Evaluación de implementaciones en software de algoritmos para la multiplicación escalar en criptografía de curvas elípticas

Vega C. Karina, Cortina R. Antonio y Morales S. Miguel

Evaluation of software implementations of algorithms for scalar multiplication in elliptic curve cryptography

***Abstract*— This work presents an evaluation of different algorithms to implement in software the most time demanding operation in ECC, the scalar multiplication. The results presented in this work could help a designer to select the most appropriate method when implementing ECC-based cryptographic schemes such as encryption or digital signatures. Different scalar multiplication algorithms for ECC defined over prime and binary fields are considered, using affine coordinates to represent the elliptic curve points. The evaluation was performed over two computing platforms. The first one is a computer powered with an Intel Core i5 processor at 2.50 GHz. The second one is a mobile device LG P500h with an ARM processor at 600 MHz. The performance evaluation consisted of comparing the timing for computing a scalar multiplication in an ECC key generation scheme. For ECC defined over binary fields, it was found that the *NAF* and *wNAF* algorithms run 1.6 times faster than the *Double&Add* method both on the PC and the mobile device. This same speed-up is observed when these two algorithms are implemented for elliptic curves defined over the prime field and executed on the PC. However, when the *wNAF* algorithm is executed on the mobile device, it is 1.2 times slower that the *Double&Add* method and 1.5 times slower than the *NAF* algorithm.**

***Keywords*— Finite fields, Cryptographic schemes based on elliptic curve, Scalar multiplication.**

***Resumen*— Este trabajo presenta una evaluación de diversas implementaciones en software de algoritmos para calcular la operación más demandante en esquemas criptográficos basados en Criptografía de Curvas Elípticas (ECC), la multiplicación escalar. Los resultados presentados en este trabajo podrán servir como un punto de referencia para quienes implementan esquemas criptográficos basados en ECC tales como esquemas de cifrado o de firma digital. En esta investigación se usaron diferentes algoritmos definidos tanto en campo primo como en campo binario empleando coordenadas afines. Las pruebas se efectuaron en una computadora de escritorio con un procesador Intel Core i5 con 2.50 GHz de velocidad, así como en el dispositivo móvil LG P500h con un procesador ARM a 600 MHz. Tras evaluar los tiempos de ejecución para la generación de una llave pública dada una llave privada previamente establecida, se encontró que los algoritmos *NAF* y *wNAF* se ejecutan 1.6 veces más rápido que el método de Suma y Doblado, tanto en la PC como en el dispositivo móvil. Sin embargo, en el caso particular del método *wNAF* sobre campo primo implementado en el dispositivo móvil, se observó que éste se ejecuta 1.2 veces más lento que el método de Suma y Doblado y 1.5 veces más lento que *NAF*.**

***Palabras clave*— Campos finitos, Esquemas criptográficos basados en curva elíptica, Multiplicación escalar**

Manuscrito recibido el 20 de Agosto de 2012. Este trabajo fue respaldado por la Universidad Politécnica de Cd. Victoria, Tamaulipas.

Vega C. Ana K. hasta la fecha se ha de desempeñado como estudiante en el programa de Maestría en Ingeniería con especialidad en Tecnologías de la Información de la Universidad Politécnica de Cd. Victoria; Av. Nuevas Tecnologías 5902 Parque TECNOTAM, Carretera Victoria - Soto la Marina Km. 5.5; Ciudad Victoria, Tamaulipas, México; C.P. 87138; (e-mail anakarynavega@hotmail.com).

Cortina R. Antonio hasta la fecha se ha de desempeñado como estudiante en el programa de Maestría en Ingeniería con especialidad en Tecnologías de la Información de la Universidad Politécnica de Cd. Victoria; Av. Nuevas Tecnologías 5902 Parque TECNOTAM, Carretera Victoria - Soto la Marina Km. 5.5; Ciudad Victoria, Tamaulipas, México; C.P. 87138;(e-mail antoniocr06@hotmail.com).

Morales S. Miguel hasta la fecha se ha desempeñado como Profesor de Tiempo Completo de la Universidad Politécnica de Cd. Victoria; Av. Nuevas Tecnologías 5902 Parque TECNOTAM, Carretera Victoria - Soto la Marina Km. 5.5 Ciudad Victoria, Tamaulipas; C.P. 87138; Tel: (834) 1720383, Fax: (834) 1720388; (e-mail mmoraless@upv.edu.mx).

I. INTRODUCCIÓN

Gracias al avance de la tecnología, hoy en día es posible contar con plataformas móviles y de escritorio con capacidades de procesamiento cada vez superiores, lo que ha dado lugar a aplicaciones más potentes, las cuales en muchas ocasiones demandan la garantía de servicios de seguridad informática, tales como confidencialidad, integridad, autenticación, etc., debido a que la información que administran es de carácter sensible, por lo que es indispensable el contar con un esquema criptográfico confiable, robusto y eficiente, que permita brindar dichos servicios. Recientemente las técnicas de criptografía basadas en curvas elípticas (ECC) y emparejamientos bilineales han demostrado la viabilidad de proveer servicios de seguridad informática de manera eficiente tanto en plataformas de escritorio como en el dominio de las aplicaciones móviles [1, 2], debido a que permiten la utilización de longitudes de llaves más cortas que esquemas tradicionales de criptografía, como el algoritmo RSA. Al usar llaves más cortas, el espacio de memoria utilizado para el almacenamiento de llaves se reduce considerablemente así como también los requerimientos de ancho de banda para la transmisión de las llaves, además de ser capaces de proveer niveles de seguridad confiables [3].

