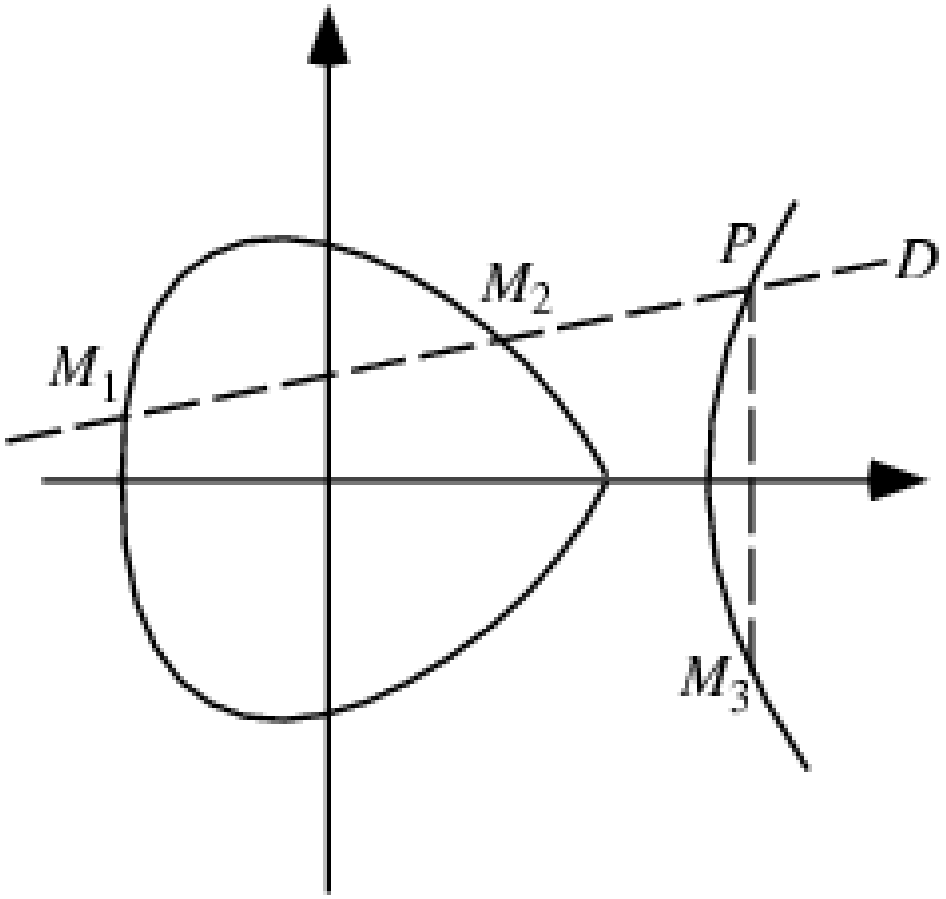


Elliptic Curves Cryptography

Cipriano Callejas Hernández
Temas Selectos de Computación II
FES Acatlán

Introducción

Las curvas elípticas se basan en el estudio de curvas o gráficas de una ecuación que tienen ciertas propiedades de las cuales la criptografía toma ventaja para implementar sus algoritmos, entre ellas, que la llave privada tenga un tamaño reducido; pero con la misma seguridad que los métodos tradicionales, como lo es RSA. En particular se verá tanto la teoría como la implementación del algoritmo de generación de llaves públicas Diffie Hellman a través del uso de curvas elípticas, esto basado en el problema del logaritmo discreto generalizado.

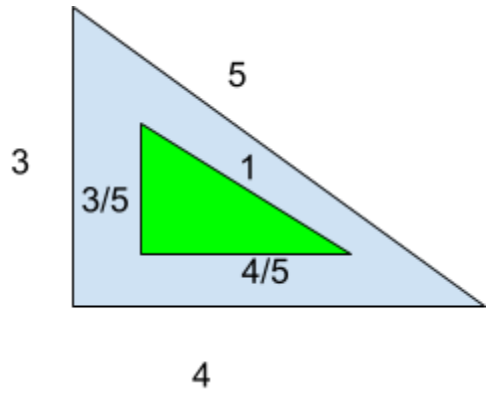
En un principio el estudio de estas curvas cuyas operaciones tienen un sentido geométrico totalmente nuevo al acostumbrado fueron un problema de estudio en la grecia antigua con el problema de los números congruentes, cuya generalización en cierta forma puede ser una introducción a las curvas elípticas dependiendo del campo o dominio en el que se trabajen. Sin embargo, a pesar de tener una teoría matemática sólida con respecto a su analiticidad, siguen siendo favoritas en la implementación de software criptográfico.[1]

Motivación

Sabemos de la relación de Pitágoras que la hipotenusa al cuadrado es igual a la suma de sus cuadrados, esto es ,

$$a^2 + b^2 = c^2 \quad \text{..... (1)}$$

Podemos pensar en el caso más sencillo, un triángulo donde los lados son 3 y 5 respectivamente, por tanto $c = 5$. Más aún, si esto es cierto ¿Qué pasa si reducimos la ecuación?.



Esto es:

$3^2 + 4^2 = 5^2$ .....(E 1)

podemos llegar a $\frac{3^2}{5^2} + \frac{4^2}{5^2} = 1$, con esto hemos encontrado una solución de puntos racionales que satisfacen la ecuación (1) . La pregunta natural sería ¿Existirán más?.

[1] Bluhm, M. & Gueron, S. J Cryptogr Eng (2015) 5: 215. https://doi.org/10.1007/s13389-015-0094-1

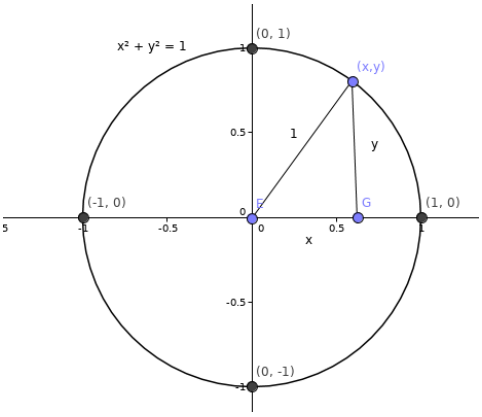
El problema de números congruentes[2] es una idea asociada para responder la pregunta anterior , por ejemplo; 6 es un número congruente por

(E 1) .

