Ciencias Matemáticas, Vol. 35, No. 1, 2021, Pag. 9-19
Recibido 11-2020, Aceptado 04-2021

Mejorando la seguridad de la implementación de la multiplicación escalar para curvas generales en forma de Weierstrass en Bouncy Castle
Improving the security of the scalar multiplication implementation for general curves in Weierstrass form in Bouncy Castle

Yessica C. Castaño Sainz[1], Carlos M. Legón Pérez[2], Teresa B. Pagés López[3]

Resumen En el presente trabajo se exponen resultados parciales de la implementación de la aritmética para curvas generales de Weierstrass en Bouncy Castle. Se hacen comparaciones de eficiencia y se presentan aspectos relacionados con la implementación de algoritmos para la aritmética para el caso en que el punto P es variable. Se valora la propuesta de aritmética para la adición de puntos por Joppe W. Bos et al. utilizando la fórmula completa para la aritmética propuesta por Joost Renes et al. Como resultado se obtuvo una implementación de algoritmos para realizar la multiplicación escalar cuando el punto es variable más segura frente a ataques de tiempo y caché manteniendo valores competitivos de eficiencia. Finalmente se describen las líneas para trabajo futuro.

Abstract The present work presents partial results of the implementation in Bouncy Castle of algorithms for the scalar multiplication for the case of general Weierstrass curves . Efficiency comparisons are made and aspects related to the implementation of scalar multiplication algorithms are presented for the case in which the point P is variable. It is studied the possibility of changing the constant time addition formula proposed by Joppe W. Bos et al. by the complete formula proposed by Joost Renes et al.. As a result, an implementation of algorithms to perform scalar multiplication when the point is variable more secure against time and cache attacks was obtained, maintaining competitive efficiency values. Finally, the lines for future work are described.

Palabras Clave
Curvas elípticas, multiplicación escalar, tiempo constante, libre de excepciones, fórmula completa

Keywords
Elliptic curves, scalar multiplication, constant time, exceptions free, complete formula

[1] *Departamento de Criptografía, Universidad de La Habana, La Habana, Cuba, yessica.castano@matcom.uh.cu, ykcastano@gmail.com*
[2] *Departamento de Criptografía, Universidad de La Habana, La Habana, Cuba, clegon58@gmail.com*
[3] *Departamento de Criptografía, Universidad de La Habana, La Habana, Cuba, teresa.bernarda@matcom.uh.cu*

Introducción

El Instituto de Criptografía de la Universidad de La Habana se ha dado a la tarea de implementar una Infraestructura de clave pública (PKI por sus siglas en inglés) basada solamente en curvas elípticas para la emisión de certificados digitales. Para ello se hace necesario el estudio de los diferentes softwares que pudieran brindar este servicio.

Uno de ellos, altamente utilizado, es EJBCA el cual utiliza la librería Bouncy Castle como proveedor criptográfico. La necesidad de añadir curvas trae como consecuencia que se haga imprescindible la definición de la aritmética a utilizar con las nuevas curvas ya que la seguridad de las implementaciones de la aritmética de curvas elípticas constituye una parte fundamental en la seguridad de los criptosistemas basados en curvas elípticas.

La seguridad de los algoritmos que implementan la aritmética se añade a la correcta selección de los parámetros de definición, trabajo previo cuyos resultados se presentan en [42].

Para una correcta implementación es necesario tener en cuenta los posibles ataques tanto desde el punto de vista matemático y de diseño de los protocolos como de los ataques de canal colateral que aunque en un primer momento no se les prestó suficiente atención, han ganado en popularidad en los últimos años dentro de la comunidad criptográfica

[31],[21],[11], [26].

A su vez, como en todas las áreas de Criptografía se deben considerar los diferentes tipos de escenarios donde se tiene la implementación, si es de software o es una implementación en una tarjeta inteligente y los ataques que son más comunes en ella ya que en la mayoría de los casos se necesita llegar a un conseso entre seguridad y eficiencia.

Es por ello que en la literatura se tienen en cuenta tres escenarios principales de explotación: Por ejemplo en [29] se consideran los escenarios: entorno protegido, entorno de red y entorno hostil.

En el primer caso se considera la implementación como una caja negra de la cual el atacante no tiene acceso a detalles específicos como el tiempo de cómputo.

El segundo caso hace referencia a los entornos en que es posible correr la implementación de los algoritmos a través de la red.

Por último, el entorno hostil incluye las implementaciones a las cuales el atacante tiene acceso completo a la implementación y a todos los posibles ataques de canal colateral, lo cual ocurre regularmente en sistemas embebidos y tarjetas inteligentes.

Varias son las propuestas de investigadores para lograr implementaciones eficientes. La gran mayoría han recurrido a otras formas de expresar la curva como las curvas de Montgomery [32] y Twisted Edwards [6] con el fin de ganar en eficiencia y seguridad frente a ataques de canal colateral. Esto se justifica por el hecho de que las curvas de Edwards pueden ser dotadas de una ley interna completa que es también relativamente eficiente (comparada con otras leyes de adición para las curvas de Edwards).

Sin embargo, la forma de Weierstras presenta beneficios que no se tienen cuando se utilizan otros modelos de curvas. Por ejemplo, la ausencia de subgrupos pequeños simplifica el proceso de validación de los puntos de entrada y sobre campos finitos primos para una longitud fija, no sacrifica bits de seguridad con respecto a ataques del problema del logaritmo discreto. Además, las curvas en forma de Weierstrass son compatibles con las implementaciones que se tienen para las curvas del NIST sobre campos finitos primos, lo cual trae como consecuencia que no se necesiten cambios en los protocolos que las utilizan.

Es por ello que en trabajos como [7] y [36] se presentan fórmulas para hacer eficiente la aritmética de las curvas elípticas en forma de Weierstrass.

Dos de los ataques más importantes contra los cuales deberíamos proteger una implementación de software lo constituyen ataques de caché y de temporización. Una propuesta de este tipo fue presentada por Joppe W. Bos et al. en [7] donde se demuestra que los algoritmos propuestos son libres de excepciones y condicionales.

El presente trabajo toma los resultados en [7] como punto de partida y explora la posibilidad de cambiar la fórmula que ellos completan por medio de enmascaramientos por una fórmula completa [36].

Se presentan resultados de la implementación en Bouncy Castle de algoritmos para la multiplicación escalar cuando el punto P es variable. Esto constituye un resultado parcial ya que se trabaja actualmente en completar este trabajo con la implementación de la aritmética para los otros dos casos a tener en cuenta para la implementación de la aritmética: Punto fijo y el caso aP+bQ.

Este trabajo demuestra que la fórmula completa propuesta en [36] es más eficiente que la variante para la adición completa utilizada en [7] y que su integración con los algoritmos para la multiplicación escalar utilizados en [7] constituye una alternativa que facilita las implementaciones de algoritmos que corran en tiempo constante y seguros frente a ataques de caché ya que elimina la necesidad de recurrir a enmascaramientos dentro de la fórmula para evitar excepciones.

