

INTRODUCCIÓN A LOS CRIPTOSISTEMAS DE CURVA ELÍPTICA

GABRIEL BELINGUERES
gaby@ieee.org

RESUMEN. En este documento veremos qué son las curvas elípticas y como pueden ser usadas para construir criptosistemas de clave pública. Se detallan nociones matemáticas, el problema del Logaritmo Discreto y su análogo sobre curvas elípticas, niveles de seguridad y eficiencia proporcionados por los sistemas ECC, primitivas y esquemas de uso común en ECC y un ejemplo de esquema de firma digital (ECDSA). Por último, se concluye que los ECC son lo suficientemente maduros como para poder emplearlos en ambientes donde, debido a sus restricciones de costo y recursos computacionales, antes no se podía incluir criptografía de clave pública. Gracias a los ECC, ahora existe una opción viable para la solución de este problema.

1. INTRODUCCIÓN

Todos los sistemas de clave pública conocidos en la actualidad basan su seguridad en la resolución de algún problema matemático que, por su gran magnitud, es casi imposible de resolver en la práctica. Pero estos problemas matemáticos en los que se basan los métodos tradicionales de criptografía de clave pública son jóvenes. Aún cuando éstos han sido estudiado por varios cientos de años por matemáticos como Euclides, Fermat o Euler, estos problemas se estudiaban para obtener otro tipo de resultados, como ecuaciones que revelen propiedades de los problemas.

Puesto que el estudio de estos problemas no estaba destinado para uso criptográfico, no se puede decir que tenemos tanta experiencia en las matemáticas subyacentes en que se basa la criptografía, como suele pasar de tanto en tanto, que se descubren nuevas propiedades o algoritmos para “romper” los criptosistemas en un tiempo menor al que se había calculado inicialmente. En efecto, a menudo los resultados que se requieren en criptografía son del tipo que cierta propiedad *no vale* o que ciertos algoritmos eficientes no se puedan desarrollar. Claro queda que este no era el objetivo de aquellos antiguos matemáticos, puesto que no solo no existía la criptografía de clave pública, presentada al mundo en 1976 por Diffie y Hellman,¹ sino que ni siquiera existían las computadoras, de modo que tratar de desarrollar un algoritmo que resuelva un problema donde los números involucrados eran de 400 cifras decimales, simplemente no era de interés.

En estos momentos, los métodos de clave pública más utilizados son el RSA, para encriptación y firma digital, el Diffie-Hellman para acuerdo de claves, y el DSA para firmas digitales.

¹En realidad, los conceptos de la criptografía de clave pública fueron desarrollados por Merkle en 1974, quien envió un artículo para publicar en *Communications of the ACM*; dicho artículo no apareció sino hasta 1978 [9].

Debido a la aparición en los últimos años de métodos que resuelven el problema matemático en que se basan los algoritmos arriba mencionados en un tiempo menor al que se había pensado (en el actual estado del conocimiento de dichos problemas cuando éstos fueron primero concebidos como método criptográfico), se necesita agrandar el espacio de claves para “emparchar” dicho sistema.

Como una opción, en 1985, Neil Koblitz y Victor Miller (independientemente) propusieron el **Elliptic Curve Cryptosystem** (ECC), o Criptosistema de Curva Elíptica, cuya seguridad descansa en el mismo problema que los métodos de Diffie-Hellman y DSA, pero en vez de usar números enteros como los *símbolos* del alfabeto del mensaje a encriptar (o firmar), usa puntos en un objeto matemático llamado *Curva Elíptica*. ECC puede ser usado tanto para encriptar como para firmar digitalmente. Hasta el momento, no se conoce ataque alguno cuyo tiempo de ejecución esperado sea sub exponencial para poder romper los ECC, esto hace que para obtener el mismo nivel de seguridad que brindan los otros sistemas, el espacio de claves de ECC sea mucho más pequeño, lo que lo hace una tecnología adecuada para utilizar en ambientes restringidos en recursos (memoria, costo, velocidad, ancho de banda, etc.)

En este documento supondremos que el lector esta familiarizado con los conceptos de los sistemas de clave pública, confidencialidad, firmas digitales, espacios de claves, clases de complejidad, teoría de números y álgebra abstracta, así como un cierto conocimiento de los sistemas más utilizados en la práctica como RSA, Diffie-Hellman o DSA. No obstante, el Apéndice A es incluido a causa de que es la base para comprender los principios en que se basan las curvas elípticas. Si Ud. no esta familiarizado con estos conceptos, puede consultarlos en este momento.

2. LAS CURVAS ELÍPTICAS

En esta sección presentamos la teoría de las curvas elípticas en forma general, y luego se estudia la teoría sobre otras estructuras algebraicas.

2.1. Generalidades. La definición general de las curvas elípticas es la siguiente [15].

Definición 2.1. Una curva elíptica es una ecuación

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5$$

donde x e y son las indeterminadas, y los a_1, \dots, a_5 son elementos constantes de un campo.

Aunque esta ecuación puede ser estudiada sobre varias estructuras algebraicas, como un anillo o campo; para nuestros propósitos, consideraremos solamente las curvas elípticas sobre un campo. En este caso, los coeficientes a_i son elementos del campo, y nuestra tarea es encontrar pares (x, y) con x e y en el campo, que satisfagan la ecuación.

Para entender de que estamos hablando, consideremos el campo de los números reales \mathbb{R} . Podemos obtener un gráfico de la curva elíptica fijando un x y resolviendo la ecuación cuadrática en y . Veamos como quedaría el grafico de la siguiente curva:

$$y^2 = x^3 - 4x + 0.67$$

donde $a_1 = a_2 = a_3 = 0; a_4 = -4; a_5 = 0.67$.

En la Figura 1 se muestra el grafico resultante.

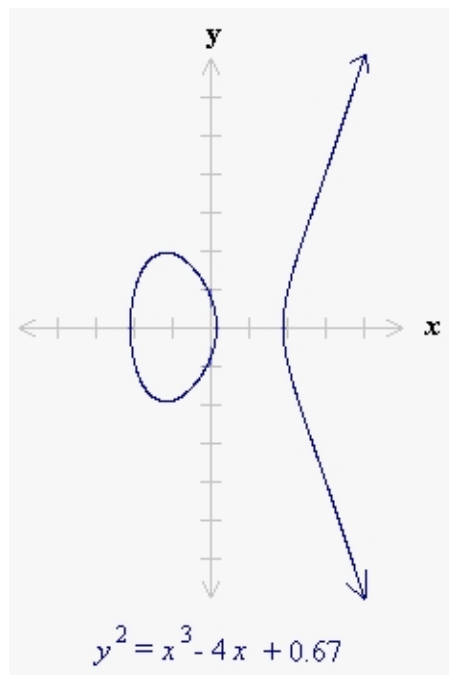


FIGURA 1. Gráfico de una curva elíptica [1].

El gráfico consiste de dos partes separadas, aunque podrían estar pegadas por un solo punto, o bien consistir solamente de una parte, en forma de “campana”.

Veremos que el conjunto de soluciones de una curva elíptica tiene propiedades interesantes. En particular, se puede hacer del conjunto un *grupo*, es decir proveer al conjunto de una operación binaria y un elemento identidad, lo cual nos habilita para hacer criptografía.

Yendo al caso de los números reales, se podría probar que si $x^3 + ax + b$ no tiene factores repetidos, o equivalentemente $4a^3 + 27b^2 \neq 0$, entonces las curvas elípticas de la forma

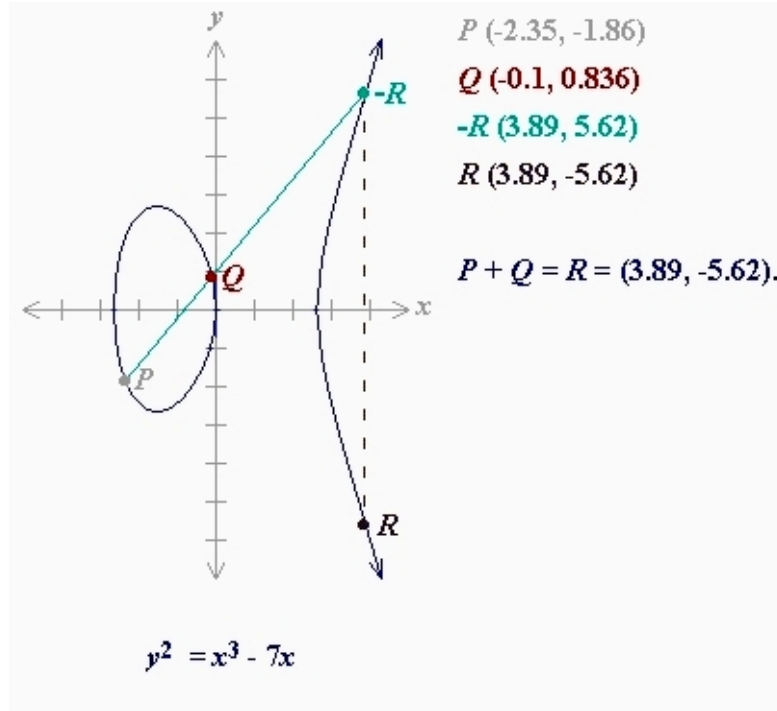
$$y^2 = x^3 + ax + b$$

forman un grupo.

Entonces, pasemos a definir la operación, siguiendo la Figura 2. Tome dos puntos P y Q de la curva, y trace la línea que pasa por ambos puntos. En el caso general esta línea siempre tiene un punto de intersección con la curva. Ahora tome este tercer punto en la intersección de la línea con la curva y trace una línea vertical. El otro punto R de intersección con la curva de esta línea vertical se define como la *suma* de P y Q , en símbolos, $R = P + Q$.

La Figura 2 también nos muestra como definimos el elemento opuesto $-R$ a un punto R de la curva: simplemente, el punto opuesto a $R = (x, y)$ es el punto $-R = (x, -y)$.

Si $P_1 = P_2$, es decir que queremos obtener $R = P + P$ (vea Figura 3), entonces la línea a ser construida en el primer paso es la tangente a la curva, el cual otra vez tiene otro punto de intersección con la curva. Note que el grupo es conmutativo.

FIGURA 2. Suma de P y Q [1].

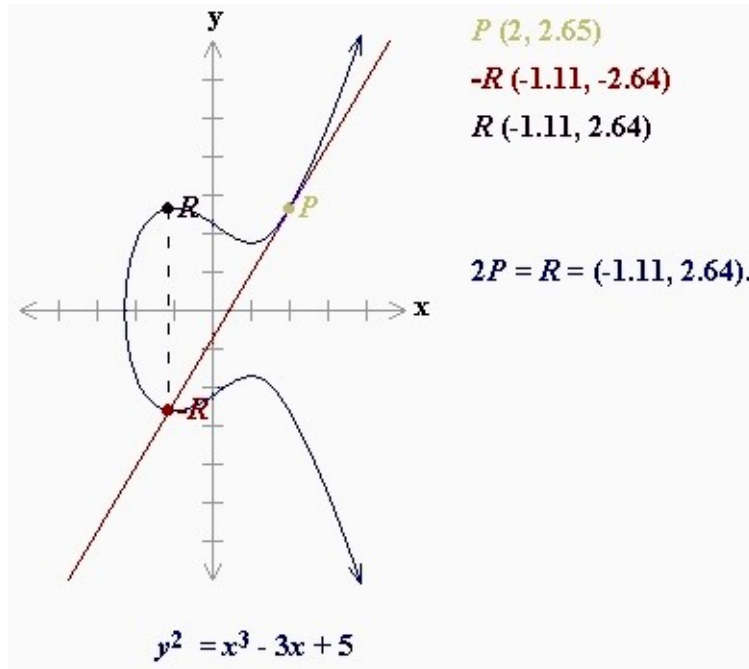
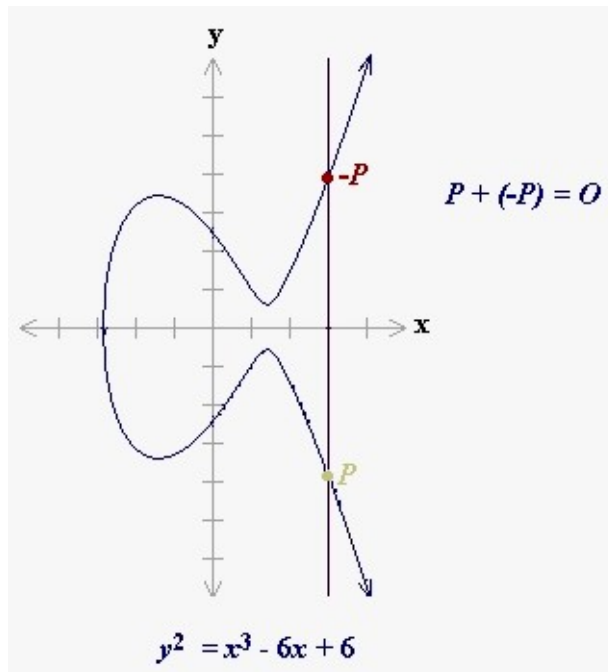
Una parte compleja es definir el elemento identidad del grupo. La pregunta a responder es: ¿Que punto de la curva sumado a un punto P cualquiera resulta en el punto P ? Solamente podemos encontrar una respuesta a esta pregunta si un punto extra es agregado a la curva. Este punto extra se llama *point at infinity* o *punto en el infinito*, y se lo designa con \mathcal{O} . El punto \mathcal{O} esta en un lugar infinitamente lejos sobre el eje vertical, y es la identidad del grupo de la curva elíptica.