Así, podemos plantear el siguiente problema:

Dado (1) dígase que $x = \frac{a}{c}$ y $y = \frac{b}{c}$ con lo que se tiene la siguiente ecuación

$$x^2 + y^2 = 1 \text{.....(2)}$$



Una ecuación que podemos reconocer fácilmente como aquella que representa un círculo centrado en el origen, con radio 1.

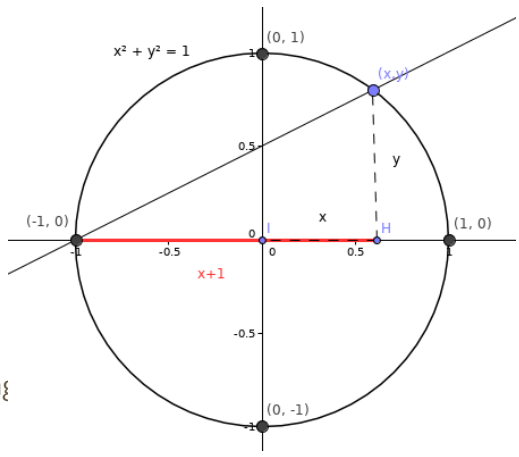
Dado un punto (x, y) en la circunferencia tenemos como referencia (como se ve en la imagen izquierda) que tiene una altura y, y distancia del origen x. Supongamos ahora que tenemos una recta, la cual podemos definir para cualesquiera dos puntos, por conveniencia uno tomaremos $(1, 0)$, siendo el otro un punto general (x, y), la ecuación que representa a la recta con pendiente m e intercepto b es:

$$\ell: y = mx + b \text{ ..... (3)}$$

De nuestra gráfica derecha podemos deducir que la pendiente de dicha recta es

$m = \frac{y}{x+1}$ ..... (4)

y para determinar los interceptos igualamos la ecuación de la circunferencia con el de la recta. Así con un poco de álgebra llegamos al siguiente resultado un polinomio de segundo grado



[2] Un número N se dice congruente si existe un triángulo rectáng
área es N.

$$x^2 + \frac{2m^2}{1+m^2}x + \frac{m^2-1}{m^2+1} = 0 \quad \ldots\ldots\ldots\ldots (5)$$

Para determinar sus soluciones podríamos hacer uso de la fórmula general, sin embargo, recordemos que conocemos una solución de (5), el punto$(-1, 0)$, es decir $x = -1$, y por lo tanto podemos representar (4) como el producto de una de sus raíces y por consiguiente determinar la segunda solución.

$$(x + 1)(x + \frac{m^2-1}{m^2+1}) = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

Finalmente sustituyendo la segunda solución en (4) tenemos las siguientes relaciones

$$x = \frac{1-m^2}{m^2+1} \quad y = \frac{2m}{m^2+1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

Con esto hemos determinado fórmulas que permiten encontrar puntos en el círculo con la característica de ser números racionales. Se puede comprobar fácilmente que con $m = 2$en (7) se llega al mismo resultado del ejemplo (E1).

Volviendo al problema de los números congruentes, se tiene que el área de un triángulo rectángulo es $A = \frac{1}{2}ab$ que por las sustituciones hechas anteriormente se sigue que:

$$A = \frac{1}{2}c^2xy \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (8)$$

Anteriormente logramos determinar fórmulas que permiten encontrar soluciones racionales de la ecuación (2), y por tanto para hallar relaciones tal que determinemos áreas congruentes, sustituimos (7) en (8). Obteniendo:

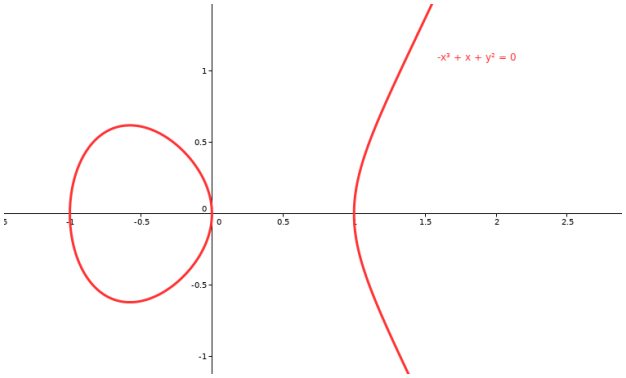
$$(\frac{m^2+1}{c})^2 = m - m^3$$

Ahora, haciendo las nuevas variables como $x = -m$ e $y = \frac{m^2+1}{c}$, se sigue la siguiente ecuación:

$$y^2 = x^3 - A^2x \text{.....(9)}$$

La siguiente pregunta natural es, ¿será posible determinar soluciones racionales para (9)? Notemos que lo más trivial es $(0, 0)$, pero nos gustaría encontrar otras. Desafortunadamente la respuesta es no, esto está justificado por el teorema de Fermat; el cual nos indica que la única solución es $y = 0$ y por tanto 1 no es un número congruente.

El manejo algebraico hecho anteriormente es un ejemplo de una curva elíptica. En criptografía es muy complicado trabajar con números racionales; por lo tanto se hace uso de curvas elípticas módulo un número primo.



La gráfica de la izquierda representa la representación visual de la ecuación (9) con $A = 1$. El nuevo objetivo es determinar soluciones o puntos en la gráfica con números primos, en concreto determinar soluciones de:

$$y^2 = x^3 - x \bmod 11 \text{.....(10)}$$

No es difícil determinar que (10) tiene 11 soluciones, o puntos en la curva que satisfacen dicha ecuación.