Además, evidencia que es posible implementar algoritmos de multiplicación escalar en Bouncy Castle que corran en tiempo constante, libre de excepciones y con contramedidas para hacer frente a ataques de caché sin perder en eficiencia, los cuales pudieran incluso ser optimizados si técnicas para mejorar la eficiencia de las operaciones del campo base fueran utilizadas.

En el primer epígrafe se introducen las curvas elípticas, los principales algoritmos utilizados para la multiplicación escalar y los principales ataques de canal colateral. Se analizan los algoritmos propuestos por Joppe W. Bos et al. en [7] y la seguridad y eficiencia de la fórmula completa propuesta en [36]. Trabajos relacionados son presentados en la sección 2. La sección 3 trata sobre la librería criptográfica Bouncy Castle y las consideraciones sobre la implementación de los algoritmos. En la sección 4 se presentan los resultados del análisis de la eficiencia de los algoritmos implementados. Finalmente, líneas de trabajos futuro son expuestas en la sección 5.

1. Sobre las curvas elípticas, su aritmética y ataques de canal colateral.

En este trabajo nos iteresan las curvas elípticas definidas sobre un campo finito $\mathbb{F}_p$ de característica prima $p > 3$. Cada curva elíptica E definida sobre un campo finito $\mathbb{F}_p$, puede ser representada en forma reducida de Weierstrass como:

$$E/\mathbb{F}_p : \ y^2 = x^3 + ax + b \tag{1}$$

Para valores arbitrarios de $a, b \in \mathbb{F}_p$ una curva elíptica sobre F_p se define como el conjunto de soluciones (x, y) de la ecuación de la curva $E/\mathbb{F}_p : \ y^2 = x^3 + ax + b$, con un punto adicional llamado punto al infinito y generalmente representado por el símbolo ∞ (o por el símbolo $\mathcal{O}$). Esos puntos forman un grupo con ∞ como elemento neutro.

Los valores para las constantes $a, b \in \mathbb{F}_p$ deben ser escogidos de forma tal que cumplan con los requisitos de seguridad que se plantean en los estándares para la criptografía de curvas elípticas [33, 12, 3, 24, 27] y en las discusiones recientes para la estandarización de nuevas curvas [18, 28, 29, 8, 14].

El orden de la curva es denotado como: $\#E(\mathbb{F}_p)$. Para curvas elípticas seguras $\#E(\mathbb{F}_p)$ tiene la forma $\#E(\mathbb{F}_p) = nh$, donde n es un primo grande y h es un número pequeño.

La criptografía de curvas elípticas se basa en la dificultad de resolver el problema del logaritmo discreto de curvas elípticas (ECDLP por sus siglas en inglés). Lo cual significa que dados dos puntos $P, Q \in E(\mathbb{F}_p)$ encontrar $k \in \mathbb{Z}$ tal que $Q = kP$, si existe, es computacionalmente difícil de resolver.

La operación de calcular $Q = kP$ es llamada multiplicación escalar y es calculada a partir de adiciones y doblados sucesivos, dependiendo de la codificación que se haga del escalar k.

Estas operaciones finalmente se traducen a multiplicaciones, inversiones y adiciones en el campo finito subyacente. Las multiplicaciones modulares son mucho más costosas que las adiciones de campo en términos de tiempo y memoria, constituyendo las inversiones de campo las más costosas de calcular.

La elección del sistema de coordenadas también influye en la eficiencia de la operación de multiplicación escalar, por ejemplo a través del uso de coordenadas proyectivas se evitan las inversiones de campo.

Sea k un campo, y sean c y d enteros positivos, se define una relación de equivalencia $\sim$ en el conjunto $k^3 \setminus \{(0,0,0)\}$ de las triplas no nulas sobre k por:

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \ si\, X_1 = \lambda^c X_2,\ Y_1 = \lambda^d Y_2,\ Z_1 = \lambda Z_2,$$

para algún $\lambda \in k^*$.

La clase de equivalencia que contiene a $(X,Y,Z) \in k^3 \setminus \{(0,0,0)\}$ es

$(X : Y : Z) = \{(\lambda^c X, \lambda^d Y, \lambda Z) : \lambda \in k^*\}$,

$(X : Y : Z)$ es llamado punto proyectivo, y (X, Y, Z) es llamado representante de $(X : Y : Z)$.

El conjunto de todos los puntos proyectivos es denotado por $\mathbb{P}(k)$. Si $Z \neq 0$, entonces $(X/Z^c.Y/Z^d, 1)$ es un representante del punto proyectivo $(X : Y : Z)$, y de hecho es el único con coordenada Z igual a 1.

Cuando $c = 1$ y $d = 1$. La ecuación para el caso de la forma de Weierstrass reducida 1 es:

$$Y^2Z = X^3 + aXZ^2 + bZ^3 \tag{2}$$

El único punto en la línea al infinito que además pertenece a la curva E es $(0 : 1 : 0)$. El punto proyectivo $(0 : 1 : 0)$ corresponde al punto al infinito.

Cuando $c = 2$ y $d = 3$ estamos en presencia de las coordenadas jacobianas, donde el punto proyectivo $(X : Y : Z), Z \neq 0$, corresponde al punto afín $(X/Z^2, Y/Z^3)$. El punto al infinito es $(1 : 1 : 0)$, mientras el punto opuesto de $(X : Y : Z)$ es $(X : -Y : Z)$.

En el caso de las coordenadas de Chudnovsky, el punto jacobiano $(X : Y : Z)$ es representado como $(Z : Y : Z : Z^2 : Z^3)$. Esta representación es utilizada para ganar en eficiencia en algunos métodos de multiplicación cuando la adición de puntos se hace en coordenadas proyectivas.

El hecho de que la seguridad y eficiencia de los Criptosistemas de Curvas Elípticas (CCE) se base en esta operación justifica el interés de la comunidad criptográfica en buscar alternativas eficientes y seguras.

1.1 Aritmética de puntos en una curva elíptica

Como se mencionó anteriormente la operación de multiplicación escalar kP domina el tiempo de ejecución de los esquemas critográficos sobre curvas elípticas. Asumiendo $\#E/\mathbb{F}_p = nh$ donde n es primo y h pequeño. $k \in [0, n-1]$, $k = (k_{t-1}, ..., k_1, k_0)$ la representación binaria de k, donde $t \approx m = \lceil \log_2 q \rceil$. La operación kP se puede realizar utilizando el algoritmo double-and-add. Se pueden hacer dos implementaciones del algoritmo double-and-add, right-to-left y left-to-right. EL algoritmo left-to-right se presenta en 1.

Algoritmo 1 Método binario de multiplicación de izquierda a derecha (Double and add left-to-right)

Entrada: $k = (k_{t-1}, ..., k_1, k_0)_2$, $P \in E(\mathbb{F}_p)$.
Salida: kP.

1: $Q = \infty$
2: **para** $i = t-1$ decreciendo hasta 0 **hacer**
3: $\quad Q = 2Q$
4: $\quad$ Si $k_i = 1$ entonces $Q = Q + P$
5: **fin para**
6: **Salida** Q.