En la literatura se han realizado estudios comparativos de algoritmos que calculan la operación más demandante en esquemas criptográficos basados en ECC, la multiplicación escalar, en base a la cantidad de operaciones que dichos algoritmos efectúan [4, 5]. En este artículo se describe el trabajo experimental para evaluar el desempeño de algoritmos para la multiplicación escalar a partir de los tiempos de ejecución, haciendo una evaluación bajo las mismas condiciones de prueba, es decir, los algoritmos fueron evaluados sobre la misma plataforma de cómputo, ya sea PC o dispositivo móvil, y usando las mismas rutinas de software.

La aportación principal de este trabajo consiste en proveer un análisis de los tiempos de ejecución de diversos algoritmos existentes para el cálculo de la multiplicación escalar en ECC, que sirva como un punto de referencia para quienes implementan esquemas criptográficos basado en ECC tanto en computadoras de escritorio como en dispositivos móviles.

Se realizó una revisión de la literatura y se seleccionaron diversos algoritmos tanto para campos binarios *GF(2^m)* como para campos primos *GF(p)*. La implementación y pruebas se efectuaron tanto en un ordenador de escritorio, como en un dispositivo móvil.

Las secciones restantes del artículo se ordenan como sigue: el marco teórico de esta investigación se describe en la sección II; La multiplicación escalar y los algoritmos para calcular esta operación se describen en la sección III. Los resultados obtenidos se detallan en la sección IV; finalmente en la sección V se presentan las conclusiones de este trabajo.

II. CRIPTOGRAFÍA DE CURVAS ELÍPTICAS (ECC)

La ECC pertenece a la criptografía asimétrica [6], esto debido a que se utilizan dos claves distintas: una pública y una privada, donde el conocimiento de la clave pública no permite determinar la clave privada.

ECC fue propuesta de manera independiente en 1985 por Neal Koblitz [7] y Victor Miller [8]. Desde entonces una gran cantidad de investigaciones se han realizado para tener implementaciones eficientes y seguras de estos esquemas criptográficos. La Criptografía de Curvas Elípticas ha permitido explorar nuevos criptosistemas, tal como la técnica de emparejamientos bilineales [1].

A. Campos finitos

Una curva elíptica sobre un campo *K* es un conjunto formado por el punto al infinito ∞ y los puntos P=(x,y)∈K×K que satisfacen la ecuación de Weistrass (1) [4]:

$$E(K): y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

La curva elíptica $E(K)$ junto con el punto al infinito (∞) forman el grupo aditivo $\{E(K) \cup \infty, +\}$, donde el punto ∞ es el elemento identidad $(P + \infty = P)$. Para aplicaciones criptográficas, el campo K es un campo finito. Si Kes un campo primo *GF(p)* la ecuación de la curva elíptica es (2):

$$E(GF(p)):\ y^2 = x^3 + ax + b \qquad (2)$$

Cuando la curva elíptica está definida en el campo binario *GF(2^m)*, la ecuación es (3):

$$E(GF(2^m)): y^2 + xy = x^3 + ax^2 + b \qquad (3)$$

B. Problema del logaritmo discreto elíptico

Dada una curva E en un campo finito, podemos representar la operación principal llamada multiplicación escalar en ECC de la siguiente manera:

$$Q = dP$$

donde:

$\boldsymbol{P}$ *y* $\boldsymbol{Q}$**:** Son puntos de una curva elíptica.

$\boldsymbol{d}$: Es un escalar secreto.

Dicho esto podemos definir al Problema de Logaritmo discreto Elíptico (PLDE) como determinar el escalar d, dado los puntos P y Q.

La seguridad basada en ECC es basada en la dificultad de resolver este problema. En general la dificultad del PLDE resulta ser más difícil que otros problemas como el de factorización de enteros y logaritmo discreto [9].

C. Estándares

Un criptosistema de curva elíptica se define a partir de una tupla *T*, la cual es un conjunto de parámetros que define un campo finito de trabajo, la ecuación de una curva elíptica (ver ecuación 1) definida sobre el campo finito sobre la que se va a trabajar, un generador de la curva elíptica, entre otros valores.

Para el campo primo *GF(p)* los parámetros son una séxtupla de valores denotados como [3]:

$$T = (p, a, b, G, n, h)$$

donde:

$\boldsymbol{p}$: Es un número primo grande.

$\boldsymbol{a}, \boldsymbol{b}$: Son coeficiente que definen la curva E en *GF(p)*.

$\boldsymbol{G}$: Es el generador de un subgrupo cíclico en la curva elíptica *E(GF(p))*.

$\boldsymbol{n}$: Es el orden de G (n es el entero más pequeño tal que $nG = \infty$).

$\boldsymbol{h}$: Es el cofactor de la curva y se define como el $número\ de\ puntos\ /n$. Este valor es opcional.

Para el campo Binario *GF(2^m)* la tupla T consiste de siete parámetros necesarios definidos como [3]:

$$T = (m, f(x)\ a, b, G, n, h)$$

donde :

$\boldsymbol{m}$: Es el entero que especifica el orden del campo finito que se está usando.

$\boldsymbol{f(x)}$: Es el polinomio irreducible de grado m.

Todos los demás valores tienen una definición similar al caso de $GF(p)$.

Existen diversos estándares que especifican un conjunto de valores para cada elemento de la tupla *T*. Dichos parámetros han sido derivados a través de métodos ampliamente estudiados y por ello se consideran seguros. Dentro de los estándares se

encuentran IEEE [10], NIST [11], ANSI [12, 13], ISO [14], SECG [15], entre otros. Los estándares de ECC en su mayoría son compatibles además de contar con vectores de prueba con los cuales se puede verificar los resultados.