Curvas Elípticas

El estudio de curvas elípticas puede ser tan extensivo como se requiera, éstas tienen una fuerte relación con la teoría de funciones elípticas de la cual se deriva su nombre. Notemos entonces que anteriormente afirmamos que 9 era una curva elíptica, sin embargo su gráfica no es la de una elipse. Así como la longitud de arco

en círculos fue lo que dio paso para conocer las funciones trigonométricas, el estudio de elipses dio paso a las integrales elípticas, éstas tienen entre sus características ser funciones multivaluadas en el plano complejo y su inversa (si es que existe), es una función elíptica. De aquí que dichas funciones elípticas tienen una fuerte relación con lo que de ahora en adelante llamaremos curvas elípticas.

Definimos una curva elíptica como la gráfica de la ecuación

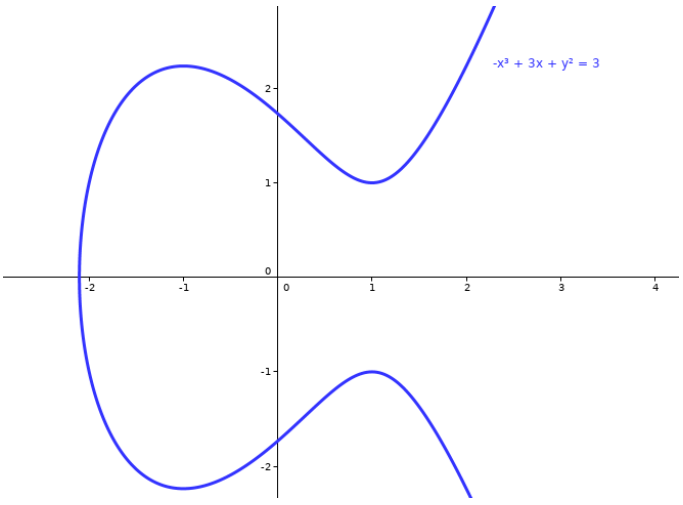
$$E\colon\ y^2 = x^3 + Ax^2 + B \qquad \text{.....(11)}$$

con A, B constantes. También llamada como la ecuación (corta) de Weierstrass.

Sin embargo, el objetivo aquí es determinar bajo qué dominio o espacio serán tanto las variables como las constantes. Por otro lado, dependiendo de este requerimiento es útil pensar en un punto infinito, el cual se encuentra en el punto máximo del eje de las ordenadas e incluso se puede pensar que tanto el punto mínimo del mismo eje se toca en algún punto (por ejemplo, detrás) con el máximo.

Otra de las características de dichas curvas, es la forma en que manejamos la aritmética de sus puntos, así introducimos las siguientes operaciones con un ejemplo en concreto para que logre ser más gráfico y menos tedioso con respecto al álgebra, sin embargo, para un caso general en procedimiento es análogo.

Suma de Puntos en E



Tomemos la ecuación siguiente

$$y^2 = x^3 - 3x + 3 \qquad \text{.....(12)}$$

cuya gráfica se muestra a la izquierda. Nuestro objetivo es determinar la operación de suma sobre los puntos de la curva (12).

Tomemos dos puntos, $P = (x_1, y_1)$ y $Q = (x_2, y_2)$, para definir un punto tercero, que será el resultado de suma P y Q, tracemos una recta ℓ en la curva (12) con P y Q, así ℓ tocará a un tercer punto R', tomamos su reflexión con respecto al eje de las abscisas y denotado por el punto R. De esta forma definiremos la suma de puntos bajo las curvas elípticas como

$$P +_E Q = R \text{.....(13)}$$

Hacemos uso de subíndice para hacer notar que esta suma no es la usual, sino una nueva operación definida en la curva. De aquí en adelante haremos uso de (13) sin el subíndice.

Se puede demostrar que la suma de puntos queda determinada por las siguientes fórmulas, donde $R = (x_3, y_3)$ y $P, Q \neq \infty$

$$x_3 = m^2 - x_1 - x_2 \qquad y_3 = m(x_1 - x_3) - y_1 \text{.....(14)}$$

De donde $m = \frac{y_2 - y_1}{x_2 - x_1}$ si $x_1 \neq x_2$ y $m = \frac{3x_1^2 + A}{2y_1}$ si $x_1 = x_2$

Más aún, tenemos que si $P = Q$ con $y_1 = 0$ entonces $p + Q = \infty$ y $P + \infty = P$.

Definimos la suma de puntos en una curva elíptica Ede una manera poco natural, pues es claro que en otro caso sumar dos puntos en una gráfica no sería más allá que hacerlo entrada por entrada, bajo las operaciones con vectores. Sin embargo la construcción de dicho operando $+_E$permite formular el siguiente teorema.

Teorema 1. La suma de puntos en una curva elíptica Esatisface las siguientes propiedades:

1. (Conmutatividad) $P + Q = Q + P$ para todo $P, Q \in E$
2. (Existencia de la identidad) $P + \infty = P$ para todo $P \in E$
3. (Existencia del inverso) Dado $P \in E$, existe $P^{-1} \in E$ tal que $P + P^{-1} = \infty$ esto es P^{-1} es el inverso de P

4. (Asociatividad) $(P + Q) + R = P + (Q + R)$ para todo $P, Q, R \in E$

El Teorema 1 nos dice que $(E, +_E)$forma un grupo abeliano con ∞como la identidad.

Observaciones:

- Cuando nos referimos a $-P$, es explícitamente al punto $P = (x, -y)$.
- Notemos que $dP = P + P + P + \ldots + P_{d\ veces}$ es decir, la multiplicación de un entero d a un punto se puede ver como la suma de ellos, d veces. Iterativamente.

Curvas Elípticas en Campos Finitos.

Hasta este punto estuvimos trabajando con curvas elípticas sobre el campo de los reales, sin embargo para su uso en criptografía es necesario ver su manipulación en un campo finito.

Recordemos la definición de campo finito.