Con el objetivo de acelerar el cálculo se puede hacer uso de la representación en forma no adyacente NAF (por sus siglas en inglés) del escalar k, ver algoritmo 3. Esta representación fue introducida como una alternativa a la representación binaria de k con el objetivo de reducir el tiempo de ejecución de la multiplicación escalar ya que permite reducir el peso de Hamming del escalar, ver [11] para otras formas de exponenciación rápida.

Además se pueden utilizar métodos de ventanas [35] para el cálculo de la representación si se dispone de memoria para precalcular algunos puntos múltiplos de P. En lugar de utilizar una representación en base 2 sería en base 2^w para w el tamaño de la ventana y el algoritmo procesaría w dígitos de k a la vez.

El método de ventana con representación wNAF está presente en varias librerías incluyendo Bitcoin, Cryptlib, OpenSSL y en Bouncy Castle.

Sin embargo, la implementación de la multiplicación escalar utilizando métodos de ventana con representación del escalar wNAF puede hacer el protocolo vulnerable frente a ataques de canal colateral [15].

En [23] M.Joye et al. proponen vías alternativas para la representación del escalar k que permiten implementaciones regulares de los algoritmos de exponenciación, lo cual constituye un requisito para lograr implementaciones resistentes a ataques simple de potencia y de temporización.

En [9] Joppe W. Bos et al. utilizan una modificación de la representación regular del escalar propuesta en [23] para obtener una representación de longitud fija para el escalar k.

Algoritmo 2 Algoritmo wNAF para la multiplicación escalar

Entrada: El ancho de la ventana w, el entero positivo k, $P \in E(\mathbb{F}_p)$.
Salida: kP.

1: Utilizar el algoritmo 3 para calcular $wNAF(k) = \sum_{i=0}^{l-1} k_i 2^i$,
2: Calcular $P_i = iP$ para $i \in \{1,3,5,...,2^{w-1}-1\}$.
3: $Q \leftarrow \infty$.
4: **para** $i = l-1$ decreciendo hasta 0 **hacer**
5: $Q = 2Q$
6: **si** $k_i \neq 0$ **entonces**
7: **si** $k_i > 0$ **entonces**
8: $Q = Q + P_{k_i}$
9: **si no**
10: $Q = Q - P_{-k_i}$
11: **fin si**
12: **fin si**
13: **fin para**
14: **Salida** Q.

Algoritmo 3 Cálculo de la representación $wNAF$ de un entero positivo

Entrada: El ancho de la ventana w, entero positivo k.
Salida: $wNAF(k)$.

1: i=0
2: **mientras** $k \geq 1$ **hacer**
3: **si** k es par **entonces**
4: $k_0 = k \mod 2^w$, $k = k - k_i$
5: **si no**
6: $k_i = 0$
7: **fin si**
8: $k = k/2$, $i = i+1$
9: **fin mientras**
10: **Salida** $k = (k_{k-1}, k_{i-2}..., k_1, k_0)$.

El algoritmo propuesto en [9] para el caso en que el punto de entrada P es variable se muestra en el algoritmo 4.

Para lograr el correcto funcionamiento del algoritmo 4 y de esa forma obtener una implementación libre de condicionales se deberán enmascar las condicionales if presentes en los pasos 6 y 15.

En [9] los autores demuestran que este algoritmo corre en tiempo constante y libre de excepciones cuando es implementado utilizando una fórmula completa para realizar la adición representada por el símbolo $\oplus$ en el paso 14.

Una ley de adición de puntos en una curva elíptica se dice que es completa si calcula correctamente la suma de dos puntos cualesquiera en el grupo de puntos de la curva elíptica. Una ley de adición completa reduce las vulnerabilidades de un criptosistema basado en curvas elípticas y facilita las implementaciones.

El hecho de que las curvas en forma de Edwards permita la definición de una ley de grupo completa y eficiente constituye la principal razón por la cual estas han ganado popularidad en

Algoritmo 4 Algoritmo de multiplicación escalar utilizando el método de ventana fija para punto base P variable.

Entrada: Escalar $k \in [0, r\rangle$ y el punto $P = (x,y) \in E(\mathbb{F}_p)$ donde $\#E(\mathbb{F}_p) = r \cdot h$ con cofactor $h \in \mathbb{Z}^+$ y r primo.
Salida: kP.

1: **si** $k = 0 \vee k \geq r$ **entonces** return (error: escalar no válido)
2: Correr el algoritmo de validación del punto P **si** retorna no válido **entonces** return (error: punto no válido)
Fase de precálculo
3: Fijar la longitud de la ventana $2 \leq w < 10 \in \mathbb{Z}^+$.
4: Calcular $P[i] = (2i+1)P$ para $0 \leq i < 2^{w-2}$, usando el algoritmo 4 de [9].
Fase de recodificación del escalar
5: Hacer odd= $k \mod 2$
6: **si** odd= 0, **entonces** $k = r - k$.
7: Recodificar $k = (k_t, ..., k_0) = (s_t \cdot |k_t|, ..., s_0 \cdot |k_0|)$, usando el algoritmo 6, donde $t = \lceil \log_2(r)/(w-1) \rceil$ y s_j son los signos de los dígitos recodificados.
Fase de evaluación
8: $Q = s_t P[(|k_t|-1)/2]$
9: **para** $i = t-1$ decreciendo hasta 1 **hacer**
10: $Q = 2^{(w-2)}Q$
11: $Q = 2Q + s_i P[(|k_i|-1)/2]$
12: **fin para**
13: $Q = 2^{(w-1)}Q$
14: $Q = Q \oplus s_0 P[(|k_0|-1)/2]$
15: **si** odd= 0 **entonces** $Q = -Q$
16: Convertir Q a coordenadas afines (x,y)
17: **Salida** Q.

Algoritmo 5 Adición completa propuesta en [7] en coordenadas proyectivas usando enmascaramiento y coordenadas Jacobianas para curvas de Weierstrass sobre campos finitos primos.

Entrada: $P,Q \in E(\mathbb{F}_p)$ tal que $P = (X_1,Y_1,Z_1)$ y $Y = (X_2,Y_2,Z_2)$ están en coordenadas Jacobianas.
Salida: $R = P+Q \in E(\mathbb{F}_p)$ en coordenadas Jacobianas. Los pasos marcados con $*$ deberán ser implementados en tiempo constante usando enmascaramiento.