Por ejemplo, observe el punto P en Figura 4. Dada la definición del elemento identidad (o elemento nulo), $P + (-P) = \mathcal{O}$. Observe también (Figura 5) el caso especial en el que queremos obtener el punto $P + P = 2P$, donde la recta tangente a la curva en el punto P es la recta vertical (lo cual sucede solo cuando $P = (x, y)$ con $y = 0$); en este caso también obtenemos $2P = \mathcal{O}$.

Puede demostrarse que la operación de suma es una operación binaria de grupo bien definida, aunque algunos de los requerimientos (por Ej. asociatividad) no son nada fáciles de probar.

Aunque arriba dimos una definición general de las curvas elípticas, no se considera práctico utilizarlas en forma general, ni tampoco utilizarla en el campo de los \mathbb{R} , debido a los errores de redondeo y truncación de los valores. Por eso, aquí se considerarán las curvas elípticas definidas sobre \mathbb{Z}_p donde p es un número primo impar y sobre los campos finitos de la forma \mathbb{F}_{2^m} , $m \geq 1$, ya que éstos conjuntos producen las implementaciones más eficientes de la aritmética de curva elíptica [8].

Antes de entrar en detalle con la descripción de los campos finitos más utilizados en criptografía, damos unas definiciones más:


 FIGURA 3. Suma del mismo punto P [1].

 FIGURA 4. Suma del punto P con su opuesto [1].

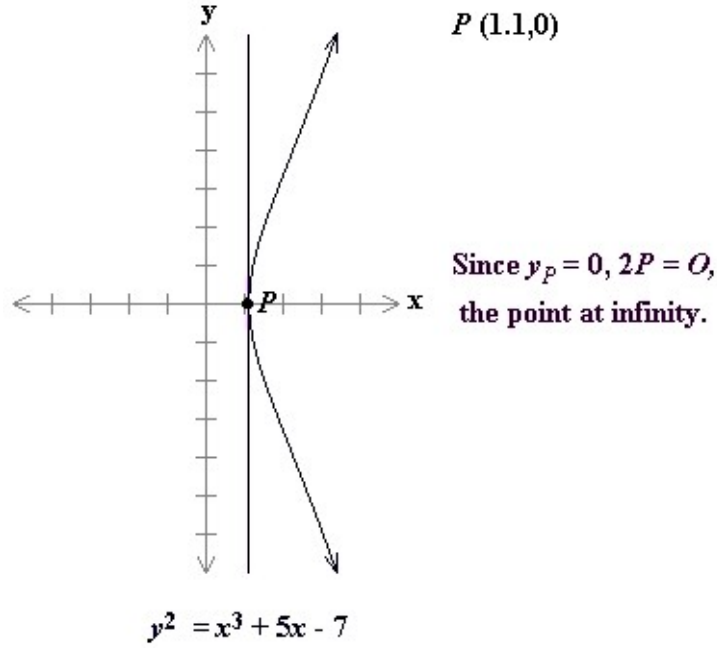


FIGURA 5. Suma del punto $P = (x, y)$ con si mismo, donde $y = 0$ [1].

Definición 2.2. El *orden* de una curva elíptica E definida sobre el campo \mathbb{F}_q es el número de puntos sobre la curva elíptica E , incluyendo \mathcal{O} . Esto es denotado por $\#E(\mathbb{F}_q)$.

Definición 2.3. Si k es un entero positivo, entonces $kP = \sum_{i=1}^k P$ denota el punto obtenido de sumar k copias del punto P a si mismo. El proceso de calcular kP a partir de P y k es llamado *multiplicación escalar*.

Definición 2.4. El *orden de un punto* P es el entero positivo más pequeño n tal que $nP = \mathcal{O}$ (el punto en el infinito).

2.2. Curvas Elípticas sobre \mathbb{Z}_p .

Definición 2.5. Una *curva elíptica* E sobre \mathbb{Z}_p , denotada $E(\mathbb{Z}_p)$ es definida por la ecuación de la forma

$$(2.1) \quad y^2 \equiv x^3 + ax + b \pmod{p}$$

donde $a, b \in \mathbb{Z}_p$, y $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, junto con el *punto en el infinito* \mathcal{O} . El conjunto $E(\mathbb{Z}_p)$ consiste de todos los puntos (x, y) , $x \in \mathbb{Z}_p$, $y \in \mathbb{Z}_p$, los cuales satisfacen la Ecuación 2.1.

Ejemplo 2.6. Si $p = 23$, y consideramos la curva elíptica $E : y^2 = x^3 + x + 1$ definida sobre \mathbb{Z}_{23} donde las constantes usadas fueron $a = b = 1$. Note que $4a^3 + 27b^2 = 4 + 4 = 8 \not\equiv 0 \pmod{23}$, de manera que en realidad es una curva elíptica.

Teorema 2.7 (Hasse). *Sea $E(\mathbb{Z}_p)$ una curva elíptica, con p primo impar, entonces $p + 1 - 2\sqrt{p} \leq \#E(\mathbb{Z}_p) \leq p + 1 + 2\sqrt{p}$.*

Aunque la operación de suma parece difícil de ejecutar, es posible encontrar una fórmula que nos da las coordenadas (x_S, y_S) de la suma $P_1 + P_2$ como función de sus coordenadas (x_1, y_1) y (x_2, y_2) . La suma de puntos se define así [11]:

Definición 2.8. La operación de *suma* de dos puntos de la curva se define según las siguientes reglas:

1. $\mathcal{O} + \mathcal{O} = \mathcal{O}$.
2. $P + \mathcal{O} = \mathcal{O} + P = P$ para todo $P \in E(\mathbb{Z}_p)$.
3. Si $P = (x, y) \in E(\mathbb{Z}_p)$, entonces $(x, y) + (x, -y) = \mathcal{O}$. (El punto $(x, -y)$ es denotado $-P$, y es llamado el *negativo* de P ; observe que $-P$ es en realidad un punto de la curva.)
4. Sea $P_1 = (x_1, y_1) \in E(\mathbb{Z}_p)$ y $P_2 = (x_2, y_2) \in E(\mathbb{Z}_p)$, donde $x_1 \neq x_2$. Entonces $P_1 + P_2 = (x_S, y_S)$ es:

$$\begin{aligned} x_S &\equiv \lambda^2 - x_1 - x_2 \pmod{p} \\ y_S &\equiv \lambda(x_1 - x_S) - y_1 \pmod{p}, \text{ donde} \\ \lambda &\equiv \frac{y_2 - y_1}{x_2 - x_1} \pmod{p}. \end{aligned}$$

5. Sea $P_1 = (x_1, y_1) \in E(\mathbb{Z}_p)$ con $y_1 \neq 0$. Entonces $P_1 + P_1 = (x_S, y_S)$, donde:

$$\begin{aligned} x_S &\equiv \lambda^2 - 2x_1 \pmod{p} \\ y_S &\equiv \lambda(x_1 - x_S) - y_1 \pmod{p}, \text{ donde} \\ \lambda &\equiv \frac{3x_1^2 + a}{2y_1} \pmod{p}. \end{aligned}$$

Ejemplo 2.9. Considere la curva elíptica definida en el Ejemplo 2.6. Sumemos los puntos $P = (3, 10)$ y $Q = (9, 7)$ (note que $P, Q \in E(\mathbb{Z}_{23})$.) Entonces $P + Q = (x_S, y_S)$ se calcula como sigue:

$$\begin{aligned} \lambda &= \frac{7 - 10}{9 - 3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \mathbb{Z}_{23}, \\ x_S &= 11^2 - 3 - 9 = 6 - 3 - 9 = -6 \equiv 17 \pmod{23}, \\ y_S &= 11(3 - (-6)) - 10 = 11(9) - 10 = 89 \equiv 20 \pmod{23}. \end{aligned}$$

Por lo tanto, $P + Q = (17, 20)$.

Ejemplo 2.10. Sea $P = (3, 10)$. Entonces $2P = P + P = (x_S, y_S)$ se calcula así:

$$\begin{aligned} \lambda &= \frac{3(3^2) + 1}{20} = \frac{5}{20} = \frac{1}{4} = 6 \in \mathbb{Z}_{23}, \\ x_S &= 6^2 - 6 = 30 \equiv 7 \pmod{23}, \\ y_S &= 6(3 - 7) - 10 = -24 - 10 = -11 \equiv 12 \pmod{23}. \end{aligned}$$

Por lo tanto, $2P = (7, 12)$.

La operación de grupo para una curva elíptica $E(\mathbb{Z}_p)$ se la llamó *suma* por razones históricas. En contraste, la operación de grupo en \mathbb{Z}_p^* es *multiplicación*. A causa de que las diferencias entre las notaciones suelen ser confusas, se incluye en Tabla 1 las correspondencias entre ambas notaciones (extraída de [8].)

Concepto	\mathbb{Z}_p^*	$E(\mathbb{Z}_p)$
Elementos de grupo	Números enteros $\{1, 2, \dots, p-1\}$	Puntos (x, y) sobre E más \mathcal{O}
Operación de grupo	Multiplicación modulo p	Suma de puntos
Notación	Elementos: g, h Multiplicación: gh Inverso: g^{-1} División: g/h Exponenciación: g^a	Elementos: P, Q Suma: $P + Q$ Negativo: $-P$ Substracción: $P - Q$ Múltiplo: aP
Problema del Logaritmo Discreto (DLP)	Dado $g \in \mathbb{Z}_p^*$ y $h = g^a \pmod{p}$, encontrar a	Dado $P \in E(\mathbb{Z}_p)$ y $Q = aP$, encontrar a

TABLA 1. Correspondencia entre las notaciones de \mathbb{Z}_p^* y $E(\mathbb{Z}_p)$.

2.3. Curvas Elípticas sobre \mathbb{F}_{2^m} . Los campos finitos de la forma \mathbb{F}_{2^m} con $m \geq 1$ son convenientes de usar, puesto que sus elementos encajan a la perfección en una palabra de datos de longitud m bits. Esto es porque los elementos de \mathbb{F}_{2^m} son representados por el conjunto de polinomios² binarios de grado $\leq m-1$:

$$\mathbb{F}_{2^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_1x + a_0 \mid a_i \in \{0, 1\}\}$$

donde estos polinomios se representan en la computadora por la palabra de datos de m bits $(a_{m-1}a_{m-2} \dots a_1a_0)$.

Definición 2.11. Sea \mathbb{F}_{2^m} un campo finito, y sean $a, b \in \mathbb{F}_{2^m}$ satisfaciendo $b \neq 0$ en \mathbb{F}_{2^m} . Entonces una curva elíptica (no supersingular) $E(\mathbb{F}_{2^m})$ sobre \mathbb{F}_{2^m} definida por los parámetros $a, b \in \mathbb{F}_{2^m}$ consiste del conjunto de soluciones o puntos $P = (x, y)$ para $x, y \in \mathbb{F}_{2^m}$ a la ecuación:

$$y^2 + xy = x^3 + ax^2 + b \text{ en } \mathbb{F}_{2^m}$$

junto con un punto extra \mathcal{O} llamado el punto en el infinito.

Comentario 2.12. La suma y resta modulo 2 se realizan haciendo el XOR bit a bit de dos palabras de datos, mientras que la multiplicación e inverso multiplicativo están definidas modulo un polinomio binario irreducible $f(x)$ de grado m .