D. Esquemas Criptográficos

Para lograr contar con los servicios de seguridad que las aplicaciones móviles y de escritorio necesitan, es indispensable utilizar esquemas criptográficos. Entre los principales basados en ECC se pueden mencionar [16]:

1) Esquema de Diffie-Hellman en Curvas Elípticas (ECDH). Esquema que permite la generación y el intercambio de llaves entre dos entidades.

2) Esquema de Cifrado Integrado en Curva Elíptica (ECIES). Utilizado para el cifrado de datos. Dentro de este esquema se utiliza ECDH para generar dos llaves simétricas, una utilizada para cifrar el texto claro y otra para autentificar el texto cifrado.

3) Algoritmo de Firma Digital de Curva Elíptica (ECDSA). Este esquema puede ser divido en dos etapas, la primera sería la generación de firma digital y la segunda la verificación de firma digital, donde en cada una de estas etapas se vería implicado el uso de una llave privada y una función hash.

Algoritmo 1: Generación de firma con ECDSA

Requiere: *(GF(q),a,b,G,n,h)* llave privada d_a, mensaje *m*
Salida: Firma digital *(r, s)*
1: Seleccionar $k \in [1, n-1]$
2: $(X_1, Y_1) \leftarrow kP$
3: $r \leftarrow X_1 \ mod \ n$. Si $r = 0$ regresar al paso 1
4: $e \leftarrow H(m)$
5: $s \leftarrow k^{-1}(e + d_a r) mod \ n$. Si $s = 0$ regresar al paso 1
6: Regresar (r, s)

Algoritmo 2: Verificación de firma con ECDSA

Requiere: *(GF(q),a,b,G,n,h)*, llave pública D_a, mensaje *m*, firma digital *(r, s)*
Salida: Aceptación o rechazo de la firma digital *(r, s)*
1: Verificar que *(r, s)* $\in [1, n-1]$
2: $e \leftarrow H(m)$
3: $w \leftarrow s^{-1} \ mod \ n$
4: $u_1 \leftarrow ew \ mod \ n$; $u_2 \leftarrow rw \ mod \ n$
5: $(X_1, Y_1) \leftarrow u_1 G + u_2 D_a$
6: $v \leftarrow X_1 \ mod \ n$
7: **Si** $v = r$ Regresar "Firma aceptada"
9: **Si no** Regresar "Firma rechazada"

III. MULTIPLICACIÓN ESCALAR

Para una comprensión más sencilla de la multiplicación escalar, su realización puede ser dividida en tres capas independientes [3].

A. Capa superior

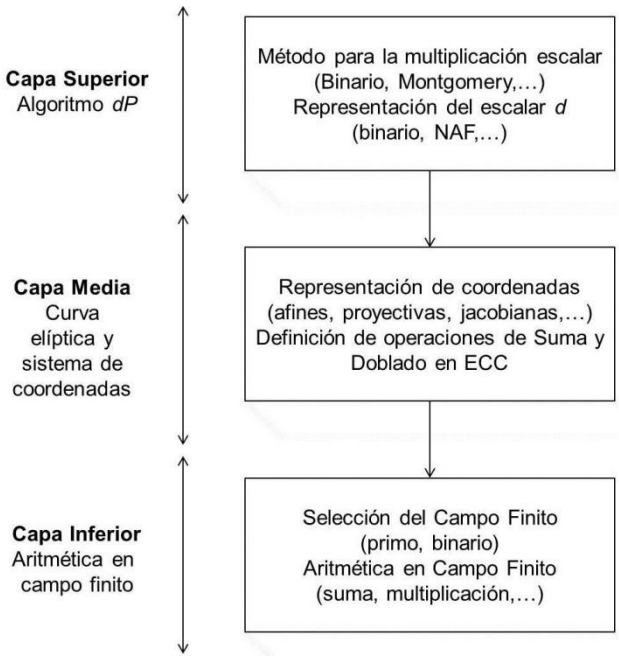

Fig. 1 Modelo de capas para la implementación de la multiplicación escalar.

En la capa superior se encuentran los diferentes métodos para efectuar la multiplicación escalar.

La multiplicación escalar dP donde d es un entero en el intervalo $[1, n-1]$ y P es un punto en la curva elíptica E definida sobre el campo finito, es el resultado de sumar $P + P + P + \cdots + P$, $d - 1$ veces [17]. Esta operación es realizada aplicando una secuencia de Sumas-ECC y Doblados-ECC.

El método básico para efectuar la multiplicación escalar para cuando P es un punto desconocido es el método de Suma y Doblado, donde la representación binaria del escalar d es usada. Existen dos algoritmos para calcular *dP* en función de cómo se recorren los bits del escalar *d*, de izquierda a derecha o de derecha a izquierda.

Multiplicación Doblado y Suma Izquierda-Derecha. El algoritmo 3 realiza en promedio $(m-1)/2$ Sumas-ECC y $m-1$ Doblados-ECC, debido a que el número esperado de unos en la representación binaria del escalar d es $t/2 \approx m/2$, donde t es el número de bits del escalar d.

