Universidad Nacional Autónoma de México

Facultad de Ciencias

Criptosistema análogo a RSA usando curvas elípticas

T E S I S

QUE PARA OBTENER EL TÍTULO EN:

Licenciado en Ciencias de la Computación

PRESENTA:

Mauricio Daniel Garza Rauda

DIRECTORA DE TESIS:

Mat. Anayanzi Delia Martínez Hernández

Ciudad Universitaria, D.F.

2014

Datos del alumno: Garza Rauda Mauricio Daniel 5741 1536 Universidad Nacional Autónoma de México Facultad de Ciencias Ciencias de la Computación 303139447
Datos de la tutora: Mat. Anayanzi Delia Martínez Hernández
Datos del sinodal 1: Dr. José de Jesús Galaviz Casas
Datos del sinodal 2: Mat. Julio César Guevara Bravo
Datos del sinodal 3: Mat. Zazil Santizo Huerta
Datos del sinodal 4: Dra. María de la Luz Gasca Soto
Datos de la tesis: Criptosistema análogo a RSA usando curvas elípticas 61 p. 2014

A mis padres.

Por el amor y cariño que me han dado,
así como por todo su apoyo en mi vida.

A Rut.

Por su apoyo, amor y amistad incondicional.

Agradecimientos

Inicialmente, quiero agradecer a mis padres por todo su apoyo y paciencia, por ser un ejemplo de vida, por sus consejos y por los sacrificios que han hecho para que pueda estar donde estoy ahora.

A Rut le agradezco estar en mi vida, por todo lo que hemos pasado juntos, por agregar sonrisas y cariño a cada día, por el simple hecho de ser ella. Te amo.

Igualmente, agradezco a la Dra. María de la Luz Gasca Soto (a.k.a. Lucy) por su guía, por ser una gran profesora y sobretodo por brindarme la oportunidad de crecer académicamente en estos años.

Agradezco a la Universidad Nacional Autónoma de México, por permitirme ser parte de ella.

Con afecto le agradezco a la H. Facultad de Ciencias por todo lo que he aprendido en mis años de estudiante.

Todos los profesores que conocí forman parte de grandes recuerdos, pues me enseñaron que todos los días se aprende algo nuevo.

He tenido compañeros y amigos que me han ayudado como Jonathan Badillo y sus valiosos e infinitos consejos de "suma un cero ahí", a mis amigas Elsa y Karen, a Juan y al Zeke que de igual forma sus aportaciones fueron valiosas y me ayudaron a resolver problemas.

Un gran apoyo e impulso, fueron los que recib de Egar García y Mario Gama, ya que me apoyaron, incluso más allá de la Facultad.

Le doy las gracias a mis amigos Arturo, Fernando, Alejandro, Lupa y Lalo.

Hacia mediados de julio del 2011 conocí a un grupo de personas, Alberto y Norma, en la jefatura de posgrado de filosofía, a ellos también les doy las gracias por permitirme formar parte de su equipo.

Un gran avance en mi forma de analizar problemas y programar fue gracias a mis profesores en la DGSCA (ahora DGTIC), en el departamento de realidad

virtual.

Formalmente agradezco a la Mat. Anayanzi Martínez por sus consejos y ayuda invaluable que permitieron la finalización de este trabajo, sin ella no se hubiera logrado. Al Dr. Octavio Paez por sus consejos, ideas y guía en esta tesis.

Han estado presentes también personas con las que no compartí aulas, pero si experiencias.

Todos los miembros del taller de violín de la facultad tienen mi gratitud, en especial a los miembros del ensamble de cuerdas, ya que sin duda ha sido toda una experiencia enriquecedora el formar parte de ese grupo.

A los sinodales que revisaron este trabajo les agradezco por su invaluable ayuda, que resultó en la mejora de mi tesis.

Grandes momentos viví en esas partidas de Magic con Hebus, Luis, George, Ricardo, Raul "el monstruo" y compañía, ya que la diversión y el entretenimiento también forman parte de las memorias que me llevo de mi tiempo como estudiante.

No puedo poner a todos los que me han impactado positivamente aquí, principalmente por mi mala memoria, si faltara alguien lo siento.

Gracias a todos.

"Decimos 'silla' pero no queremos decir 'silla', y nos entienden.
O por lo menos nos entienden aquellos a quienes
está secretamente destinado el mensaje, ***críptico****,*
pasando indemne a través de las multitudes indiferentes y hostiles."

Sobre héroes y tumbas, Ernesto Sabato.

Índice

Índice de tablas

Índice de algoritmos

1 Introducción

"El problema de todas las historias es
que se cuentan después de que hayan pasado."
Nana, Chuck Palahniuk

El objetivo de esta tesis es explorar el esquema criptográfico de Demytko propuesto en [3] y dar una implementación a éste. También se analizan los métodos elaborados para ver, en términos de complejidad computacional, su desempeño.

El trabajo elaborado se distribuye de la siguiente forma:

- Capítulo 1. Se revisan algunas definiciones básicas en terminología y notación que se utilizarán en el trabajo.
- Capítulo 2. Se presentan definiciones y propiedades de curvas elípticas sobre Z_p y Z_n. Notar que se usará a p como un número primo, lo que implica que Z_p es el campo finito de p elementos.
- Capítulo 3. Se enuncian las definiciones de los criptosistemas RSA y el propuesto por Demytko [3].
- Capítulo 4. Se revisan conceptos en el análisis de algoritmos, que son necesarios para el análisis de la implementación dada del criptosistema.

- Capítulo 5. Se revisan los detalles de la implementación junto con el análisis correspondiente a los métodos expuestos, así como los detalles concernientes a la selección de la curva y el mapeo entre el espacio de mensajes y los puntos de la curva.

- Capítulo 6. Se presentan resultados encontrados con las pruebas realizadas usando la implementación dada.

1.1. Criptología

La criptología, de sus raices griegas *krypto*: oculto y *logos*: estudio, palabra, es la disciplina que se encarga de estudiar 'mensajes ocultos'. Ésta a su vez se puede dividir en otras disciplinas más especializadas: la criptografía y el criptoanálisis. "El criptógrafo busca encontrar métodos que aseguren la privacidad y/o autenticidad de mensajes. El criptoanalista busca deshacer el trabajo de un criptógrafo, usando debilidades en la regla de cifrado o la generación de mensajes que se hagan pasar como auténticos"[14], a esto último se le conoce como *romper el criptosistema*.

1.1.1. Criptografía

El principal objetivo de la criptografía, *graphos*: escribir, es permitir que dos entidades, $\mathcal{A}$ y $\mathcal{B}$, se comuniquen a través de un canal no seguro (este canal puede ser público), de tal forma que una entidad no autorizada, digamos $\mathcal{C}$, que tenga acceso al canal no pueda entender los mensajes entre $\mathcal{A}$ y $\mathcal{B}$.

El proceso de comunicación se puede describir como sigue: a la información que $\mathcal{A}$ mandará a $\mathcal{B}$ se le llama *mensaje*. $\mathcal{A}$ aplica un algoritmo para transformar este $mensaje$ en un $mensaje\ cifrado$ (el *mensaje cifrado* no es comprensible para $\mathcal{C}$), a esto se le conoce como $cifrar$ el $mensaje$. $\mathcal{A}$ envía el $mensaje\ cifrado$ a $\mathcal{B}$ usando el canal. $\mathcal{B}$ recibe el $mensaje\ cifrado$ y le aplica un algoritmo para recuperar el $mensaje$ original, a esto se le conoce como $descifrar$ el $mensaje$.

Del párrafo anterior surge la siguiente definición:

Definición: Un sistema criptográfico Cr (Stinson [15]), o criptosistema, es una quíntupla (M,f_e,e,f_d,d) donde

- M es el conjunto de todos los posibles *mensajes*.
- f_e es el algoritmo usado para cifrar el *mensaje*.
- e es la llave que usa el algoritmo de cifrado de tal forma que si m es un *mensaje* a cifrar, entonces $f_e(m) = c$, c es el *mensaje cifrado* correspondiente a m bajo Cr.
- f_d es el algoritmo usado para descifrar el *mensaje*.
- d es la llave que usa el algoritmo de descifrado de tal forma que si c es el *mensaje cifrado* correspondiente a un *mensaje* m bajo Cr, entonces $f_d(c) = m$.

Si $e = d$ entonces se dice que Cr es un criptosistema simétrico, de otro modo es asimétrico. Los *mensajes* (espacio de mensajes) que resultan del cifrado en general son números dentro de un rango (para los criptosistemas que se definirán más adelante se trata de enteros en Z_n, con $n = p * q$ con p y q números primos).

1.1.2. Criptoanálisis

El criptoanálisis se encarga de estudiar los criptosistemas con el objetivo de usar propiedades de los datos/algoritmos que usa el criptosistema para recuperar el *mensaje* original sin el conocimiento de información secreta. La hipótesis general es que $\mathcal{C}$ conoce el criptosistema, esto se conoce como el principio de Kerckhoffs [7]. Claro que el hecho de que $\mathcal{C}$ no conozca el criptosistema usado hace su trabajo más complicado, pero no se quiere basar la seguridad de un criptosistema en la premisa de que $\mathcal{C}$ no conoce el criptosistema usado.

Los ataques más comunes a criptosistemas (ver [4]), son los siguientes:

- **Sólo *mensaje cifrado***: $\mathcal{C}$ posee una cadena de *mensaje cifrado*.
- ***Mensaje* conocido**: $\mathcal{C}$ posee una cadena de *mensaje* y su correspondiente *mensaje cifrado*.
- ***Mensaje* elegido**: $\mathcal{C}$ obtuvo acceso a la regla de cifrado de manera temporal, y ha elegido una cadena de *mensaje* y construido el *mensaje cifrado correspondiente*.

- ***Mensaje cifrado* elegido**: $\mathcal{C}$ obtuvo acceso a la regla de descifrado y ha elegido una cadena de *mensaje cifrado* y construido el *mensaje* correspondiente.

Se dice que los ataques tienen como objetivo *romper* el criptosistema, es decir, poder descifrar los *mensajes cifrados* generados por el criptosistema y en algunos casos también el generar *mensajes cifrados* (esto último en la generación de $firmas$, ver capítulo 3).

2
Curvas Elípticas

"El joven escultor afirmó que la geometría de la ciudad de sus sueños era anormal, no euclidiana, y que sugería esferas y dimensiones distintas de las nuestras."
La llamada de Cthulhu, H. P. Lovecraft.

En este capítulo se revisarán algunos conceptos y resultados referentes a curvas elípticas que se necesitan para la implementación del criptosistema que define Demytko [3].

2.1. Curvas elípticas sobre Z_p

Las curvas elípticas que son usadas en esta tesis son las comprendidas dentro de la siguiente definición.

Definición: Sea $p > 3$ un número primo y $a, b \in Z_p$ constantes tales que $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$. Definimos como **curva elíptica** $E_p(a, b)$ al conjunto de soluciones $(x, y) \in Z_p \times Z_p$ de la congruencia

$$y^2 \equiv x^3 + ax + b \quad \text{mód } p, \tag{2.1}$$

junto con un punto especial $\mathcal{O}$ llamado punto al infinito. Los elementos de este conjunto son llamados **puntos de la curva**.

Ejemplo: La curva $E_{11}(4, 9)$ (que son las soluciones $(x, y) \in Z_p \times Z_p$ de la congruencia $y^2 \equiv x^3 + 4x + 9 \mod 11$), será el conjunto: $\{\mathcal{O}$, (0,3), (0,8), (1,5), (1,6), (2,5), (2,6), (3,2), (3,9), (4,1), (4,10), (5,0), (8,5), (8,6), (9,2), (9,9), (10,2), (10,9)$\}$.

A una curva elíptica $E_p(a, b)$ se le puede dar estructura de grupo abeliano definiendo una operación entre sus puntos. Esta operación es escrita aditivamente, y está definida como sigue (todas las operaciones aritméticas se realizan en Z_p).

Sea

$$P + \mathcal{O} = \mathcal{O} + P = P$$

para todo punto $P \in E_p(a, b)$.

Si P y Q son puntos de $E_p(a, b)$ con $P = (x_1, y_1)$ y $Q = (x_2, y_2)$, entonces se tienen los siguientes casos:

- $x_1 = x_2$ y $y_1 \neq y_2$, esto implica que $y_1 = -y_2$ ya que $(-y_2)^2 = {y_2}^2$, lo que solucionaría la ecuación para x_1, para este caso la operación se define como

  $$P + Q = \mathcal{O}.$$

  Se dice que Q es el **inverso de** P y se denota como $-P$.

- $x_1 = x_2$ y $y_1 = y_2 = 0$, implica que $P = -P = Q = -Q$, y por el anterior la operación es

  $$P + (-P) = \mathcal{O}.$$

- En cualquier otro caso

  $$P + Q = (x_3, y_3), \text{ con}$$

$$
\begin{aligned}
x_3 &= \lambda^2 - x_1 - x_2 \quad \text{mód } p \\
y_3 &= \lambda(x_1 - x_3) - y_1 \quad \text{mód } p \\
\text{donde } \lambda &= \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{si } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{si } P = Q \end{cases}
\end{aligned}
$$

Teorema: $E_p(a, b)$ con la operación + que se ha definido es un grupo abeliano con elemento identidad $\mathcal{O}$.

Demostración:

- Neutro: por definición el elemento $\mathcal{O}$ es neutro y único.

- Inverso: para todo punto (x, y) de la curva existe otro punto $(x, -y)$ tal que es su inverso (ya que la ecuación 2.1 es cuadrática), hay que notar que los inversos son únicos.

- Cerradura: en [13] se pueden encontrar detalles de esta justificación.

- Conmutatividad: la demostración de $P + Q = Q + P$ sólo tiene un caso interesante ya que los demás están por definición: $P \neq Q$.
  Suponiendo que $P = (x_1, y_1)$, $Q = (x_2, y_2)$ y $P+Q = (x_3, y_3)$, entonces se tiene que para la suma $P + Q$, $\lambda_1 = \frac{y_2 - y_1}{x_2 - x_1}$ y $x_3 = \frac{{y_2}^2 - 2y_2y_1 + {y_1}^2}{{x_2}^2 - 2x_2x_1 + {x_1}^2} - x_1 - x_2$, pero estos elementos están en Z_p que se sabe es conmutativo, por lo que x_3 se puede reescribir como $x_3 = \frac{{y_1}^2 - 2y_1y_2 + {y_2}^2}{{x_1}^2 - 2x_1x_2 + {x_2}^2} - x_2 - x_1$, y esto último corresponde a la coordenada x_3 pero usando $\lambda_2 = \frac{y_1 - y_2}{x_1 - x_2}$ que es de la suma $Q + P$. Se tiene entonces que $P + Q = (x_3, y)$ y $Q + P = (x_3, y')$.