Ejemplo 2.13. Podemos definir una curva elíptica $E(\mathbb{F}_{2^4})$, donde $a, b \in \mathbb{F}_{2^4}$. Es decir que $a = (a_3a_2a_1a_0)$ y $b = (b_3b_2b_1b_0) \neq (0000)$ son palabras binarias de 4 bits. La curva E la definen las palabras $x, y \in \mathbb{F}_{2^4}$ que satisfagan la ecuación $y^2 + xy = x^3 + ax^2 + b$, donde las operaciones (salvo la suma) se hacen modulo el polinomio irreducible en \mathbb{F}_{2^4} de grado 4 $f(x) = x^4 + x + 1$.

Definición 2.14. Una *Curva de Koblitz* es una curva elíptica sobre \mathbb{F}_{2^m} con $a, b \in \{0, 1\}$.

²A esta representación se la llama *representación polinomial*. Existe otra representación, conocida como *representación de base optima*, la cual es más eficiente realizando multiplicaciones. Puesto que este documento trata solo los conceptos teóricos relacionados con la ECC, esta representación es irrelevante.

Ejemplo 2.15. Si el campo es \mathbb{F}_{2^4} , $a = (0)_{10} = (0000)_2$, $b = (1)_{10} = (0001)_2$, $f(x) = x^4 + x + 1$, entonces $E : y^2 + xy = x^3 + 1$ es una curva elíptica de Koblitz no supersingular.

También se puede usar una notación muy conveniente ya que evita tener que lidiar con $f(x)$ cada vez que se quieran realizar multiplicaciones y divisiones. Recuerde la Definición A.46 sobre los polinomios irreducibles primitivos. Cada elemento de $\mathbb{F}_{2^m}^*$ puede escribirse como el polinomio $g^i = x^i \bmod f(x)$, $0 \leq i < 2^m - 1$, si $f(x)$ es irreducible primitivo. Luego, dados números i, j tenemos que $g^i g^j = g^{(i+j) \bmod (2^m-1)}$ y $(g^i)^j = g^{ij} = g^{ij \bmod (2^m-1)}$.

Ejemplo 2.16. Sea E la curva definida sobre \mathbb{F}_{2^4} , $E : y^2 + xy = x^3 + g^4 x^2 + g^0$, donde $a = g^4$, $b = g^0 = 1$. Veremos que el punto (g^6, g^8) es un punto en la curva:

$$\begin{aligned} (g^8)^2 + g^6 g^8 &= (g^6)^3 + g^4 (g^6)^2 + g^0 \\ g^{16} + g^{14} &= g^{18} + g^{16} + g^0 \\ g + g^{14} &= g^3 + g + g^0 \\ (0010) + (1001) &= (1000) + (0010) + (0001) \\ (1011) &= (1011) \end{aligned}$$

Es necesario decir que en criptografía las únicas curvas elípticas sobre \mathbb{F}_{2^m} de interés son las curvas elípticas no supersingulares.

Teorema 2.17 (Hasse). *Sea $E(\mathbb{F}_{2^m})$ una curva elíptica, con $m \geq 1$, entonces $2^m + 1 - 2\sqrt{2^m} \leq \#E(\mathbb{F}_{2^m}) \leq 2^m + 1 + 2\sqrt{2^m}$.*

De vuelta es posible definir una regla para sumar puntos en E . La suma se define como sigue:

Definición 2.18. La operación de *suma* de dos puntos de la curva se define según las siguientes reglas:

1. $\mathcal{O} + \mathcal{O} = \mathcal{O}$.
2. $P + \mathcal{O} = \mathcal{O} + P = P$ para todo $P \in E(\mathbb{F}_{2^m})$.
3. Si $P = (x, y) \in E(\mathbb{F}_{2^m})$, entonces $(x, y) + (x, x+y) = \mathcal{O}$. (El punto $(x, x+y)$ es denotado $-P$, y es llamado el *negativo* de P ; observe que $-P$ es en realidad un punto de la curva.)
4. Sea $P_1 = (x_1, y_1) \in E(\mathbb{F}_{2^m})$ y $P_2 = (x_2, y_2) \in E(\mathbb{F}_{2^m})$, donde $x_1 \neq x_2$. Entonces $P_1 + P_2 = (x_S, y_S)$ es:

$$\begin{aligned} x_S &= \lambda^2 + \lambda + x_1 + x_2 + a \text{ en } \mathbb{F}_{2^m} \\ y_S &= \lambda(x_1 + x_S) + x_S + y_1 \text{ en } \mathbb{F}_{2^m}, \text{ donde} \\ \lambda &= \frac{y_1 + y_2}{x_1 + x_2} \text{ en } \mathbb{F}_{2^m}. \end{aligned}$$

5. Sea $P_1 = (x_1, y_1) \in E(\mathbb{F}_{2^m})$ con $x_1 \neq 0$. Entonces $P_1 + P_1 = (x_S, y_S)$, donde:

$$\begin{aligned} x_S &= \lambda^2 + \lambda + a \text{ en } \mathbb{F}_{2^m} \\ y_S &= x_1^2 + (\lambda + 1)x_S \text{ en } \mathbb{F}_{2^m}, \text{ donde} \\ \lambda &= x_1 + \frac{y_1}{x_1} \text{ en } \mathbb{F}_{2^m}. \end{aligned}$$

3. EL PROBLEMA DEL LOGARITMO DISCRETO (DLP)

El protocolo de acuerdo de claves Diffie-Hellman (nombre debido a sus inventores) proporcionó la primer solución práctica al problema de distribución de claves, permitiendo que dos partes establezcan un secreto compartido intercambiando mensajes sobre un canal inseguro. Su seguridad descansa en la intratabilidad del **Discrete Logarithm Problem (DLP)**, o **Problema de los Logaritmos Discretos**. Antes de definir el problema, debemos definir qué es un *logaritmo discreto*.

Definición 3.1. Sea G un grupo cíclico finito de orden n . Sea α un generador de G , y sea $\beta \in G$. El *logaritmo discreto de β a la base α* , denotado $\log_\alpha \beta$, es el único entero x , $0 \leq x \leq n - 1$, tal que $\beta = \alpha^x$.

Ejemplo 3.2. El logaritmo discreto en el grupo \mathbb{Z}_p^* con p primo y la multiplicación modulo p es un problema de interés en criptografía. Sin embargo, no lo es cuando el grupo es \mathbb{Z}_n con la suma modulo n , puesto que es un problema fácil de resolver en todos los aspectos.

Definición 3.3. El *problema de los logaritmos discretos generalizado (GDLP)* es el siguiente: dados un grupo cíclico finito G de orden n , un generador $\alpha \in G$, y un elemento $\beta \in G$, encontrar el entero x , $0 \leq x \leq n - 1$, tal que $\alpha^x = \beta$.

En el GDLP, el único requerimiento que se le hace a G para que tenga un uso práctico en criptografía es que el GDLP sea un problema difícil de resolver en G , mientras que la exponenciación sea una operación eficiente de ejecutar.

Aquí hemos definido la generalización del DLP extendida a cualquier grupo cíclico finito. El DLP es una “instancia” del GDLP, en el cual $G = \mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$, donde p es un número primo y el orden de \mathbb{Z}_p^* es $p-1$, de manera que dado un generador $\alpha \in \mathbb{Z}_p^*$, y un elemento $\beta \in \mathbb{Z}_p^*$, el problema es encontrar el entero x , $0 \leq x \leq p-2$, tal que $\alpha^x \equiv \beta \pmod{p}$. La operación del grupo \mathbb{Z}_p^* es la multiplicación modulo p . Por supuesto, también el GDLP puede instanciarse usando como grupo campos finitos de la forma \mathbb{F}_{p^m} con p primo y $m \geq 1$.

Como ejemplo de uso del DLP, en el protocolo Diffie-Hellman básico³, dos partes A y B quieren compartir un secreto. A genera un primo p y generador $\alpha \in \mathbb{Z}_p^*$, ($2 \leq \alpha \leq p-2$), genera un número aleatorio (secreto) x , $1 \leq x \leq p-2$, calcula $\alpha^x \pmod{p}$, y envía a B el mensaje $(p, \alpha, \alpha^x \pmod{p})$.⁴

Una vez que B recibió el mensaje, elige un número aleatorio y , $1 \leq y \leq p-2$, y envía el mensaje $(\alpha^y \pmod{p})$ hacia A. B luego calcula el secreto compartido haciendo $K = (\alpha^x)^y \pmod{p}$.

Por otra parte, A recibe el mensaje de B, y calcula el secreto compartido haciendo $K = (\alpha^y)^x \pmod{p}$.

Aunque el problema real a que se enfrenta un adversario en encontrar $\alpha^{xy} \pmod{p}$, dado que conoce el primo p , un generador α de \mathbb{Z}_p^* , y elementos $\alpha^x \pmod{p}$ y $\alpha^y \pmod{p}$, se cree que la resolución de este problema tiene la misma complejidad que la del DLP.

³El protocolo básico no provee ni autenticación de las partes involucradas en la comunicación, ni autenticación de los secretos generados.

⁴Este modo de operación se lo conoce como Diffie-Hellman *efímero*, puesto que los parámetros son regenerados cada cierto lapso de tiempo. En contraste, se puede incluir el valor público $\alpha^x \pmod{p}$ en un certificado firmado, los cuales están disponibles *a priori* entre las partes, y entonces no se necesita enviar ningún mensaje.

Entonces, en el caso del protocolo Diffie-Hellman, la intención de un oponente es encontrar los valores aleatorios privados, es decir x e y , o bien xy . Si un atacante logra esto, el cálculo del secreto compartido es eficiente.

Otros criptosistemas han sido propuestos cuya seguridad se basa en el DLP, entre ellos están:

- Los esquemas de acuerdo de claves derivados del Diffie-Hellman, tal como el de ElGamal, la familia de protocolos MTI y el protocolo STS (Station-to-Station).
- El esquema de encriptación ElGamal.
- El esquema de firma digital ElGamal y sus variantes, como DSA (Digital Signature Algorithm)⁵, el esquema de firma de Schnorr, y el esquema ElGamal con recuperación de mensaje de Nyberg-Rueppel.

Todos estas variantes del DLP se pueden consultar en [9].

Además, se puede *generalizar* la definición del GDLP para uso con *cualquier* grupo G y elementos $\alpha, \beta \in G$, donde el problema es encontrar un entero x tal que $\alpha^x = \beta$, si es que tal entero existe. Aquí no se requiere que G sea cíclico, ni tampoco se requiere que α sea un generador de G . Se cree que este problema es mucho más difícil de resolver.

Entonces, otros grupos son de interés para criptografía, como el grupo de puntos de una curva elíptica definida sobre un campo finito. Luego, podemos instanciar el problema con el grupo de las curvas elípticas sobre un campo finito:

Definición 3.4 (ECDLP). Sea E una curva elíptica definida sobre un campo finito \mathbb{F}_q , y sea $G \in E(\mathbb{F}_q)$ un punto sobre E de orden n (número primo y grande). El ECDLP es, dados E , G , y un múltiplo escalar Q de G , determinar un entero l tal que $Q = lG$.

Note que la similitud de las definiciones hace que todos los criptosistemas basados en el DLP puedan ser adaptados utilizando curvas elípticas. Así, tenemos variantes de los protocolos y esquemas anteriores convertidos a curvas elípticas, y entonces tenemos ECDSA, EC Diffie-Hellman, encriptación EC ElGamal, etc. Algunas leves modificaciones técnicas son necesarias para adaptarlos al grupo de las curvas elípticas, pero los principios subyacentes son los mismos que para los otros sistemas basados en el DLP.

Dado que el ECDLP es similar al GDLP, la pregunta que todos se hacen es: Si el problema del logaritmo discreto es definido idénticamente sobre \mathbb{Z}_p^* , $\mathbb{F}_{p^m}^*$, $E(\mathbb{Z}_p)$ o $E(\mathbb{F}_{2^m})$: ¿Cuál es la ganancia en seguridad que se obtiene al usar curvas elípticas, en vez de los ya bastante investigados esquemas y protocolos sobre los grupos \mathbb{Z}_p^* y $\mathbb{F}_{p^m}^*$, tales como RSA [13] o los otros basados en el DLP?

La diferencia esta en los tipos de ataques que se pueden implementar contra el mismo problema⁶ pero sobre distintos grupos.

A modo de resumen, en las siguientes subsecciones presentamos una revisión de los algoritmos conocidos para atacar los problemas matemáticos en los que se basan los métodos de clave pública más conocidos.

⁵Desde que el gobierno de U.S. reconoció como estándar el esquema DSA, también se lo conoce como DSS (Digital Signature Standard).

