+01_{16}x^2+01_{16}x+02_{16} \qquad (3)$$

Para descifrar o invertir esta transformación, se realiza el mismo procedimiento, pero con el polinomio d(x), que es el inverso de c(x).

3.4. *Addroundkey*

En esta última transformación se realiza una operación OR-Exclusiva entre la matriz de estado que proviene de la transformación anterior (función MixColumn) y la subclave generada a partir de la clave del sistema para esa ronda. El bloque resultante de esta transformación es la nueva matriz de estado para la siguiente ronda, hallando el bloque o dato de salida del algoritmo, si es la última ronda.

3.5. Función de expansión de clave

Esta función permite generar bytes útiles como subclaves a partir de la clave de sistema K. Además, se puede describir como un arreglo lineal, denominado W, de palabras de 4 bytes y con una longitud de Nb*(Nr+1). Las primeras Nk palabras de este arreglo contienen la clave de cifrado, ya que la clave del usuario se mapea en el arreglo W, mientras que el resto de palabras se van generando a partir de estas primeras Nk palabras [6].

3.6. Proceso de cifrado

El algoritmo para cifrar un bloque realiza primero la transformación Addrounkey de adición con la primera clave, luego se llevan a cabo varias rondas, cuyo número *n* es 9, si tanto el bloque como la clave son de 128 bits; 12 si la clave o el bloque son de 192 bits y ninguno de ellos supera esta longitud, y 13 si la clave o el bloque son de 256 bits. Se realizan las cuatro etapas que constituyen el ciclo básico de Rijndael, que son: Bytesub, Shiftrow, Mixcolumn y Addroundkey. Además se realiza un ciclo extra compuesto por las transformaciones Bytesub, Shiftrow, Addroundkey, pero no aparece la transformación Mixcolumn. Para obtener las subclaves que se aplican en cada ciclo se genera una serie de expansiones de la clave original, cada una del mismo tamaño que ésta [6].

3.7. Proceso de descifrado

En el proceso de descifrado se procede en el orden inverso al cifrado, realizando también en cada paso la operación inversa. La transformación Addroundkey, por estar constituida por una operación XOR, es inversa de sí misma, pero en el caso de las otras tres transformaciones es necesario usar la operación inversa.

4. Resultados

Las implementaciones se realizaron sobre un dispositivo XV5LX110T de Xilinx, usando una tarjeta de desarrollo XUPV5-LX110T de Digilent. Para el desarrollo y simulación se empleó el entorno ISE Design Suite 10.1 de Xilinx.

4.1. AES

La implementación hardware del algoritmo AES-Rijndael corresponde al tipo iterativo, pues se tiene un bloque Round único, el cual es usado para procesar iterativamente los datos obtenidos en cada una de las rondas. El funcionamiento comienza con el texto y la clave originales, luego un registro soporta el almacenamiento parcial realimentado, hasta llegar a completar las 10 iteraciones o rondas, contempladas para AES.

El boque Round consiste en la interconexión en cascada de los bloques que implementan cada una de las operaciones internas, de esta manera no se tiene ningún tipo de retardo para obtener la salida de la ronda. También es necesario contar con la clave de ronda correspondiente; esto se logra con el bloque KeySch (Key Schedule).

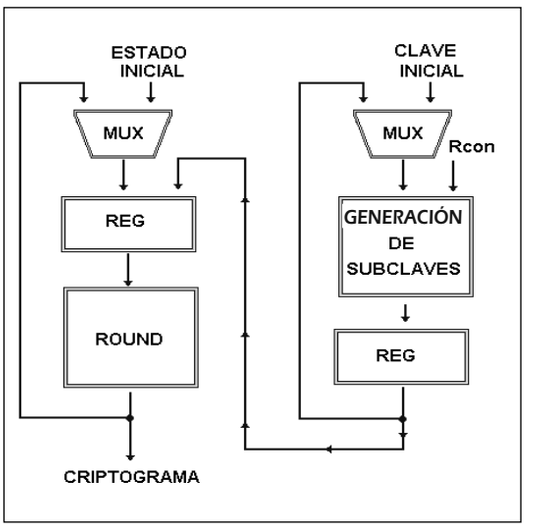
El boque Round es modular, totalmente desarrollado en VHDL lenguaje para descripción y modelado de circuitos, junto con el módulo InvRound, que implementa las rondas en el proceso de descifrado. El boque Round recibe un bloque de 128 bits correspondiente al estado del proceso, junto con la clave de ronda correspondiente y un bit que indica la realización de la transformación MixColumns, si está en 1, o la ausencia de esta transformación, en caso contrario.