  Si $y = 0$ entonces $y' = 0$, ya que la primera coordenada es igual.

  En otro caso, se tiene para $P + Q$:

$$
\begin{aligned}
y_3 &= \tfrac{y_2-y_1}{x_2-x_1}(x_1-x_3)-y_1 = \\
&= \tfrac{x_1y_2-x_3y_2-x_1y_1+x_3y_1}{x_2-x_1}-y_1 = \\
&= \tfrac{x_1y_2-x_3y_2-x_1y_1+x_3y_1-x_2y_1+x_1y_1}{x_2-x_1} = \\
&= \tfrac{x_1y_1-x_2y_1+x_3y_1-x_3y_2}{x_2-x_1} = \\
&= \tfrac{y_1(x_1-x_2)+x_3(y_1-y_2)}{x_2-x_1} = \\
&= -\big[\tfrac{y_1(x_1-x_2)+x_3(y_1-y_2)}{x_1-x_2}\big] = \\
&= -\big[\tfrac{y_1(x_1-x_2)}{x_1-x_2}+\tfrac{x_3(y_1-y_2)}{x_1-x_2}\big] = \\
&= -[y_1+x_3\lambda_2]. \text{ (Ec1)}
\end{aligned}
$$

Por otro lado, para la suma $Q+P$ se tiene que:

$$
\begin{aligned}
y_3 &= \tfrac{y_1-y_2}{x_1-x_2}(x_2-x_3)-y_2 = \\
&= \tfrac{x_2y_1-x_3y_1-x_2y_2+x_3y_2}{x_1-x_2}-y_2 = \\
&= \tfrac{x_2y_1-x_3y_1-x_2y_2+x_3y_2-x_1y_2+x_2y_2}{x_1-x_2} = \\
&= \tfrac{x_2y_1-x_3y_1+x_3y_2-x_1y_2}{x_1-x_2} = \\
&= -\big[\tfrac{-x_2y_1+x_3y_1-x_3y_2+x_1y_2}{x_1-x_2}\big] = \\
&= -\big[\tfrac{x_3(y_1-y_2)}{x_1-x_2}+\tfrac{x_1y_2-x_2y_1}{x_1-x_2}\big] = \\
&= -[x_3\lambda_2+\tfrac{x_1y_2-x_2y_1}{x_1-x_2}]. \text{ (Ec2)}
\end{aligned}
$$

Si se realiza la resta entre (Ec1) y (Ec2) se tiene que:

$$\begin{aligned}(\text{Ec1}) - (\text{Ec2}) &= -[x_3\lambda_2 + \tfrac{x_1y_2 - x_2y_1}{x_1 - x_2}] - (-[y_1 + x_3\lambda_2]) = \\ &= -[x_3\lambda_2 + \tfrac{x_1y_2 - x_2y_1}{x_1 - x_2}] + [y_1 + x_3\lambda_2] = \\ &= -x_3\lambda_2 - \tfrac{x_1y_2 - x_2y_1}{x_1 - x_2} + y_1 + x_3\lambda_2 = \\ &= -\tfrac{x_1y_2 - x_2y_1}{x_1 - x_2} + y_1 = \\ &= \tfrac{x_1y_1 - x_2y_1 - x_1y_1 + x_2y_1}{x_1 - x_2} = 0.\end{aligned}$$

Por lo que la segunda coordenada debe ser igual y por lo tanto la operación es conmutativa.

- Asociaividad: en [10] se puede ver una prueba de la asociatividad.

Corolario: con la operación descrita anteriormente, $\{\mathcal{O}\} \cup \{P = (x, y) | P \in E_p(a, b)\ , y = 0\}$ forma un subgrupo de la curva, es decir, un subconjunto de los puntos de la curva que con la operación + y $\mathcal{O}$ como elemento identidad, a su vez es un grupo.

Ejemplo: En la curva $E_{11}(4, 9)$, lo puntos $\mathcal{O}$ y (5,0) forman un subgrupo, a saber que $(5, 0) + (5, 0) = \mathcal{O}$ ya que es su propio inverso.

En este conjunto se puede definir una operación de multiplicación escalar como sigue:

$$kP = M,$$

donde k es un entero mayor a 0, $P \in E_p(a, b)$ y M es la suma de P consigo mismo k-veces.

Se verá que para el criptosistema que se definirá es importante conocer el número de puntos de la curva, y no sólo esto, también es de interés que éste sea un número *grande*.

Para calcular de manera sencilla la cardinalidad de una curva el símbolo de Legendre es una herramienta muy útil ya que nos dice si un entero a es un residuo cuadrático módulo un primo p o no, y este resultado sirve ya que si $x^3 + ax + b$ en es residuo cuadrático módulo p, entonces existe un valor y tal

que $y^2 = x^3 + ax + b$ en Z_p.

Definición: Sea $p > 2$ un primo impar y a un entero. El **símbolo de Legendre** de a en p, denotado como $\binom{a}{p}$, esta definido de la siguiente manera:

$$\binom{a}{p} = \begin{cases} 0 & \text{si } p|a \\ 1 & \text{si } a \text{ es un residuo cuadrático} \quad \text{mód } p \\ -1 & \text{si } a \text{ no es un residuo cuadrático} \quad \text{mód } p \end{cases}$$

Definición: Sea $a \in Z, n \in Z, n \geq 3$ e impar con factorización canónica $n = {p_1}^{e_1}{p_2}^{e_2} \cdots {p_k}^{e_k}$, donde $e_i \in Z,\ i = 1, ..., k$. Entonces el **símbolo de Jacobi** de a en n, denotado como $\binom{a}{n}$, se define como:

$$\binom{a}{n} = \binom{a}{p_1}^{e_1}\binom{a}{p_2}^{e_2} \cdots \binom{a}{p_k}^{e_k}$$

Observación: Si n es primo, entonces el símbolo de Jacobi es justo el símbolo de Legendre, pues la factorización de p es $p^1 = p$ y el símbolo de Jacobi de a en p es

$$\binom{a}{p}.$$

De este punto en adelante, p y q representan números primos impares mayores que 3.

Teorema 2.1.1 *El símbolo de Legendre se calcula como:*

$$\binom{a}{p} \equiv a^{\frac{p-1}{2}} \quad \text{mód } p, \text{ con } a \text{ en } Z_p.$$

Demostración: Si $a = 0$ ó $a = kp$, con $k \in Z$ entonces se tiene:

$$\binom{0}{p} \equiv 0^{\frac{p-1}{2}} \quad \text{mód } p \equiv 0 \quad \text{mód } p$$
$$\binom{kp}{p} \equiv kp^{\frac{p-1}{2}} \quad \text{mód } p \equiv 0 \quad \text{mód } p.$$

Por otro lado, el teorema de Fermat nos dice que para $a \in Z$ se tiene que:

$$1 \quad \equiv \quad a^{p-1} \quad \text{mód } p,$$

que se puede reescribir como

$$0 \equiv a^{p-1} - 1 \mod p.$$

como p es impar entonces se tiene que $p-1$ es par, por lo que podemos factorizar a $a^{p-1} - 1$ como

$$0 \equiv (a^{\frac{p-1}{2}} + 1)(a^{\frac{p-1}{2}} - 1) \mod p,$$

ya que $\frac{p-1}{2}$ esta bien definido. Esto implica que p debe dividir a $(a^{\frac{p-1}{2}} + 1)$ o a $(a^{\frac{p-1}{2}} - 1)$, pero no ambos ya que de ser así p dividiría a la diferencia que es 2, por lo tanto se tiene que

$$\begin{aligned} 1 &\equiv a^{\frac{p-1}{2}} \mod p, \text{ o} \\ -1 &\equiv a^{\frac{p-1}{2}} \mod p. \end{aligned}$$

Por otro lado si a es un residuo cuadrático entonces existe una x tal que $x^2 \equiv a$ mód p de donde:

$$\begin{aligned} a^{\frac{p-1}{2}} \mod p &\equiv (x^2)^{\frac{p-1}{2}} \mod p \equiv \\ &\equiv x^{p-1} \mod p \equiv \\ &\equiv 1 \mod p. \end{aligned}$$

es decir, los residuos cuadráticos son soluciones a la congruencia

$$1 \equiv a^{\frac{p-1}{2}} \mod p.$$

Para finalizar sabemos que la congruencia anterior no tiene más soluciones ya que el grado es $\frac{p-1}{2}$ y en Z_p hay exactamente $\frac{p-1}{2}$ residuos cuadráticos [9]. Por lo tanto, los residuos no cuadráticos son soluciones a la congruencia

$$-1 \equiv a^{\frac{p-1}{2}} \mod p. \qquad \square$$

Con el símbolo de Legendre podemos verificar qué valores de Z_p son residuos cuadráticos y así calcular el orden de la curva, esto se formaliza en el siguiente teorema.

Teorema 2.1.2 *Sea $E_p(a, b)$ una curva elíptica sobre Z_p y sea h el orden o cardinalidad de $E_p(a, b)$. Entonces h se calcula como sigue*

$$h = |E_p(a, b)\ | = 1 + \sum_{x=0}^{p-1}(\left(\frac{x^3 + ax + b}{p}\right) + 1),$$

donde $\binom{*}{*}$ es el símbolo de Legendre.

Demostración: Para toda curva $E_p(a,b)$ se tiene el punto especial $\mathcal{O}$, por lo que al menos el orden para toda curva es 1. Ahora, para cada $x \in Z_p$ hay que revisar si existe solución a la ecuación (2.1), para esto sólo se necesita revisar si $z = x^3+ax+b$ es residuo cuadrático módulo p, de serlo entonces existe un valor y que soluciona la ecuación y por tanto (x,y) y $(x,-y \mod p)$ son puntos de la curva. Usando el símbolo de Legendre de $\binom{z}{p}$ se tienen tres posibilidades:

- $\binom{z}{p} = 1$: z es residuo cuadrático módulo p, entonces para la ecuación se tienen dos soluciones módulo p, (x,y) y $(x,-y)$.
- $\binom{z}{p} = 0$: z es un múltiplo de p, la ecuación sólo tiene una solución, $(x,0)$.
- $\binom{z}{p}$ = -1: z no es un residuo cuadrático módulo p, por lo que para x no hay y tal que (x,y) sea solución.

Estimando el número de puntos de $E_p(a,b)$ con lo anterior se tiene que si todos los enteros en Z_p fueran residuos cuadráticos se tendrían a lo más $2p$ puntos de la forma (x,y) más el punto al infinito, es decir, un limite superior sería:

$$h = |E_p(a,b)\ | \leq 2p+1 \qquad \square$$

El siguiente teorema presenta un refinamiento de esta cota y se le conoce como cota de *Hasse*.

Teorema 2.1.3 (Hasse) *Sea $E_p(a,b)$ una curva elíptica, entonces*

$$h = |E_p(a,b)\ | = p+1-t, \text{ con } |t| \leq \lfloor 2\sqrt{p} \rfloor.$$

La demostración de este teorema queda fuera del alcance de esta tesis, pero se puede encontrar en [13].

Definición: Diremos que la curva $E_p(a,b)$ es maximal si cumple con la siguiente igualdad

$$h = p+1+\lfloor 2\sqrt{p} \rfloor.$$

Definición: Diremos que la curva $E_p(a,b)$ es minimal si cumple que

$$h = p+1-\lfloor 2\sqrt{p} \rfloor.$$

Ejemplo: Para la curva $E_{11}(4,9)$, $h = 18$ y por el teorema (2.1.3) se tiene que $|t| \leq 2\sqrt{11} \cong 2 * 3{,}3 \cong 6{,}6$, por otro lado $18 = p+1-t = 11+1-t = 12-t$ lo que implica que $t = -6$, entonces tenemos que no sólo cumple el teorema, sino que se trata de una curva con el mayor número posible de puntos con $p = 11$. La siguiente tabla muestra algunas de las curvas con $p = 11$ y su orden.

Curva	Orden
$E_{11}(0,1)$	12
$E_{11}(1,0)$	12
$E_{11}(1,1)$	14
$E_{11}(1,3)$	18
$E_{11}(1,8)$	6
$E_{11}(4,9)$	18
$E_{11}(5,7)$	16
$E_{11}(9,0)$	12
$E_{11}(9,4)$	18
$E_{11}(10,10)$	14

Cuadro 2.1: Muestra de curvas sobre Z_{11} y su orden.

Teorema 2.1.4 *Sea $E_p(a,b)$ una curva elíptica, entonces para todo $P \in E_p(a,b)$ y todo $k \in Z$ se tiene lo siguiente*

$$(k \ * \ h) * P = \mathcal{O}.$$

Demostración: h es el orden de la curva, por lo que $h * P = \mathcal{O}$, por lo tanto

$$(k * h) * P = k * (h * P) = k * \mathcal{O} = \mathcal{O}. \qquad \square$$

Es de interés criptográfico tener un espacio de mensajes igual a Z_p o Z_n con n un número no necesariamente primo en lugar de $E_p(a,b)$, ya que de otro modo se puede dar una relación explícita entre el espacio de los mensajes y los puntos de la curva. Los detalles de como un mensaje se representa como puntos de la curva se ven en el capítulo 5.

Sea $E_p(a,b)$ una curva elíptica definida por la ecuación (2.1), definiremos una estructura de grupo sobre un subconjunto de $Z_p \times \{u\sqrt{v}, u \in Z_p\}$ con v fijo y además no residuo cuadrático módulo p.

Definición: Sea $p > 3$ un primo y $a, b \in Z_p$. v es un no residuo cuadrático fijo módulo p, entonces $\overline{E_p(a,b)}_v$ es el conjunto de todas las parejas $(x, y) \in Z_p \times Z_p$ donde $y = u\sqrt{v} \pmod p$, $u \in Z_p$ que satisfacen

$$y^2 \equiv x^3 + ax + b \ (mod\ p) \tag{2.2}$$

junto con un punto especial $\mathcal{O}$.

A $\overline{E_p(a,b)}_v$ se le llama **curva complementaria** de $E_p(a,b)$ con respecto de v (definido por Demytko [3]).