⁶La seguridad del esquema RSA no esta basada en el DLP, sino en el Integer Factorization Problem (IFP) o Problema de la Factorización Entera. Los ECC no proveen ninguna ventaja en seguridad cuando los métodos basados en el IFP usan como grupo las curvas elípticas.

3.1. Ataques conocidos al IFP. El método RSA utiliza \mathbb{Z}_n^* como grupo, donde $n = pq$ con p y q números primos grandes. Existen algoritmos que resuelven el problema de la factorización de n de orden sub-exponencial. Estos algoritmos explotan propiedades de los números enteros para lograr su objetivo. Por ej., los números enteros tienen una propiedad conocida llamada *smoothness*. Un número n es B -smooth si todos sus factores primos son $\leq B$ (note que esta propiedad es de los números enteros, no del criptosistema). Existen algoritmos que explotan esta propiedad para factorizar números enteros grandes en forma eficiente. Los algoritmos más eficientes son el Quadratic Sieve (QS) y el Number Field Sieve (NFS). Estos algoritmos utilizan (además de explotar la propiedad de smoothness) una base de datos de números primos, llamada base de factores (*factor base*); esta base de datos hace que aumente la probabilidad de encontrar factores del número a factorizar (el modulo n en RSA). En especial, NFS utiliza dos bases de factores y es fácil de paralelizar.

3.2. Ataques conocidos al DLP. En cuanto al DLP sobre \mathbb{Z}_p^* y $\mathbb{F}_{p^m}^*$, los algoritmos más conocidos son el Pollard- ρ (Pollard's rho) que tiene un tiempo esperado de ejecución de $O(\sqrt{n})$ operaciones, donde n es el orden del grupo cíclico. Note que este algoritmo no es sub-exponencial. El algoritmo de Pohlig-Hellman es eficiente solo cuando el orden n del grupo es smooth. El algoritmo más eficiente que no explota ninguna característica de los elementos del grupo es el index-calculus. En este método, una base de datos de primos pequeños y sus logaritmos correspondientes es calculada, esto sirve para poder calcular eficientemente logaritmos de elementos arbitrarios del grupo. El algoritmo index-calculus y todas sus variantes, como el algoritmo de Coppersmith y el Number Field Sieve (adaptado para logaritmos) tienen un tiempo de ejecución sub-exponencial y son fácilmente paralelizables.

Como se puede ver, el principio de funcionamiento del Number Field Sieve es el mismo tanto para resolver el IFP como para el DLP. Por esta razón, si una mejora en los algoritmos para el IFP o DLP es encontrada, entonces poco después es esperado que un algoritmo similar mejorado se encuentre para el otro problema [3].

3.3. Ataques conocidos al ECDLP. No se conocen algoritmos sub-exponenciales generales (es decir, que no exploten ninguna propiedad de las puntos sobre curvas elípticas) para resolver el ECDLP. Más específicamente, no se conoce un algoritmo index-calculus para el ECDLP, puesto que no se conoce el concepto de smoothness de un punto sobre una curva elíptica⁷. Por esta razón, se cree que el ECDLP es mucho más difícil de resolver que el IFP o el DLP, en el sentido que no se conocen algoritmos de propósito general y tiempo sub-exponencial para resolverlo.

Los mejores algoritmos generales conocidos a la fecha están basados en el Pollard- ρ y el Pollard- λ . El Pollard- ρ toma alrededor de $\sqrt{\pi n/2}$ pasos (cada paso representa la suma de dos puntos en una curva elíptica), y el Pollard- λ toma alrededor de $2\sqrt{n}$ pasos. Ambos métodos pueden ser paralelizados.

Recientemente se mostró que el método Pollard- ρ puede ser acelerado por un factor de $\sqrt{2}$. Entonces el tiempo esperado de ejecución del método Pollard- ρ con esta mejora es de $\sqrt{\pi n/4}$ pasos [6].

⁷Muchos expertos se han mostrado cautelosos sobre el uso masificado de los ECC, argumentando que alguien definirá en algún momento el concepto de smoothness y la ventaja en seguridad que se supone que presentan los ECC quedaría invalidada [14].

También se mostró que esta aceleración puede ser mejorada cuando E es una curva elíptica sobre $\mathbb{F}_{2^e d}$, el cual está definido sobre \mathbb{F}_{2^e} . En este caso se muestra que el método Pollard- ρ puede ser acelerado por un factor de $\sqrt{2d}$. Por ej., la curva de Koblitz $E : y^2 + xy = x^3 + x^2 + 1$ sobre $\mathbb{F}_{2^{163}}$ tiene la propiedad que $\#E(\mathbb{F}_{2^{163}}) = 2n$ donde n es un número primo de 162 bits. Como un resultado de la mejora al método Pollard- ρ , el ECDLP en $E(\mathbb{F}_{2^{163}})$ puede ser resuelto en unos 2^{77} pasos en contraste con los 2^{81} requeridos para resolver el ECDLP para una curva aleatoria de orden similar.

Finalmente, aunque no se conocen algoritmos sub-exponenciales generales para resolver el ECDLP, dos clases de curvas son susceptibles a algoritmos de propósito especial. Primero, las curvas elípticas E sobre \mathbb{F}_q con n (el orden del punto base G) dividiendo a $q^B - 1$ para B pequeño (estas son las curvas *supersingulares*) son susceptibles a ataques. Estos ataques reducen eficientemente el ECDLP sobre esas curvas al DLP tradicional en una extensión de grado pequeño de \mathbb{F}_q . Segundo, curvas elípticas E sobre \mathbb{F}_q con $\#E(\mathbb{F}_q) = q$ (estas curvas también son llamadas curvas *anómalas*) son susceptibles a ataque. Este ataque mapea eficientemente la curva elíptica E dentro del grupo aditivo de \mathbb{F}_q . Ambas clases débiles de curvas son excluidas en este documento y también son fácilmente detectables, de manera que nunca se las genera en la práctica.

4. COMPARACIÓN CON OTROS CRIPTOSISTEMAS

Antes de introducir al lector al aspecto de cómo hacer criptografía con las curvas elípticas, compararemos como impactan los tipos de ataques conocidos en las seguridad y eficiencia de los métodos basados en el IFP, DLP y ECDLP.

4.1. Seguridad. El primer paso en el proceso de generación de los parámetros de un esquema de ECC es determinar el nivel de seguridad que se intenta alcanzar. Varios factores podrían influenciar esta determinación, como el valor de la información, el tiempo que debe ser protegida, el tamaño de los parámetros que serán usados, el nivel de seguridad provisto por el esquema, etc.

Sin embargo, varias reglas prácticas podrían ser utilizadas. Por Ej., se sabe que (al momento en que se escribió este documento) el esquema RSA con módulo de 512 bits provee seguridad solo a corto plazo, 1024 bits de módulo es generalmente adecuada para uso general, y 2048 bits de módulo provee buena seguridad a mediano y largo plazo (con el actual nivel de conocimiento sobre RSA).

En la Tabla 2 vemos una comparación del tamaño necesario de los parámetros de varios esquemas para alcanzar un nivel comparable de seguridad. En la primera columna se presenta el tamaño de un parámetro en bits. La segunda columna lista los tamaños de claves en un esquema simétrico “ideal”, luego los tamaños de un esquema basado en ECC, y después para los esquemas RSA y DSA (utilizan el IFP y DLP respectivamente.) Cada fila representa un nivel de seguridad comparable entre los distintos métodos. La conclusión que sacamos de esta tabla es que para sistemas de clave pública, esquemas basados en ECC brindan la misma seguridad que los ya tradicionales, pero con un costo más reducido en su empleo, debido al tamaño reducido de los parámetros que utiliza (el parámetro de seguridad es el tamaño de n , el orden del punto base G , como veremos más adelante.)

4.2. Eficiencia. Cuando se habla sobre eficiencia de un criptosistema de clave pública, hay que tener en cuenta tres factores:

Nivel de Seguridad	Esquema Simétrico (tamaño de clave)	Esquema basado en ECC (tamaño de n)	DSA/RSA (tamaño del módulo)
56	56	112	512
80	80	160	1024
112	112	224	2048
128	128	256	3072
192	192	384	7680
256	256	512	15360

TABLA 2. Tamaños de clave comparables (en bits). (extraído de [11])

- Sobrecarga en cálculos: Cuanta computación se requiere para ejecutar las transformaciones de clave privada y clave pública.
- Tamaño de clave: Cuantos bits se requieren para almacenar el par de claves y algunos otros parámetros del sistema.
- Ancho de banda: Cuantos bits deben ser comunicados para transferir un mensaje encriptado o una firma.

Efectuaremos las comparaciones de eficiencia de un criptosistema ECC de 160 bits con un RSA y DSA de 1024 bits, ya que ambos proveen un nivel comparable de seguridad [2].

4.2.1. *Sobrecarga de cálculos.* En RSA, un exponente público corto puede ser empleado (aunque esto presenta algunos riesgos de seguridad) para acelerar la encriptación y la verificación de firma (por ej., suele emplearse el exponente $e = 3$ o $e = 2^{16} + 1$).⁸ En DSA y ECC, una gran proporción de la generación de una firma y transformaciones de encriptación pueden ser precomputada. También, varias bases especiales para los campos finitos \mathbb{F}_{2^m} pueden ser empleadas para ejecutar más rápidamente la aritmética modular involucrada en la operación de ECC, como por ejemplo usar una representación normal óptima o cambiar los puntos de la curva a coordenadas proyectivas. Buenas implementaciones de los sistemas muestran que con todas estas eficiencias incorporadas, ECC es un orden de magnitud más rápido que RSA o DSA.

4.2.2. *Tamaño de clave.* En la Tabla 3 se muestran los tamaños de clave pública y privada, y de parámetros del sistema. Es claro que los de ECC son más cortos que los de RSA o DSA.

4.2.3. *Ancho de banda.* Aquí consideramos solo el caso cuando mensajes cortos están siendo transformados, ya que los criptosistemas de clave pública son a menudo empleados para transmitir mensajes cortos, como claves de sesión para algoritmos de encriptación de clave simétrica. Para hacer una comparación concreta, supongamos que cada sistema esta siendo usado para firmar un mensaje de 2000 bits, o para encriptar un mensaje de 100 bits.⁹ Tablas 4 y 5 comparan las longitudes de las firmas y mensajes encriptados respectivamente (observe que en Tabla 5, se compara

⁸Note que la representación binaria de e tiene solo dos 1's, lo cual hace que el algoritmo square-and-multiply sea muy eficiente.

	Parámetros del sistema	Clave Pública	Clave Privada
RSA	n/a	1088	2048
DSA	2208	1024	160
ECC	481	161	160

TABLA 3. Tamaño de los parámetros del sistema y par de claves (en bits). (extraído de [2])

	Tamaño de firma
RSA	1024
DSA	320
ECC	320

TABLA 4. Tamaños de firma (en bits). (extraído de [2])

	Tamaño mensaje encriptado
RSA	1024
ElGamal	2048
ECC	321

TABLA 5. Tamaños de mensajes encriptados (en bits). (extraído de [2])

la encriptación con el método de ElGamal, en vez de DSA. Esta comparación tiene sentido puesto que ambos se basan en el mismo algoritmo). Se ve que ECC ofrece un ahorro de ancho de banda considerable sobre los otros tipos de sistemas de clave pública.

Por lo tanto, todos estos ahorros redundan en velocidades más altas, menor consumo de energía, y reducciones en el tamaño del código.

5. APLICACIONES DE ECC

Una implementación de ECC es beneficiosa particularmente en aplicaciones donde el ancho de banda, capacidad de procesamiento, disponibilidad de energía o almacenamiento están restringidos. Tales aplicaciones incluyen transacciones sobre dispositivos inalámbricos, computación en dispositivos handheld como PDAs o teléfonos celulares, broadcasting y smart cards. [4] describe las restricciones que presenta el uso de criptografía de clave pública en smart cards, y presenta algunas opciones de implementación.

6. HACIENDO CRIPTOGRAFÍA CON ECC

Esta sección discute la provisión de los parámetros del dominio de curvas elípticas. Describe cuales son los parámetros de dominio y cómo deberían ser generados (§6.1), cuales son las claves públicas y privadas (§6.2) y algunas primitivas (§6.3).

⁹Este valor parece estar desactualizado. En la práctica, el protocolo más utilizado sobre Internet que utiliza sistemas de clave pública es SSL (o TLS), pero este protocolo a menudo encripta más de 100 bits, a saber, unos $48 * 8 = 384$ bits [5].