Algoritmo 3: Multiplicación Doblado y Suma Izquierda-Derecha

Requiere: $d = (d_{t-1}, \ldots, d_1, d_0)_2$, $P \in E(GF_q)$
Salida: dP
1: $Q \leftarrow P$
2: **Para** $i = t-2$ hasta 0 **hacer**
3: $Q \leftarrow 2Q$
4: **Si** $d_i = 1$ **entonces**
5: $Q \leftarrow Q + P$
6: **Fin Si**
7: **Fin Para**
8: Regresar (Q)

Ya que este método itera de izquierda a derecha y el bit más significativo del escalar d es uno, el ciclo se realiza desde t-2 hasta 0, ahorrando así una Suma-ECC y un Doblado-ECC.

Multiplicación Doblado y Suma Derecha-Izquierda. El algoritmo 4 realiza en promedio $m/2$ Sumas-ECC y m Doblados-ECC. Debido a que este método se realiza de derecha a izquierda, se debe comenzar a iterar desde 0 hasta $t-1$.

Algoritmo 4: Multiplicación Doblado y Suma Derecha-Izquierda
Requiere: $d = (d_{t-1}, \dots, d_1, d_0)_2$, $P \in E(GF_q)$
Salida: dP
1: $Q \leftarrow \infty$
2: **Para** $i = 0$ hasta $t-1$ **hacer**
3: **Si** $d_i = 1$ **entonces**
4: $Q \leftarrow Q + P$
5: **Fin Si**
6: $P \leftarrow 2P$
7: **Fin Para**
8: Regresar (Q)

En los algoritmos 3 y 4, se efectúa un Doblado-ECC de manera incondicional y la operación de Suma-ECC se lleva a cabo solo si el bit d_i es uno.

Para mejorar el tiempo de ejecución de la multiplicación escalar se han diseñado nuevos métodos, algunos de los cuales logran disminuir la cantidad de Sumas-ECC, al reducir el número de dígitos diferentes de cero del escalar d, como por ejemplo los métodos *NAF* [18] y *wNAF* [19, 20], que recodifican d de forma que se incrementan los símbolos con los que se puede representar d.

Multiplicación Binaria NAF. El método *NAF* (Non-Adjacent Form) de un entero positivo d es una expresión de la forma $d = \sum_{i=0}^{l-1} d_i 2^i$ donde $d_i \in \{0, \pm 1\}, d_{l-1} \neq 0$ y al menos dos bits consecutivos d_i son diferentes de cero. La longitud de la representación *NAF* es representada por l.

Algunas de las propiedades de la representación *NAF* de un entero positivo *d* son [4]:

1. d tiene una única representación NAF denotada por $NAF(d)$
2. $NAF(d)$ tiene menos dígitos diferentes de cero que la representación binaria de d
3. La longitud de $NAF(d)$ es a lo más un bit mayor que la longitud de la representación binaria de d
4. Si la longitud de $NAF(d)$ es l, entonces $2^l/3 < d < 2^{l+1}/3$
5. La densidad promedio de dígitos diferentes de cero de $NAF(d)$ es aproximadamente 1/3

El valor $NAF(d)$ puede ser calculado eficientemente usando el algoritmo 5.

Algoritmo 5: Representación NAF
Requiere: Entero positivo d
Salida: $NAF(d)$
1: $i \leftarrow 0$
2: **Mientras** $d \geq 1$ **hacer**
3: **Si** d es impar **entonces**
4: $d_i \leftarrow 2 - (d \bmod 4), d \leftarrow d - d_i$
5: **Sino**
6: $d_i \leftarrow 0$
7: $d \leftarrow d/2, i \leftarrow i + 1$
8: **Fin Si no**
9: **Fin Mientras**
10: Regresar $(d_{i-1}, d_{i-2}, \dots, d_1, d_0)$

Posteriormente el algoritmo 6 se puede utilizar para obtener la multiplicación escalar. Este algoritmo realiza en promedio $(m-1)/3$ Sumas-ECC o restas de puntos y $m-1$ Doblados-ECC, gracias a que la densidad de dígitos diferentes de cero es de dos ceros por cada dígito diferente cero.

Algoritmo 6: Multiplicación binaria NAF de Izquierda-Derecha
Requiere: d, $P \in E(GF_q)$
Salida: dP
1: Obtener $NAF(d) = \sum_{i=0}^{l-1} d_i 2^i$
2: $Q \leftarrow P$
3: **Para** $i = l-2$ hasta 0 **hacer**
4: $Q \leftarrow 2Q$
5: **Si** $d_i = 1$ **entonces** $Q \leftarrow Q + P$
6: **Si** $d_i = -1$ **entonces** $Q \leftarrow Q - P$
7: **Fin Para**
8: Regresar (Q)

Como el método va de izquierda a derecha, nuevamente puede ahorrarse una Suma-ECC o una resta de puntos y un Doblado-ECC al igual que el algoritmo 1.

Multiplicación Binaria wNAF. El método *wNAF* (window Non-Adjacent Form) permite reducir aún más la densidad de dígitos diferentes de cero del escalar d, por lo que disminuye la cantidad de Sumas-ECC requeridas para multiplicación escalar.

Algunas de las propiedades de la representación *wNAF* de un entero d son [4]:

1. d tiene una única representación *wNAF* denotada por $NAF_w(d)$
2. $NAF_2(d) = NAF(d)$
3. La longitud de $NAF_w(d)$ es a lo más un dígito mayor que la longitud de la representación binaria de d
4. La densidad promedio de dígitos diferentes de cero de $NAF_w(d)$ es aproximadamente $1/(w+1)$

El valor $NAF_w(d)$ puede ser calculado mediante el algoritmo 7.