El bloque KeySch se implementa completamente en paralelo, logrando una optimización significativa al no presentar retardos en el cálculo de la subclave correspondiente. A este bloque ingresan la clave anterior y el valor de Rcon correspondiente a la ronda, teniendo como salida la subclave que se lleva al bloque Round para su operación interna. Los registros de almacenamiento de matriz estado y subclave se habilitan mediante una señal activa alta por flanco, aprovechando la capacidad de describir este tipo de componentes en VHDL. El proceso completo de operación de las 10 rondas se controla mediante una máquina de estado finito FSM, que se encarga de las siguientes operaciones:

- En el estado inicial coloca el enable del registro de estado en 0 y el de subclave en 1. Igualmente genera el valor de Rcon para el bloque KeySch y coloca el bit de habilitación de la transformación MixColumns en 1, con la cual se efectúa la misma.
- En el estado siguiente coloca el *enable* de estado en 1 y el de subclave en 0. En este punto la subclave correspondiente queda almacenada en el registro.
- En el siguiente estado coloca ambos bit de *enable* en 0, terminando el ciclo correspondiente a una ronda.

Este proceso se repite para las 10 rondas, con una diferencia fundamental en la última, en la cual coloca un 0 en el habilitador de MixColumns, con lo que no se realiza esta transformación.

Figura 2. Arquitectura para el cifrado AES.



Para el proceso de descifrado se diseñó una arquitectura semejante, con las siguientes consideraciones:

- La entrada del bloque es el texto cifrado y la clave de cifrado original. A partir de esta se calcula la décima subclave (10), con la cual se inicia un proceso similar al del cifrado, hasta llegar a la clave inicial. Este proceso se implementa conectando en cascada 10 bloques KeySch y almacenando la salida en el registro correspondiente.
- El proceso interno del bloque Ronda, en el caso del descifrado, en inverso al del cifrado.
- La FSM de control es similar a la de cifrado, con excepción de la generación de la señal Rcon, que va en orden inverso, y del uso del bit de habilitación de MixColumns, que también se maneja en orden inverso, ya que en la primera ronda no se realiza dicha operación.

La Tabla 1 presenta la comparación de los resultados obtenidos con otras implementaciones reportadas.

Es posible considerar algunos factores que influyen en los resultados obtenidos. Existen ventajas inherentes a los dispositivos FPGA, dadas su arquitectura paralela y la posibilidad de procesamiento concurrente. Este hecho se evidenció en la implementación del prototipo, en el cual predominan los bloques paralelos y las conexiones masivas de elementos en cascada, logrando flujo de datos sin retardos significativos. De igual manera, se debe tener en cuenta la optimización lograda en el bloque KeySch, el cual se implementó completamente en paralelo, logrando la generación de la subclave directamente y sin retardos. Otro punto importante de optimización es el uso de multiplicadores paralelos en la transformación *mixcolumns* del descifrado, logrando también reducir el área ocupada por dicho bloque. Estos multiplicadores se desarrollan teniendo en cuenta las características del campo finito empleado.

4.2. Curvas elípticas

Se ha implementado en primer lugar la aritmética de campos finitos, específicamente sobre el campo GF(2163). Este campo está estandarizado por NIST y se considera aún seguro

Tabla 1. Comparación con otras implementaciones.

Autor	Cifrado		Descifrado	
	Throughput Mbps	CLs	Throughput	CLs
Liberatori [7]	637,24	1.584	500,28	2.506
Standaert et al. [8]	1450	542	No implementa	NI
Segredo et al. [9]	417	496	NI	NI
López [10]	6,8	2.500	4,5	2.400
J. López et al. [11]	1067,62	633		
Este trabajo	660	410	660	840

Figura 3. Algoritmo de López-Dahab.

ALGORITHM: LÓPEZ-DAHAB POINT MULTIPLICATION.

Input: $k=(k_{t-1}, ..., k_1, k_0)_2$, $k_{t-1} = 1$, $P(x, y) \in E(GF(2^{163}))$.
Output: *kP*.
1. If k = 0, or x = 0 then kP=(0,0) and stop
2. $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$.
   {compute $x(P)$, $x(2P)$}
3. ***For*** i from t-2 downto 0 do:
4. if $k_i = 1$ then:
5. $T \leftarrow Z_1$, $Z_1 \leftarrow (X_1 Z_2 + X_2 Z_1)^2$,
   $X_1 \leftarrow x Z_1 + X_1 X_2 T Z_2$
6. $T \leftarrow X_2$, $X_2 \leftarrow X_2^4 + b Z_2^4$, $Z_2 \leftarrow T^2 Z_2^2$
7. else:
8. $T \leftarrow Z_2$, $Z_2 \leftarrow (X_1 Z_2 + X_2 Z_1)^2$,
   $X_2 \leftarrow x Z_2 + X_1 X_2 T Z_1$
9. $T \leftarrow X_1$, $X_1 \leftarrow X_1^4 + b Z_1^4$, $Z_1 \leftarrow T^2 Z_1^2$
10. if $Z_1 = 0$, then $kP = (0,0) = 0_\infty$
11. $x_3 \leftarrow X_1 / Z_1$
12. if $Z_2 = 0$, then $kP = -P$ and stop
11. $y_3 \leftarrow (x + X_1/Z_1)[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1 Z_2)](xZ_1 Z_2)^{-1} + y$

para implementaciones prácticas. La operación de suma en el campo finito simplemente es una operación XOR bit a bit, entre dos palabras de 163 bits, operación que se implementa completamente en paralelo aprovechando las ventajas de los dispositivos FPGA. Para la multiplicación en el campo finito, se implementa un multiplicador bit-serial [12], el cual realiza la operación en 163 ciclos de reloj.