6.1. Parámetros del Dominio de Curvas Elípticas. Los parámetros del dominio de las curvas elípticas son los valores básicos necesarios para definir el campo finito a usar, los valores a y b que definen la curva, etc. Aunque pueden ser generados por cualquier entidad, en Internet seguramente los generará un Certificate Authority (CA). Debe notarse que estos parámetros deben ser compartidos por las partes que quieren comunicarse, de manera que en general se trata de utilizar siempre los mismos parámetros recomendados por las organizaciones productoras de estándares (por ej. ver [12].)

En esta subsección se presentan primero los métodos de generación de los parámetros de dominio de curva para \mathbb{F}_p (es decir, \mathbb{Z}_p), y luego para \mathbb{F}_{2^m} .

6.1.1. Parámetros de dominio para curvas sobre \mathbb{F}_p . Los parámetros que definen el dominio de las curvas elípticas sobre \mathbb{F}_p son una séxtupla:

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

consistiendo de un número entero primo p que especifica el campo finito \mathbb{F}_p , dos elementos $a, b \in \mathbb{F}_p$ especificando una curva elíptica $E(\mathbb{F}_p)$ definida por la ecuación

$$E : y^2 \equiv x^3 + ax + b \pmod{p},$$

un *punto base* $G = (x_G, y_G) \in E(\mathbb{F}_p)$, un número primo n el cual es el orden de G , y un número entero h que es el cofactor $h = \#E(\mathbb{F}_p)/n$.

La función que cumplen p, a, b ya la hemos visto en el transcurso del documento, y son las constantes que definen el tamaño en bits de los puntos y la forma de la curva.

El punto base G es un punto de referencia que se proporciona para poder hacer criptografía. Es el análogo del generador α en el DLP. El número n es el orden de G , o sea n es tal que $nG = \mathcal{O}$.

Note que n es un número primo grande. Algo muy importante que no siempre se dice es que el orden de G debe ser lo suficientemente grande como para que sea impracticable buscar todos los múltiplos $G, 2G, \dots, (n-1)G$ de G . De hecho, el GDLP depende, además de otros factores, de que el orden n de G sea grande (en el DLP, el orden del elemento generador queda implícitamente definido por el tamaño de p .)

El cofactor h es un factor de $\#E(\mathbb{F}_p)$, puesto que $\#E(\mathbb{F}_p) = nh$, y se utiliza como primitiva básica para ofrecer una resistencia eficiente a ataques como los ataques de subgrupo pequeño, en el cual un adversario sustituye la clave pública de una de las partes con un punto de orden pequeño en un intento de coersionar a la otra parte a calcular un elemento de grupo que sea predecible usando una de las primitivas. No necesariamente este valor se utiliza en todos los esquemas de ECC.

El proceso para generar la séxtupla $T = (p, a, b, G, n, h)$ es el siguiente:

1. Elegir el nivel de seguridad aproximado para los parámetros de dominio de curva elíptica. Este debe ser un entero $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$ de manera que calcular logaritmos sobre la curva generada tome aproximadamente 2^t operaciones. Luego, generar los parámetros de dominio de la curva elíptica como sigue:
2. Seleccionar un número primo p tal que $\lceil \log_2 p \rceil = 2t$ si $t \neq 256$ y tal que $\lceil \log_2 p \rceil = 521$ si $t = 256$ para determinar el campo finito \mathbb{F}_p .
3. Seleccionar elementos $a, b \in \mathbb{F}_p$ para determinar la curva $E(\mathbb{F}_p)$ definida por la ecuación $E : y^2 \equiv x^3 + ax + b \pmod{p}$, un punto base $G = (x_G, y_G)$ sobre

$E(\mathbb{F}_p)$, un número primo n el cual es el orden de G , y un cofactor entero $h = \#E(\mathbb{F}_p)/n$, sujeto a las siguientes restricciones:

- (i) $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$.
- (ii) $\#E(\mathbb{F}_p) \neq p$.
- (iii) $p^B \not\equiv 1 \pmod{n}$ para todo $1 \leq B < 20$.
- (iv) $h \leq 4$.

4. retornar (p, a, b, G, n, h) .

Comentario 6.1. Dependiendo del nivel de seguridad que se intenta conseguir, el número primo p de \mathbb{Z}_p se suele elegir de manera que:

$$[\log_2 p] \in \{112, 128, 160, 192, 224, 256, 384, 521\}$$

Esto representa el tamaño en bits de p . Note la aparición del número 521, en vez de 512; esto es un requerimiento que aparece en parámetros estándares de seguridad del gobierno de U.S.

Comentario 6.2. Con respecto a las restricciones que deben cumplir los demás parámetros, (i) se pide para que se cumpla de definición que dimos de curvas elípticas sobre \mathbb{F}_p . La condición (ii) se pide para evitar las curvas anómalas, y la (iii) para evitar las curvas supersingulares (a esta condición se la conoce como *condición MOV* debido a sus descubridores: Menezes, Okamoto, Vanstone). La condición (iv) pide que $h \leq 4$; esto es para que $\#E(\mathbb{F}_p)$ sea un número “cerca” de un primo grande.

Comentario 6.3. Dado un punto G , es fácil calcular n (y h) cuando se conoce $\#E(\mathbb{F}_p)$, siempre que el orden $\#E(\mathbb{F}_p)$ de la curva tenga un primo grande. Encontrar $\#E(\mathbb{F}_p)$ involucra el cálculo del algoritmo de Schoof, o en métodos basados en Multiplicación Compleja, o en métodos basados en el Teorema de Weil (no los veremos aquí, puesto que no son necesarios para entender ECC.)

Comentario 6.4. Para las entidades involucradas en el uso de parámetros de dominio de curva elíptica, es recomendable que validen la veracidad de los mismos, ya sea autenticándolos contra alguna Tercera Parte Confiable (tal como un Certificate Authority (CA)) o recalculando los parámetros para asegurarse que encajan dentro de las condiciones.

6.1.2. *Parámetros de dominio para curvas sobre \mathbb{F}_{2^m} .* Los parámetros de dominio de una curva elíptica sobre \mathbb{F}_{2^m} son una séptupla

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

consistiendo de un número entero m que define el campo finito \mathbb{F}_{2^m} , un polinomio binario irreducible $f(x)$ de grado m especificando la representación de \mathbb{F}_{2^m} , dos elementos $a, b \in \mathbb{F}_{2^m}$ especificando la forma de la curva elíptica $E(\mathbb{F}_{2^m})$ definida por la ecuación:

$$y^2 + xy = x^3 + ax^2 + b,$$

un punto base $G = (x_G, y_G) \in E(\mathbb{F}_{2^m})$, un número primo n que es el orden de G , y un número entero h el cual es el cofactor $h = \#E(\mathbb{F}_{2^m})/n$.

El proceso para generar una séptupla $T = (m, f(x), a, b, G, n, h)$ es el siguiente:

1. Seleccionar el nivel de seguridad (en bits) apropiado para los parámetros de curva elíptica — este debe ser un número entero $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$ de manera que calcular logaritmos sobre la curva elíptica asociada tome aproximadamente 2^t operaciones.

Campo	Polinomio(s) irreducibles
$\mathbb{F}_{2^{113}}$	$f(x) = x^{113} + x^9 + 1$
$\mathbb{F}_{2^{131}}$	$f(x) = x^{131} + x^8 + x^3 + x^2 + 1$
$\mathbb{F}_{2^{163}}$	$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$
$\mathbb{F}_{2^{193}}$	$f(x) = x^{193} + x^{15} + 1$
$\mathbb{F}_{2^{233}}$	$f(x) = x^{233} + x^{74} + 1$
$\mathbb{F}_{2^{239}}$	$f(x) = x^{239} + x^{36} + 1$ ó $f(x) = x^{239} + x^{158} + 1$
$\mathbb{F}_{2^{283}}$	$f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$
$\mathbb{F}_{2^{409}}$	$f(x) = x^{409} + x^{87} + 1$
$\mathbb{F}_{2^{571}}$	$f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

TABLA 6. Representaciones de \mathbb{F}_{2^m} .

2. Sea t' el número entero más pequeño que sea $> t$ en el conjunto $\{64, 80, 96, 112, 128, 192, 256, 512\}$. Seleccionar $m \in \{113, 131, 163, 193, 233, 239, 283, 409, 571\}$ tal que $2t < m < 2t'$ para determinar el campo finito \mathbb{F}_{2^m} .
3. Seleccionar un polinomio binario irreducible $f(x)$ de grado m de la Tabla 6 para determinar la representación de \mathbb{F}_{2^m} .
4. Seleccionar elementos $a, b \in \mathbb{F}_{2^m}$ para determinar la curva elíptica $E(\mathbb{F}_{2^m})$ definida por la ecuación $E : y^2 + xy = x^3 + ax^2 + b$ en \mathbb{F}_{2^m} , un punto base $G = (x_G, y_G) \in E(\mathbb{F}_{2^m})$, un número primo n el cual es el orden de G , y un entero h , que es el cofactor $h = \#E(\mathbb{F}_{2^m})/n$, sujetos a las siguientes restricciones:
 - (i) $b \neq 0$ en \mathbb{F}_{2^m} .
 - (ii) $\#E(\mathbb{F}_{2^m}) \neq 2^m$.
 - (iii) $2^{mB} \not\equiv 1 \pmod{n}$ para todo $1 \leq B < 20$.
 - (iv) $h \leq 4$.
5. retornar $(m, f(x), a, b, G, n, h)$.

Comentario 6.5. Veamos cómo escogemos el número m de \mathbb{F}_{2^m} para fines criptográficos. m debería ser tal que:

$$m \in \{113, 131, 163, 193, 233, 239, 283, 409, 571\}$$

y la suma y multiplicación en \mathbb{F}_{2^m} debería ser ejecutada usando uno de los polinomios de grado m en la Tabla 6 (extraída de [11]). Esta restricción está diseñada para facilitar la interoperabilidad entre implementaciones, y para que éstas sean eficientes con requerimientos de seguridad comunes.

Además, se trata de evitar siempre que m sea un número compuesto, debido a preocupaciones expresadas por algunos expertos respecto a la seguridad de las curvas elípticas definidas sobre \mathbb{F}_{2^m} con m compuesto.

Comentario 6.6. La regla para elegir al polinomio irreducible $f(x)$ es elegir un trinomio (polinomio con tres términos) de la forma $f(x) = x^m + x^k + 1$, con $m > k \geq 1$, y k tan pequeño como sea posible. Si un trinomio irreducible no existe, se opta por un pentanomio (polinomio con cinco términos) de la forma $f(x) = x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$, $m > k_3 > k_2 > k_1 \geq 1$, con (1) k_3 tan pequeño como sea posible, (2) k_2 tan pequeño como sea posible dado k_3 , y (3) k_1 tan pequeño como sea posible dado k_3 y k_2 . Estos polinomios habilitan el cálculo eficiente de las

operaciones del campo, ya que son los polinomios irreducibles sobre \mathbb{F}_{2^m} de menor cantidad de términos posibles (ver Teorema A.39).

Para \mathbb{F}_{2^m} valen las mismas restricciones que se describen en Comentario 6.2, 6.3 y 6.4, reemplazando \mathbb{F}_p por \mathbb{F}_{2^m} .

6.2. Pares de Claves para Curvas Elípticas. Dado los parámetros de dominio de curva elíptica $T = (p, a, b, G, n, h)$ o $T = (m, f(x), a, b, G, n, h)$, un par de claves de curva elíptica (d, Q) asociado con T consiste de una clave secreta de curva elíptica d la cual es un entero en el intervalo $[1, n - 1]$, y una clave pública de curva elíptica $Q = (x_Q, y_Q)$ la cual es el punto $Q = dG$.

Para generar un par de claves, una entidad procede como sigue, dados los parámetros (válidos) de dominio $T = (p, a, b, G, n, h)$ o $T = (m, f(x), a, b, G, n, h)$ según corresponda:

1. Aleatoriamente o pseudo aleatoriamente seleccionar un número entero d en el intervalo $[1, n - 1]$.
2. Calcular $Q = dG$.
3. retornar (d, Q) .

6.2.1. Validación de Claves Públicas. Para comprobar que la clave pública que una entidad quiere utilizar es válida, puede proceder a ejecutar una serie de tests sobre la clave pública (usando los parámetros de dominio) para validar que la clave pública efectivamente fue derivada correctamente. Usualmente sobre Internet la clave pública estará embebida dentro de un certificado PKI (o X.509) emitido por un CA confiable, autenticado apropiadamente, con lo cual, la entidad solo tiene que comprobar que la clave en el certificado sea autentica ejecutando la verificación de una firma.

No obstante, para una entidad a veces es suficiente que una clave pública de curva elíptica sea *parcialmente* válida, en contraste a “completamente” válida, como se describió en el párrafo anterior.