Definición 1: Un campo (finito) es un conjunto (finito) F asociado con dos operaciones binarias $(+, \cdot)$, tal que se cumple lo siguiente:

- $\boldsymbol{F}$ es un grupo abeliano con respecto a la operación $+$. Denotamos a 0 como el elemento neutro.
- $\boldsymbol{F}\backslash\{0\}$es un grupo abeliano con respecto a la operación $\cdot$, Denotamos a 1 como el elemento neutro
- $\boldsymbol{F}$ cumple la Distributividad del producto sobre la suma, esto es, para todo $a, b \in \boldsymbol{F}$ se tiene que $a \cdot (b + c) = a \cdot b + a \cdot c$

En general hay cuestiones algebraicas que permiten caracterizar a los campos finitos como Campos de Galois, por notación todo campo finito de la forma

$\mathbb{Z}_p = \{0,\ 1,\ \ldots, p\ -\ 1\}$ con p un número primo se denotará como F_p. Más aún, si F_p es un campo finito de orden p^m, entonces la característica de F_p es p, por esto mismo a los campos de Galois de la forma $GF(p^m)$ se les conoce como un campo extendido de F_p de grado m .

Para introducir la manipulación de curvas elípticas en F_p consideremos algunos ejemplos.

Ejemplo 1:

$E(F_5)$: $y^2 = x^3 + x + 1 \ mod\ 5$ .....(E 2)

x	$x^3 + x + 1$	y^2	Puntos
0	1	0	$(0,1),\ (0,4)$
1	3	1	No hay puntos $x = 3$
2	1	4	$(2,1),\ (2,4)$
3	1	4	$(3,4), (3,1)$
4	4	1	$(4,3),\ (4,2)$
∞		∞	∞

De esta forma el orden de F_5 es 9 , el número de puntos.

Para la suma de puntos, es natural pensar que las fórmulas (14) tienen que ser modificadas modularmente, es decir:

$$x_3 = m^2 - x_1 - x_2 \ mod\ p \qquad y_3 = m(x_1 - x_3) - y_1 mod\ p \text{.....(15)}$$

De donde $m = \frac{y_2 - y_1}{x_2 - x_1} \, mod \, p$ si $x_1 \neq x_2$ y $m = \frac{3x_1^{\,2} + A}{2y_1} \, mod \, p$ si $x_1 = x_2$

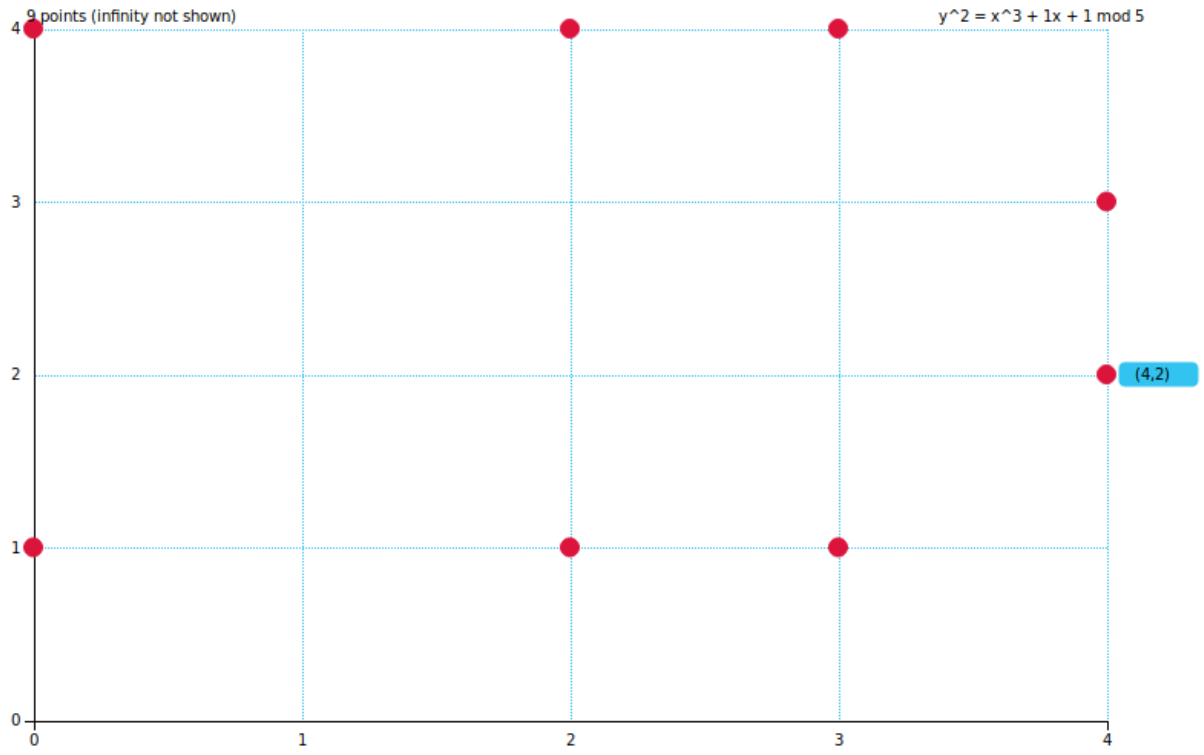
Así tomemos el siguiente ejemplo.

Ejemplo 2: Verificar que $(3, 1) + (2, 4) = (4, 2)$

Haciendo uso de las relaciones modificadas (15) tenemos que son entradas diferentes, así $m = \frac{4-1}{2-3} = \frac{3}{-1} = -3 \equiv 2 \, mod \, 5$, luego

$x = (2)^2 - 3 - 2 = -1 \equiv 4 \, mod \, 5$ y $y = 2(3 - 4) - 1 = -3 \equiv 2 \, mod \, 5$ por lo tanto el punto resultante de la suma es $(4, 2)$.

La siguiente gráfica muestra los 9 puntos de la curva.