Teorema 2.1.5 *Sea $E_p(a,b)$ una curva elíptica y $\overline{h}$ que denota al orden o cardinalidad de $\overline{E_p(a,b)}$, entonces:*

$$\overline{h} = |\overline{E_p(a,b)}_v| \ = 1 + \textstyle\sum_{x=0}^{p-1}(1 - \left(\dfrac{x^3 + ax + b}{p}\right)),$$

donde $\left(\frac{*}{*}\right)$ es el símbolo de Legendre.

Demostración: Para toda curva $\overline{E_p(a,b)}_v$ se tiene el punto especial $\mathcal{O}$, por lo que al menos el orden es 1, ahora, para cada $x \in Z_p$ hay que revisar si existe solución a la ecuación (2.2), para esto sólo se necesita revisar si $z = x^3 + ax + b$ no es residuo cuadrático módulo p. Usando el símbolo de Legendre de $\left(\frac{z}{p}\right)$ se tienen tres posibilidades:

- $\left(\frac{z}{p}\right) = 1$: z es residuo cuadrático módulo p, entonces la ecuación no tiene soluciones módulo p, ya que nos estamos fijando en $y = u\sqrt{v}$.
- $\left(\frac{z}{p}\right) = 0$: z es un múltiplo de p, la ecuación sólo tiene una solución, $(x, 0)$.
- $\left(\frac{z}{p}\right) =$ -1: z no es un residuo cuadrático módulo p, entonces para la ecuación se tienen dos soluciones módulo p, (x, y) y $(x, -y)$.

Teorema 2.1.6 *Sea la curva $E_p(a,b)$ y $\overline{E_p(a,b)}$ su curva complementaria, con $h = |\ E_p(a,b)\ |$ y $\overline{h} = |\overline{E_p(a,b)}|$, entonces*

$$h + \overline{h} = 2p + 2$$

Demostración: De los teoremas (2.1.2) y (2.1.5) tenemos lo siguiente:

$$\begin{aligned} h+\overline{h} &= 1+\textstyle\sum_{x=0}^{p-1}(\left(\frac{x^3+ax+b}{p}\right)+1)+1+\sum_{x=0}^{p-1}(1-\left(\frac{x^3+ax+b}{p}\right)) \\ &= 2+\textstyle\sum_{x=0}^{p-1}2 \\ &= 2+2p.\square \end{aligned}$$

Del teorema 2.1.6 se observa que si la curva $E_p(a,b)$ es maximal, entonces la curva complementaria relacionada a esta es minimal.

Teorema 2.1.7 *Sea un primo p fijo y sea n el número de curvas maximales sobre p, entonces existen n curvas complementarias minimales. Análogamente si hay k curvas minimales sobre p entonces hay k curvas complementarias maximales.*

Demostración: Sea $E_p(a,b)$ una curva y sea $\overline{E_p(a,b)}$ su curva complementaria. Si $E_p(a,b)$ es maximal entonces

$$h = p+1+\lfloor 2\sqrt{p}\rfloor,$$

y por el teorema 2.1.6 se tiene que

$$\overline{h} = 2p+2-h = 2p+2-(p+1+\lfloor 2\sqrt{p}\rfloor) = p+1-\lfloor 2\sqrt{p}\rfloor,$$

y por lo tanto, $\overline{E_p(a,b)}$ es minimal. En caso de que $E_p(a,b)$ sea minimal es análogo. $\square$

2.2. Curvas elípticas sobre Z_n

Sean n un entero positivo mayor que 3 y $a, b \in Z_n$ constantes tales que $4a^3+27b^2 \not\equiv 0 \pmod{n}$. Se define a $E_n(a,b)$ como el conjunto de soluciones $(x,y) \in Z_n \times Z_n$ a la congruencia

$$y^2 \equiv x^3+ax+b \pmod{n}$$

Se puede definir una operación como la de $E_p(a,b)$, pero se requiere de una multiplicación de inversos para el cálculo de λ y los inversos multiplicativos no siempre están definidos sobre Z_n.

Supongamos que $n = pq$, con p y q primos. Sea

$$\tilde{E}_n(a,b) := E_p(a,b) \times E_q(a,b)$$

"Se tiene que $\tilde{E}_n(a,b)$ es un grupo ya que es un producto cartesiano de grupos.

Utilizando el teorema chino del residuo cada elemento x de Z_n puede ser representado por una única pareja $[x_p,x_q]$, donde $x \equiv x_p \bmod p$ y $x \equiv x_q \bmod q$. Por lo que cada punto $P = (x,y) \in E_n(a,b)$ puede ser representado por una única pareja $[P_p, P_q] = [(x_p, y_p), (x_q, y_q)]$, con $P_p \in E_p(a,b)$ y $P_q \in E_q(a,b)$. Así que cada elemento de $E_n(a,b)$ corresponde a un único elemento de $\tilde{E}_n(a,b)$. Por otro lado cada elemento de $\tilde{E}_n(a,b)$ corresponde a un único elemento de $E_n(a,b)$, excepto por los puntos (P,Q) de $\tilde{E}_n(a,b)$ donde $P = \mathcal{O}_p$ ó $Q = \mathcal{O}_q$, pero no ambos. Entonces podemos caracterizar al conjunto $\tilde{E}_n(a,b)$ de la siguiente forma

$$\tilde{E}_n(a,b) = E_n(a,b) \cup \{[P_p, \mathcal{O}] : P_p \in E_p(a,b)\} \cup \{[\mathcal{O}, P_q] : P_q \in E_q(a,b)\}$$

Si la operación de grupo en $E_n(a,b)$ está definida, ésta coincide con la operación en $\tilde{E}_n(a,b)$", ver [1].

En este nuevo conjunto la multiplicación escalar se define como:

$$k[P_p, P_q] = [kP_p, kP_q],$$

con $k \in Z$.

Teorema 2.2.1 *Sea $n = pq$, con p y q primos y sean las curvas $E_p(a,b)$ y $E_q(a,b)$ tales que $h_p = \mid E_p(a,b) \mid$, $h_q = \mid E_q(a,b) \mid$ y $N_n = mcm(h_p, h_q)$[1]. Entonces para todo $P \in E_n(a,b)$ y $k \in Z$*

$$(kN_n \ + \ 1)P = P \ en \ E_n(a,b).$$

Demostración: Sea $P = (x,y) \in E_n(a,b)$ y $k \in Z$. Reescribimos a N_n como:

$$N_n = \alpha \ * \ h_p = h_q \ * \ \beta,$$

ya que N_n es $mcm(h_p, h_q)$. Definimos $P_p = (x \mod p, y \mod p)$ y $P_q = (x \mod q, y \mod q)$, luego usando el teorema (2.1.4):

$$\begin{aligned} (k \ * \ N_n \ + \ 1)P_p &= (k * \alpha * h_p + 1)P_p \\ &= (k * \alpha * h_p)P_p + P_p \\ &= \mathcal{O} + P_p \\ &= P_p \end{aligned}$$

[1]mcm denota al mínimo común múltiplo

Análogamente para P_q con $E_q(a,b)$ se tendrá que:

$$(k * N_n + 1)P_q = P_q.$$

Entonces $(k * N_n + 1)P = [P_p, P_q] = P$ $\square$.

El teorema anterior se puede extender como sigue [1]:

Teorema 2.2.2 *Sea $n = pq$, con p y q primos. Sean $a, b \in Z_n$ con $4a^3+27b^2 \not\equiv 0$ mod n. Sea v un residuo no cuadrático módulo p, y w un residuo no cuadrático módulo q. Sea E_n^c la curva compuesta entre $E_1 \in \{E_p(a,b), \overline{E_p(a,b)_v}\}$ y $E_2 \in \{E_q(a,b), \overline{E_q(a,b)_w}\}$. Sea $N_n = mcm(h_1, h_2)$, con $h_1 = |E_1|$ y $h_2 = |E_2|$. Entonces para todo $P \in E_n^c$ y $k \in Z$*

$$(kN_n + 1)P = P \text{ en } E_n^c,$$

si la multiplicación está definida.

Demostración: Sea $P = (x, y) \in E_n(a, b)$ y $k \in Z$. Reescribimos N_n como:

$$N_n = \alpha * h_1 = h_2 * \beta.$$

Aquí se tienen distintos casos:

- $E_1 = E_p(a,b)$, $E_2 = E_q(a,b)$, definimos $P_p = (x \bmod p, y \bmod p)$ y $P_q = (x \bmod q, y \bmod q)$, luego usando el teorema (2.1.4) se tiene:

$$\begin{aligned}(k * N_n + 1)P_p &= (k * \alpha * h_1 + 1)P_p \\ &= (k * \alpha * h_1)P_p + P_p \\ &= \mathcal{O} + P_p \\ &= P_p\end{aligned}$$

  Análogamente para P_q. Entonces $(k * N_n + 1)P = [P_p, P_q] = P$.

De la elección de E_1 y E_2 se obtienen otros 3 casos:

- $E_1 = E_p(a,b)$, $E_2 = \overline{E_q(a,b)_w}$,
- $E_1 = \overline{E_p(a,b)_v}$, $E_2 = E_q(a,b)$,
- $E_1 = \overline{E_p(a,b)_v}$, $E_2 = \overline{E_q(a,b)_w}$,

los cuales serán análogos al primero, ya que son grupos y al multiplicar por el orden siempre da el elemento identidad, además de que si

$$|E_p(a,b)| = 1 + p + t \text{ entonces } |\overline{E_p(a,b)_v}| = 1 + p - t,$$

y si

$$|E_q(a,b)| = 1 + q + r \text{ entonces } |\overline{E_q(a,b)_v}| = 1 + q - r. \quad \square$$

3
Criptosistema

"Toda tecnología lo suficientemente avanzada es indistinguible de la magia."
Peligros de la profecía: la falta de imaginación, Arthur C. Clarke.

En este capítulo se revisan algunos conceptos y la definición de dos criptosistemas: RSA y su análogo propuesto por Demytko.

El concepto de *criptografía de llave pública* fue introducido por Whitfield Diffie y Martin Hellman [5]. Este concepto se basa en que las llaves de un criptosistema son diferentes: una llave es pública mientras que la otra llave es privada y sólo el propietario (alguno de los participantes en el intercambio de *mensajes*) tiene acceso a la llave privada. Otra característica es que la llave privada debe ser difícil de obtener a partir de la llave pública.

Se conocen pocos critosistemas que sean seguros y prácticos, en términos de la implementación y el tiempo de cifrado/descifrado. Los criptosistemas están basados generalmente en un problema difícil en el contexto matemático y computacional. De los criptosistemas de llave pública que son seguros y prácticos destacan tres por su versatilidad para cifrar y generar firmas digitales: RSA, ElGamal y Rabin.

La seguridad que ofrecen estos sistemas es relativa, ya que si se conoce el espacio de mensajes y la función de cifrado f se puede construir una tabla de dos columnas que cada renglón contenga en el primer registro posible mensaje x y en el segundo el resultado de cifrar x, es decir $f(x)$, para todos los posibles mensajes. Teniendo esta tabla se puede romper el sistema en cuestión simplemente haciendo una búsqueda de diccionario (la tabla resulta ser el diccionario). Si se quiere saber que mensaje origino a y sólo se debe buscar dentro de la segunda columna de la tabla el valor y, el valor en la primer columna de ese renglón corresponde a el mensaje original.

Al proceso de lograr obtener sistemáticamente los *mensajes* originales a partir de los $cifrados$ por un criptosistema se le llama *romper* el criptosistema.

3.1. Criptosistema RSA

El criptosistema RSA, nombrado así por sus inventores Ron Rivest, Adi Shamir y Leonard Adleman, es tal vez el más popular y desde su publicación en 1977. Ha soportado años de criptoanálisis sin que los criptoanalistas hayan probado que es o no seguro, en términos de qué tan fácil/rápido se puede recuperar el mensaje original sin conocer la llave privada, lo que sugiere un grado de confianza al cirptosistema.

Para generar las dos llaves que se necesitan en este criptosistema, se escogen dos números primos distintos p y q mayores que 2, para mayor seguridad que sean grandes en catidad de dígitos y que no sean cercanos, es decir que la resta $|p-q|$ no sea tan pequeña [1]. Despúes se calcula el producto:

$$n = pq.$$

El espacio de *mensajes* de RSA es Z_n (el conjunto de enteros en el intervalo $[0, n-1]$). Una vez fijado n, se selecciona la llave de cifrado e tal que e y $(p-1)(q-1)$ sean primos relativos. Finalmente se calcula la llave de descifrado d tal que

$$ed \equiv 1 \mod (p-1)(q-1),$$

en otras palabras, d es el inverso multiplicativo de e módulo $(p-1)(q-1)$

[1] Google, por ejemplo, usa llaves de 256 bits para cifrar mensajes en su servicio de correo "gmail".

$$d \equiv e^{-1} \mod (p-1)(q-1).$$

Esta llave d, se puede calcular usando el algoritmo de Euclides extendido. Hay que notar que d es primo relativo a $(p-1)(q-1)$.

La llave pública es (n, e) mientras que la llave privada es d. Los primos p y q deben permanecer secretos.

Se puede ver que si el criptoanalista logra factorizar n entonces es capaz de romper RSA calculando d. Sin embargo aún no se ha comprobado que si se logra romper RSA por otros medios entonces se puede tener una factorización de n, por ejemplo el ataque de Wiener cuando la llave secreta es pequeña [16].

Teniendo lo anterior, el criptosistema se reduce a lo siguiente: sea m el mensaje que $\mathcal{B}$ mandará a $\mathcal{A}$. Suponiendo que la llave pública de $\mathcal{A}$ ha sido generada y es (n, e), $\mathcal{B}$ debe seguir los siguientes pasos:

1. $\mathcal{B}$ debe obtener la llave pública de $\mathcal{A}$, (n, e).
2. $\mathcal{B}$ debe representar el mensaje m en el espacio de *mensajes* de RSA.
3. $\mathcal{B}$ debe calcular $c = m^e \mod n$.
4. $\mathcal{B}$ debe enviar c a $\mathcal{A}$.