Definición 6.7. Una clave pública de curva elíptica Q se dice que es *parcialmente válida* si Q es un punto sobre la curva elíptica asociada pero no necesariamente es cierto que $Q = dG$ para algún entero d .

Por ejemplo, el esquema de acuerdo de claves MQV y el esquema Diffie-Hellman¹⁰ usando la primitiva Diffie-Hellman de cofactor son esquemas diseñados para proveer seguridad aún cuando las entidades solo chequean que las claves públicas involucradas son parcialmente válidas.

El atractivo de la validación parcial de claves públicas de curva elíptica radica en el hecho que es computacionalmente más eficiente que la validación completa. En efecto, solo hay que chequear (1) que $Q = (x_Q, y_Q) \neq \mathcal{O}$, (2) que $x_Q, y_Q \in [1, p - 1]$ (x_Q, y_Q sean polinomios de grado a lo sumo $m - 1$ para \mathbb{F}_{2^m}) y (3) que $Q \in E(\mathbb{F}_p)$ (o $Q \in E(\mathbb{F}_{2^m})$, según corresponda.)

¹⁰Esquemas de acuerdo de claves y de encriptación no son tratados en este texto debido a que, al momento en que se escribió este documento, los esquemas basados en ECC de más interés son los de firma digital.

6.3. Primitivas Diffie-Hellman de Curvas Elípticas. Las primitivas Diffie-Hellman de curvas elípticas son la base para operación del ECAES (Elliptic Curve Augmented Encryption Scheme), y el Esquema Diffie-Hellman de curvas elípticas.

Se especifican dos primitivas: la primitiva Diffie-Hellman de curva elíptica y la primitiva Diffie-Hellman de cofactor de curva elíptica. La idea básica de ambos esquemas es la misma — generar un valor secreto compartido a partir de una clave privada adueñada por una entidad U y una clave pública adueñada por otra entidad V de manera que si ambas entidades ejecutan la primitiva simultáneamente con las claves correspondientes, podrán recuperar el mismo valor secreto compartido.

Sin embargo, las dos primitivas son sutilmente diferentes: la primitiva Diffie-Hellman de curva elíptica es el análogo directo del bien conocido protocolo de acuerdo de claves Diffie-Hellman (§3), mientras que la primitiva Diffie-Hellman de cofactor de curva elíptica incorpora el cofactor h en el calculo del valor secreto compartido para proveer una resistencia eficiente contra ataques de subgrupo pequeño.

A partir de estas definiciones, podemos ver que la construcción de un protocolo de acuerdo de claves Diffie-Hellman basado en ECC es directa:

6.3.1. Primitiva Diffie-Hellman de Curva Elíptica. Para que una entidad U pueda calcular un valor secreto compartido con la entidad V usando la primitiva Diffie-Hellman de curva elíptica, se debe ejecutar el proceso siguiente, que toma como entrada parámetros validados de curva elíptica $T = (p, a, b, G, n, h)$ o $T = (m, f(x), a, b, G, n, h)$, la clave privada d_U asociada con T perteneciente a U , y una clave pública Q_V (al menos parcialmente validada) asociada con T perteneciente a V .

1. Calcular el punto $P = (x_P, y_P) = d_U Q_V$ de la curva elíptica.
2. Chequear que $P \neq \mathcal{O}$. Si $P = \mathcal{O}$, retornar “invalido” y parar.
3. retornar $z = x_P$, el elemento del campo finito que es secreto y compartido.

6.3.2. Primitiva Diffie-Hellman de cofactor de Curva Elíptica. El calculo del valor secreto para esta primitiva es esencialmente el mismo que para la primitiva Diffie-Hellman descrita en §6.3.1. La única diferencia radica en el paso de la multiplicación escalar (paso 1). En vez de solo multiplicar por la clave privada d_U , la entidad U ahora debe calcular el punto $P = (x_P, y_P) = h d_U Q_V$.

7. ESQUEMAS DE FIRMA DIGITAL

Dada la aceptación de los sistemas basados en curvas elípticas, recientemente el gobierno de U.S., bajo recomendaciones del NIST (National Institute of Standards and Technology) estandarizó la segunda versión de su estándar de DSS (Digital Signature Standard) [10], que describe ahora tanto el algoritmo de firma digital DSA como también el ECDSA.

El algoritmo DSA es una variante del esquema de firmas de ElGamal. Este explota pequeños subgrupos en \mathbb{Z}_p^* de manera de reducir el tamaño de las firmas producidas. A continuación se describen los procedimientos de generación de claves, generación de firma y verificación para el algoritmo DSA [7]:

Generación del claves DSA. Cada entidad A hace lo siguiente:

1. Seleccionar un número primo q tal que $2^{159} < q < 2^{160}$. (q es un primo de 160 bits.)
2. Seleccionar un número primo p de 1024 bits con la propiedad que $q|p-1$. (El estándar DSS obliga a que p sea un primo tal que $2^{511+64t} < p < 2^{512+64t}$ donde $0 \leq t \leq 8$. Si $t = 8$ entonces p es un primo de 1024 bits.)
3. Seleccionar un elemento $h \in \mathbb{Z}_p^*$ y calcular $g = h^{p-1/q} \bmod p$; repetir hasta que $g \neq 1$. (g es un generador del único subgrupo cíclico de orden q en \mathbb{Z}_p^* .)

DSA	ECDSA
q	n
g	P
x	d
y	Q

TABLA 7. Correspondencias entre las notaciones de DSA y ECDSA

4. Seleccionar aleatoriamente un entero x en el intervalo $[1, q - 1]$.
5. Calcular $y = g^x \bmod p$.
6. La clave pública de A es (p, q, g, y) . La clave privada de A es x .

Generación de una firma en DSA. Para firmar un mensaje m , A hace lo siguiente:

1. Seleccionar aleatoriamente un entero k en el intervalo $[1, q - 1]$.
2. Calcular $r = (g^k \bmod p) \bmod q$.
3. Calcular $k^{-1} \bmod q$.
4. Calcular $s = k^{-1} \{h(m) + xr\} \bmod q$, donde h es el Secure Hash Algorithm (SHA-1).
5. Si $s = 0$ entonces volver al paso 1. (Si $s = 0$, entonces $s^{-1} \bmod q$ no existe; s^{-1} es requerido en el paso 2 de la verificación de la firma.)
6. El firma para el mensaje m es el par de enteros (r, s) .

Verificación de una firma en DSA. Para verificar la firma (r, s) de A sobre el mensaje m , B debería hacer lo siguiente:

1. Obtener una copia autentica de la clave pública (p, q, g, y) de A .
2. Verificar que r y s sean enteros en el intervalo $[1, q - 1]$.
3. Calcular $w = s^{-1} \bmod q$ y $h(m)$. (h es SHA-1.)
4. Calcular $u_1 = h(m)w \bmod q$ y $u_2 = rw \bmod q$.
5. Calcular $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$.
6. Aceptar la firma si y solo si $v = r$.

Ya que r y s son ambos enteros menores que q , las firmas DSA son de 320 bits de longitud. La seguridad de DSA descansa sobre dos problemas de logaritmos discretos distintos, pero relacionados. Uno es el DLP donde el algoritmo Number Field Sieve (NFS) es aplicable (este algoritmo tiene un tiempo sub-exponencial). Si p es un primo de 1024 bits, entonces DSA no es vulnerable a un ataque con el NFS. El segundo problema de logaritmo discreto trabaja a la base g : Dados p, q, g e y , encontrar x tal que $y \equiv g^x \pmod{p}$. Para p grande (por ej. 1024 bits), el mejor algoritmo conocido para este problema es el Pollard- ρ , y toma alrededor de $\sqrt{\pi q/2}$ pasos. Si $q \approx 2^{160}$ entonces $\sqrt{\pi q/2}$ representa un monto de computación inviable; luego el DSA no es vulnerable a este ataque.

7.1. Elliptic Curve DSA (ECDSA). ECDSA es el análogo sobre curvas elípticas al DSA. Esto es, en vez de trabajar sobre el subgrupo de orden q en \mathbb{Z}_p^* , se trabaja en un grupo de curva elíptica $E(\mathbb{Z}_p)$. Como dijimos anteriormente, el ECDSA fue estandarizado por el NIST, pero también están en este proceso los comités de estándares ANSI X9F1 e IEEE P1363. La Tabla 7 muestra la correspondencia entre algunas notaciones matemáticas usadas en DSA y ECDSA.

En concreto, el ECDSA es un esquema de firma con apéndice basado en ECC. Esta diseñado para ser existencialmente infalsificable, aún ante la presencia de un adversario capaz de lanzar ataques de mensajes elegidos por él (es decir, un chosen-message attack).

Definición 7.1. Un esquema de firma es *existencialmente infalsificable* si es inviable para un adversario falsificar una firma sobre algún mensaje que no ha sido previamente firmado por el usuario legítimo del esquema.

Existe un interés especial en este algoritmo, puesto que su estructura es muy similar al DSA, el cual ha sido muy estudiado estos últimos años. Menos interés ha habido en estandarizar un análogo sobre curvas elípticas del esquema de Schnorr, o el de Nyberg-Rueppel, y es por eso que aquí los omitimos.

A continuación se dan los procedimientos para generar los pares claves, generar una firma y la verificación de una firma usando ECDSA:

Generación de claves ECDSA. Cada entidad A hace lo siguiente:

1. Seleccionar una curva elíptica E definida sobre \mathbb{Z}_p^* . El número de puntos en $E(\mathbb{Z}_p)$ debería ser divisible por un número primo grande n .
2. Seleccionar un punto $P \in E(\mathbb{Z}_p)$ de orden n .
3. Seleccionar un entero d estadísticamente único e impredecible en el intervalo $[1, n-1]$.
4. Calcular $Q = dP$.
5. La clave pública de A es (E, P, n, Q) ; la clave privada de A es d .

Generación de una firma ECDSA. Para firmar un mensaje m , A hace lo siguiente:

1. Seleccionar un entero k estadísticamente único e impredecible en el intervalo $[1, n-1]$.
2. Calcular $kP = (x_1, y_1)$ y $r = x_1 \bmod n$. (Aquí x_1 es tratado como a un número entero, por ejemplo por conversión de su representación binaria.)
Si $r = 0$, entonces volver al paso 1. (Esta es una condición de seguridad: si $r = 0$, entonces la ecuación de firma $s = k^{-1}\{h(m) + dr\} \bmod n$ no involucra a la clave privada d !)
3. Calcular $k^{-1} \bmod n$.
4. Calcular $s = k^{-1}\{h(m) + dr\} \bmod n$, donde h es el Secure Hash Algorithm (SHA-1).
5. Si $s = 0$, entonces volver al paso 1. (Si $s = 0$ entonces $s^{-1} \bmod n$ no existe; s^{-1} es requerido en el paso 2 de la verificación de firma.)
6. La firma para el mensaje m es el par de enteros (r, s) .

Verificación de firma ECDSA. Para verificar la firma (r, s) de A sobre el mensaje m , B debería hacer:

1. Obtener una copia autentica de la clave pública (E, P, n, Q) de A .
2. Verificar que r y s sean enteros en el intervalo $[1, n-1]$.
Si alguno no está en el intervalo $[1, n-1]$ retornar “invalida” y parar.
3. Calcular $w = s^{-1} \bmod n$ y $h(m)$.
4. Calcular $u_1 = h(m)w \bmod n$ y $u_2 = rw \bmod n$.
5. Calcular $R = u_1P + u_2Q = (x_0, y_0)$ y $v = x_0 \bmod n$.
Si $R = \mathcal{O}$, retornar “invalida” y parar.
6. Aceptar la firma si y solo si $v = r$. Si $v \neq r$ retornar “invalida”.

El estándar ANSI X9.62 obliga a que $n > 2^{160}$. Para obtener un nivel de seguridad similar al de DSA (con q de 160 bits y p de 1024 bits), el parámetro n debería tener alrededor de 160 bits. En este caso, las firmas producidas por DSA y ECDSA tienen la misma longitud de 320 bits.

En vez de que cada entidad genere su propia curva elíptica, las entidades podrían elegir usar la misma curva E sobre \mathbb{Z}_p , y punto P de orden n ; estas cantidades luego son llamadas *parámetros del sistema*. (En DSA, los parámetros de sistema análogos serían p , q y g .) En este caso, la clave pública de una entidad consiste solamente del punto Q . Esto resulta en claves públicas de menor tamaño.