Una operación importante en campos finitos, que se requiere para la multiplicación escalar en curvas elípticas, es el cálculo del inverso multiplicativo. Para esta operación se implementa el algoritmo de Itoh-Tsuji [13]. Dado que este algoritmo emplea elevaciones al cuadrado, se implementa esta operación a través de una simplificación del multiplicador en campos finitos.

En cuanto a la aritmética de curvas elípticas, se implementa el algoritmo de López-Dahab para realizar la operación kP (multiplicación escalar). Este algoritmo emplea operaciones en el campo finito en cada una de sus etapas.

En la Figura 4 se presenta la arquitectura desarrollada.

El funcionamiento de la arquitectura se soporta en una (FSM) que controla las señales y el flujo de los datos requeridos.

Figura 4. Arquitectura del procesador EC.

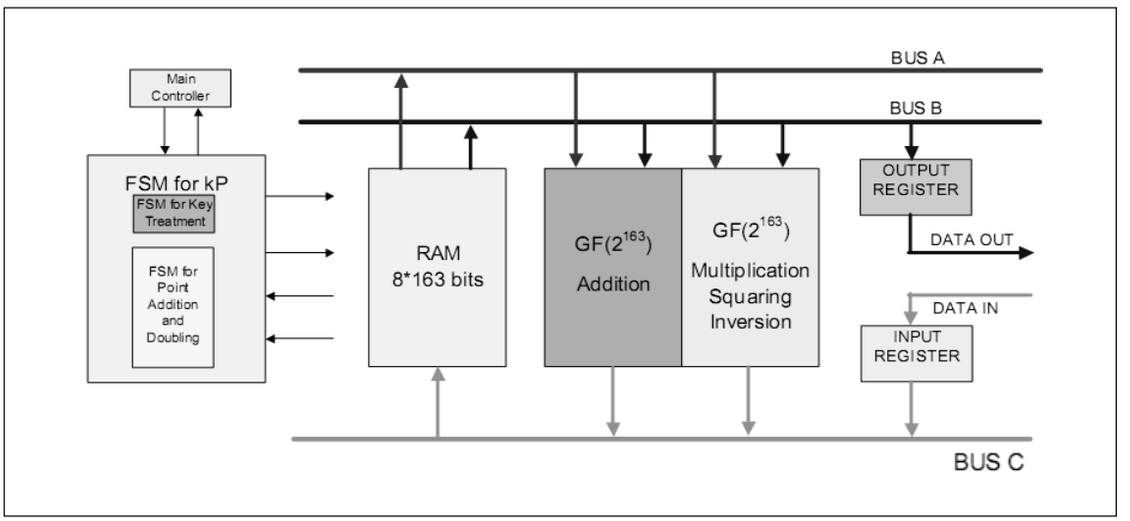


Tabla 2. Comparación con otras implementaciones de EC.

Autor	Dispositivo	Bits	kP (ms)	Gates (K)
Leung et al. [14]	FPGA XCV300	113	3,7	320
Ernst et al. [15]	FPGA XC4085XLA	155	1,3	180
Okada et al. [16]	FPGA EPF10K	163	80,7	310
Orlando et al. [17]	FPGA XCV400E	167	0,21	570
Trujillo et al. [18]	FPGA EP2A70B724C7	163	0,819	110
Este trabajo	FPGA XV5LX110T	163	0,75	100

5. Conclusiones

En el presente trabajo se obtuvieron importantes resultados implementando la arquitectura iterativa del algoritmo AES-Rijndael y las operaciones sobre curvas elípticas. En este caso se pudieron notar las ventajas de trabajar en hardware, sobre todo aprovechando las características de los dispositivos FPGA.

Si se compara el *throughput* y el desempeño en general de las aplicaciones en software y hardware reportadas, es evidente la superioridad de estas últimas, pues permiten una mayor tasa de transferencia de información para los procesos de cifrado y descifrado.

La mayor parte de la optimización resultante en las aplicaciones hardware se debe al uso del paralelismo, como en el caso de la transformación Bytesub de AES, que se realiza sin ningún tipo de retardo, mientras que en una aplicación software se requieren 16 operaciones secuenciales para AES-128, ya que está definida en el nivel de byte.