Para que $\mathcal{A}$ descifre el mensaje c, debe hacer lo siguiente:

1. Usar la llave privada d y calcular $m = c^d \mod n$.

La funcionalidad de RSA se sustenta en lo siguiente:
Dado que $ed \equiv 1 \mod (p-1)(q-1)$, existe un entero k tal que

$$ed = 1 + k(p-1)(q-1).$$

Ahora, si m y p son primos relativos, entonces por el teorema de Fermat se tiene que

$$m^{p-1} \equiv 1 \mod p.$$

Si elevamos ambos lados de la congruencia a la $k(q-1)$ y luego multiplicamos ambos lados por m obtenemos

$$m^{1+k(p-1)(q-1)} \equiv m \mod p.$$

Se puede reescribir a $m^{1+k(p-1)(q-1)}$ como sigue:

$$m^{ed} \equiv m \mod p.$$

Análogamente para q:

$$m^{ed} \equiv m \mod q.$$

Finalmente, como p y q son primos distintos, del teorema chino del residuo se sigue que:

$$m^{ed} \equiv m \mod n,$$

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \mod n. \quad \square$$

RSA se puede usar tanto para cifrar *mensajes* como para firmar *mensajes*. A continuación se ve cómo es el esquema para firmas.

Supongamos que $\mathcal{A}$ quiere mandar un *mensaje* firmado a $\mathcal{B}$, nuevamente $\mathcal{A}$ tiene su llave privada y $\mathcal{B}$ tiene la llave pública de$\mathcal{A}$. Supongamos que el *mensaje* que $\mathcal{A}$ desea enviar es m y suponiendo que la llave pública de $\mathcal{A}$ ha sido generada y es (n, e) mientras que la privada es (n, d)

1. $\mathcal{A}$ calcula $f = m^d \mod n$, que es el cifrado de m usando la llave privada d.
2. $\mathcal{A}$ envía la pareja (m, f) a través del canal a $\mathcal{B}$.
3. $\mathcal{B}$ cifra la firma f usando la llave pública e obteniendo $f' = f^e \mod n$.
4. $\mathcal{B}$ compara f' y m, si son iguales entonces puede asegurar que nadie modificó el *mensaje* que $\mathcal{A}$ envió.

3.2. Criptosistema de Demytko

Demytko [3] introduce un criptosistema basado en el uso de curvas elípticas sobre Z_n que es análogo al criptosistema RSA.

Se seleccionan dos números primos distintos, p y q, de la misma forma que en el criptosistema RSA y se calcula $n = pq$, luego se seleccionan los valores a y b en Z_n tales que $4a^3 + 27b^2 \not\equiv 0 \mod n$.

Sean

$$N_1 = h_1 = |\underline{E_p(a,b)}| = 1 + p + \alpha,$$
$$N_2 = h_2 = |\overline{E_p(a,b)}| = 1 + p - \alpha,$$
$$N_3 = h_3 = |\underline{E_q(a,b)}| = 1 + q + \beta,$$
$$N_4 = h_4 = |\overline{E_q(a,b)}| = 1 + p - \beta.$$

Donde $\overline{E_p(a,b)}$ y $\overline{E_q(a,b)}$ denotan al grupo complementario de las curvas $E_p(a,b)$ y $E_q(a,b)$ correspondientemente, $\alpha \in [0, \lfloor 2\sqrt{p} \rfloor]$ y $\beta \in [0, \lfloor 2\sqrt{q} \rfloor]$, estas últimas 2 por el teorema 2.1.3.

La regla de cifrado está definida por

$$(s,t) = e * (m,y) \bmod n.$$

Donde m es el *mensaje* representado por la coordenada de un punto de la curva, así s denota al *mensaje cifrado* y $e * (m,y)$ denota al punto $P = (m,y)$ (en la curva correspondiente sobre Z_n) multiplicado por e, recordando que la multiplicación de un punto P por i está definido como la suma de P consigo mismo i veces.

La regla de descifrado define como:

$$(m,y) = d_i * (s,t) \bmod n.$$

Donde

$$\begin{gathered} e \in Z_n \text{ con } mcd(e, N_i) = 1, \text{ para } i = 1,2,3,4, \\ \text{y } d_i, \text{ con } i = 1,2,3,4 \text{ tal que} \\ ed_1 \equiv 1 \mod mcm(N_1, N_3), \\ ed_2 \equiv 1 \mod mcm(N_1, N_4), \\ ed_3 \equiv 1 \mod mcm(N_2, N_3), \\ ed_4 \equiv 1 \mod mcm(N_2, N_4). \end{gathered}$$

El índice i se selecciona de acuerdo a lo siguiente:

$$i = \begin{cases} 1 & \text{si } \left(\frac{w}{p}\right) = \left(\frac{w}{q}\right) = 1; \\ 2 & \text{si } \left(\frac{w}{p}\right) = 1, \left(\frac{w}{q}\right) = -1; \\ 3 & \text{si } \left(\frac{w}{p}\right) = -1, \left(\frac{w}{q}\right) = 1; \\ 4 & \text{si } \left(\frac{w}{p}\right) = \left(\frac{w}{q}\right) = -1. \end{cases}$$

Con $\left(\frac{*}{*}\right)$ el símbolo de Legendre y $w = s^3 + as + b \bmod n$.

"Este algoritmo está bien definido ya que,

- Si $\left(\frac{w}{p}\right) = 1$ entonces $s^3 + as + b$ es un residuo cuadrático módulo p, por lo que hay un punto en $E_p(a, b)$ con coordenada $x = s$, el cual es un múltiplo de un punto en $E_p(a, b)$ con coordenada-x igual a m multiplicado por e.
- Si $\left(\frac{w}{p}\right) = -1$, entonces $s^3 + as + b$ no es un residuo cuadrático módulo p, por lo que hay un punto en $\overline{E_p(a, b)}$ con coordenada $x = s$, el cual es un múltiplo de un punto en $\overline{E_p(a, b)}$ con coordenada x igual a m multiplicado por e.
- Si $\left(\frac{w}{q}\right) = 1$ entonces $s^3 + as + b$ es un residuo cuadrático módulo q, por lo que hay un punto en $E_q(a, b)$ con coordenada $x = s$, el cual es un múltiplo de un punto en $E_q(a, b)$ con coordenada-x igual a m multiplicado por e.
- Si $\left(\frac{w}{q}\right) = -1$, entonces $s^3 + as + b$ no es un residuo cuadrático módulo q, por lo que hay un punto en $\overline{E_q(a, b)}$ con coordenada $x = s$, el cual es un múltiplo de un punto en $\overline{E_q(a, b)}$ con coordenada x igual a m multiplicado por e"[1].

Sin pérdida de generalidad tomemos $i = 1$, como

$$ed_1 \equiv 1 \mod (mcm(N_1, N_3)),$$

existe un entero k tal que

$$ed_1 = 1 + k(mcm(N_1, N_3)).$$

Por otro lado tenemos que para un punto $P_1 \in E_p(a, b)$ se tiene:

$$(h_1) * P_1 \quad = \quad \mathcal{O} \text{ en } E_p(a, b) \ ,$$

ya que h_1 es el orden de la curva $E_p(a, b)$. Si multiplicamos ambos lados por $k * mcm(N_1, N_3)/h_1$ tenemos

$$(k * (mcm(N_1, N_3))) * P_1 \quad = \quad \mathcal{O},$$

luego sumando P_1 a cada lado

$$(k * (mcm(N_1, N_3)) + 1) * P_1 \quad = \quad P_1,$$

renombramos $mcm(N_1, N_3)$ como N_n y usando el teorema 2.2.2 tenemos que

$$(k * N_n + 1) * P_1 \quad = \quad P_1 \text{ en } E_n^c.$$

Por otro lado si $P_1 \in E_q(a, b)$ se tiene

$$(h_3) * P_1 \quad = \quad \mathcal{O} \text{ en } E_q(a, b) \ ,$$

ya que h_3 es el orden de la curva $E_q(a, b)$. Si multiplicamos ambos lados por $k * mcm(N_1, N_3)/h_3$ tenemos

$$(k * (mcm(N_1, N_3))) * P_1 \quad = \quad \mathcal{O},$$

luego sumando P_1 a cada lado

$$(k * (mcm(N_1, N_3)) + 1) * P_1 \quad = \quad P_1,$$

renombramos $mcm(N_1, N_3)$ como N_n y usando el teorema 2.2.2 tenemos que

$$(k * N_n + 1) * P_1 \quad = \quad P_1 \text{ en } E_n^c.$$

Análogamente para $i = 2, 3, 4$. □

Observación: en la sección 5 del artículo de Demytko [3] se presenta la siguiente forma (equivalente) de calcular $i * (x, y)$ sobre $E_p(a, b)$ (o $\overline{E_p(a, b)}$).

Sea $(x_i, y_i) \equiv i * (x, y) \mod p$ y $y_i \not\equiv 0 \mod p$, entonces:

$$x_{2i} \equiv \frac{(x_i^2 - a)^2 - 8bx_i}{4(x_i^3 + ax_i + b)} \mod p,$$

Además, si $x \not\equiv 0 \mod p$ y $x_i \not\equiv x_{i+1}$,

$$x_{2i+1} \equiv \frac{(a - x_i x_{i+1})^2 - 4b(x_i + x_{i+1})}{x(x_i - x_{i+1})^2} \mod p,$$

Esta última congruencia se puede reescribir como

$$x_{2i+1} \equiv \frac{4b + 2(a - x_i x_{i+1})(x_i + x_{i+1})}{(x_i - x_{i+1})^2} - x \mod p.$$

La complejidad está en los cálculos módulo $n = pq$, ya que no siempre está definida la división módulo n, lo que puede ocasionar un error durante el proceso de cifrado o descifrado. Para solucionar esto, se reescriben las ecuaciones usando una variable auxiliar Z para evitar la división hasta el último paso del proceso de cifrado o descifrado, de tal forma que al final del proceso Z tenga el acumulado del denominador. Se trata de reescribir como sigue:

$$x \equiv \frac{X}{Z} \bmod p$$

$$y \equiv \frac{Y}{Z} \bmod p.$$

Sea $(x_i, y_i) \equiv (X_i/Z_i, Y_i/Z_i) \equiv i * (X/Z, Y/Z) \bmod n$, entonces las ecuaciones anteriores se pueden reescribir como:

$$X_{2i} \equiv (X_i^2 - aZ_i^2)^2 - 8bX_iZ_i^3 \bmod n$$
$$Z_{2i} \equiv 4Z_i(X_i^3 + aX_iZ_i^2 + bZ_i^3) \bmod n$$
$$X_{2i+1} \equiv$$
$$Z[4bZ_i^2Z_{i+1}^2 + 2(aZ_iZ_{i+1} + X_iX_{i+1})(X_iZ_{i+1} + X_{i+1}Z_i)] - X(X_iZ_{i+1} - X_{i+1}Z_i)^2$$
$$\bmod n$$
$$Z_{2i+1} \equiv Z(X_iZ_{i+1} - X_{i+1}Z_i)^2 \bmod n$$

En términos de esta notación, las reglas de cifrado y descifrado pueden reescribirse de la siguiente forma.

$$s \equiv x_e \equiv X_e/Z_e \bmod n$$
$$x \equiv s_{d_i} \equiv S_{d_i}/Z_{d_i} \bmod n$$

Con $X = x$, $S = s$, $Z = 1$ y e, d_i las llaves de cifrado y descifrado respectivamente.

En resumen, este criptosistema esta basado en el problema de la factorización de $n = pq$, es decir, si se logra factorizar a n el único problema restante es calcular el orden de las curvas $E_p(a, b)$ y $E_q(a, b)$ para poder calcular las llaves de descifrado. En este criptosistema el mensaje se representa como la primera coordenada (o la coordenada x) de un punto $P = (x, y)$ de una curva elíptica, $y^2 \equiv x^3 + ax + b \pmod{n}$, con parámetros fijos a y b.

En el capítulo 6 se presenta un ejemplo del cifrado y descifrado realizado por el programa que se describirá en el capítulo 5.

4
Conceptos en algoritmos

"Contar es la más simple y primitiva de las narrativas
- 1 2 3 4 5 6 7 8 9 10 -
una historia con un principio, un medio y un final."
Drowning By Numbers, Peter Greenaway.

En este capítulo se revisan conceptos necesarios para realizar análisis de algoritmos, con estos conceptos se analizará la implementación que se ofrece en esta tesis de los algoritmos necesarios para el criptosistema de Demytko [3].

Definición: Un problema es una cuestión para la cual se busca una respuesta.

Definición: Los parámetros de un problema son los datos (variables) que este puede contener y que no posean valores específicos al enunciar el problema.

Definición: Un ejemplar de un problema se define como un problema y una asignación de valores específica para sus parámetros.

Definición: Una solución al problema es una respuesta a la cuestión hecha por el problema.

Definición: Un algoritmo es un proceso que toma un dato (*entrada*), o conjunto de datos, y mediante una serie de pasos bien definidos obtiene un valor (*salida*), o conjunto de valores, como resultado. Un algoritmo tiene las siguientes características:

1. El número y tipo de datos de *entrada* está bien especificado, así como las condiciones iniciales de estos para obtener una ejecución exitosa del algoritmo. A estas condiciones se les llama *PreCondiciones*.

2. En cada paso del algoritmo no hay ambigedad sobre las acciones a realizar.

3. El algoritmo resolve el problema para el cual fue creado.

4. Se garantiza que para toda *entrada* el algoritmo termina su ejecución después de un número finito de pasos.

5. La *salida* o efecto del algoritmo se puede expresar como una serie de condiciones. A estas condiciones se les llama *PostCondiciones*.

Se puede ver a los algoritmos como herramientas para resolver problemas bien definidos. Estos problemas determinan, en términos generales, la relación deseada que hay entre la *entrada* y la *salida*.

Supongamos que tenemos dos algoritmos A y B que solucionan el problema P, y deseamos saber cual algoritmo es mejor. Se puede hacer la comparación bajo dos conceptos: *tiempo de ejecución* y *espacio en memoria*. El *tiempo de ejecución* se refiere a cuanto tiempo toma el algoritmo en completar una ejecución, desde inicio hasta fin, mientras que *espacio en memoria* se refiere a la cantidad de memoria que se necesita para realizar la ejecución del algoritmo. En esta tesis el enfoque tomado es bajo *tiempo de ejecución*.

4.1. Análisis de un algoritmo

El análisis de un algoritmo (Cormen et al. [2]) implica decir que para cada ejemplar de un problema, el algoritmo tiene tiempo de ejecución $T_A(E)$, donde A es el algorimo que se analiza y E es el ejemplar del problema que resuelve. Ya que el conjunto de todos los ejemplares E es muy grande, se toman familias de ejemplares del mismo tamaño, E es de tamaño n si tiene n datos.