ECDSA tiene un número de similitudes con DSA, de las cuales las más importantes son [7]:

1. Ambos algoritmos están basados en el esquema de firmas de ElGamal y usan la misma ecuación de firma: $s = k^{-1}\{h(m) + dr\} \bmod n$.
2. En ambos algoritmos, los valores que son difíciles de generar son los parámetros del sistema (p , q y g para DSA; p , E , P y n para ECDSA) los cuales son públicos; su generación puede ser auditada y chequeada por validez en forma independiente. Esto ayuda a mostrar que no fueron producidos para reunir ningún criterio secreto (por Ej. proveer a los parámetros con alguna propiedad “trapdoor”). Generar una clave privada, dado un conjunto de parámetros de sistema es relativamente simple y generar la clave pública asociada es directo.

Contraste esto con el algoritmo RSA, donde los valores que son difíciles de generar (los primos p y q) deben ser mantenidos secretos.

3. En su versión actual, tanto DSA como ECDSA usan SHA-1 como única opción para la función de hash. Sin embargo, está planeado el soporte para SHA-2 a medida que SHA-2 pase por los procesos de estandarización de ANSI y NIST.

No obstante, también existen algunas diferencias entre DSA y ECDSA, a saber:

1. La clave privada d y el valor efímero k generado por cada firma en ECDSA están definidos para que sean *estadísticamente únicos e impredecibles* en vez de sencillamente *aleatorio* como en DSA. Esta es una clarificación importante y es una mejor definición de los requerimientos de seguridad. Si k puede ser determinado o si k se repite entonces un adversario puede recuperar d , la clave privada. Por supuesto, el uso de un valor aleatorio está explícitamente siendo permitido; no obstante arquitecturalmente es preferible definir los requerimientos en vez de decir una forma en particular de reunir los requerimientos. Por Ej., dar los requerimientos permite a una implementación de alta seguridad filtrar los valores k para asegurar que no se repitan. Esta posibilidad no está permitida si k es requerido que sea aleatorio. También, definir los requerimientos da más guía a los implementadores y usuarios respecto a qué constituye una preocupación de seguridad.
2. En ECDSA, un método llamado *compresión de punto* permite a un punto sobre la curva elíptica (por ej., una clave pública Q) ser representado en forma compacta por un elemento del campo y un bit adicional, en vez de dos elementos del campo.¹¹ Luego, por Ej., si $p \approx 2^{160}$ (de manera que los elementos en \mathbb{Z}_p son cadenas de 160 bits), las claves públicas pueden ser representadas como cadenas de 161 bits. Esto puede llevar a una reducción sustancial en el tamaño de un certificado de clave pública en el orden de un 25% cuando es comparado con otros algoritmos asimétricos.
3. En el DSA, hay un chequeo de límites opcional durante la generación de la firma sobre los componentes de la firma digital (r y s). Esto resulta en una probabilidad extremadamente baja que un sistema conformista el cual no haga el chequeo de límites genere una firma que no se pudiera verificar. Sin embargo, esto significa que código escrito genéricamente (es decir, código que no tome en cuenta conocimiento de la implementación subyacente de DSA) debe verificar la firma que generó el mismo, si es que debe haber un 100% de garantía que la firma se pueda verificar. El chequeo de límites análogo ha sido hecho obligatorio en ECDSA; el 100% de las firmas digitales que son generadas se pueden verificar. Note, no obstante, que aún podría haber situaciones donde es una sabia decisión verificar explícitamente una firma recién generada, tal como cuando se sabe que la firma será distribuida ampliamente. La verificación explícita ayudará a detectar implementaciones dudosas y errores de aplicación.
4. La prueba de primalidad en DSA es probabilística. Como la tecnología de curva elíptica tiene la capacidad para una prueba de primalidad determinística, tal prueba fue incluida como una opción en el estándar X9.62, mientras que

¹¹Esto se puede hacer debido a que toda curva elíptica es simétrica con respecto al eje x , entonces solo se representa la coordenada x del punto; la coordenada y correspondiente es calculada en vez de almacenada. El bit extra es para indicar si el punto correcto está del lado positivo del eje y , o del lado negativo del eje y .

también retiene la prueba probabilística. Esto le permite a una aplicación con altos requerimientos de seguridad (por Ej., un Certificate Authority) verificar que los valores primos presentados son en realidad primos, aunque a un costo computacional adicional.

5. ECDSA especifica los pasos de un procedimiento para verificar los parámetros del sistema generados, mientras que DSA no. En un escenario común, uno recibiría los parámetros del sistema DSA de una Tercer Parte Confiable (TTP) donde no hay necesidad de validar los parámetros del sistema. Sin embargo, otro escenario útil es donde una entidad ha generado sus propios parámetros personales del sistema. En el último escenario uno podría querer verificar que los parámetros del sistema provisto realmente reúnan todos los requerimientos de seguridad antes de usarlos. Tal procedimiento de validación de parámetros del sistema podría ser agregado a DSA.
6. Se han demostrado debilidades teóricas en DSA basadas en el hecho que la función de hash realmente usada en DSA es $\text{SHA-1} \bmod q$, no simplemente SHA-1, donde q es el número primo de 160 bits. Esta debilidad permite la falsificación de un mensaje si el adversario puede seleccionar los parámetros del sistema. Esta debilidad no existe en DSA si los parámetros del sistema son seleccionados como se especifica en el estándar X9.30. El análogo de q en ECDSA es n , el orden del punto base P . Si $n > 2^{160}$, entonces no existen colisiones y tal ataque no es posible.
7. DSA especifica que la integridad del dato firmado es dependiente de la prevención de la revelación no autorizada, modificación, sustitución, inserción y borrado de la clave privada x o el valor k (particular) de una firma. Sin embargo, mientras quizás sea una implicación, el uso no autorizado no es explícitamente prohibido. En ECDSA, el requerimiento para uso autorizado de una clave privada ha sido hecho explícito.

8. CONCLUSIONES

Como con todos los criptosistemas, y especialmente con criptosistemas de clave pública, toma años de evaluación pública antes de que un razonable nivel de confianza en un nuevo sistema sea establecido. ECCs parecen haber alcanzado ese nivel ahora, y en los últimos años, las primeras implementaciones comerciales fueron apareciendo, como bibliotecas de funciones, pero también en aplicaciones reales, como seguridad en e-mail, smart cards, etc.

Un factor que aún queda grandemente incierto es su seguridad: como con todos los criptosistemas de clave pública usados en la práctica, su seguridad no ha sido probada, sino que esta basada en la incapacidad de encontrar ataques. Si existen ataques sobre alguno de esos sistemas, podrían ser descubiertos tarde o temprano. En tal caso, se debería cambiar a alguna otra alternativa, como los ya tradicionales criptosistemas usados en la práctica basados en el IFP y el DLP.

No obstante, los ECC pueden ser implementados eficientemente, y tienen un número de ventajas que pueden hacerlo la mejor elección en un rango de aplicaciones, como aquellas en las que los recursos disponibles (memoria, procesamiento, energía, ancho de banda, etc.) son reducidos.

Además, con un número de estándares estando en preparación, la interoperabilidad entre los diferentes productos será mucho más fácil de obtener, como en el caso

del algoritmo de firma digital ECDSA, el cual ya tiene un estándar establecido en U.S.

APÉNDICE A. ALGUNOS CONCEPTOS DE ÁLGEBRA ABSTRACTA

A continuación se darán algunas definiciones y resultados que le ayudarán a entender las matemáticas detrás de las curvas elípticas. Todas ellas fueron extraídas de [9].

Definición A.1. Una *operación binaria* $*$ sobre un conjunto S es un mapeo de $S \times S$ a S . Esto es, $*$ es una regla que asigna a cada par ordenado de elementos de S un elemento de S .

Ejemplo A.2. La operación $+$ en \mathbb{Z} . Dado $a, b \in \mathbb{Z}$, $(a + b) \in \mathbb{Z}$.

Definición A.3. Un *grupo* $(G, *)$ consiste de un conjunto G con una operación binaria $*$ sobre G satisfaciendo los siguientes tres axiomas.

- (i) La operación del grupo es *asociativa*. Esto es, $a * (b * c) = (a * b) * c$ para todo $a, b, c \in G$.
- (ii) Existe un elemento $1 \in G$, llamado el *elemento identidad*, tal que $a * 1 = 1 * a = a$ para todo $a \in G$.
- (iii) Para cada $a \in G$ existe un elemento $a^{-1} \in G$, llamado el *inverso* de a , tal que $a * a^{-1} = a^{-1} * a = 1$.

Un grupo G es *abeliano* (o *conmutativo*) si además,

- (iv) Para todo $a, b \in G$, $a * b = b * a$.

Ejemplo A.4. $(\mathbb{Z}_n, +)$ es un grupo, puesto que la suma entre enteros es conmutativa, existe el número 0 que es la identidad, y el inverso de a es $-a$. También $(\mathbb{Z}_p^*, *)$ con p primo es un grupo con la operación de multiplicación modulo p , donde la identidad es el número 1, y el inverso de a siempre existe dado que p es primo.

Definición A.5. Un subconjunto no vacío H de un grupo G es un *subgrupo* de G si H es en si mismo un grupo con respecto a la operación de G . Si H es un subgrupo de G y $H \neq G$, entonces H es llamado un subgrupo *propio* de G .

Ejemplo A.6. Si consideramos el grupo \mathbb{Z}_{19}^* , el conjunto $H = \{1, 7, 11\}$ es un subgrupo de \mathbb{Z}_{19}^* , ya que todas las combinaciones de cálculos mod 19 con elementos de H da resultados en H : $1*1 = 1$; $1*7 = 7$; $1*11 = 11$; $7*7 = 11$; $7*11 = 1$; $11*11 = 7$.

Definición A.7. Un grupo $(G, *)$ es *cíclico* si existe un elemento $\alpha \in G$ tal que para cada $b \in G$ existe un número entero i con $b = \alpha^i = \overbrace{\alpha * \alpha * \dots * \alpha}^{\text{iveces}}$. Tal elemento α es llamado un *generador* de $(G, *)$.

Ejemplo A.8. El conjunto \mathbb{Z}_p^* con p primo es un grupo cíclico ya que por el Teorema de Fermat, $a^{p-1} \equiv 1 \pmod{p}$, multiplicando por a cada miembro queda $a^p \equiv a \pmod{p}$.

De aquí en adelante, omitiremos la operación del grupo, si ésta es irrelevante o sobreentendida, de acuerdo al contexto en que se encuentre.

Teorema A.9. Si G es un grupo y $a \in G$, entonces el conjunto de todas las potencias de a forma un subgrupo cíclico de G , llamado el subgrupo generado por a , y denotado por $\langle a \rangle$.

Ejemplo A.10. El conjunto $H = \{1, 7, 11\}$ es un subgrupo cíclico de \mathbb{Z}_{19}^* . Además $H = \langle 7 \rangle = \langle 11 \rangle$.

Definición A.11. Sea G un grupo y $a \in G$. El *orden* de a es definido como el menor entero positivo t tal que $a^t = 1$, si es que tal entero existe. Si tal t no existe, entonces el orden de a es definido como ∞ .

Ejemplo A.12. En el conjunto \mathbb{Z}_{16}^* , el orden del número 3 es $t = 4$, ya que: $3^1 \bmod 16 = 3$, $3^2 \bmod 16 = 9$, $3^3 \bmod 16 = 11$, $3^4 \bmod 16 = 1$.

Definición A.13. Un grupo G es *finito* si $|G|$ es finito. El *orden* de G es el número de elementos de G , si éste es finito.

Ejemplo A.14. \mathbb{Z}_n tiene orden n , mientras que \mathbb{Z}_p^* tiene orden $p - 1$ si p es primo.

Teorema A.15. Sea G un grupo, sea $a \in G$ un elemento de orden finito t . Entonces $|\langle a \rangle| = t$ (el orden de a es la cantidad de elementos de $\langle a \rangle$.)

Ejemplo A.16. En \mathbb{Z}_{19}^* , el orden del elemento 7 es $|\langle 7 \rangle| = 3$.

Teorema A.17 (Lagrange). Si G es un grupo finito y H es un subgrupo de G , entonces $|H|$ divide a $|G|$. Por lo tanto, si $a \in G$, el orden de a divide a $|G|$.

Ejemplo A.18. \mathbb{Z}_{19}^* es un grupo de orden $18 = 2 \cdot 3^2$, luego si $a \in \mathbb{Z}_{19}^*$, entonces $|\langle a \rangle| \in \{1, 2, 3, 6, 9, 18\}$.