1: $T[0] = \infty$ $\{T[i] = (\widehat{X}_i,\widehat{Y}_i,\widehat{Z}_i)\}$ para $0 \le i < 5$
2: $T[1] = Q$
3: $T[4] = P$
4: $t_2 = Z_1^2$
5: $t_3 = Z_1 \cdot t_2$
6: $t_1 = X_2 \cdot t_2$
7: $t_4 = Y_2 \cdot t_3$
8: $t_3 = Z_2^2$
9: $t_5 = Z_2 \cdot t_3$
10: $t_7 = X_1 \cdot t_3$
11: $t_8 = Y_1 \cdot t_5$
12: $t_1 = t_1 - t_7$
13: $t_4 = t_4 - t_8$
14: index=3
15: **si** $t_1 = 0$ **entonces** index=0 $\{R = \infty\}$ $[*]$.
16: **si** $t_4 = 0$ **entonces** index=2 $\{R = 2P\}$ $[*]$.
17: **si** $P = \infty$ **entonces** index=1 $\{R = Q\}$ $[*]$.
18: **si** $Q = \infty$ **entonces** index=4 $\{R = P\}$ $[*]$.
19: mask =0
20: **si** index=3 **entonces** mask=1 (Caso $P+Q$).
21: $t_3 = X_1 + t_2$
22: $t_6 = X_1 - t_2$
23: **si** mask=0 **entonces** $t_2 = Y_1$ **si no** $t_2 = t_1$ $[*]$
24: $t_5 = t_2^2$
25: **si** mask=0 **entonces** $t_7 = X_1$ $[*]$
26: $t_1 = t_5 \cdot t_7$
27: $\widehat{Z}_2 = Z_1 \cdot t_2$
28: $\widehat{Z}_3 = Z_2 \cdot \widehat{Z}_2$
29: **si** mask $\neq 0$ **entonces** $t_3 = t_2$ $[*]$
30: **si** mask $\neq 0$ **entonces** $t_6 = t_5$ $[*]$
31: $t_2 = t_3 \cdot t_6$
32: $t_3 = t_2/2$
33: $t_3 = t_2 + t_3$
34: **si** mask $\neq 0$ **entonces** $t_3 = t_4$ $[*]$
35: $t_4 = t_3^2$
36: $t_4 = t_4 - t_1$
37: $\widehat{X}_2 = t_4 - t_1$
38: $\widehat{X}_3 = \widehat{X}_2 - t_2$
39: **si** mask=0 **entonces** $t_4 = \widehat{X}_2$ **si no** $t_4 = \widehat{X}_3$ $[*]$
40: $t_1 = t_1 - t_4$
41: $t_4 = t_3 \cdot t_1$
42: **si** mask=0 **entonces** $t_1 = t_5$ **si no** $t_1 = t_8$ $[*]$
43: **si** mask=0 **entonces** $t_2 = t_5$ $[*]$
44: $t_3 = t_1 \cdot t_2$
45: $\widehat{Y}_2 = t_4 - t_3$
46: $\widehat{Y}_3 = \widehat{Y}_2$
47: $R = T[\text{index}] = (\widehat{X}_{index},\widehat{Y}_{index},\widehat{Z}_{index})$
48: **Salida** R.

Algoritmo 6 Algoritmo protegido de codificación de escalares (solo impares) para el método de multiplicación de ventana fija.

Entrada: Entero impar $k \in [1,r\rangle$ y la longitud de la ventana $w \ge 2$, donde r es el orden (primo) del subgrupo de la curva elíptica con la que se está trabajando.
Salida: $(k_t,...,k_0)$, donde $k_i \in \{\pm 1,\pm 3,...,\pm(2^{w-1}-1)\}$ y $t = \lceil \log_2(r)/(w-1) \rceil$.
1: $t = \lceil \log_2 r/(w-1) \rceil$
2: **para** $i = 0$ hasta $t-1$ **hacer**
3: $k_i = (k \mod 2^w) - 2^{w-1}$
4: $k = (k-k_i)/2^{w-1}$
5: **fin para**
6: $k_t = k$
7: **Salida** $(k_t,...,k_0)$.

los últimos años. No obstante las curvas en forma de Weierstrass presentan beneficios sobre las otras formas de curvas elípticas propuestas incluyendo por ejemplo el hecho de que se pueden considerar grupos de puntos de curvas elípticas con cofactor uno, lo cual se traduce en beneficios de seguridad ya que se evitan ataques de subgrupo pequeño [16],[39].

Los resultado en [9] sin dudas constituyen una vía para lograr implementaciones eficientes, de tiempo constante y libre de excepciones al completar la fórmula de adición mediante técnicas de enmascaramiento, logrando valores de eficiencia equivalentes a las fórmulas dedicadas.

La elección de Joppe W. Bos et al. de utilizar técnicas de enmascaramiento sin recurrir a las fórmulas completas, conocidas dede 1955 en [38] se justifica por ganar en eficiencia. Sin embargo, en [36] se propone una mejora significativa a las fórmulas propuestas en [38] logrando reducir el número de operaciones de campo necesarias. El algoritmo 7 muestra las operaciones de campo para calcular la fórmula completa en coordenadas proyectivas.

Teniendo en cuenta la complejidad desde el punto de vista de la implementación que estas técnicas de enmascaramiento le añaden, el hecho de reemplazarla por una fórmula completa con una eficiencia similar sería no solo de gran ayuda para la implementación sino también más segura, por ser de menor envergadura las transformaciones que los compiladores puedan ocasionar al código del programa.

Como parte de este trabajo se implementó la fórmula completa de [36], la cual podrá ser utilizada para sustituir la adición del paso 14 del algoritmo 4.

1.2 Resistencia frente a atques de canal colateral

Hoy en día los ataques de canal colaterar, primeramente introducidos por Kocher [25] han ganado en popularidad entre la comunidad criptográfica, debido a la existencia de resultados que confirman su fortaleza y los posicionan como una amenaza real para la seguridad de los criptosistemas tanto de clave simétrica como de clave pública.

Disímiles son los trabajos que se centran en proponer

Algoritmo 7 Fórmula completa en coordenadas proyectivas para curvas en forma de Weierstrass propuesta en [36].

Entrada: $P, Q \in E(\mathbb{F}_p)$ tal que $P = (X_1 : Y_1 : Z_1)$ y $Q = (X_2 : Y_2 : Z_2)$.
Salida: $(X_2 : Y_2 : Z_2) = P + Q \in E(\mathbb{F}_p)$ en coordenadas Proyectivas.

1: $t_0 = X_1 \cdot X_2$
2: $t_1 = Y_1 \cdot Y_2$
3: $t_2 = Z_1 \cdot Z_2$
4: $t_3 = X_1 + Y_1$
5: $t_4 = X_2 + Y_2$
6: $t_3 = t_3 \cdot t_4$
7: $t_4 = t_0 + t_1$
8: $t_3 = t_3 - t_4$
9: $t_4 = Y_1 + Z_1$
10: $X_3 = Y_2 + Z_2$
11: $t_4 = t_4 \cdot X_3$
12: $X_3 = t_1 + t_2$
13: $t_4 = t_4 - X_3$
14: $X_3 = X_1 + Z_1$
15: $Y_3 = X_2 + Z_2$
16: $X_3 = X_3 \cdot Y_3$
17: $Y_3 = t_0 + t_2$
18: $Y_3 = X_3 - Y_3$
19: $Z_3 = b \cdot t_2$
20: $X_3 = Y_3 - Z_3$
21: $Z_3 = X_3 + X_3$
22: $X_3 = X_3 + Z_3$
23: $Z_3 = t_1 - X_3$
24: $X_3 = t_1 + X_3$
25: $Y_3 = b \cdot Y_3$
26: $t_1 = t_2 + t_2$
27: $t_2 = t_1 + t_2$
28: $Y_3 = Y_3 - t_2$
29: $Y_3 = Y_3 - t_0$
30: $t_1 = Y_3 + Y_3$
31: $Y_3 = t_1 + Y_3$
32: $t_1 = t_0 + t_0$
33: $t_0 = t_1 + t_0$
34: $t_0 = t_0 - t_2$
35: $t_1 = t_4 \cdot Y_3$
36: $t_2 = t_0 \cdot Y_3$
37: $Y_3 = X_3 \cdot Z_3$
38: $Y_3 = Y_3 + t_2$
39: $X_3 = t_3 \cdot X_3$
40: $X_3 = X_3 - t_1$
41: $Z_3 = t_4 \cdot Z_3$
42: $t_1 = t_3 \cdot t_0$
43: $Z_3 = Z_3 + t_1$