Se define $T_A(n)$ como la complejidad computacional de A para ejemplares de tamaño n. $T_A(n)$ puede ser analizado como una función del peor de los casos de la siguiente manera:.

$$T_A(n) = T_A(E^*), \text{ donde } T_A(E^*) = max\{T_A(E) \text{ tal que } |E| = n\}.$$

De igual forma se puede definir $T_A(n)$ como el análisis del mejor caso

$$T_A(n) = T_A(E^*), \text{ donde } T_A(E^*) = min\{T_A(E) \text{ tal que } |E| = n\}.$$

Definición [2]: Notación-Θ. Dada una función $g(n)$, se denota a $\Theta(g(n))$ como el conjunto

$$\Theta(g(n)) = \{f(n)|\exists c_1, c_2, n_0, \text{ constantes positivas tales que } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ para toda } n \geq n_0\}.$$

Definición [2]: Notación-O. Dada una función $g(n)$, se denota $O(g(n))$ al conjuno de funciones

$$O(g(n)) = \{f(n)|\exists c, n_0, \text{ constantes positivas tales que } 0 \leq f(n) \leq cg(n) \text{ para toda } n \geq n_0\}.$$

Definición [2]: Notación-Ω. Dada una función $g(n)$, se denota a $\Omega(g(n))$ como el conjunto

$$\Omega(g(n)) = \{f(n)|\exists c, n_0, \text{ constantes positivas tales que } 0 \leq cg(n) \leq f(n) \text{ para toda } n \geq n_0\}.$$

Dadas f y g dos funciones,
- f es $O(g)$ si g es una cota superior para f.
- f es $\Omega(g)$ si g es $O(f)$, es decir, g es una cota inferior para f.
- f es $\Theta(g)$ si f es $O(g)$ y $\Omega(g)$.

Con estas notaciones se pueden acotar las funciones de forma asintótica, de forma ajustada o no, por ejemplo, $2n^2 \in O(n^2)$ es una cota ajustada pero $2n \in O(n^2)$ no lo es. Otro ejemplo de esto pero con Ω es el siguiente, $n^2/2 \in \Omega(n^2)$ es justa pero $n^2/2 \in \Omega(n)$ no lo es. Ante esto se definen las siguientes notaciones.

Definición [2]: Notación-o. Dada una función $g(n)$, se denota $o(g(n))$ al conjuno de funciones

$$o(g(n)) = \{f(n)|\forall c, \text{ constante positiva, } \exists n_0, \text{ constante positiva tal que } 0 \leq f(n) < cg(n) \text{ para toda } n \geq n_0\}.$$

Definición [2]: Notación-ω. Dada una función $g(n)$, se denota a $\omega(g(n))$ como el conjunto

$$\omega(g(n)) = \{f(n)|\forall c, \text{ constante positiva } \exists n_0, \text{ constante positiva tal que } 0 \leq cg(n) < f(n) \text{ para toda } n \geq n_0\}.$$

El objetivo detrás de todas estas notaciones es dar cotas asintóticas para la función que describe el tiempo de ejecución del algoritmo.

Ejemplo: "supongamos que $f(n) = an^2 + bn + c$ con $a > 0$, se pueden tomar constantes $c_1 = a/4$, $c_2 = 7a/4$ y $n_0 = 2 * max(|b|/a, \sqrt{|c|/a})$, se puede ver que $0 \leq c_1 n^2 \leq an^2 + bn + c \leq c_2 n^2$ para toda $n \geq n_0$"[2].

El ejemplo anterior se puede generalizar para cualquier polinomio

$$p(n) = \sum_{i=0}^{d} a_i n^i, \text{ con } a_i \text{ constantes y } a_d > 0,$$

entonces se tiene que $p(n)$ es $\Theta(n^d)$. Dado que una constante es un polinomio de grado 0, se puede expresar cualquier constante como $\Theta(n^0)$, o $\Theta(1)$ con respecto de la variable n.

Para acotar asintóticamente el tiempo de ejecución de un algoritmo, $T_A(n)$, sólo importa el orden de la magnitud, ya que, tanto los términos de orden menor y la constante del término de mayor orden pueden ser ignorados en la cota.

Por último sólo falta la forma en que se obtienen estos polinomios a acotar. Dado un algoritmo A y su ejemplar a resolver E de tamaño n, se deben contar las operaciones que realiza A en generar la *salida*. Si A requiere x operaciones para resolver E, se debe buscar una función f tal que $f(n) = x$.

Ejemplo: Se desea conocer el valor que resulta de sumar todos los números enteros en el intervalo [1,n]. Un algoritmo para resolver este problema es el siguiente:

Para calcular $T_{suma}(n)$ que es el tiempo de ejecución de este algoritmo, se tiene que en la primera linea hay una inicialización, que cuenta como una operación ($\Theta(1)$), en la segunda línea se tiene un ciclo, que tiene a cada iteración una

Algoritmo 4.1 *suma*

Entrada: Un entero no negativo n
Salida: La suma de los enteros entre 1 y n

```
res ← 0
para todo i ∈ [1, n] hacer
  res ← res + i
fin para
devolver res
```

asignación para i y dentro de este hay una suma y una asignación, por último se tiene que se regresa el resultado ($\Theta(1)$). Por lo tanto el tiempo de ejecución es $1 + T_{ciclo}(n) + 1$, el ciclo se repite n veces, por lo que $T_{ciclo}(n) = 3n$ (ya que se trata de 3 operaciones, dos asignaciones y una suma repetidas n veces) por lo que $T_{suma}(n) = 1 + 3n + 1 = 3n + 2$. Ahora sólo falta acotar esta función. Sea $g(n) = n$, $c = 4$ y $n_0 = 2$ teniendo $0 \leq 3n + 2 \leq 4g(n)$, $\forall n \geq n_0 = 2$, concluyendo que $3n + 2 \in O(n)$. Análogamente para Ω, concluyendo que este algoritmo tiene orden $\Theta(n)$.

Los órdenes de complejidad clásicos, de menor a mayor, son:

$$\Theta(1),\ \Theta(\log_2 n),\ \Theta(n),\ \Theta(n \log_2 n),\ \Theta(n^2),\ \Theta(2^n),\ \Theta(n!)$$

Por simplicidad se dice que las operaciones aritméticas $+, -, *, /$, la operación módulo, las asignaciones, instrucciones de retorno (*return*) y accesos a localidades dentro de un arreglo son de orden constante, es decir $\Theta(1)$.

5
Implementación

"Todas las obras de arte deben empezar por el final."
Edgar Allan Poe.

En este capítulo se revisan los modelos de curvas elípticas y el criptosistema dentro de la implementación, así como las formas de mapear información dentro del espacio de mensajes y funciones auxiliares.

5.1. Modelo

Los puntos de las curvas estan dados como parejas (x, y) o como $\mathcal{O}$, este será el primer modelo a tratar.

Nombre:	
	ECPoint
Atributos :	
	x:entero
	y:entero
	infinity:booleano
Constructores:	
	ECPoint()
	ECPoint(x,y)

El constructor ECPoint() servirá para crear un nuevo punto al infinito poniendo el valor de $infinity$ en verdadero, mientras que el otro constructor crea un nuevo punto con coordenadas (x, y) poniendo $infinity$ en falso. Las operaciones de este modelo sólo son las operaciones de recuperación y actualización de atributos y comparación con otros objetos de tipo ECPoint.

Ya definido el modelo ECPoint se puede definir el modelo EllipticCurve que es el encargado de representar a las curvas.

Nombre:
EllipticCurve
Atributos :
a:entero
b:entero
p:entero
q:entero
card:entero
cardComplementaria:entero
puntos: lista ligada
Constructores:
EllipticCurve(a,b,p,q)

El constructor EllipticCurve(a,b,p,q) crea la curva elíptica $E_n(a, b)$ con $a, b \in Z_n$ y $n = pq$, si $q = 1$ entonces la curva creada es $E_p(a, b)$ y si $p = 1$ entonces es $E_q(a, b)$. Los atributos a, b, p, q son los parámetros que definen la ecuación $y^3 \equiv x^2 + ax + b \mod (p * q)$. El atributo *card* es el orden de la curva, mientras que *cardComplementaria* (sólo si $p = 1$ ó $q = 1$) es el orden de la curva complementaria. Por último el atributo *puntos* es la lista (explícita) de los puntos de la curva.

Las operaciones de la curva son las de recuperación de datos, calcular órdenes (tanto de la curva como de la complementaria de ser posible), suma de puntos en la curva, la multiplicación de un punto por un entero k (la suma del punto consigo mismo k veces), cálculo de sus puntos mediante dos estrategias: por cálculo de residuos cuadráticos y por cálculo de órbitas de puntos ya conocidos.

Se puede ver otra propuesta de modelo usando C++ en [6].

Por último se necesita modelar el criptosistema.

Nombre:
Demytko
Atributos :
ec:curva elíptica
N1:entero
N2:entero
N3:entero
N4:entero
Constructores:
Demytko(ec)

El constructor Demytko(ec) crea un nuevo criptsistema a partir de la curva elíptica dada. El atributo *ec* es la curva elíptica mientras que Ni con $i = 1, 2, 3, 4$ es el mcm entre los órdenes de las curvas y las curvas complementarias, que servirán para calcular las llaves de descifrado. Las operaciones definidas aquí son de recuperación de datos, cálculo de las Ni, generación de llaves de cifrado y descifrado, y función de cifrado y descifrado.

Las operaciones de estos tres modelos se ven a detalle en la siguiente sección.

5.2. Algoritmos

Sean P y Q dos puntos en $E_p(a, b)$, para realizar la suma $P + Q$ se deben revisar los siguientes casos: alguno de ellos es $\mathcal{O}$, $P = -Q$, $P = Q$, ninguno de los anteriores. La menor cantidad de operaciones necesarias para realizar la suma es 2 y sucede cuando uno de los puntos es $\mathcal{O}$. La mayor cantidad de operaciones necesarias es 16 cuando $P = Q$. Por lo que sumar dos puntos en $E_p(a, b)$ es $\Omega(1)$ y $O(1)$, lo que implica que la suma es $\Theta(1)$.

En $E_p(a, b)$ se define la multiplicación de un punto P por n como la suma de P consigo mismo n veces. Este se muestra en el algoritmo 5.1.

El desempeño computacional de *multiplicar* depende del tamaño de n, los casos base del algoritmo representan el mejor caso de éste, y para ellos se tiene sólo una operación, que es regresar ya sea un error, el punto original o el punto al infinito, teniendo $\Omega(1)$.

Algoritmo 5.1 *multiplicar*

Entrada: Un punto P en la curva y un entero no negativo n
Salida: Un punto R en la curva tal que $R = n * P$

```
si n < 0 entonces
  error
si no, si n = 0 entonces
  devolver O
si no, si n = 1 entonces
  devolver P
si no, si n = 2k entonces
  devolver multiplicar(P + P, n/2)
si no
  devolver P + multiplicar(P + P, n/2)
fin si
```

Si n es par, entonces se tiene una suma entre puntos de la curva, una división y el tiempo que tarde en resolver la llamada recursiva de la función más la instrucción de retorno. El último caso es si n es impar, en este casi se tienen 2 sumas entre puntos de la curva, una división, lo que tarde la llamada recursiva y la instrucción de retorno. Esto se puede expresar mediante la relación siguiente:

$$\begin{aligned} T_{multiplicar}(0) &= 3 \\ T_{multiplicar}(1) &= 4 \\ T_{multiplicar}(n = 2k) &= T_{multiplicar}(k) + 8 \\ T_{multiplicar}(n = 2k + 1) &= T_{multiplicar}(k) + 9 \end{aligned}$$

Como ya se mostró, la suma de dos puntos de la curva, $P + Q$ es $\Theta(1)$. Como cada llamada de la función depende del la mitad del valor del parámetro n de la llamada anterior, entonces se tiene que el algoritmo tiene una complejidad de $O(\log_2 n)$.

Para calcular el orden de una curva usando el teorema (2.2) antes necesitamos poder calcular el símbolo de Legendre, que por el teorema (2.1) se sabe que se calcula usando potencias. Para calcular de forma directa $a^b \mod m$ se necesitan b multiplicaciones y un módulo.

Proposición: Usando las leyes de los exponentes (válidas para curvas elípticas) se puede calcular a^b usando $\log_2 b$ multiplicaciones:

$$a^b = (a^2)^{b/2} \text{ si b es par,}$$

$$a^b = a * (a^2)^{(b-1)/2} \text{ si b es impar.}$$

Algoritmo 5.2 *potencia*

Entrada: Un entero a, un entero no negativo b y un entero m mayor que 2
Salida: $x = a^b \mod m$

```
si b = 0 entonces
  devolver 1
si no, si b = 1 entonces
  devolver x = a mód m
si no, si b = 2k entonces
  devolver potencia((a * a) mód m, b/2, m) mód m
si no
  devolver a * potencia((a * a) mód m, b/2, m) mód m
fin si
```

Esta función depende del tamaño b, que es el número de veces que se elevará el término a módulo m, además de que se revisa si b es de la forma $2k$ ó $2k+1$. Siguiendo un análisis similar al anterior, se tiene lo siguiente:

$$\begin{aligned} T_{potencia}(0) &= 2; \\ T_{potencia}(1) &= 3; \\ T_{potencia}(2k) &= T_{potencia}(k) + 9; \\ T_{potencia}(2k+1) &= T_{potencia}(k) + 10. \end{aligned}$$

Los casos bases no se contemplarán como el mejor caso para el análisis, ya que se espera que b sea al menos 3, por lo que sólo es posible entrar a ellos dentro del mismo algoritmo. Se concluye que *potencia* es $\Omega(\log_2 b)$ y $O(\log_2 b)$, lo que implica que es $\Theta(\log_2 b)$.

Con el algoritmo *potencia* ya definido, se puede difinir *legendre* en el algoritmo 5.3. Esta función depende de *potencia*, por lo que tiene desempeño:

$$T_{legendre}(p) = T_{potencia}(p).$$

Teniendo como orden de complejidad $\Theta(\log_2 p)$.

Con el algoritmo 5.3 se puede definir la función para calcular la cardinalidad, o el orden, de la curva $E_p(a, b)$. Esta misma función se usará para calcular la cardinalidad de la curva complementaria de $E_p(a, b)$, $\overline{E_p(a, b)}$.