Definición A.19. Un *anillo* $(R, +, \times)$ consiste de un conjunto R con dos operaciones binarias denotadas arbitrariamente $+$ (adición) y \times (multiplicación) sobre R , satisfaciendo los siguientes axiomas.

- (i) $(R, +)$ es un grupo conmutativo con identidad denotada 0.
- (ii) La operación \times es asociativa. Esto es, $a \times (b \times c) = (a \times b) \times c$ para todo $a, b, c \in R$.
- (iii) Existe una identidad multiplicativa denotada 1, con $1 \neq 0$, tal que $1 \times a = a \times 1 = a$ para todo $a \in R$.
- (iv) La operación \times es *distributiva* sobre $+$. Esto es, $a \times (b + c) = (a \times b) + (a \times c)$ y $(b + c) \times a = (b \times a) + (c \times a)$ para todo $a, b, c \in R$.

El anillo es un *anillo conmutativo* si $a \times b = b \times a$ para todo $a, b \in R$.

Ejemplo A.20. \mathbb{Z}_n con la suma y multiplicación modulo n es un anillo conmutativo.

Definición A.21. Un elemento a de un anillo R es llamado una *unidad* o un *elemento invertible* si existe un elemento $b \in R$ tal que $a \times b = 1$.

Ejemplo A.22. En \mathbb{Z}_n , existe el inverso multiplicativo de $a \neq 0$ si $\gcd(a, n) = 1$.

Definición A.23. Un *campo* es un anillo conmutativo en el cual todos los elementos no nulos tienen inversos multiplicativos.

Ejemplo A.24. Si p es primo, \mathbb{Z}_p es un campo.

Definición A.25. Un subconjunto F de un campo E es un *subcampo* de E si F es en sí mismo un campo con respecto a las operaciones de E . Si este es el caso, E se dice que es un *campo extensión* de F .

Definición A.26. Un *campo finito* es un campo F el cual contiene un número finito de elementos. El *orden* de F es el número de elementos de F .

Ejemplo A.27. Si p es primo, \mathbb{Z}_p es un campo finito de orden p .

Teorema A.28. (*existencia y unicidad de los campos finitos*)

- (i) Si F es un campo finito, entonces F contiene p^m elementos para algún número primo p y entero $m \geq 1$.
- (ii) Para todo orden de potencia prima p^m , existe un único campo finito (hasta el isomorfismo) de orden p^m . Este campo es denotado por \mathbb{F}_{p^m} , o a veces por $GF(p^m)$.¹²

En otras palabras, dos campos del mismo orden son *isomorfos*, es decir, son estructuralmente el mismo, aunque la representación de sus elementos del campo pudieran ser diferentes. Note que si p es un número primo entonces \mathbb{Z}_p es un campo, y por lo tanto todo campo de orden p es isomorfo a \mathbb{Z}_p . A menos que se indique explícitamente, el campo finito \mathbb{F}_p será identificado con \mathbb{Z}_p .

Teorema A.29 (Subcampos de un campo finito). Sea \mathbb{F}_q un campo finito de orden $q = p^m$. Entonces todo subcampo de \mathbb{F}_q tiene orden p^n , para algún n que es un divisor positivo de m . Y viceversa, si n es un divisor positivo de m , entonces existe exactamente un subcampo de \mathbb{F}_q de orden p^n ; un elemento $a \in \mathbb{F}_q$ está en el subcampo \mathbb{F}_{p^n} si y solo si $a^{p^n} = a$.

Ejemplo A.30. Ya que $1|m$, donde p es un número primo y $m \geq 1$, entonces \mathbb{F}_{p^m} contiene una copia de \mathbb{Z}_p como un subcampo. Por lo tanto \mathbb{F}_{p^m} puede ser visto como un campo extensión de \mathbb{Z}_p de grado m .

Definición A.31. Los elementos no nulos de \mathbb{F}_q forman un grupo bajo la operación de multiplicación llamado el *grupo multiplicativo* de \mathbb{F}_q , denotado por \mathbb{F}_q^* .

Ejemplo A.32. Si p es primo, el grupo multiplicativo de \mathbb{Z}_p es $\mathbb{Z}_p^* = \mathbb{Z}_p - \{0\}$.

Teorema A.33. \mathbb{F}_q^* es un grupo cíclico de orden $q - 1$. Luego, $a^q = a$ para todo $a \in \mathbb{F}_q^*$

Ejemplo A.34. En \mathbb{Z}_q^* con $q = p^m$, para $m = 1$ tenemos el Teorema de Fermat.

Definición A.35. Si R es un anillo conmutativo, el *anillo polinomial* $R[x]$ es el anillo formado por el conjunto de todos los polinomios en la indeterminada x teniendo coeficientes en R . Las dos operaciones son la suma y multiplicación estándar de polinomios, con coeficientes aritméticos que operan en el anillo R .

Ejemplo A.36. $f(x) = x^3 + x + 1$ y $g(x) = x^2 + x$ son elementos del anillo polinomial $\mathbb{Z}_2[x]$, donde los coeficientes pueden ser 0 o 1, y las operaciones son modulo 2. $f(x) + g(x) = x^3 + x^2 + 1$ y $f(x)g(x) = x^5 + x^4 + x^3 + x$.

Sea \mathbb{Z}_p el campo finito de orden p , con p primo. La Teoría de Números se puede llevar al anillo polinomial $\mathbb{Z}_p[x]$, y más generalmente al anillo polinomial $F[x]$, donde F es cualquier campo. En otras palabras, los polinomios se pueden factorizar, dividir, calcular el gcd, calcular el algoritmo de Euclides, definir congruencias, etc. de manera similar a los elementos de \mathbb{Z}_p .

Definición A.37. Sea $f(x) \in F[x]$ un polinomio de grado al menos 1. Entonces $f(x)$ se dice que es *irreducible sobre F* si este no puede ser escrito como el producto de dos polinomios en $F[x]$, cada uno de grado positivo.

¹²La F viene de la palabra en inglés *field* (campo). GF viene de *Galois Field*.

Ejemplo A.38. $f(x) = x^2 + x + 1$ es irreducible sobre \mathbb{Z}_2 .

Teorema A.39. *Un polinomio $f(x)$ irreducible sobre \mathbb{F}_{2^m} tiene un número impar de términos.*

Demostración. Esto es así puesto que si tuviera un número par de términos, entonces debe ocurrir alguna de dos posibilidades:

1. Si $a_0 = 0$, entonces $f(x) = x^{k_1} + x^{k_2} + \cdots + x^{k_{2t}}$. Luego $f(0) = 0 + 0 + \cdots + 0 = 0$, por lo tanto $x|f(x)$.
2. Si $a_0 = 1$, entonces $f(x) = x^{k_1} + x^{k_2} + \cdots + x^{k_{2t-1}} + 1$. Luego $f(1) = 1 + 1 + \cdots + 1 + 1 = 0$ puesto que hay un número par de 1's, por lo tanto $(x+1)|f(x)$.

Entonces debe ser que un polinomio irreducible sobre \mathbb{F}_{2^m} tiene un número impar de términos. \square

Como dijimos arriba, en $F[x]$ se pueden definir congruencias modulo un polinomio $f(x)$. Al igual que en \mathbb{Z}_n , esta congruencia define una relación de equivalencia en $F[x]$, y así $F[x]$ queda particionado en clases de equivalencia.

Definición A.40. $F[x]/(f(x))$ denota el conjunto de (clases de equivalencia de) los polinomios en $F[x]$ de grado menor que $n = \deg f(x)$. La suma y multiplicación se ejecutan modulo $f(x)$.

Teorema A.41. $F[x]/(f(x))$ es un anillo conmutativo.

Teorema A.42. Si $f(x)$ es irreducible sobre F , entonces $F[x]/(f(x))$ es un campo.

Ejemplo A.43. $f(x) = x^2 + x + 1$ es irreducible sobre \mathbb{Z}_2 . Luego $\mathbb{Z}_2[x]/(f(x))$ es un campo, donde las operaciones son modulo $f(x)$.

Una representación comúnmente usada para los elementos de un campo finito \mathbb{F}_q , donde $q = p^m$ y p es un número primo, es una *representación de base polinomial*. Si $m = 1$, entonces \mathbb{F}_q es simplemente \mathbb{Z}_p y la aritmética es ejecutada modulo p .

Teorema A.44. Para cada $m \geq 1$, existe un polinomio monico irreducible de grado m sobre \mathbb{Z}_p . Por lo tanto, todo campo finito tiene una representación de base polinomial.

Entonces, los elementos de un campo finito \mathbb{F}_{p^m} serán representados por polinomios en $\mathbb{Z}_p[x]$ de grado $< m$, y con coeficientes en \mathbb{Z}_p .

Ejemplo A.45. El campo finito \mathbb{F}_{2^4} se puede representar como el conjunto de polinomios sobre \mathbb{F}_2 de grado < 4 . Esto es,

$$\mathbb{F}_{2^4} = \{a_3x^3 + a_2x^2 + a_1x + a_0 \mid a_i \in \{0, 1\}\}.$$

Por conveniencia, el polinomio $a_3x^3 + a_2x^2 + a_1x + a_0$ es representado por el vector $(a_3a_2a_1a_0)$ de longitud 4, y así

$$\mathbb{F}_{2^4} = \{(a_3a_2a_1a_0) \mid a_i \in \{0, 1\}\}.$$

Lo cual es muy adecuado para representación en una computadora. También el polinomio $f(x) = x^4 + x + 1$ es irreducible sobre \mathbb{Z}_2 , entonces las operaciones se hacen modulo $f(x)$.

A veces es conveniente utilizar un polinomio primitivo para definir un campo finito:

Definición A.46. Un polinomio irreducible $f(x) \in \mathbb{Z}_p[x]$ de grado m es llamado un *polinomio primitivo* si x es un generador de $\mathbb{F}_{p^m}^*$, el grupo multiplicativo de todos los elementos no nulos en \mathbb{F}_{p^m} .

Ejemplo A.47. $f(x) = x^4 + x + 1$ es un polinomio primitivo, o equivalentemente, el elemento $x = (0010)$ del campo es un generador de $\mathbb{F}_{2^4}^*$. Es decir que $\mathbb{F}_{2^4}^* = \{x^i \bmod f(x) \mid 0 \leq i < 2^4 - 1\}$.

REFERENCIAS

1. Certicom, *Certicom ecc tutorials*, <http://www.certicom.com/ecc/enter/index.htm>.
2. ———, *Current public-key cryptographic systems*, Whitepaper 2, Certicom Corp., <http://www.certicom.com>, April 1997.
3. ———, *Remarks on the security of elliptic curve cryptosystem*, Whitepaper 3, Certicom Corp., <http://www.certicom.com>, September 1997.
4. ———, *The elliptic curve cryptosystem for smart cards*, Whitepaper 7, Certicom Corp., <http://www.certicom.com>, May 1998.
5. T. Dierks and C. Allen, *The tls protocol version 1.0*, RFC 2246, IETF, <http://www.ietf.org>, January 1999.
6. A. E. Escott, J. C. Sager, A. P. L. Selkirk, and D. Tsapakidis, *Attacking elliptic curve cryptosystems using the parallel pollard rho method*, *CryptoBytes* **4** (1999), no. 2, 15–19.
7. D. B. Johnson and A. J. Menezes, *Elliptic curve dsa (ecdsa): An enhanced dsa*, Whitepaper, Certicom Corp., <http://www.certicom.com>, 1997.
8. A. Jurisic and A. J. Menezes, *Elliptic curves and cryptography*, Whitepaper, Certicom Corp., <http://www.certicom.com>, 1997.
9. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*, CRC Press, Boca Raton, Florida, 1997.
10. NIST, *Digital signature standard*, FIPS PUB 186-2, NIST, <http://csrc.nist.gov/fips/>, January 2000.
11. Certicom Research, *Sec 1: Elliptic curve cryptography*, Working Draft 0.5, Standards for Efficient Cryptography, <http://www.secg.org>, September 1999.
12. ———, *Sec 2: Recommended elliptic curve domain parameters*, Working Draft 0.6, Standards for Efficient Cryptography, <http://www.secg.org>, October 1999.
13. R. L. Rivest, A. Shamir, and L. M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, *Communications of the ACM* **21** (1978), 120–126.
14. Bruce Schneier, *Criptogram*, boletín por e-mail, November 1999.
15. E. De Win and B. Preneel, *Elliptic curve public key cryptosystems - an introduction*, {erik.devin,bart.preneel}@esat.kuleuven.ac.be.

CHACABUCO - BUENOS AIRES - ARGENTINA

E-mail address: gaby@ieee.org

URL: <http://www.geocities.com/belingueres>