contramedidas para contrarrestar su aplicación [10], [36], [13], pero como se demuestra en [17], [1], [26], el proceso de diseñar algoritmos seguros frente a las diversas propuestas de ataques de canal colateral nunca termina, debido a la constante aparición de propuestas de ataques que invalidan las medidas que hasta ese momento parecían seguras [4], [11], [2].

Los ataques de canal colateral explotan fugas de información crítica sobre los parámetros secretos de un criptosistema durante pasos intermedios de sus corridas y a diferencia del criptoanálisis convencional, donde un atacante tenía acceso solamente a las parejas de entrada/salida de texto, los ataques de canal colateral utilizan información como: el tiempo de cómputo, el consumo de potencia, las emisiones electromagnáticas de los dispositivos corriendo los algoritmos o la inserción de fallas para ganar información sobre datos intermedios u operaciones.

De esta forma a través de la información obtenida por canales colaterales se puede lograr recuperar la clave secreta sin tener que romper matemáticamente las primitivas criptográficas.

Las algoritmos criptográficos pueden ser implemetados en software o en hardware. Las implementaciones en software tienen la caraterística de que son flexibles ya que pueden ser actualizados fácilmente a diferencia de las implementaciones en hardware para las cuales su reconfiguración puede conllevar un alto costo. Sin embargo las aplicaciones en hardware pueden alcanzar mayor rendimiento.

Para asegurar una implementación contra ataques de canal colateral, el primer paso es determinar en qué escenario estará corriendo el algoritmo y cuáles son las mayores amenazas reales que este podría enfrentar. Lo cierto es que es bien conocido que en Criptografía la mayoría de las veces tenemos que hacer concesiones en cuanto a criterios de seguridad/eficiencia para ganar en eficiencia/seguridad y en algunos casos protegiendo contra un ataque se puede introducir una vulnerabilidad frente a otro.

Por ejemplo, una implementación resistente a un ataque de canal colateral de temporización puede que no sea resistente a ataques del tipo DPA (Differential Power Analysis) y viceversa pudiera ocurrir que al proteger una implementación con contramedidas para ser resistente a ataques DPA se vuelva vulnerable contra ataques de temporización. Un ejemplo de esto lo constituye las contramedidas implementadas en [13] para complementar la fórmula completa propuesta en [36].

Los ataques de canal colateral se pueden clasificar en pasivos y activos. En el caso de los pasivos el atacante no manipula o modifica los datos sino que analiza de forma pasiva la fuga de bits de datos tales como los bits del escalar k. Estos ataques se aprovechan de la diferencia del tiempo de corrida de las operaciones, el consumo de potencia y las radiaciones electromagnéticas. Los ataques activos utilizan la inserción de errores para revelar los bits de k y se conocen también como ataques de fallas. Estos ataques incluyen los ataques de punto y curva no válidos, los cuales pueden ser mitigados mediante la correcta elección de los parámetros de definición de la curva y la realización de las validaciones pertinentes en la implementación de los algoritmos.

En este trabajo, como se mencionó desde la introducción nos interesa la seguridad de las implementaciones con respecto a ataques de caché y temporización, estos ataques pueden ser realizados a través de una conexión de redes de computadoras y constituyen una amenaza real para implementaciones de software.

Ataque de temporización: Los ataques de temporización expuestos por primera vez en [25] explotan la variación de tiempo consumido para diferentes entradas y pueden estar

presentes en implementaciones donde existan sentencias condicionales que dependan de bits de valores secretos.

Ataques de tiempo de acceso a caché: Información sobre elementos secretos, se obtiene a partir de la medición del tiempo de acceso a caché. [22],[10], [40], [41].

Con el objetivo de contrarrestar ataques de canal colateral pasivos se debe eliminar la relación de los datos secretos y las fugas físicas, es decir para lograr una implementación de tiempo constante en el caso de la multiplicación escalar en curvas elípticas un algoritmo debe realizar las mismas operaciones cualquiera sea la representación del escalar, esto no quiere decir que el algoritmo como un todo tenga que correr en el mismo tiempo, pero si la parte de este que involucra datos secretos.

Para el caso de los ataques de caché se deberá garantizar que el acceso a tablas de precálculo se haga de forma tal que no se pueda relacionar el punto extraído de la tabla con bits de la representación del secreto, en este caso el escalar k.

Por otra parte, las implementaciones pueden ser susceptibles a ataques simple de potencia, posiblemente los más mencionados, si las trazas de potencia muestran patrones distintivos de operaciones dependientes de la clave secreta. El valor de la clave puede ser revelado si el adversario puede distinguir la diferencia entre una adición y un doblado de puntos a partir de una traza del consumo de potencia. El algoritmo Double-and-add-always fue introducido para contrarrestar la aplicación de ataques simple de potencia al realizar una adición y un doblado en cada paso del algoritmo. Sin embargo este algoritmo no es resistente frente ataques más avanzados de potencia como DPA o RPA (Refined Power Analysis), ver [19], [1].

No es objetivo de la presente investigación afirmar que los algoritmos implementados sean resistentes a ataques simple de potencia, ni mucho menos a ataques más sofisticados de potencia. La afirmación de que una implementación es resistente a ataques de potencia requiere de otro tipo de análisis y mediciones físicas que en estos momentos no nos encontramos en condiciones de realizar. Sin embargo, como se menciona en [7], la forma que tienen los algoritmos los hace ser buenos candidatos para ser resistentes a ataques de potencia.

2. Trabajos relacionados

La existencia de trabajos haciendo referencia a ataques prácticos [11],[40],[41] de caché y temporización ha fomentado el estudio y propuesta de alternativas de implementaciones y liberías para las cuales uno de los objetivos fundamentales es lograr implementaciones de tiempo constante. Tal es el caso de las librerías bearssl, Bitcoin cryptography library [34] y MSR ECCLib [30].

En el caso de Bouncy Castle, aunque la comunidad criptográfica reconoce el hecho de que no se encuentra protegida frente a este tipo de ataques, hasta la última versión de la librería, versión 1.66 [37], esta no incluye medidas para mitigar los ataques de temporización. La librería implementa lookup tables para el acceso a tablas de precálculo de forma segura pero esto no es utilizado como parte del algoritmo de ventana utilizado para el cálculo de la multiplicación escalar en el caso de punto variable.