Algoritmo 5.3 *legendre*

Entrada: Un entero no negativo n y un primo p mayor que 2
Salida: El símbolo de legendre $\binom{n}{p}$

```
si p ≤ 2 o p no es un primo entonces
  error
si no
  leg ← potencia(n, (p − 1)/2, p)
  si leg = p − 1 entonces
    devolver -1
  si no
    devolver leg
  fin si
fin si
```

Algoritmo 5.4 *cardinalidad*

Entrada: Los parámetros a, b, p de la curva $E_p(a, b)$
Salida: $h = |E_p(a, b)\ |$

```
card ← 1, z ← 0, leg ← 0
para todo i ∈ [1, p] hacer
  z ← i^3 + ai + b  mód p
  leg ← legendre(z, p)
  card ← card + leg + 1
fin para
devolver card
```

Esta función tiene tres inicializaciones, luego un ciclo que se realizará p veces y el regreso del valor calculado. El ciclo tiene a su vez tres asignaciones a variables locales, también tiene tres multiplicaciones, cuatro sumas y una reducción modular, además, en cada una de las p iteraciones del ciclo se calculará el símbolo de Legendre. Se puede escribir la función que caracteriza al tiempo de ejecución de *cardinalidad* como:

$$\begin{aligned} T_{cardinalidad}(E_p(a,b)\ ) &= 3 + T_{ciclo}(p) + 1 = \\ &= 3 + [7p + p(\log_2 p + 1) + 3p] + 1 = \\ &= 11p + p(\log_2 p) + 4. \end{aligned}$$

El término mayor es $p(\log_2 p)$, por lo que se tiene que este algoritmo tiene orden $O(p(\log_2 p))$ y $\Omega(p(\log_2 p))$, ya que no importa el orden de la curva pues se verifican todos los números entre 1 y p, por lo que se tiene un orden de $\Theta(p(\log_2 p))$.

Para calcular la cardinalidad de la curva complementaria de $E_p(a,b)$, es decir $\overline{E_p(a,b)}$, el algoritmo es el siguiente:

Algoritmo 5.5 *cardinalidadComp*

Entrada: Los parámetros a, b, p de la curva $\overline{E_p(a,b)}$
Salida: $\overline{h} = \overline{E_p(a,b)}$
  $card \leftarrow 1, z \leftarrow 0, leg \leftarrow 0$
  para todo $i \in [1,p]$ **hacer**
    $z \leftarrow i^3 + ai + b \mod p$
    $leg \leftarrow legendre(z,p)$
    $card \leftarrow card + (1 - leg)$
  fin para
  devolver $card$

El análisis de *cardinalidadComp* es análogo al del algoritmo 5.4 concluyendo en $\Theta(p \log_2 p)$.

Se puede dar una mejora a este algoritmo como sigue: de los teoremas (2.2), (2.3) y (2.5) se tiene que $h = 1+p-t$, entonces $\overline{h} = 1+p+t$, y si ya se conoce h (usando el algoritmo anterior), entonces calcular $\overline{h}$ se reduce a encontrar el valor de t y sustituir en $\overline{h} = 1 + p + t$, teniendo complejidad $\Theta(1)$.

Algoritmo 5.6 *cardinalidadFuerte*

Entrada: El parámetro p de la curva $E_p(a,b)$ y k, que es el valor de h o el de $\overline{h}$ (ya que se trata de calcular un valor a partir del otro)
Salida: $\overline{h}$ si la entrada fue h, $\overline{h}$ en otro caso
  $tmp \leftarrow k - (p+1)$
  si $tmp < 0$ **entonces**
    devolver $(p+1) + (-1) * tmp$
  si no
    devolver $(p+1) - tmp$
  fin si

Ahora que se sabe el orden de la curva $E_p(a,b)$, se pueden calcular sus puntos.

El algoritmo, *puntosECRC*, calculará los puntos de la curva verificando que las clases residuales que se tengan sean residuos cuadráticos y si lo son, se obtendrá el punto correspondiente.

Algoritmo 5.7 *puntosECRC*

Entrada: Los parámetros a, b, p de la curva $E_p(a, b)$
Salida: La lista L que contiene a los puntos de la curva $E_p(a, b)$

```
Lista L, z ← 0, leg ← 0, Arreglo A de tamaño (p + 1)/2
L.agregar(O)
para todo i ∈ [0, (p − 1)/2] hacer
  A[i] ← i^2 mód p
fin para
para todo i ∈ [0, p − 1] hacer
  z ← i^3 + ai + b mód p
  leg ← legendre(z)
  si leg ≠ −1 entonces
    para todo j ∈ [0, (p + 1)/2] hacer
      si A[j] = z entonces
        L.agregar((i, j))
        si A[j] ≠ 0 entonces
          L.agregar((i, −j mód p))
        fin si
      fin si
    fin para
  fin si
fin para
devolver L
```

En esta función se aprovecha la propiedad de que $-k \equiv p-k \mod p$, además de que $k^2 \mod p \equiv (-k)^2 \mod p$. Usando esto se crea un arreglo auxiliar donde se guardan los resultados de los cuadrados de los enteros $0, 1, 2, \cdots, (p-1)/2$. Luego, se calcula $z = i^3 + ai + b \mod p$ para toda $i \in [0, p-1]$ y se compara a los resultados almacenados en el arreglo. En caso de que para un índice j del arreglo se cumple $A[j] = z$, entonces quiere decir que la pareja (i, j) corresponde a un punto de la curva, si además $j \neq 0$ entonces también
$(i, -j \mod p)$ es un punto de la curva.

El método *agregar* de la lista tiene orden $O(n)$, con n el tamaño de la lista,

por lo que en esta función el tamaño máximo de la lista esta acotado por el orden de $E_p(a,b)$, sin embargo, de momento no es de importancia el orden en que son agregados los puntos de la curva, por lo que podemos agregar al inicio de la lista teniendo un orden de $\Theta(1)$.

El primer ciclo se ejecutará $(p-1)/2$ veces, teniendo $\Theta(p)$. El segundo ciclo se ejecutará p veces, lo que implica que se tiene un orden de $\Theta(p)$ para este ciclo.

Para el calculo de *legendre* se tiene un orden de $\Theta(\log_2 p)$ (algoritmo 5.3).

El último ciclo se ejecutará $(p+1)/2$ veces, teniendo lo siguiente:

$$\begin{aligned} T_{puntosECRC}(E_p(a,b)\ ) &= 4+1+T_{ciclo1}(p)+T_{ciclo2}(p)+1 \\ &= 6+[4((p-1)/2)]+[7p+p((p+1)/2)(5)], \end{aligned}$$

para el peor caso, o se tiene

$$\begin{aligned} T_{puntosECRC}(E_p(a,b)\ ) &= 4+1+T_{ciclo1}(p)+T_{ciclo2}(p)+1 \\ &= 6+[4((p-1)/2)]+[7p+p((p+1)/2)], \end{aligned}$$

para el mejor caso.

Se puede ver que no importa que se trate del mejor o peor caso, el término de mayor magnitud es cuadrático, por lo que podemos concluir que este algoritmo tiene orden $O(p^2)$ y $\Omega(p^2)$ lo que implica que todo el algoritmo es de orden $\Theta(p^2)$.

Por último, falta el algoritmo que suma un punto de una curva consigo mismo k-veces ("múltiplicar"k por el punto).

Para el siguiente algoritmo se usan las ecuaciones definidas en la sección 5 del artículo de Demytko [3]. La forma en que están escritas las ecuaciones (usando coordenadas homogéneas) es recursiva y si se intenta hacer una implementación directa sobre estas, se pueden generar errores muy fácilmente o se puede gastar mucha memoria y hacer cálculos que ya se habían realizado antes (en la misma ejecución), por lo que, en esta implementación, se optó por un algoritmo iterativo.

La idea es calcular los índices necesarios de las coordenadas a calcular y sólo calcularlas una vez y almacenarlas para usarlas despúes en el cálculo final.

Algoritmo 5.8 *multXK*

Entrada: La coordenada x del punto, P, a multiplicar, un entero positivo k y los parámetros a, b, n de la curva sobre Z_n

Salida: X y Z tales que $X/Z \mod n$ es la coordenada x de $k * P$

```
Lista index, i ← 0
index.agregar(k)
mientras i < tamaño(index) y 1 < index(i) hacer
  nkey ← index(i)
  si nkey = 2t entonces
    nkey2 ← nkey/2
    si index no contiene a nkey2 entonces
      index.agregar(nkey2)
    fin si
  si no
    nkey2 ← (nkey − 1)/2
    si index no contiene a nkey2 + 1 entonces
      index.agregar(nkey2 + 1)
    fin si
    si index no contiene a nkey2 entonces
      index.agregar(nkey2)
    fin si
  fin si
  i ← i + 1
fin mientras
tmp ← tamaño(index), X, Z        // X, Z son arreglos de tamaño tmp
X[0] ← x, Z[0] ← 1

  para todo i ∈ [1, tmp − 1] hacer
    si index(tmp − i − 1) = 2t entonces
```

$X_i \leftarrow 0, Z_i \leftarrow 0$
si $i = 0$ **entonces**
$X_i \leftarrow X[i-1], Z_i \leftarrow Z[i-1]$
si no
$X_i \leftarrow X[i-2], Z_i \leftarrow Z[i-2]$
fin si
$X_{2i} \leftarrow (X_i^2 - aZ_i^2)^2 - 8bX_iZ_i^3 \mod n$
$Z_{2i} \leftarrow 4Z_i(X_i^3 + aX_iZ_i^2 + bZ_i^3) \mod n$
$X[i] = X_{2i}$
$Z[i] \leftarrow Z_{2i}$
si no
$X_i \leftarrow 0,\ Z_i \leftarrow 0,\ X_{i\_1} \leftarrow 0,\ Z_{i\_1} \leftarrow 0$
si $index(tmp - i) = ((index(tmp - i - 1)/2) + 1)$ **entonces**
$X_i \leftarrow X[i-2],\ Z_i \leftarrow Z[i-2]$
$X_{i\_1} \leftarrow X[i-1],\ Z_{i\_1} \leftarrow Z[i-1]$
si no
$X_i \leftarrow X[i-3],\ Z_i \leftarrow Z[i-3]$
$X_{i\_1} \leftarrow X[i-2],\ Z_{i\_1} \leftarrow Z[i-2]$
$X_{2i\_1} \leftarrow Z[0][4bZ_i^2Z_{i\_1}^2 + 2(aZ_iZ_{i\_1} + X_iX_{i\_1})(X_iZ_{i\_1} + X_{i\_1}Z_i)] - X[0](X_iZ_{i\_1} - X_{i\_1}Z_i)^2 \mod n$
$Z_{2i\_1} \leftarrow Z[0](X_iZ_{i\_1} - X_{i\_1}Z_i)^2 \mod n$
$X[i] \leftarrow X_{2i\_1}$
$Z[i] \leftarrow Z_{2i\_1}$
fin si
fin si
fin para
devolver $X[tmp-1]$ **y** $Z[tmp-1]$

Este algoritmo se puede analizar por partes (los dos ciclos). El primer ciclo cálcula los índices de los elementos que se deben calcular para generar $k * P$, la forma de hacer esto es ir dividiendo el valor actual de $nkey$ entre 2 y ver si los resultados de la división ya fueron agregados a la lista lo que implica que este ciclo se repetirá $log_2 k$ veces. Revisar si la lista ya contiene a un elemento es de orden lineal sobre el tamaño de la lista. En la primera iteración se hace a lo más 1 comparación, en la segunda 2 comparaciones, en la j-ésima iteración se hacen a lo más j comparaciones. Lo que termina en la suma:

$$T_{ciclo\_mientras}(k) = 1 + 2 + ... + j = \frac{j(j+1)}{2} \in \Theta(j^2).$$

Ahora sólo falta, en este ciclo, determinar qué tan grande es j. Resulta que $j = \lfloor \log_2 k \rfloor$ ya que en cada iteración sólo se agrega a lo más un elemento a la lista si $nkey$ es par, o a lo más dos elementos si $nkey$ es impar, y como se itera $\log_2 k$ veces entonces la lista crece en razón de $\log_2 k$. Este ciclo es de orden $\Theta((\log_2 k)^2)$.

El ciclo "para todo"se repite $\log_2 k$ ya que tmp es el tamaño de la lista $index$. Dentro de este ciclo sólo hay operaciones aritméticas, accesos a localidades de arreglos y asignaciones, por lo que realiza un número constante de operaciones a cada iteración por lo que el orden del ciclo es $\Theta(log_2 k)$.

Todas las operaciones fuera de los ciclos son de orden $\Theta(1)$ implicando que el orden de todo el algoritmo es $\Theta((\log_2 k)^2)$.

5.3. Otras funciones

Además de implementar el criptosistema también se han implementado otras funciones sobre las curvas elípticas, las cuales permiten:

1. *showEC*: Mostrar una curva elíptica específica, junto con sus propiedades.

2. *report*: Dado un primo p, obtener todas las curvas elípticas sobre Z_p con sus propiedades, además de reportar que $E_p(a, b)$ tiene el mayor orden y cual el menor. Toda esta información es volcada en un archivo de texto.

Algoritmo 5.9 *showEC*

Entrada: Los parámetros a, b, p, q de la curva a mostrar (si existe)
Salida: Se muestran las propiedades de la curva especificada
si la condición $4a^3 + 27b^2 \not\equiv 0 \mod p * q$ no se satisface **entonces**
devolver mensaje de error
fin si
Generar los puntos de la curva con el algoritmo 5.7
imprimir Los datos de la curva, así como h, $\overline{h}$, listas de puntos con sus tiempos de generación en milisegundos

El tiempo de ejecución de *showEC* es el siguiente

$$T_{showEC} = 8 + T_{puntosECRC} + T_{getPointsByOrbits} + T_{mostrar}$$

Diremos, por simplicidad, que $T_{mostrar}$ es de $\Theta(1)$, en la sección anterior se mostró cual es el orden para $T_{puntosECRC}$ y $T_{getPointsByOrbits}$. Por lo que se tiene que T_{showEC} es de orden $\Theta(p^2)$.