Fuente: http://www.graui.de/code/elliptic2/

Incluso para un campo de orden 9 (E 2) la aritmética y los puntos generados puede ser computacionalmente lento, y muchas veces podemos necesitar determinar el orden de otro campo sin tener que hacer los cálculos de todos los

posibles puntos, aquí es donde introducimos el Teorema de Hasse, el cual permite tener cotas para el orden de una curva elíptica sobre un campo F_p.

Teorema 2: (Hasse) Sea E una curva elíptica sobre un campo finito F_p. Entonces el orden de $E(F_p)$, es decir, el número de puntos denotado por $\#E$ está acotado por

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p}$$

Entendamos que el Teorema de Hasse informalmente significa que el orden es aproximadamente p pues es éste más una cantidad muy pequeña. La necesidad de conocer el orden de las curvas elípticas se basa en prevenir ciertos ataques en criptografía de dichas curvas. Más aún, computacionalmente determinar manualmente el orden es difícil.

Criptografía de Curvas Elípticas (ECC)

ya

Ejemplo 3: $(5, 1) + (5, 16)^{-1} = \infty$ en módulo 17

Es claro que la primera coordenada es la misma y $-1 \equiv 16 \, mod \, 17$.

Como se había mencionado,ECC se basa en el problema del logaritmo discreto generalizado, así pues tratemos de determinar los ingredientes de la Definición 2 en términos de curvas elípticas:

Bajo ciertas condiciones los puntos de una curva elíptica sobre un campo finito forman un grupo cíclico, sin embargo dado un elemento en la curva (un punto) P podemos considerar el subconjunto de$E$ tal que son generados por P, a este conjunto subgrupo cíclico generado por P.

Ejemplo 4: Sea E: $y^2 = x^3 + 2x + 2 \; mod \; 17$ cuyo orden es 19, por la teoría de grupos, cuando éste es de orden primo entonces el conjunto de puntos en la curva forma un grupo cíclico. Se puede probar [3] que si tomamos un punto $P = (5, 1)$ haciendo $P,\ 2P,\ 3P,\ ...,\ \#E\ P$se tienen todos los puntos (elementos) del grupo.

Tabla 1[4]:

$2P = (5,1) + (5,1) = (6,3)$	$6P = (16,13)$
$3P = 2P + P = (6,3) + (5,1) = (10,6)$	$7P = (0,6)$
$4P = (3,1)$	$8P = (13,7)$
$5P = (9,16)$	$13P = (16,4)$

Ahora podemos describir concretamente el problema de logaritmo discreto en términos de curvas elípticas.

Definición 4: Dada una curva elíptica E. Considera un elemento primitivo P y otro cualquiera T. El problema de logaritmo discreto es encontrar el entero tal que $1 \leq d \leq \#E$, tal que $P + P + ... + P_{\ d\ veces} = dP = T$.

Ejemplo 5: Sea E: $y^2 = x^3 + 2x + 2 \; mod \; 17$, el generado $P = (5, 1)$ y el punto $(7, 9)$, sabemos por la Tabla 1 que $d = 9$.

Aplicación: Generador de llave pública con Diffie Hellman

Antes de comprender cómo las curvas elípticas son usadas en la Técnica de Diffie Hellman debemos introducir dicha técnica con un enfoque general.

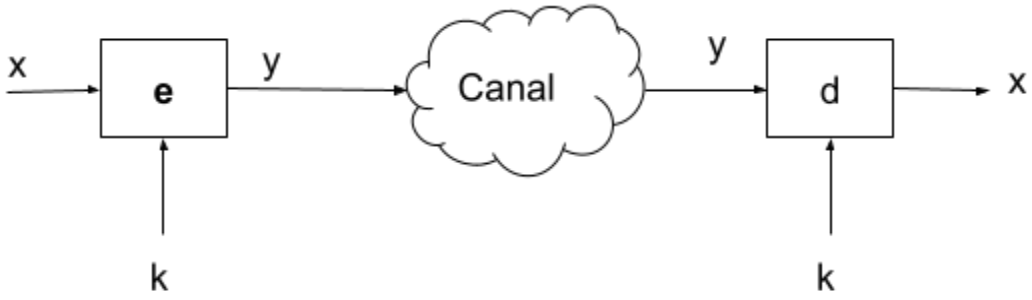
pg 247

Diffie Hellman

[3] Consulte [Paar, pag. 247]
[4] Tabla tomada de Paar, página 246.

La historia de Diffie y Hellman se remonta a 1976, año en que publicaron su artículo llamado “New directions in Cryptography”, en cual observaron que el mundo tiene comportamientos *asimétricos*, en particular, existen ciertas acciones que son fáciles de hacer pero no de deshacer; por ejemplo, romper un vaso de vidrio es muy fácil, pero es casi imposible volver a reconstruir el mismo. En RSA, es muy fácil multiplicar dos números primos grandes pero recuperarlos es muy difícil computacionalmente. Esto dio inicio al intercambio de llaves públicas, es decir, permitir dos usuarios (generalmente computadoras o dispositivos) compartir una llave secreta a través de un canal público con la característica de que estos puedan usarla para operar, pero no un tercero.

De esta forma, como mencionan Diffie y Hellman, se puede tomar ventaja de la perspectiva asimétrica de las cosas. Así el intercambio de llaves Diffie Hellman permite, a través de una llave privada, generar una pública.



Suponga que Alice y Bob desean enviar información encriptada entre ellos, para esto necesitan de una llave que permite hacer funcionar las funciones de encriptación y desencriptado de los algoritmos elegidos, respectivamente. Sin embargo, para ambos necesitan una llave en común, antiguamente se haría en físico, con lo que la opción de generar una llave pública era difícil. En contraste Diffie y Hellman proponen el siguiente algoritmo:

Se conocen los siguientes parámetros públicamente[5], estos comúnmente son propuestos por organizaciones que han probado su seguridad. Entonces suponga que p y α fueron acordados entre ambos.