3. Sobre la implementación en la librería Bouncy Castle

Con el objetivo de hacer uso de técnicas para optimizar la implementación de la operación de multiplicación escalar en curvas elípticas se distinguen los tres casos: Punto variable, Punto fijo y el caso ECDSA $aP + bQ$.

Para el caso de punto variable Bouncy Castle utiliza por defecto el multiplicador WNafL2RMultiplier, el cual implementa el algoritmo de multiplicación WNAF (Window Non-Adjacent Form), algoritmo 3.36 de [20].

Como se mencionó en la introducción, este trabajo es parte de un esfuerzo por añadir nuevas curvas a la librería Bouncy Castle y dotarlas con una airtmética segura frente a ataques de temporización y caché. Hasta el momento se han implementado los multiplicadores UHwNafL2RMultiplier y UHwNafL2RMultiplierComplete.

El multiplicador UHwNafL2RMultiplier implementa el algoritmo utilizado por Patrick Longa et al. en [7] para realizar la multiplicación escalar de punto variable. En el caso del multiplicador UHwNafL2RMultiplierComplete el algortimo para realizar la última adición de puntos es modificado para hacerla utilizando la fórmula completa propuesta por Renes et al. en [36].

El multiplicador a utilizar para realizar la multiplicación escalar de punto variable se define como una propiedad de la curva a utilizar. Se creó una carpeta uhCurves en donde se podrán añadir las curvas y la clase uhPoint. Hasta el momento se utiliza la aritmética de campo finito que trae por defecto Bouncy Castle, pero esta también constituye un área en la cual se pretende trabajar.

Para asegurar que el acceso a las tablas sea seguro frente a ataques de caché Bouncy Castle implementa la función *createCacheSafeLookupTable* dentro de la clase ECCurve. Esta función fue sobrescrita y modificada en las curvas de prueba añadidas para almacenar puntos en coordenadas de Chudnovsky.

Además se modifica el punto al infinito, el cual es considerado en Bouncy Castle como $(x = null, y = null, z = 0)$ para ser $(0,1,0)$ en coordenadas proyectivas. Esto se hace porque la forma en que el punto al infinito es tratado en Bouncy Castle conlleva de inmediato a excepciones. El algoritmo para realizar la última adición en el algoritmo de multiplicación escalar requiere guardar en una tabla un arreglo de puntos que incluye el punto al infinito y se debe hacer de forma tal que no existan condicionales. De igual forma la implementación de la fórmula completa propuesta en [36], ver algoritmo 7, necesita este tipo de representación del punto al infinito.

4. Análisis de los resultados parciales

A partir del análisis de los resultados, hasta la fecha, sobre ataques de canal colateral a implementaciones de software, así como de las diferentes contramedidas que han sido propuestas e implementadas por diferentes grupos de investigación, se decidió modificar la librería Bouncy Castle para trabajar en la implementación de una aritmética que sea resistente a ataques de temporización y de caché con el objetivo de garantizar la seguridad de las nuevas curvas que fuesen añadidas.

Aunque este constituye un trabajo en proceso los resultados obtenidos hasta el momento nos permiten hacer comparaciones de eficiencia que nos pueden brindar una idea de cuánto sería el costo en cuanto a eficiencia de las nuevas medidas implementadas.

Bouncy Castle implementa curvas definidas en los estándares (defined curves) y curvas o implementaciones de estas que no han sido incluidas en estándares (custom curves). Para las curvas de SECG (Las mismas propuestas por el NIST) se cuenta con dos implementaciones, una dentro de la clase SECNamedCurves.java en el paquete org.bouncycastle.asn1.sec y la otra como custom curve en la clase específica para cada curva propuesta. En particular para la curva secp192r1 se implementa en la clase SecP192R1Curve.java dentro del paquete org.bouncycastle.math.ec.custom.sec.

La curva secp192r1 definida en SECNamedCurves.java utiliza la aritmética de campos finitos implementada por defecto en Bouncy Castle, la misma que hasta el momento utiliza la implementación de este trabajo. Por el contrario la implementación custom de esta curva en SecP192R1Curve.java utiliza una aritmética de campo finito específica para ganar en eficiencia, teniendo en cuenta las características del campo primo base. Lo mismo se tiene para las otras curvas definidas por SECG y estandarizadas por el NIST.

Para hacer el análisis comparativo en cuanto a la eficiencia se implementaron cinco curvas de SECG (secp192r1, secp224r1, secp256r1, secp384r1, secp521r1) para utilizar los multiplicadores:

UHwNafL2RMultiplier y UHwNafL2RMultiplierComplete.

Como se dijo anteriormente el multiplicador se define como una propiedad de la curva elíptica.

Esto nos permite comparar cuánto se estaría perdiendo en eficiencia al utilizar la aritmética de campo finito por defecto de Bouncy Castle, fórmulas completas y la adición de medidas de enmascaramiento para evitar la existencia de condicionales.

Las pruebas se corrieron en una Máquina Virtual con Linux Mint 19.1, kernel 4.15.0-20-generic arquitectura x86-64, con 4Gb de RAM y 8 cores de procesamiento a un máximo de 3.4 GHz. Se usó VMware Workstation 15 como software the virtualización.

En la tabla 1 se compara la eficiencia de las fórmulas para la adición (para las curvas de SECG) propuestas en [7], [36] y la adición para las implementaciones custom y standard de estas curvas. Se calculó la cantidad de operaciones de adición que tenían lugar durante 10 segundos, proceso que fue repetido 100 veces y promediado.

Como era de esperarse la adición de tiempo constante propuesta por Joppe W. Bos et al. en [7] es la que más tiempo consume con un menor número de operaciones realizadas en promedio. Esto se justifica por los enmascaramientos que son necesarios realizar para lograr tiempo constante y las operaciones necesarias para distinguir si es un doblado o una adición lo que necesita hacerse.

También se sigue de la tabla 1 que la que mejor se comporta es la adición para la implementación custom de las curvas de SECG, lo cual también se esperaba debido al hecho de que estas fórmulas no son completas, han sido optimizadas y además utilizan una aritmética de campo finito optimizada a partir de la explotación de las bondades de los primos que definen el campo finito base.

Luego, es válido destacar la eficiencia de la fórmula completa propuesta por [36], la cual permite implementar la multiplicación escalar en tiempo constante y libre de excepciones a costa de un factor de pérdida de eficiencia con respecto a las implementaciones eficientes de las curvas de SECG acotado por 1.82; factor que pudiera ser disminuido si se contara con implementaciones eficientes de la aritmética de campo. Este es uno de los aspectos en los que debemos seguir trabajando. Optimizar la multiplicación, disminuyendo complejidad, quizás con operaciones más simples o desplazamientos [5].

Además se calculó la cantidad de operaciones de multiplicación que tenían lugar durante 10 segundos (proceso que, al igual que en el caso de las fórmulas para la adición, fue repetido 100 veces y promediado). Los resultados comparativos se muestran en la tabla 2.