Algoritmo 5.10 *report*

Entrada: $p > 3$

Salida: Un archivo con la información de todas las curvas sobre Z_p

```
para todo (a, b) ∈ Z_p × Z_p hacer
  si 4a^3 + 27b^2 ≢ 0  mód p entonces
    Generar los puntos de la curva con el algoritmo 5.7
    Agregar los datos de la curva al archivo (parámetros, h, h̄, lista de
      puntos con sus correspondientes tiempos de generación en milisegundos)
    Agregar los datos de las curvas maximal y minimal por separado,
    junto con el total de curvas y promedios de los tiempos en generar
    las listas de puntos
  fin si
fin para
```

Para la función *report* supondremos que escribir un archivo es de orden $\Theta(1)$. Este algoritmo tiene un doble ciclo para generar las parejas $(a, b) \in Z_p \times Z_p$, el cual es $\Theta(p^2)$, si además dentro de este ciclo se hacen todos los demás cálculos se ve que el algoritmo es de orden $\Theta(p^4)$ ya que $T_{puntosECRC}$ es $\Theta(p^2)$.

5.4. Detalles acerca de la implementación

Algunos detalles importantes a tomar en cuenta al realizar esta implementación son: como interpretar la información en enteros, la forma en que se hace el mapeo de estos enteros a cifrar dentro del espacio de mensajes, y el propio espacio de mensajes (las curvas que se usarán).

5.4.1. Mapeo información-enteros

Toda la información o archivos en formato digital están formados por conjuntos de bytes, cada byte se puede ver como un número en base 2 en el intervalo (decimal) [0,255].

Supongamos que el archivo F es el conjunto de bytes $b_1 b_2 \cdots b_k$, se puede dar un mapeo σ tal que a cada byte b_i le corresponda su valor en decimal e_i, con $i \in [1, k]$. El archivo ahora queda representado como $e_1 e_2 \cdots e_k$. Con estos enteros ya se podría cifrar, sin embargo de esta manera se tiene poca seguridad ya que se tiene el caso en que se puede construir una tabla "diccionario" y romper así el criptosistema.

Supongamos que F es un archivo de texto, o simplemente un texto, al hacer el mapeo anterior se tiene una secuencia de tamaño k que tiene a lo más 26 enteros distintos, lo que al cifrar produce una secuencia que de igual manera esta formada por una combinacin de a lo más 26 enteros distintos. Si se conoce el idioma en el que esta escrito el texto, se puede hacer un análisis de frecuencias sobre el texto cifrado y se podría recuperar el texto original sin necesidad de conocer el criptosistema que se usó para el cifrado.

El mapeo se puede mejorar si en lugar de que a b_i se le asigne su entero e_i, se toman dos bytes $b_i b_{i+1}$ y a partir de ellos usando una funcin invertible se genera un sólo entero $e_{\bar{i}}$. Una forma de lograrlo es, que internamente se trabaje con enteros en el intervalo [0,65535] (el número más grande que se puede representar con 2 bytes es 65536). Escribimos el mapeo de la siguiente forma

$$\sigma(b_i, b_{i+1}) \quad = \quad t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 \quad = \quad e_{\bar{i}},$$

con $t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8$ la expansión en binario de b_i y $r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8$ la expansión de b_{i+1}. Con esta nueva σ se tiene una mayor seguridad que en la anterior, pero sigue siendo débil pues es sensible al mismo tipo de ataque antes mencionado. A pesar de esto, esta es la interpretación de la información que se usa en esta tesis por su simplicidad de implementación. Se mencionan mejoras a ésta σ.

La siguiente mejora se hace al cambiar el orden en que se toman los bytes, es decir, en lugar de calcular $\sigma(b_i, b_{i+1})$ se puede calcular $\sigma(b_i, b_j)$, con j no necesariamente igual a $i + 1$.

Otra forma es hacer que todos los bytes de la información representen a un sólo entero, lo que sería calcular $\sigma(F) = e_F$, para luego sólo cifrar e_F. Este proceso se puede hacer de igual forma al no tomar los bytes en su orden dentro de F.

5.4.2. Mapeo en el espacio de mensajes

Nuevamente, se opta por un mapeo sencillo dentro de la tesis.

Algoritmo 5.11 *mapeoEnteroAMensaje*

Entrada: Los bytes b_i y b_{i+1} a cifrar, y los parámetros de la curva a, b, p, q
Salida: La coordenada x de un punto dentro de la curva, y un valor auxiliar *cont*.

$pt \leftarrow \sigma(b_i, b_{i+1})$, $cont \leftarrow 0$, $jac \leftarrow -1$, $t \leftarrow 0$
$t \leftarrow pt^3 + a * pt + b \pmod{p * q}$
$jac \leftarrow$ símbolo de jacobi de t en $p * q$
mientras $jac \neq 1$ **hacer**
  $cont \leftarrow cont + 1$, $pt \leftarrow pt + 1$
  $t \leftarrow pt^3 + a * pt + b \pmod{p * q}$
  $jac \leftarrow$ símbolo de jacobi de t en $p * q$
fin mientras
devolver *pt* y *cont*

El símbolo de Jacobi, en este caso, se define como la multiplicación del símbolo de Legendre $\binom{x}{p}$ con el símbolo de Legendre $\binom{x}{q}$. Aquí se usa para saber si *pt* es una coordenada x válida dentro de la curva.

Todas las instrucciones fuera del ciclo "mientras"son de orden $\Theta(1)$ y la función σ puede ser implementada usando operadores a nivel de bit (corrimiento a la izquierda y operación OR). El cálculo del símbolo de Legendre es de orden $\Theta(\log_2 p)$ y $\Theta(\log_2 q)$, para p y q correspondientemente. Por último, el ciclo "mientras"se repetirá pocas veces, pues se tiene que en Z_p hay $(p-1)/2$ residuos cuadráticos (y $(q-1)/2$ en Z_q), por lo que hay una probabilidad de 1/4 de que *pt* sea un residuo cuadrático módulo $p*q$ (si se toma a *pt* de forma aleatoria). Dentro del mismo ciclo hay más asignaciones y nuevamente el cálculo del símbolo de Jacobi. Así, el tiempo de ejecución de este mapeo es de orden $\Theta(max\{\log_2 p, \log_2 q\})$.

Koblitz [8], describe otra forma de representar a los mensajes como puntos dentro de una curva elíptica.

Dentro del programa los resultados de los cifrados son coordenadas de puntos de la curva, separadores para distinguir un punto del valor auxiliar calculado

en el algoritmo 5.11 y separadores para distinguir un punto de otro. Esto es, los dígitos del 0 al 9, , (coma) y ; (punto y coma), que en total son 12 símbolos.

Con 4 bits se pueden representar 16 símbolos, por lo que se puede usar un byte para almacenar a 2 símbolos del cifrado. Así se define el siguiente mapeo.

$$\alpha(s) = \begin{cases} s & \text{en binario a 4 bits si } 0 \leq s \leq 9 \\ 10 & \text{en binario a 4 bits si } s \text{ es una coma} \\ 11 & \text{en binario a 4 bits si } s \text{ es un punto y coma} \end{cases}$$

Con lo que el cifrado se puede comprimir usando este mapeo de tal forma que 2 símbolos consecutivos del cifrado, c_i y c_{i+1}, pueden ser vistos como la concatenación de sus mapeos bajo α, es decir $\alpha_1(c_i, c_{i+1}) = \alpha(c_i)\alpha(c_{i+1})$.

5.4.3. Selección de la curva

Para seleccionar la curva a usar en el criptosistema hay que tomar en cuenta la cantidad de mensajes posibles. Para esta implementación se tienen 65536 posibles mensajes, por lo que una primera aproximación a p y q sería usar los primos más cercanos a la raíz cuadrada de 65536, pero con $p * q > 65536$.

De la sección anterior se tiene el mapeo

$$\sigma(b_i, b_{i+1}) = t_1t_2t_3t_4t_5t_6t_7t_8r_1r_2r_3r_4r_5r_6r_7r_8$$

siendo $t_1t_2t_3t_4t_5t_6t_7t_8r_1r_2r_3r_4r_5r_6r_7r_8$ la representación en binario de juntar dos bytes. Del lado izquierdo (t_i) se tienen 256 formas de escoger los bits, y de la derecha (r_i) también 256, por lo que se tienen $256 * 256 = 65536$ posibilidades de formar a $\sigma(b_i, b_{i+1})$, entonces los primos que se deben buscar en una primera aproximación son los cercanos a 256. Estos resultan ser 257 y 263, $257 * 263 = 67591$.

Esta p y q no son muy buenas al cifrar, pues están muy cercanas, y $n = p * q$ resulta fácil de factorizar. Supongamos que $p < q$, entonces se puede reescribir n de la siguiente forma:

$$\begin{aligned} n &= p * q = \\ &= p * (p + m) = \\ &= p^2 + pm, \end{aligned}$$

teniendo una ecuación cuadrática en p, que se puede resolver con la ecuación

$$p = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

donde $c = 0$ y $a = 1$, por lo que se tendría que probar con distintos valores para b, estos intentos no son muchos ya que p y q son cercanos y por lo tanto m es pequeña, cuando $b = m$ se habrá resuelto la ecuación y por tanto, factorizado n, lo que culminaría en la ruptura del criptosistema. Al conocer p y q, se pueden calcular los órdenes de las curvas que se usan y calcular las llaves de descifrado.

Se concluye, 257 y 263 no son buenas opciones para p y q, pero sirven de referencia como primos mínimos a partir de los cuales escoger.

6
Resultados

"Now these points of data make a beautiful line."
Still Alive, Jonathan Coulton.

En este capítulo se revisan algunas de las pruebas, tanto para curvas elípticas, el criptosistema y funciones extras. Así como algunos resultados encontrados durante el proceso. Las pruebas se realizaon en una computadora con procesador a 3.42 GHz y 8 Gb de memoria RAM.

6.1. Pruebas y conteos sobre curvas elípticas

A continuación se presentan algunas de las pruebas y conteos involucrando las propiedades de las curvas elípticas bajo la implementación dada.

En cuadro 6.1 se muestran algunas de las curvas maximales sobre Z_p, para toda $p \in [257, 700]$, algunas de las cuales se usaron como referencia para probar el criptosistema.

En el cuadro 6.2 se muestran algunas de las curvas minimales sobre Z_p, para toda $p \in [257, 700]$.

p	Curva max.	Orden	p	Curva max.	Orden
257	$E_{257}(3,0)$	290	467	$E_{467}(2,361)$	511
263	$E_{263}(5,32)$	296	479	$E_{479}(1,128)$	523
269	$E_{269}(7,103)$	302	487	$E_{487}(0,5)$	532
271	$E_{271}(1,9)$	304	491	$E_{491}(1,33)$	536
277	$E_{277}(2,61)$	311	499	$E_{499}(1,188)$	544
281	$E_{281}(1,11)$	315	503	$E_{503}(1,80)$	548
283	$E_{283}(2,251)$	317	509	$E_{509}(1,70)$	555
293	$E_{293}(2,22)$	328	521	$E_{521}(1,200)$	567
307	$E_{307}(0,5)$	343	523	$E_{523}(1,411)$	569
311	$E_{311}(1,50)$	347	541	$E_{541}(0,1)$	588
313	$E_{313}(0,10)$	349	547	$E_{547}(1,344)$	594
317	$E_{317}(5,86)$	353	557	$E_{557}(3,132)$	605
331	$E_{331}(1,54)$	368	563	$E_{563}(2,363)$	611
337	$E_{337}(1,94)$	374	569	$E_{569}(6,39)$	617
347	$E_{347}(2,96)$	385	571	$E_{571}(0,3)$	619
349	$E_{349}(0,9)$	387	577	$E_{577}(5,0)$	626
353	$E_{353}(3,87)$	391	587	$E_{587}(1,54)$	636
359	$E_{359}(1,187)$	397	593	$E_{593}(1,94)$	642
367	$E_{367}(1,215)$	406	599	$E_{599}(1,124)$	648
373	$E_{373}(0,8)$	412	601	$E_{601}(0,9)$	651
379	$E_{379}(2,78)$	418	607	$E_{607}(0,2)$	657
383	$E_{383}(1,91)$	423	613	$E_{613}(2,189)$	663
389	$E_{389}(1,23)$	429	617	$E_{617}(17,63)$	667
397	$E_{397}(3,130)$	437	619	$E_{619}(0,4)$	669
401	$E_{401}(6,0)$	442	631	$E_{631}(3,267)$	682
409	$E_{409}(1,56)$	450	641	$E_{641}(2,0)$	692
419	$E_{419}(1,69)$	460	643	$E_{643}(1,105)$	694
421	$E_{421}(0,18)$	463	647	$E_{647}(1,479)$	698
431	$E_{431}(1,296)$	473	653	$E_{653}(1,123)$	705
433	$E_{433}(1,96)$	475	659	$E_{659}(1,91)$	711
439	$E_{439}(0,15)$	481	661	$E_{661}(3,274)$	713
443	$E_{443}(2,132)$	486	673	$E_{673}(1,242)$	725
449	$E_{449}(1,163)$	492	677	$E_{677}(3,0)$	730
457	$E_{457}(1,165)$	500	683	$E_{683}(1,282)$	736
461	$E_{461}(1,36)$	504	691	$E_{691}(1,21)$	744
463	$E_{463}(0,4)$	507			

Cuadro 6.1: Muestra de algunas curvas maximales indicando p y su orden.

p	Curva min.	Orden	p	Curva max.	Orden
257	$E_{257}(156, 0)$	226	467	$E_{467}(396, 128)$	425
263	$E_{263}(63, 129)$	232	479	$E_{479}(466, 41)$	437
269	$E_{269}(32, 167)$	238	487	$E_{487}(0, 480)$	444
271	$E_{271}(117, 224)$	240	491	$E_{491}(473, 123)$	448
277	$E_{277}(275, 218)$	245	499	$E_{499}(246, 50)$	456
281	$E_{281}(121, 84)$	249	503	$E_{503}(157, 352)$	460
283	$E_{283}(213, 232)$	251	509	$E_{509}(29, 75)$	465
293	$E_{293}(155, 261)$	260	521	$E_{521}(162, 244)$	477
307	$E_{307}(0, 170)$	273	523	$E_{523}(493, 1)$	479
311	$E_{311}(28, 171)$	277	541	$E_{541}(0, 89)$	496
313	$E_{313}(90, 212)$	279	547	$E_{547}(1, 203)$	502
317	$E_{317}(143, 40)$	283	557	$E_{557}(291, 201)$	511
331	$E_{331}(24, 302)$	296	563	$E_{563}(254, 6)$	517
337	$E_{337}(232, 89)$	302	569	$E_{569}(295, 319)$	523
347	$E_{347}(273, 59)$	311	571	$E_{571}(72, 395)$	525
349	$E_{349}(2, 148)$	313	577	$E_{577}(500, 0)$	530
353	$E_{353}(214, 15)$	317	587	$E_{587}(56, 318)$	540
359	$E_{359}(276, 284)$	323	593	$E_{593}(60, 161)$	546
367	$E_{367}(245, 29)$	330	599	$E_{599}(460, 536)$	552
373	$E_{373}(45, 254)$	336	601	$E_{601}(0, 568)$	553
379	$E_{379}(90, 213)$	342	607	$E_{607}(571, 222)$	559
383	$E_{383}(305, 161)$	345	613	$E_{613}(403, 23)$	565
389	$E_{389}(183, 164)$	351	617	$E_{617}(516, 45)$	569
397	$E_{397}(394, 250)$	359	619	$E_{619}(395, 305)$	571
401	$E_{401}(170, 0)$	362	631	$E_{631}(362, 9)$	582
409	$E_{409}(74, 0)$	370	641	$E_{641}(110, 0)$	592
419	$E_{419}(49, 40)$	380	643	$E_{643}(432, 5)$	594
421	$E_{421}(0, 418)$	381	647	$E_{647}(484, 39)$	598
431	$E_{431}(358, 23)$	391	653	$E_{653}(519, 88)$	603
433	$E_{433}(102, 393)$	393	659	$E_{659}(656, 534)$	609
439	$E_{439}(1, 148)$	399	661	$E_{661}(144, 611)$	611
443	$E_{443}(280, 64)$	402	673	$E_{673}(162, 422)$	623
449	$E_{449}(176, 365)$	408	677	$E_{677}(674, 0)$	626
457	$E_{457}(85, 247)$	416	683	$E_{683}(638, 6)$	632
461	$E_{461}(68, 457)$	420	691	$E_{691}(16, 38)$	640
463	$E_{463}(0, 157)$	421			

Cuadro 6.2: Muestra de algunas curvas minimales indicando p y su orden.