Alice elige aleatoriamente un entero[6] $a \in \{2, 3, \ldots, p-2\}$, después hace $A = \alpha^a \bmod p$ y se la envía a Bob. Por otro lado, Bob elige también de manera aleatoria $b \in \{2, 3, \ldots, p-2\}$ y hace $B = \alpha^b \bmod p$, después de lo envía a Alice.

De esta manera se ha generado una clave pública, la cual es de la siguiente:

$$k = \alpha^{ab} \bmod p$$

La cual es el resultado de que Alice y Bob realicen:

$k = B^a \bmod p$, $k = A^b \bmod p$, respectivamente.

La fortaleza del algoritmo reside en que a pesar de que se conocen p, α, A, B el **problema de Diffie Hellman** (DHP) puede ser concretado matemáticamente como: Dado un grupo cíclico y finito G_p, un generador $\alpha \in G_p$ y dos elementos cualesquiera del grupo $\alpha^a, \alpha^b \in G_p$. Determinar α^{ab}.[7]

Esto es, en el dominio público se tienen α^a, α^b . Una manera natural sería multiplicarlos, es decir, α^{a+b} lo cual es diferente a la llave buscada.

Diffie Hellman y Curvas Elípticas.

El algoritmo es sumamente similar al de Diffie Hellman sobre un grupo cíclico, la diferencia ahora es el dominio o conjunto de elementos del que se extraerán los parámetros.

[5] En inglés se les conoce como *Domain Parameters*
[6] Tanto a como b , son elementos que claramente deben mantenerse privados. Sin embargo, pertenecen al mismo conjunto en el que tanto Alice como Bob acordaron trabajar.
[7] También conocido como Problema del Logaritmo discreto.

El algoritmo es el siguiente:

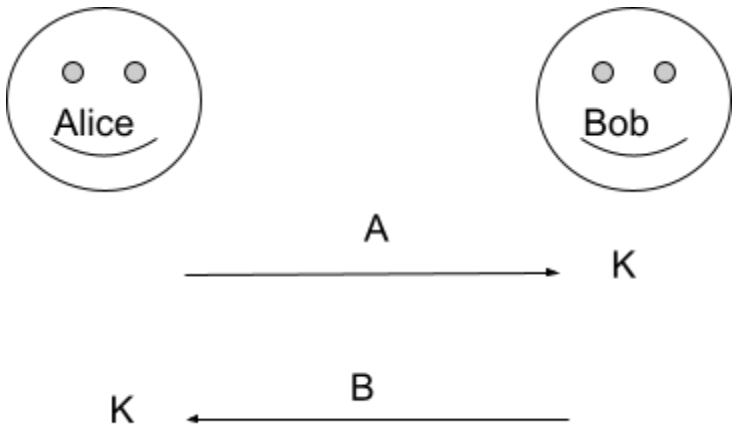
Se tienen los parámetros conocidos $\{p,\ a,\ b, G,\ n,\ h\}$ de donde

- p es un número primo (usualmente grande)
- a,b representan los parámetros de la ecuación (11) para una curva elíptica.
- G es un punto en la curva, con la característica de ser el generador.[8]
- n es el orden de la curva, es decir, el número de puntos cuyas entradas pertenezcan al conjunto F^p.
- h es más que un parámetro público, un requisito recomendable, $h \leq 4$ de donde $h = \frac{1}{2}\left|E(F_p)\right|$. Preferentemente[9] $h = 1$.

Luego,

Volviendo al problema de generar una llave pública tanto para Bob como Alice se hace de la siguiente manera.

Se obtiene un entero aleatorio dentro del campo F_p. Alice toma $d_A \in \{2,\ldots,\ \#E - 1\}$ y calcula $A = d_A G$. De igual manera, Bob escoge $d_B \in \{2,\ldots,\ \#E - 1\}$, $B = d_B G$. Ambos envían A y B, respectivamente.



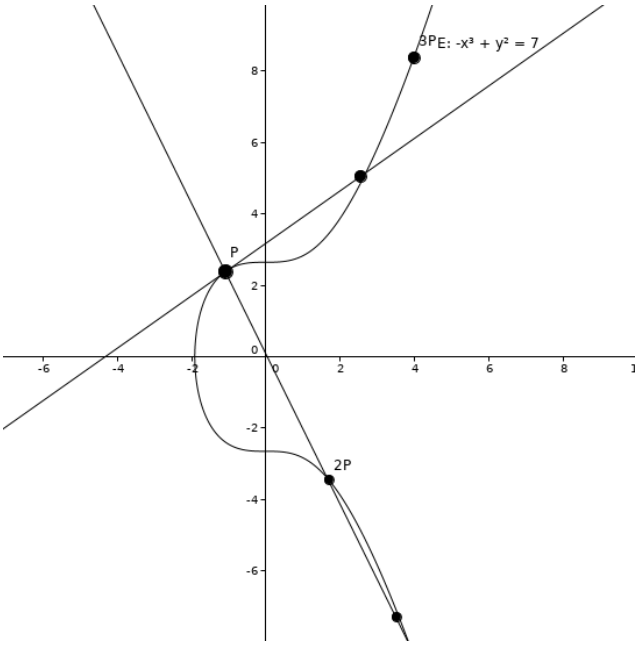
De esta manera ambos tienen una llave de la forma $k = d_B d_A G$.

De nuevo, desde la perspectiva asimétrica es sencillo en términos computacionales de hacer los cálculos anteriores, pues la llave generada no es más que una suma iterada del

[8] Esto es, aquel que multiplicado por la cardinalidad de la curva da como resultado, la identidad, o el punto al infinito (identidad en las curvas elípticas)
[9] Fuente https://en.wikipedia.org/wiki/Elliptic-curve_cryptography#Domain_parameters

punto, para el cual existe la técnica “Add and doubling” que optimiza dicha operación.

Por otro lado, la resistencia contra ataques reside en el problema generalizado del logaritmo discreto, esto es, dado dP un punto en al curva, determinar el número de adiciones que te llevaron ahí ,d.