Algoritmo 7: Representación wNAF
Requiere: Entero positivo d
Salida: $NAF_w(d)$
1: $i \leftarrow 0$
2: **Mientras** $d \geq 1$ **hacer**
3: **Si** d es impar **entonces**
4: $d_i \leftarrow d \bmod 2^w, d \leftarrow d - d_i$
5: **Si no**
6: $d_i \leftarrow 0$
7: $d \leftarrow d/2, i \leftarrow i + 1$
8: **Fin Si no**
9: **Fin Mientras**
10: Regresar $(d_{i-1}, d_{i-2}, \dots, d_1, d_0)$

La función *mods* es definida de manera distinta en el campo primo y binario. Los algoritmos 8 y 9 calculan esta función.

Algoritmo 8: Función mods campo primo
Requiere: d, w
Salida: $d \bmod 2^w$
1: **Si** $d \bmod 2^w \geq 2^w/2$

2: Regresar (d mod 2^w) $- 2^w$
3: **Si no**
4: Regresa d mod 2^w

Algoritmo 9: Función mods campo binario
Requiere: d, w
Salida: $d\ mods\ 2^w$
1: **Si** d $\geq 2^{w-1}$
2: Regresar (d mod 2^w) $- 2^w$
3: **Si no**
4: Regresa d mod 2^w

El algoritmo 10 muestra la multiplicación escalar de izquierda a derecha utilizando la representación *wNAF*. Este algoritmo realiza en promedio $m/(w+1)$ Sumas-ECC o restas de puntos y m Doblados-ECC.

Algoritmo 10: Multiplicación wNAF de Izquierda-Derecha
Requiere: ancho w, entro positivo $d, P \in E(GF_q)$
Salida: dP
1: Obtener $wNAF(d) = \sum_{i=0}^{l-1} d_i 2^i$
2: Precalcular $P_i = iP$ para $i \in \{1,3,\dots,2^{w-1}-1\}$
3: $Q \leftarrow \infty$
4: **Para** $i = l-1$ hasta 0 **hacer**
5: $Q \leftarrow 2Q$
6: **Si** $d_i \neq 0$ **entonces**
7: **Si** $d_i > 0$ **entonces**
8: $Q \leftarrow Q + P_{ki}$
9: **Si no**
10: $Q \leftarrow Q - P_{-ki}$
11: **Fin Si**
12: **Fin Si**
13: **Fin Para**
14: Regresar (Q)

B. Capa media

La capa media corresponde al sistema de coordenadas en las cuales los puntos de la curva elíptica son representados. Existen diferentes tipos de coordenadas con las que es posible representar los puntos en la curva elíptica. Las más populares son las coordenadas afines, en donde cada punto en la curva es representado por el par *(x, y).* En este trabajo se utiliza este tipo de representación.

Esta capa además define como son realizadas las operaciones de suma y doblado en la curva elíptica.

Cuando la curva está definida sobre el campo primo $E(GF(p))$, las operaciones Suma-ECC y Doblado-ECC son definidas como sigue [21]:

Sea $a, b \in GF(p)$ que satisfacen la ecuación $4a^3 + 27b^2 \neq 0$. Sea $P, Q, R \in E(GF(p))$. $P = (x_1, x_1)$

$Q = (x_2, x_2)$ $R = (x_3, x_3)$

1. $P + Q = \infty$, si $P = \infty$ o $Q = \infty$
2. $Suma - ECC(P \neq \pm Q)$

$R = P + Q$, donde

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

3. $Doblado - ECC(P = Q)$

$R = P + Q = 2P$, donde

$$x_3 = \lambda^2 - 2_{x_1}$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

Cuando la curva elíptica está definida sobre el campo finito $E(GF(2^m))$, las operaciones de Suma-ECC y Doblado-ECC se realizan como sigue [21]:

Dado los puntos:

$P = (x_1, x_1), Q = (x_2, x_2),$ $R = (x_3, x_3) \in E(GF(2^m))$

1. $P + Q = \infty$, si $P = \infty$ o $Q = \infty$
2. $Suma - ECC(P \neq \pm Q)$

$R = P + Q$, donde

$$x_3 = \lambda^2 + \lambda + a$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$
$$\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

3. $Doblado - ECC(P = Q)$

$R = P + Q = 2P$, donde

$$x_3 = \lambda^2 + \lambda + a$$
$$y_3 = x_1^2 + \lambda x_3 + x_3$$
$$\lambda = x_1 + \frac{y_1}{x_1}$$

C. Capa inferior

Finalmente la capa inferior involucra aritmética en campo finito. La implementación eficiente de estas operaciones aritméticas impactan el desempeño de la multiplicación escalar.

Para realizar la Suma-ECC y el Doblado-ECC de la capa media se requieren operaciones de la capa inferior como suma, multiplicación, inversión (puede ser sustituida por la división) y elevación al cuadrado.

Para $GF(2^m)$ la implementación de las operaciones en campo finito dependen de una base, la cual puede ser polinomial, normal o dual [21]. En bases polinomiales, los elementos de $GF(2^m)$ son vistos como polinomios $A(x)$ de grado $m-1$, con coeficientes en $GF(2) = \{1,0\}$. Una base en $GF(2^m)$ es representada por $\{1, t, t_1, t_2, \dots, t_{m-1}\}$, donde t es un cuadrado de un polinomio irreducible $F(x)$ de grado m [9] y fue la base utilizada en la realización de pruebas de este artículo. La aritmética en $GF(2^m)$ con base polinomial es aritmética de polinomios modulo $f(x)$. Para $GF(p)$, la aritmética es implementada como aritmética de enteros modulo p.