Los resultados que se muestran en la tabla 2 referente a las implementaciones del algoritmo de multiplicación escalar, algoritmo 4, pueden ser valorados como alentadores ya que se cuenta con una implementación que se espera se ejecute en tiempo constante, libre de excepciones y resistente a ataques de caché y que mantiene valores competitivos de eficiencia.

Conclusiones

En este trabajo se presentaron los resultados de la implementación de algoritmos resistentes a ataques de tiempo y caché para la realización de la multiplicación escalar en Bouncy Castle.

Se demostró la eficiencia de la fórmula completa para la adición de [36] en comparación con el algoritmo completo de [7].

Se logró un aumento en la seguridad de los algoritmos para realizar la multiplicación escalar cuando el punto es variable y se demuestra que es posible contar con la implementación de algoritmos diseñados para correr en tiempo constante, libre de excepciones y resistentes frente a ataques de caché, sin tener que perder en eficiencia.

Es por ello que recomendamos el uso de los algoritmos presentados en este trabajo en lugar de los que se encuentran implementados en BouncyCastle no solo para las propuestas de nuevas curvas que se quieran incluir sino también para ser utilizados con las curvas estandarizadas por el NIST, aspecto

Tabla 1. Número de operaciones de adición durante 10 segundos (proceso que fue repetido 100 veces y promediado) para cinco curvas estandarizadas por el NIST y SECG, usando la fórmula de tiempo constante y libre de excepciones propuesta en [7], ver algoritmo 5, la fórmula completa propuesta en [36] y de la adición por defecto de las implementaciones estándares y custom de estas curvas en Bouncy Castle. El factor de eficiencia muestra cuántas veces es más eficiente el algoritmo de adición utilizando la fórmula completa con respecto a la implementación custom de las curvas de SECG, de donde se sigue que no es tanta la diferencia y que incluso se pudiera llegar a mejorar utilizando una aritmética de campo eficiente.

	No. de operaciones en 10s				
	secp192r1	secp224r1	secp256r1	secp384r1	secp521r1
Adición tiempo constante [7]	792250	1416606	1513035	587296	410568
Fórmula completa [36] $*$	5333161	7806364	6661628	2208379	900369
SECG implementación Standard	1123956	1135107	1039974	822106	893315
SECG implementación Custom $\star$	8955665	5612091	5077553	2457831	1637391
Factor de eficiencia $\star/*$	1,68	0,72	0,76	1,11	1,82

Tabla 2. Número de operaciones de multiplicación escalar durante 10 segundos (proceso que fue repetido 100 veces y promediado) para cinco curvas estandarizadas por el NIST y SECG, usando los multiplicadores UHwNafL2RMultiplier, UHwNafL2RMultiplierComplete y el multiplicador WNafL2RMultiplier para las implementaciones estándares y custom de estas curvas en Bouncy Castle. Los multiplicadores UHwNafL2RMultiplier y UHwNafL2RMultiplierComplete solo difieren en la realización de la última operación de adición en el algortimo 4.

	No. de operaciones en 10s				
Multiplicador	secp192r1	secp224r1	secp256r1	secp384r1	secp521r1
UHwNafL2RMultiplier	2474454	2505308	2310984	2307156	2154232
UHwNafL2RMultiplierComplete	2465627	2530054	2325716	2292887	2148574
Bouncy Castle WNafL2RMultiplier (SECG Custom)	2474131	2506776	2312614	2302902	2160116
Bouncy Castle WNafL2RMultiplier (SECG Standard)	2481960	2501171	2296028	2299768	2164471

que influirá positivamente en el robustecimiento del módulo criptográfico del EJBCA y consecuentemente de los servicios que se presten con la PKI.

Igualmente, se espera que el hecho de contar con la fómula completa de [36] sea mucho más significativo en la eficiencia de la implementación del algoritmo de multiplicación escalar libre de excepciones para el caso en que el punto P es variable.

5. Trabajos futuros

Como trabajos futuros tendremos que enfocarnos en completar la implementación de los algoritmos para realizar la multiplicación escalar para el caso en que el punto es fijo. Para este caso se seguirá la idea en [7] pero evaluar reemplazar el algoritmo completo (utilizando enmascaramientos) que se utiliza para la adición de puntos por la fórmula completa propuesta en [36].

También se planea evaluar la aritmética de campo para asegurar que de igual forma esta se ejecute en tiempo constante y la posibilidad de ganar en eficiencia desde el punto de vista de la implementación de los algoritmos.

Referencias

[1] Abarúa, Rodrigo, Claudio Valencia y Julio López: *Survey for Performance Security Problems of Passive Side-channel Attacks Countermeasures in ECC*. Cryptology ePrint Archive, Report 2019/010, 2019. https://eprint.iacr.org/2019/010.

[2] Aldaya, Alejandro Cabrera, Cesar Pereida García y Billy Bob Brumley: *From A to Z: Projective coordinates leakage in the wild*. IACR Cryptol. ePrint Arch., 2020:432, 2020. https://eprint.iacr.org/2020/432.

[3] ANSI, ANSI: *X9. 62: 2005: Public Key Cryptography for the Financial Services Industry*. The elliptic curve digital signature algorithm (ECDSA), 2005.

[4] Aranha, Diego F., Felipe Rodrigues Novaes, Akira Takahashi, Mehdi Tibouchi y Yuval Yarom: *LadderLeak: Breaking ECDSA With Less Than One Bit Of Nonce Leakage*. IACR Cryptol. ePrint Arch., 2020:615, 2020. https://eprint.iacr.org/2020/615.

[5] Bernstein, Daniel J.: *Multidigit multiplication for mathematicians*. URL: http://cr.yp.to/papers.html.

[6] Bernstein, Daniel J., Peter Birkner, Marc Joye, Tanja Lange y Christiane Peters: *Twisted Edwards Curves*. En *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings* [6], páginas 389–405. http://dx.doi.org/10.1007/978-3-540-68164-9_26.

[7] Bos, Joppe W., Craig Costello, Patrick Longa y Michael Naehrig: *Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis*. Informe técnico, Microsoft Research, 2014.

[8] Bos, Joppe W., Craig Costello, Patrick Longa y Michael Naehrig: *Selecting elliptic curves for cryptography: an efficiency and security analysis*. J. Cryptographic Engineering, 6(4):259–286, 2016. http://dx.doi.org/10.1007/s13389-015-0097-y.

[9] Bos, Joppe W., Craig Costello, Patrick Longa y Michael Naehrig: *Selecting elliptic curves for cryptography: an efficiency and security analysis.* J. Cryptographic Engineering, 6(4):259–286, 2016. http://dblp.uni-trier.de/db/journals/jce/jce6.html#BosCLN16.

[10] Braun, Benjamin A., Suman Jana y Dan Boneh: *Robust and Efficient Elimination of Cache and Timing Side Channels*. CoRR, abs/1506.00189, 2015. http://arxiv.org/abs/1506.00189.

[11] Brumley, Billy Bob y Nicola Tuveri: *Remote Timing Attacks Are Still Practical*. En Atluri, Vijay y Claudia Diaz (editores): *Computer Security – ESORICS 2011*, páginas 355–371, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg, ISBN 978-3-642-23822-2.