Geométricamente el problema es muy parecido como en Diffie Hellman estándar donde ahora tienes un punto dado en un reloj α^{ab},el problema consiste en determinar cuántas vueltas se hicieron para llegar ahí, ab.

Implementación Python

Notemos que el hecho de hacer dP es equivalente a sumar d veces el punto, más aún, no es realmente el mismo, es decir, los primeros dos sumandos son $P + P = 2p$ lo que equivale a una suma de puntos iguales, por tanto en una primera función debemos considerar *point doubling* como una primera función, luego en la siguiente suma $P + P + P = 2P + P$ lo que implica una suma de dos puntos diferentes (segunda función). Más detalladamente, para construir estas funciones haremos uso de las fórmulas (15), donde tenemos una división modular, esto es, $\frac{a}{b}\ mod\ n$ lo cual también es equivalente a hacer $ab^{-1}\ mod\ n$, con lo que tenemos la siguiente función, el algoritmo extendido de Euclides para

determinar inversos multiplicativos. Finalmente la función que calculará dPse basará en el algoritmo de "*Add and doubling*".

Algoritmo Add and doubling para multiplicación por escalar.

Entrada: Una curva elíptica y un punto generador P, donde el escalar d se descompone en una combinación lineal de factores binarios, es decir:

$$d = \sum_{i=0}^{t} d_i 2^i; \; d_i \in \{0, 1\}$$

Salida: $dP = T$

Algoritmo:

Se inicia $T = P$

Para i de 1 hasta t hacer

$T = T + T$

Si $d_i = 1$entonces $T = T + P$

Fin Si

Fin Para.

Código Python. [10]

Entrada: Definimos los parámetros del dominio.

```
Pcurve=17
N=19
Acurve=2; Bcurve=2
Gx=5; Gy=1
Gpoint=(Gx,Gy)
privKey=13
```

[10] El código puede encontrarse en https://github.com/CiprianHdz/Criptograf-a-/blob/master/Diffie%20Hellman%20.ipynb

Función 1: Algoritmo Extendido de Euclides.

```
def ECDiv(a, n=Pcurve): #Regresa el inverso de a módulo n
    r1,r2=n,a
    t1,t2=0,1
    while r1 >1:
        q=r1 // r2
        r=r1-q*r2
        r1,r2=r2,r
        t=t1-q*t2
        t1,t2=t2,t

    return t1%n #Este último es para evitar valores negativos
```

Para su implementación el código se basó en el algoritmo tomado de [Multiplicative Inverse]

Función 2: Suma de puntos diferentes.

```
def ECSum(a,b): #Suma de puntos en la curva elíptica
    m=( ((b[1]-a[1])% Pcurve) * ECDiv((b[0]-a[0])%Pcurve, Pcurve) ) % Pcurve
    x=(m*m-a[0]-b[0]) % Pcurve
    y=(m*(a[0]-x)-a[1]) % Pcurve
    return (x,y)
```

Función 3: Suma de puntos iguales o Doubling.

```
def ECDoub(a): #Suma iterada del punto
    m= ((3*(a[0]*a[0])+Acurve)* ECDiv(2*a[1], Pcurve))% Pcurve
    x= ((m*m)-2*a[0]) % Pcurve
    y= (m*(a[0]-x)-a[1]) % Pcurve
    return (x,y)
```

Observación: Tanto para las funciones 2 y 3, notemos que sólo fue usar las fórmulas (15) y hacer pequeñas modificaciones, por ejemplo, para que se respetara la conmutatividad, i.e. , $P + Q = Q + P$ al momento de obtener el inverso se tenía que hacer módulo antes de ser pasado a la función , esto para evitar valores negativos ya que de la fórmula (15) el signo de $m = \frac{y_2 - y_1}{x_2 - x_1}$sí depende del orden en que se ingresen a la función.

Función 4: Multiplicación de un punto por un escalar.

```
def ECMult(Genpoint, key): #Operación de dQ, Entrada: P , d
    if key ==0 or  key >= N: raise Exception("Llave inválida") #Evitar errores
    keyBin=str( bin(key) ) [2:] #Pasamos d en su representación binaria
    Q=Genpoint #Inicialización
    for i in range(1,len(keyBin)):
        Q = ECDoub(Q)
        if keyBin[i] =='1':
            Q=ECSum(Q,Genpoint)
    return Q
```

Ejemplo 6: En un primer ejemplo usaremos el que hemos trabajado manualmente (Ejemplo 4).

Sean los parámetros

- E: $y^2 = x^3 + 2x + 2 \, mod \, 17$ es decir, $a = 2 = b$ y $p = 17$
- El punto generador ya lo habíamos obtenido siendo $P = (5, 1)$ y el orden también $n = 19$
- Sea la llave privada $d = 13$

El objetivo es computar $13P = 13 * (5, 1)$

Definimos las variables con los nuevos datos:

```
Pcurve=17
N=19
Acurve=2; Bcurve=2
Gx=5; Gy=1
Gpoint=(Gx,Gy)
privKey=13
```

Y simplemente hacemos uso de la función ECMult con los parámetros $Gpoint$ y $privKey$. Obteniendo:

```
print "La llave pública es:", ECMult(Gpoint,privKey)
```

```
La llave pública es: (16, 4)
```

Que coincide con el valor que se tenía.

Internamente lo que se está haciendo es lo siguiente:

Usando el método de *"Add and doubling" primero hacemos una trasnformación de la llave privada d en binario. Es decir.*

$$13P = 1101P = d_1 d_2 d_3 d_4 P$$

En principio: $Q = P$

Para $i = 1$; (*Doubling*) $Q = Q + Q = (5,1) + (5,1) = (6,3)$, como $d_1 = 1$ entonces hacemos $Q = Q + P = (6,3) + (5,1) = (10,6)$(*Adding*)

Para $i = 2$; (*Doubling*) $Q = Q + Q = (10,6) + (10,6) = (16,13)$, como 0 no hacemos nada.