IV. IMPLEMENTACIÓN Y RESULTADOS

La implementación de la multiplicación escalar y la realización de pruebas se llevo a cabo en una computadora de escritorio y en un dispositivo móvil, cuyas características se describen en la tabla I.

TABLA I. CARACTERÍSTICAS DE CÓMPUTO DE LA PC Y EL DISPOSITIVO MÓVIL

Parámetros	**Especificaciones**	
	PC	Móvil
Procesador	Intel Core i5 2.5GHz	ARM 600 MHz
SO	Windows 7	Android 2.2
Memoria RAM	4 GB	170 MB

TABLA II. ESPECIFICACIÓN DE PARÁMETROS DE LA CURVA SECP256R1

Parámetro	Valor
$\boldsymbol{p}$	FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF
$\boldsymbol{a}$	FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC
$\boldsymbol{b}$	5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B
$\boldsymbol{G}$	(6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296, 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5)
$\boldsymbol{n}$	FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551
$\boldsymbol{h}$	01

La plataforma de programación utilizada fue Java Edición Estándar (o Java SE 7) y el entorno de desarrollo utilizado fue Eclipse.

Un aspecto fundamental al momento de seleccionar las curvas elípticas con las que se realizó la etapa de implementación y pruebas es el nivel de seguridad que éstas ofrecen, es por ello que siguiendo la recomendación del SECG [15], este trabajo se enfocó en desarrollar un análisis de un nivel de seguridad de 128 bits tanto para campo primo como para campo binario. Los resultados alcanzados pudieron ser corroborados en los vectores de prueba que el estándar SECG [15] proporciona.

Para la obtención de resultados, y debido a la variación del tiempo respecto a la generación de la llave pública, se optó por seguir el teorema del límite central el cual menciona que para obtener una distribución normal es suficiente evaluar un mínimo de 30 muestras [22]. Para la realización de pruebas, en este artículo se tomaron 32 muestras de cada uno de los cuatro algoritmos analizados para el cálculo de la multiplicación escalar (ver tablas III y V), y posteriormente se obtuvo el promedio de cada uno de ellos.

A. Campo primo

Para evaluar los métodos de multiplicación escalar definido sobre el campo primo, para la tupla $T = (p, a, b, G, n, h)$ se utilizaron los valores recomendados para la curva secp256r1, la cual está recomendada en el estándar SECG [15], y es compatible con IEEE [10] y esta también recomendada en ANSI [12, 13] y NIST [11]. Los valores para cada elemento de la tupla *T* se muestran en la tabla II.

Para la implementación de la multiplicación escalar sobre campo primo se utilizaron clases incluidas en la versión 1.2 de la Arquitectura Criptográfica de Java o JCA, específicamente las clases incluidas en el paquete java.security.spec el cual brinda soporte para esquemas criptográficos y que fue utilizado para implementar las operaciones de la capa superior y la capa media de la multiplicación escalar. También, se utilizó la clase `BigInteger` con la cual fue posible implementar las operaciones requeridas en la capa inferior de la multiplicación escalar, ya que permite realizar operaciones sobre números grandes necesarios para la representación de parámetros criptográficos.

TABLA III. TIEMPO DE EJECUCIÓN DE LA MULTIPLICACIÓN ESCALAR SOBRE CAMPO PRIMO

	Tiempo de ejecución (ms)	
	PC	Móvil
S&D Izq-Der	26.50	678.16
S&D Der-Izq	27.03	742.16
NAF	21.09	556.61
wNAF	10.59	885.97

TABLA IV. ESPECIFICACIÓN DE PARÁMETROS DE LA CURVA SECT283R1

Parámetro	Valor
$\boldsymbol{f(x)}$	$x^{283} + x^{12} + x^7 + x^5 + 1$
$\boldsymbol{a}$	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001
$\boldsymbol{b}$	027B680A C8B8596D A5A4AF8A 19A0303F CA97FD76 45309FA2 A581485A F6263E31 3B79A2F5
$\boldsymbol{G}$	(05F93925 8DB7DD90 E1934F8C 70B0DFEC 2EED25B8 557EAC9C 80E2E198 F8CDBECD 86B12053 03676854, FE24141C B98FE6D4 B20D02B4 516FF702 350EDDB0 826779C8 13F0DF45 BE8112F4)
$\boldsymbol{n}$	03FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFEF90 399660FC 938A9016 5B042A7C EFADB307
$\boldsymbol{h}$	02

En la Tabla III se muestran los tiempos de ejecución para la multiplicación escalar definida sobre campo primo, tanto en la computadora de escritorio como en el dispositivo móvil.

Como puede apreciarse en la Tabla III el método que menor tiempo de ejecución demanda en el ordenador de escritorio es el *wNAF*, mientras que en el dispositivo móvil es el *NAF*. En la mayoría de los casos la multiplicación escalar se calcula en menos de un segundo sobre el procesador ARM, el cual es un tiempo que puede ser tolerado en la mayoría de las aplicaciones móviles.

B. Campo binario

Para evaluar los métodos de multiplicación escalar definido sobre el campo binario, para la tupla $T = (m, f(x)\ a, b, G, n, h)$ se utilizaron los valores recomendados para la curva sect283r1, la cual está recomendada en el estándar SECG [15], y es compatible con ANSI [12, 13], IEEE [10] y esta también recomendada en NIST [11]. Los valores para cada elemento de la tupla T se muestran en la tabla IV.