En la implementación de AES se lograron optimizaciones adicionales, principalmente por una comprensión de la naturaleza estructural de ciertos módulos, como en el caso del proceso de Key Schedule (generación de subclaves); esta mejor implementación redunda en una reducción de los recursos usados en la FPGA (área ocupada).

En la implementación de curvas elípticas se observa aún más la ventaja del uso de hardware reconfigurable, ya que las operaciones se realizan sobre bloques de 163 bits, sin posibilidad de subdividir en bloques más pequeños, tarea que se dificulta al usar las ALU de los microprocesadores convencionales. De esta manera se logra un mejor rendimiento en las diferentes operaciones, tanto en el campo finito como en la aritmética de curvas elípticas.

6. Trabajo futuro

El trabajo desarrollado hace parte de un proyecto en el cual se pretende implementar una arquitectura segura de voto telemático, soportada en FPGA, incluyendo una PKI integrada por AES, ECC y SHA (*Secure-hash algorithm*). El trabajo continuará integrando las implementaciones AES y EC logradas, aplicadas en un servidor web soportado en un sistema embebido sobre FPGA y un dispositivo USB que soporta el acceso de los clientes a través de Internet desde cualquier PC.

Una operación importante en campos finitos, que se requiere para la multiplicación escalar en curvas elípticas, es el cálculo del inverso multiplicativo.

Referencias

[1] M. Merino Martínez. "Una introducción a la criptografía. El criptosistema R.S.A". I.E.S Cardenal López de Mendoza, 2004.

[2] N. Koblitz. "Elliptic curve cryptosystems". En *Mathematics of Computation*. Vol. 48. Boston, American Mathematical Society, 1987, pp. 203-209.

[3] V. Miller. "Uses of elliptic curves in cryptography". *Advances in Cryptology.* Crypto 85, Proceedings, Lecture Notes in Computer Science, 218, Springer-Verlag, pp. 417-426, 1986.

[4] D. Hankerson, A. Menezes y S. Vanstone. "Guide to elliptic curve cryptography". Springer-Verlag, 2004.

[5] J. Lopez y R. Dahab. "Fast multiplication on elliptic curves over GF(2n) without precomputation". *Cryptographic Hardware and Embedded Systems*. CHES`99, Lecture Notes in Computer Science, 1717, 2002, pp. 316-327.

[6] J. Daemen y V. Rijmen. "AES Proposal: Rijndael". Rijndael Tecjnical Report, 1999. En línea: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf

[7] M. Liberatori. "Desarrollo de encriptado AES en FPGA". Tesis de Maestría en Redes de Datos, Universidad de la Plata, 2004.

[8] F. Standaert. "Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs". Cryptographic Hardware and Embedded Systems. CHES 2003, 2003, pp. 334-350.

[9] A. Segredo. "Diseño de un procesador criptográfico Rijndael en FPGA". X Workshop Iberchip, 2004, p. 64.

[10] E. López. "Implementación eficiente en FPGA del modo CCM usando AES". Tesis de Maestría en Ciencias, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México, 2005.

[11] J. López et al. "Implementación en hardware del algoritmo Rijndael". Grupo de Bioelectrónica y Nanoelectronica, Escuela IEEE, Universidad del Valle, 2003.

[12] D. Hankerson, A. Menezes y S. Vanstone. "Guide to Elliptic Curve Cryptography". Springer-Verlag, 2004, pp. 230-233.

[13] T. Itoh y S. Tsujii. "A fast algorithm for computing multiplicative inverses in GF(2m) using nomal bases". *Information and Computation*, 78 (1988): 171-177.

[14] K. H. Leung, K. W. Ma, W. K. Wong y P. H. W. Leong. "FPGA Implementation of a microcoded elliptic curve cryptographic processor". Proc. IEEE FCCM 2000, pp. 68-76, Napa Valley.

[15] M. Ernst, S. Klupsch, O. Hauck y S. A. Huss. "Rapid prototyping for hardware accelerated elliptic curve public-keycryptosystems". Proc. 12th

IEEE Workshop on Rapid System Prototyping (RSP01), Monterey, CA, 2001.

[16] S. Okada, N. Torii, K. Itoh y M. Takenaka. "Implementation of Elliptic Curve Cryptographic Coprocessor over GF(2m) on an FPGA". C. K. Koc y C. Paar (eds.). Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science, 1965, Springer-Verlag, 2000, pp. 25-40.

[17] G. Orlando. "Efficient elliptic curve processor architectures for field programmable logic". Tesis, Worcester Polytechnic Logic, 2002.

[18] V. Trujillo et al. "Design of an elliptic curve cryptoprocessor over GF(2^{163})". XI Workshop Iberchip, Salvador de Bahía, marzo de 2005.