[12] Certicom Research: *Standards for Efficient Cryptography 2 (SEC 2): Recommended Elliptic Curve Domain Parameters. Technical report, Certicom Corp*, 2010.

[13] Chmielewski, L., Massolino P.M.C., J. Vliegen, L. Batina y N. Mentens: *Completing the complete ECC formulae with countermeasures*. Journal of Low Power Electronics and Applications, vol. 7, 2017.

[14] Costello, Craig, Patrick Longa y Michael Naehrig: *A brief discussion on selecting new elliptic curves*. Informe técnico, Microsoft Research, 2014.

[15] De Micheli, Gabrielle, Rémi Piau y Cécile Pierrot: *A Tale of Three Signatures: Practical Attack of ECDSA with wNAF*. En Nitaj, Abderrahmane y Amr Youssef (editores): *Progress in Cryptology - AFRICACRYPT 2020*, páginas 361–381, Cham, 2020. Springer International Publishing, ISBN 978-3-030-51938-4.

[16] Fan, Junfeng, Benedikt Gierlichs y Frederik Vercauteren: *To Infinity and Beyond: Combined Attack on ECC Using Points of Low Order*. En Preneel, Bart y Tsuyoshi Takagi (editores): *Cryptographic Hardware and Embedded Systems – CHES 2011*, páginas 143–159, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg, ISBN 978-3-642-23951-9.

[17] Fan, Junfeng, Xu Guo, Elke De Mulder, Patrick Schaumont, Bart Preneel y Ingrid Verbauwhede: *State-of-the-art of Secure ECC Implementations: A Survey on Known Side-channel Attacks and Countermeasures*. En Plusquellic, Jim y Ken Mai (editores): *HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 13-14 June 2010, Anaheim Convention Center, California, USA*, páginas 76–87. IEEE Computer Society, 2010. https://doi.org/10.1109/HST.2010.5513110.

[18] Flori, Jean-Pierre, Jérôme Plût, Jean-René Reinhard y Martin Ekerå: *Diversity and Transparency for ECC*. IACR Cryptology ePrint Archive, 2015:659, 2015. http://eprint.iacr.org/2015/659.

[19] Goubin, Louis: *A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems*. En Desmedt, Yvo (editor): *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volumen 2567 de *Lecture Notes in Computer Science*, páginas 199–210. Springer, 2003. https://doi.org/10.1007/3-540-36288-6_15.

[20] Hankerson, D., A. Menezes y S. Vanstone: *Guide to Elliptic Curve Cryptography*. first ed. 2004.

[21] Jancar, Jan, Vladimir Sedlacek, Petr Svenda y Marek Sys: *Minerva: The curse of ECDSA nonces*. Cryptology ePrint Archive, Report 2020/728, 2020. https://eprint.iacr.org/2020/728.

[22] Jayasinghe, D., J. Fernando, R. Herath y R. Ragel: *Remote Cache Timing Attack on Advanced Encryption Standard and countermeasures*. En *2010 Fifth International Conference on Information and Automation for Sustainability*, páginas 177–182, 2010.

[23] Joye, Marc y Michael Tunstall: *Exponent Recoding and Regular Exponentiation Algorithms*. En Preneel, Bart (editor): *Progress in Cryptology – AFRICACRYPT 2009*, páginas 334–349, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg, ISBN 978-3-642-02384-2.

[24] Kerry, Cameron F., Acting Secretary y Charles Romine Director: *FIPS PUB 186-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS)*, 2013.

[25] Kocher, P. C.: *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems.* En Koblitz, N. (editor): *Crypto 1996*, volumen 1109 de *Lecture Notes in Computer Science*, páginas 104–113. Springer, 1996.

[26] Legón, C.M., Y. Delgado, I. Millán, Camilo Denis González y P. Bouza: *Side Channel attacks: a real threat to cryptographic algorithms implementations . Ataques de canal colateral : Una amenaza real a las implementaciones de algoritmos criptograficos.* Agosto 2016.

[27] Lochter, Manfred y Johannes Merkle: *Elliptic Curve Cryptography (CCE) Brainpool Standard Curves and Curve Generation*. RFC 5639 (Informational), 2010.

[28] Lochter, Manfred, Johannes Merkle y Jöne Marc Schmidt: *Requirements for Elliptic Curves for High-Assurance Applications*. Torsten Schütze, 2015.

[29] Lochter, Manfred, Johannes Merkle, Jörn-Marc Schmidt y Torsten Schütze: *Requirements for Standard Elliptic Curves*. IACR Cryptology ePrint Archive, 2014:832, 2014. http://eprint.iacr.org/2014/832.

[30] M, Research.: *MSR Elliptic Curve Cryptography Library (MSR ECCLib)*, 2014. http://research.microsoft.com/en-us/projects/nums.

[31] Micheli, Gabrielle De, RÃ©mi Piau y CÃ©cile Pierrot: *A Tale of Three Signatures: practical attack of ECDSA with wNAF*. Cryptology ePrint Archive, Report 2019/861, 2019. https://eprint.iacr.org/2019/861.

[32] Montgomery, P. L.: *Speeding the Pollard and elliptic curve methods of factorization*. Mathematics of computation, 48(177):243–264, 1987.

[33] National Institute of Standards and Technology: *FIPS 186-2. Digital Signature Standard*. Informe técnico, NIST, 2000.

[34] Nayuki: *Bitcoin Cryptography Library*. https://www.nayuki.io/page/bitcoin-cryptography-library.

[35] Okeya, Katsuyuki y Tsuyoshi Takagi: *The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks*, páginas 328–342. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, Germany, 2003, ISBN 3540008470.

[36] Renes, Joost, Craig Costello y Lejla Batina: *Complete Addition Formulas for Prime Order Elliptic Curves*. En *EUROCRYPT*, 2016.

[37] The Legion of the Bouncy Castle: *Bouncy Castle Crypto package*, accedido septiembre 2020. https://www.bouncycastle.org/releasenotes.html.

[38] W, Bosma. y H. W. Lenstra: *Complete systems of two addition laws for elliptic curves*. Journal of Number Theory, 53(2):229-240, 1995.

[39] Wronski, Michal: *Combined small subgroups and side-channel attack on elliptic curves with cofactor divisible by 2m*. International Journal of Electronics and Telecommunications, vol. 65(No 2):203–209, 2019.

[40] Yarom, Yuval y Naomi Benger: *Recovering OpenSSL ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack*. IACR Cryptology ePrint Archive, 2014:140, 2014. https://eprint.iacr.org/2014/140.

[41] Yarom, Yuval y Katrina Falkner: *FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack*. En *23rd USENIX Security Symposium (USENIX Security 14)*, páginas 719–732, San Diego, CA, Agosto 2014. USENIX Association, ISBN 978-1-931971-15-7. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom.

[42] Yessica Caridad Castaño Sainz: *Generación de curvas elípticas con buenas propiedades criptográficas sobre campos finitos primos*. Tesis de Maestría, Universidad de la Habana, Enero 2018.