En cuanto a la implementación de la multiplicación escalar sobre campo binario se utilizó el paquete java.security.spec que proporciona clases e interfaces para la especificación de parámetros de algoritmos criptográficos y la librería de código

TABLA V. TIEMPO DE EJECUCIÓN DE LA MULTIPLICACIÓN ESCALAR SOBRE CAMPO BINARIO

	Tiempo de ejecución (ms)	
	PC	Móvil
S&D Izq-Der	86.18	7408.5
S&D Der-Izq	91.93	8285.3
NAF	60.25	4977.9
wNAF	59.09	4947.7

abierto FlexiProvider [23], ya que brinda soporte para operaciones de polinomios, indispensables en este tipo de campo por lo cual fue posible realizar las operaciones respectivas a la capa media y la capa inferior del modelo de capas de la multiplicación escalar.

En la Tabla V se muestran los tiempos necesarios para la ejecución de la multiplicación escalar, tanto en la computadora de escritorio como en el dispositivo móvil.

Como se puede apreciar el método con un menor tiempo de ejecución sobre campo binario es *wNAF* al realizarse la implementación en el ordenador de escritorio al igual que para el dispositivo móvil. Este valor incluye además el tiempo de generación de la representación *wNAF* del escalar *d*, lo que permite confirmar que al reducir la densidad de dígitos diferentes de cero del escalar *d* mediante el método *wNAF* es posible lograr incrementar la eficiencia en el desempeño de la multiplicación escalar.

De acuerdo a los resultados obtenidos la Criptografía de Curvas Elípticas sobre campo primo resulta ser más eficiente en comparación con el campo binario, tanto para PC como para móvil. En los resultados obtenidos en la PC se puede apreciar que la mayoría de los métodos implementados en campo binario son en promedio 2 o 3 veces más lentos que en campo primo exceptuando el método *wNAF* el cual es 5 veces más lento que su versión en campo primo. Mientras que en los resultados obtenidos del dispositivo móvil se puede observar que los métodos en campo binario son entre 8 y 11 veces más lentos que los métodos en campo primo con excepción del método *wNAF* el cual es 5 veces más lento que su versión en campo primo. Estas diferencias en el tiempo de ejecución de los algoritmos para el cálculo de la multiplicación escalar nos muestran que dentro de las implementaciones en software es mejor opción el uso del campo primo, sin embargo también es importante la elección correcta del método para la multiplicación escalar dado a que el uso de pre-caculos como en el caso de *wNAF* indica mayor uso de recursos.

V. CONCLUSIONES

Debido al gran interés del uso de esquemas criptográficos basados en curva elíptica, resulta crucial el análisis de las operaciones subyacentes para la implementación de dicha técnica, con la finalidad de lograr el mejor desempeño posible.

En este artículo presentamos los resultados obtenidos por la implementación de cuatro diferentes técnicas algorítmicas para el cálculo de la multiplicación escalar de curva elíptica sobre campos finitos primos y binarios

Los resultados obtenidos reflejan que los métodos *NAF* y *wNAF* son más rápidos, debido a que en ambos métodos el escalar *d* es transformado y la densidad de dígitos diferentes de cero resulta menor que en su representación binaria, que es la que usan los métodos de Suma y Doblado. Esto implica que se necesiten menos operaciones aritméticas para el cálculo de la multiplicación escalar, ya que al disminuir la densidad de dígitos diferentes de cero en el escalar *d*, el número de Sumas-ECC se reduce.

Se realizó un análisis de los tiempos de ejecución del ordenador de escritorio contra los obtenidos en el dispositivo móvil. Podemos mencionar que al ejecutar los algoritmos para la multiplicación escalar utilizando el campo primo, el tiempo de ejecución en el dispositivo móvil fue en promedio 26 veces más lento que el tiempo consumido por la PC con excepción del método *wNAF* que fue 83 veces más lento en el móvil en comparación con la PC. Al ejecutar los algoritmos para la multiplicación escalar utilizando el campo binario, el tiempo requerido por el móvil fue en promedio 85 veces más lento que la PC. Gracias a estas comparaciones es que podemos concluir que el uso del campo primo en implementaciones de software sobre plataformas móviles es más adecuado que el uso del campo binario.

Con los resultados obtenidos podemos puntualizar que múltiples aplicaciones móviles pueden soportar el cálculo de las operaciones sobre curvas elípticas, lo cual permite el uso de ECC en estos dispositivos con características de procesamiento restringidas.

Como trabajo futuro se explorarán algoritmos para cálculo de la multiplicación escalar definidos sobre otros campos finitos, tales como GF(3^m). También se sugiere probar técnicas algorítmicas más complejas que realicen el pre-procesamiento del escalar *d*, tales como el método *Frac-wNAF* (fractional window Non-Adjacent Form) [24], el método *Fixed-base NAF windowing* [4], o el método *Sliding window* [5]. Además de un análisis detallado de los métodos y las capas de la multiplicación escalar para su implementación sobre arquitecturas paralelas tales como los GPUs (Graphics Processing Unit).

REFERENCIAS

[1] Y. Kawahara, T. Takagi, and E. Okamoto, Efficient Implementation of Tate Pairing on a Mobile Phone Using Java. In Computational Intelligence and Security, vol. 2, pp. 1247 - 1252, Berlin, 2006.

[2] A. Weimerskirch, C. Paar, and S. Chang Shantz. Elliptic Curve Cryptography on a Palm OS Device. In V. Varadharajan and Y. Mu, editors, The 6th Australasian Conference on Information Security and Privacy, vol. LNCS 2119, pp. 502-513, Berlin, 2001.