$Para\ i = 3$; (*Doubling*) $Q = Q + Q = (16,13) + (16,13) = (0,11)$, como $d_1 = 1$entonces hacemos $Q = Q + P = (0,11) + (5,1) = (16,4)$(*Adding*)

Lo que es equivalente a

Para $i = 1$; (Doubling) $2P = 01P$, (Adding) $2P + P = 3P = 11P$

Para $i = 2$; (Doubling) $6P = 011P$

Para $i = 3$; (Doubling) $12\,P = 0111\,P$, (Adding)$12P + P = 13P = 1101P$

Ejemplo 7:

De esta manera el código puede ser usado para determinar llaves públicas, usando parámetros conocidos, para esto nos referimos a los estándares para una criptografía eficiente, SEC2: Recommender Elliptic Curve Domain Parameters.[11]

[11] Disponible en http://www.secg.org/sec2-v2.pdf

En particular tenemos la curva para 256 bits, dada por

$$y^2 = x^2 + 7 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (16)$$

Es decir, los parámetros[12] dados son:

```
Pcurve=2**256-2**32-2**9-2**8-2**7-2**6-2**4-1
N=0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
Acurve=0; Bcurve=7
privKey=0xE9873D79C6D87DC0FB6A5778633389F4453213303DA61F20BD67FC233AA33262
```

En el formato, cabe notar que sólo se da el valor de x para el punto generado, sin embargo, esto no es un inconveniente pues de (16) podemos obtener el valor de y en particular obtendremos el positivo[13].

```
Gx=int(0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798)
Gx
```

```
55066263022277343669578718895168534326250603453777594175500187360389116729240L
```

```
import math

Gy=int(math.sqrt((Gx*Gx*Gx)+7))
Gy
```

```
129219604432978288608916481582924718196751077013132353612479945684186185257286635210403707677311726174163585004994 56L
```

```
Gpoint=(Gx,Gy)
```

Y finalmente se tiene la llave pública:

```
print "La llave pública es:", ECMult(Gpoint,privKey)
```

```
La llave pública es: (13849298010291207813366229657240249850107058795911473616135054469160306592672L, 702610329180192388283758227745263282849802589222798531744820233670217093346 34L)
```

Aplicación:

Uno de los usos de generar una llave pública puede ser para el comercio de bitcoins, pues si se visita la página

[12] La llave privada fue tomada de https://en.bitcoin.it/wiki/Private_key

[13] Esto pues, la ecuación 16 es un curva que es simétrica con respecto al eje de las abscisas y por tanto podemos hacer uso del valor negativo o positivo.

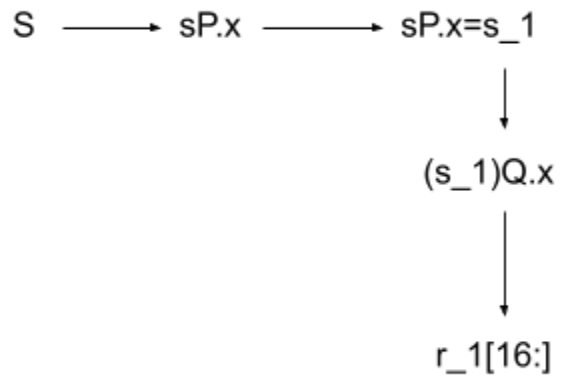
https://www.bitaddress.org/bitaddress.org-v3.3.0-SHA256-dec17c07685e1870960903d8f58090475b25af946fe95a734f88408cef4aa194.html

se puede jugar con las llaves privadas que genera y al mismo tiempo generar tus propias llaves públicas.

Otras aplicaciones de las curvas elípticas (PRNG).

Hasta el momento el uso de curvas elípticas se basa en la dificultad de determinar cuántas veces un punto se movió en la curva, determinar el d en dP dado. Por consiguiente la mayoría de aplicaciones tiene como principal base éste mismo problema, es decir, hacen uso del problema de logaritmo discreto generalizado (ECDLP).

En particular tenemos lo que en un momento logró controversia, la generación de número pseudoaleatorios con curvas elípticas[14]. El cual tiene el siguiente esquema.



Dado dos puntos $P, Q \in E$ y una semilla[15] s uno puede ir generando números aleatorios de la siguiente manera:

Se toma una semilla, se hace la multiplicación con el punto se genera una segunda semilla con la coordenada x, que se toma como el proceso interno para determinar el bit random como la coordenada x de la multiplicación del resultado anterior por el punto Q, r_1. Finalmente se toman los últimos 16 bits significativos.

[14] También conocido como Dual Elliptic Curve RNG
[15] Generalmente la semilla es lo más cercano a un proceso aleatorio, por ejemplo, el movimiento del mouse.

La controversia de dicho generador comenzó como una posibilidad de determinar qué es lo que pasa internamente, consecuentemente generar o conocer el nuevo bit “aleatorio” . Esto basado en la posibilidad de un punto sea el múltiplo de otro, es decir, $Q = sP$.

Si esto es así, una vez determinados los 16 dígitos más significativos haciendo fuerza bruta, entonces $r_1 = (x, y)$ es un punto. Más aún, si se conoce s_1, podemos computar $r_1 = s_1 Q$.

Más aún podemos suponer que $dr_1 = ds_1 Q = s_1 dQ = s_1 P$ que es el proceso interno, con el que se puede generar s_2. La implicación que tiene esto, conocido como Backdoor Dual EC, es que desde un número aleatorio se puede determinar el siguiente (conocer el proceso interno).

Bibliografía

Blake, I. F., Seroussi, G., & Smart, N. P. (1999). *Elliptic curves in cryptography*. Cambridge : Cambridge University Press, 1999.

Aumasson, J. P. (2017). Serious Cryptography: A Practical Introduction to Modern Encryption.

Paar, C., & Pelzl, J. (2009). *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media.

Washington, L. C. (2003). *Elliptic curves: number theory and cryptography*. Chapman and Hall/CRC.