[3] M. Morales-Sandoval, A reconfigurable and interoperable hardware architecture for elliptic curve cryptography, Tesis de Doctorado, Instituto Nacional de Astrofísica, Óptica y Electrónica, México, 2008.

[4] D. Hankerson, A. J. Menezes, and S. Vanstone. Guide to Elliptic Curve Cryptography, USA, 2003, Springer-Verlag New York, Inc.

[5] P. Longa, Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields, Tesis de Maestría, School of Information Technology and Engineering University of Ottawa, Canada, 2007

[6] N. Ferguson and B. Schneier, Practical Cryptography. Wiley, 2003.

[7] N. Koblitz. Elliptic curve in cryptography. American Mathematical Society J. Comput. Math., pp. 207–209, 1987.

[8] V. S Miller. Use of elliptic curves in cryptography. In Lecture notes in computer sciences; 218 on Advances in cryptology CRYPTO 85, pp. 417–426, USA, 1986. Springer-Verlag New York, Inc.

[9] A. J. Menezes, T. Okamoto, and S. A. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, IEEE Transactions on Information Theory , 1639 (1993).

[10] IEEE P1363. Standard Specifications for Public-Key Cryptography. Institute of Electrical and Electronics Engineers, 2000.

[11] NIST, Recommended Elliptic Curves for Federal Government Use, 1999. [En línea] Disponible: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf
[12] ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA). American Bankers Association, 1999.
[13] ANSI X9.63-199x: Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography. American Bankers Association, October, 1999. Working Draft.
[14] ISO/IEC 18033-3:2005: International Standard. [En línea] Disponible: http://webstore.iec.ch/p-preview/info_isoiec18033-3%7Bed1.0%7Den.pdf
[15] D. R. Brown, SEC 2: Recommended Elliptic Curve Domain Parameters, Certicom Research, 2010. [En línea] Disponible: http://www.secg.org/collateral/sec2_final.pdf
[16] A. Trujillo-Vázquez, Criptografía basada en ECC y AES para dispositivos con recursos restringidos, Tesis de Maestría, Universidad Politécnica de Cd. Victoria, México, 2011.
[17] G. L. Hernández, Paralelización de la multiplicación escalar en curvas elípticas en una arquitectura multinúcleo de Intel, Tesis de Maestría, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México, 2010.
[18] Harsandeep Brar and Rajpreet Kaur. Design and Implementation of Block Method for Computing NAF. International Journal of Computer Applications pp. 37-41, 2011.
[19] Fan R. On The Efficiency Analysis of wNAF and wMOF, Tesis de Maestría. Technische Universität Darmstadt,Alemania, 2005.
[20] K. Okeya, and T. Takagi, "The Width-w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks," CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003, USA, 2003.
[21] Z. Cheng, Simple Tutorial on Elliptic Curve Cryptography, School of Computing Science, 2004.
[22] J. Gil-Flores, Aplicación Del Método Bootstrap al Contraste De Hipótesis En La Investigación Educativa, Universidad de Sevilla *Revista de Educación* núm. 336, pp. 251-265, 2005.
[23] FlexiProvider 2012. [En línea] Disponible: http://www.flexiprovider.de/overview.html
[24] Bodo Moller: Improved Techniques for Fast Exponentiation. In: ICISC 2002, LNCS, vol.2587, pp. 298-312. Springer Heidelberg, 2003.

Karina Vega Castillo, nació en Cd Victoria, Tamaulipas el 15 de agosto de 1986. Obtuvo el grado de Ingeniero en Sistemas Computaciones por el Instituto Tecnológico de Ciudad Victoria en el estado de Tamaulipas en el año 2009. Ella actualmente, se encuentra desarrollando su proyecto de investigación para obtener el grado de Maestra en Ingeniería en la Universidad Politécnica de Ciudad Victoria, Tamaulipas. Su área de especialidad es la seguridad informática y el desarrollo de esquemas criptográficos.

Antonio Cortina Reyes nació en Cd. Victoria, Tamaulipas el 06 de abril de 1988. Obtuvo el grado de Ingeniero en Sistemas computacionales en el Instituto Tecnológico de Cd. Victoria en el estado de Tamaulipas en el año 2011.El actualmente se encuentra desarrollando su proyecto de investigación para obtener el grado de Maestro en Ingeniería en la Universidad Politécnica de Ciudad Victoria, Tamaulipas. Su área de especialidad es la seguridad informática y el desarrollo de esquemas criptográficos.

Miguel Morales Sandoval nació en Calpan, Puebla el 07 de octubre de1978. Recibió el grado de licenciado en Ciencias de la Computación en 2002 por la Universidad Autónoma de Puebla. Recibió el grado de Maestro y Doctor en Ciencias en 2004 y 2008 respectivamente, ambos por el Instituto Nacional de Astrofísica, Óptica y Electrónica, ubicado en Tonantzintla, Puebla. Desde diciembre de 2008 es profesor investigador en la Universidad Politécnica de Victoria, en Ciudad Victoria, Tamaulipas donde dirige tesis de maestría e imparte diversas materias de posgrado y licenciatura en el área de las Tecnologías de Información. Desde 2003 realiza proyectos de investigación en las áreas de la criptografía, con especialidad en Criptografía de Curvas Elípticas y el diseño digital de arquitecturas hardware usando dispositivos reconfigurables y lenguajes de descripción de hardware. Ha publicado alrededor de 20 artículos en revistas y congresos especializados, y participa en el desarrollo de proyectos de investigación aplicada. El Dr. Morales-Sandoval es miembro del Sistema Nacional de Investigadores y cuenta con la distinción de perfil deseable por PROMEP.