Los resultados de ambos cuadros fueron generados por el método que reporta todas las curvas posibles para un p fijo. Usando los datos generados por este mismo método también se contó el número total de curvas existentes para un p fijo, resultando en que hay $p*(p-1)$ curvas en Z_p. También se hizo un conteo sobre el número de curvas maximales y minimales totales, a pesar que no se pudo llegar a una relación que indicara en términos de p cuántas curvas hay, si se encontró que hay el mismo número de maximales que minimales para un p dado.

6.2. Pruebas sobre el criptosistema

Hay que notar que al usar curvas complementarias en el criptosistema lo que se hace es tener un espacio de mensajes igual a Z_n ($n = pq$), de forma que si un elemento r no corresponde a una coordenada válida de un punto de la curva $E_p(a, b)$ o $E_q(a, b)$, entonces corresponde a una coordenada en las curvas complementarias correspondientes. También hay que notar que si se selecciona una curva maximal, la curva complementaria será minimal (teorema 2.1.6).

Al cifrar texto en claro (caracteres imprimibles) se hicieron pruebas calculando el tiempo en que se cifró el texto y midiendo el tamaño en bits del texto cifrado.

Usando las curvas $E_{461}(1, 36)$, $\overline{E_{461}(1, 36)}$, $E_{571}(1, 36)$ y $\overline{E_{571}(1, 36)}$, con órdenes 504, 420, 560 y 584 respectivamente, con la llave de cifrado 451 se obtiene el texto cifrado *213591,0;173706,2;201500,2;16614,0;40368,0;142310,6;173778,1; 244041,5;37279,6;173884,3;169611,3;251053,4;237021,1;209929,1;106243,6; 171698,1;* (coordenadas x de puntos en la curva $E_{263231}(1, 36)$) en ∼30 milisegundos donde el texto original es *qwertyuiopasdfghjklzxcvbnm123456*. Y al descifrar se obtuvo la secuencia original en ∼10 milisegundos.

El texto *qwertyuiopasdfghjklzxcvbnm123456* corresponde a 256 bits a cifrar (32 caracteres), mientras que el texto cifrado asociado con las curvas mencionadas es de 1136 bits. Aproximadamente 4 veces el tamaño de la entrada.

Secuencias de 1024 bits de tamaño se cifran en ∼60 milisegundos en cifrados de al rededor de 4500 bits, mientras que decifrar 4500 bits toma ∼50 milisegundos.

Al cifrar archivos se tiene la opción de comprimir el archivo cifrado usando el mapeo α de la sección 5.4.2. Se tiene que al cifrar archivos usando este crip-

tosistema el tamaño del archivo cifrado sin comprimir es aproximadamente 4 veces el tamaño del archivo original, en bytes (así como cifrar sólo texto el tamaño aproximado es 4 veces mayor) y con el mapeo α es aproximadamente 2 veces más grande. Los tiempos tienen cambios con respecto a la cantidad de operaciones por ciclo, es decir, mientras más operaciones se hacen por iteración al cifrar mayor es el tiempo que toma el cifrado del archivo, la tabla 6.3 muestra más detalles de esto.

Bytes totales	Bytes por ciclo	Tiempo aprox./ciclo	Tiempo total
40869	500	7 ms	497 ms
40869	2000	35 ms	692 ms
40869	20000	1300 ms	2609 ms
40869	50000	5739 ms	5739 ms
1578010	500	20 ms	60396 ms
1578010	2000	39 ms	30249 ms
1578010	20000	1112 ms	86731 ms
1578010	50000	9515 ms	294965 ms
3112296	500	35 ms	216959 ms
3112296	2000	50 ms	79045 ms
3112296	20000	1370 ms	212279 ms
3112296	50000	10826 ms	671223 ms

Cuadro 6.3: Ejemplo de tamaño de archivos - operaciones por ciclo - tiempo requerido

De estas pruebas se encontró que lo mejor en este caso es cifrar en bloques de 2000 bytes. Sin embargo, los tiempos empiezan a crecer mucho para archivos más grandes, este cuadro tiene datos para archivos de a lo más 3Mb de tamaño, en archivos de $\sim$20 Mb o más ya comienza a tomar tanto tiempo que se puede dudar de si el programa continúa o se ha trabado.

De lo anterior se puede concluir que este cifrado es lo bastante lento como para no querer usarlo para cifrar archivos muy grande, pero es lo bastante rápido como para cifrar llaves que sirvan para cifrar con otros esquemas de encriptación. Por ejemplo, el servicio de correo de Google (gmail) usa llaves de cifrado de 256 bits que se pueden cifrar y el resultado pasarse a otro esquema como el proyecto Chromium OS [12], que en una etapa del proceso de protección de

datos del usuario cifra una llave de 256 bits usando RSA para luego usar el resultado para cifrar con otro esquema.

En cuanto a ataques posibles, se tiene que si se logra obtener los factores p y q, entonces se pueden calcular los órdenes de las curvas, ya que los parámetros a y b son públicos, y depúes continuar a descifrar los mensajes cifrados sin tener autorización a ello.

Otra vulnerabilidad resulta de elegir a p y q cercanos, como se mencionó en el capítulo anterior obtener los factores resultaría sencillo.

Pinch, en [11], analiza una vulnerabilidad más al seleccionar llaves secretas pequeñas, y poder extender la estrategia de Wiener para RSA a este esquema.

Apéndices

A
Teoría de Números

A.1. Definiciones

Definición: en los enteros un número primo es aquel que sólo puede ser dividido entre la unidad y si mismo.

Definición: si p es un primo entonces Z_p denota al campo finito de p elementos, es decir, al conjunto de los números $\{0, 1, ..., p-1\}$.

Definición: se denota como Z_p^* al conjunto $Z_p \backslash \{0\}$.

Definición (módulo): Si $x, y, m, r \in Z$, con $0 \leq r < m$ entonces

$$x \equiv r \mod m$$
$$\text{si y sólo si}$$
$$x = ym + r.$$

Definición (residuo cuadrático): Se dice que $a \in Z_n$ es un residuo cuadrático, con n no necesariamente un primo, si existe $x \in Z_n$ tal que

$$x^2 \equiv a \mod n.$$

Definición: Sea $x \in Z_n$, el inverso multiplicativo de x módulo n es un entero k tal que

$$xk \equiv 1 \mod n.$$

Si k existe entonces es único y se denota como x^{-1}.

Definición: Sean $a, b \in Z_n$, la división a/b módulo n esta definida como el producto de ab^{-1} módulo n, si b es invertible módulo n.

Definición [9]: "Sea $d = mcd(a, n)$[1]. La congruencia

$$ax \equiv b \mod n,$$

tiene solución x si y sólo si d divide a b, en tal caso hay exactamente d soluciones entre 0 y n-1; estas soluciones son todas congruentes módulo n/d".

A.2. Teoremas

Teorema A.2.1 (Fermat) *Si $a, p \in Z$, con p un primo entonces*

$$1 \equiv a^{p-1} \mod p.$$

Teorema A.2.2 (El Chino del Residuo) *Sean los enteros $n_1, n_2, ..., n_k$ tales que $mcd(n_i, n_j) = 1$ con $1 \leq i, j \leq k$ (son primos relativos entre si), entonces el sistema de congruencias*

$$\begin{aligned} x &\equiv a_1 \mod n_1, \\ x &\equiv a_2 \mod n_2, \\ &\vdots \\ x &\equiv a_k \mod n_k, \end{aligned}$$

tiene una solución única módulo $n = n_1 n_2 \cdots n_k$.

[1]Se denota al máximo común divisor como *mcd*

A.3. Algoritmo de Euclides

El algoritmo de Euclides es una herramienta útil en el cálculo del máximo común divisor entre 2 enteros positivos, a y b, que no necesita del conocimiento de la factorización de estos, ya que se basa en el hecho de que si $a > b$ entonces $mcd(a, b) = mcd(b, a \mod b)$.

Algoritmo A.1 *euclides*

Entrada: Dos enteros positivos a, b tales que $a \geq b$.
Salida: $d = mcd(a, b)$.
  mientras $b \neq 0$ **hacer**
    $r \leftarrow a \mod b, a \leftarrow b, b \leftarrow r$.
  fin mientras
  devolver a

Este algoritmo puede ser extendido de tal forma que no sólo de el máximo común divisor d entre a y b, si no que también de enteros r y s tales que $ar + bs = d$.

Algoritmo A.2 *euclides extendido*

Entrada: Dos enteros positivos a, b tales que $a \geq b$.
Salida: $d = mcd(a, b)$ y enteros r y s tales que $ar + bs = d$.
  si $b = 0$ **entonces**
    $d \leftarrow a, r \leftarrow 1, s \leftarrow 0$.
    devolver (d, r, s).
  fin si
  $r_2 \leftarrow 1, r_1 \leftarrow 0, s_2 \leftarrow 0, s_1 \leftarrow 1$.
  mientras $b \geq 0$ **hacer**
    $q \leftarrow \lfloor a/b \rfloor, t \leftarrow a - qb, r \leftarrow r_2 - qr_1, s \leftarrow s_2 - s_1$.
    $a \leftarrow b, b \leftarrow t, r_2 \leftarrow r_1, r_1 \leftarrow r, s_2 \leftarrow s_1, s_1 \leftarrow s$.
  fin mientras
  $d \leftarrow a, r \leftarrow r_2, s \leftarrow s_2$.
  devolver $(d.r.s)$

El algoritmo A.2 es de orden $O((\log a)^2)$ (ver [9]).

B
Teoría de Grupos

B.1. Definiciones

Definición: Sea $\mathcal{G}$ un conjunto y sea $+$ una operación binaria definida sobre $\mathcal{G}$ con las siguientes propiedades:

- Sean $a, b \in \mathcal{G}$, se cumple que el resultado de la operación $a + b$ también está en $\mathcal{G}$.
- Existe un elemento $e_I \in \mathcal{G}$ tal que para todo elemento $a \in \mathcal{G}$ se cumple la ecuación $a + e_I = e_I + a = a$.
- Para todo elemento $a \in \mathcal{G}$ existe un único elemento $b \in \mathcal{G}$ tal que $a + b = b + a = e_I$
- Para cualesquiera $a, b, c \in \mathcal{G}$ se cumple que $a + (b + c) = (a + b) + c$.

A $\mathcal{G}$ con la operación $+$ se le llama grupo y se suele denotar $(\mathcal{G}, +)$.

Definición: Sea $(\mathcal{G}, +)$ un grupo, si para todo $a, b \in \mathcal{G}$ se cumple que

$$a + b = b + a,$$

se dice que el grupo es conmutativo o abeliano.

Definición: Sea $(\mathcal{G}, +)$ un grupo y $\mathcal{H}$ un subconjunto de $\mathcal{G}$, si $\mathcal{H}$ cumple las propiedades de grupo con la operación $+$, es decir, $(\mathcal{H}, +)$ es un grupo, entonces se dice que $(\mathcal{H}, +)$ es un subgrupo de $(\mathcal{G}, +)$.

Definición: Sea $(\mathcal{G}, +)$ un grupo y $a \in \mathcal{G}$,se define la operación a^t como la suma de a consigo mismo t-veces.

Teorema B.1.1 *Sea $(\mathcal{G}, +)$ un grupo, existe un entero k tal que para todo $a \in \mathcal{G}$ se cumple*

$$a^k = e_I.$$

Bibliografía

[1] J. Borst. Public key cryptosystems using elliptic curves. Master's thesis, Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven, 1997.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[3] N. Demytko. A new elliptic curve based analogue of rsa. In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 1993.

[4] Dorothy E. Denning, Peter, and J. Denning. Cryptography and data security, 1982.

[5] W. Diffie and M. E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, volume 22, pages 644–654, 1976.

[6] Shen Guicheng and Zheng Xuefeng. Research on the implementation of elliptic curve cryptosystem based on object-oriented method. 2008.

[7] A. Kerckhoffs. La cryptographie militaire. In *Journal des sciences militaires*, volume vol. IX, 1883. URL `http://www.petitcolas.net/fabien/kerckhoffs/`.

[8] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, 1994.

[9] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[10] Department of Mathematics: Rice University. An elementary proof of the group law for elliptic curves, 2009. URL `http://math.rice.edu/ friedl/papers/AAELLIPTIC.PDF`.

[11] R.G.E. Pinch. Extending the wiener attack to rsa-type cryptosystems, 1995.

[12] Chromium Projects. Chromium os, 2008. URL `http://www.chromium.org/chromium-os`.

[13] Joseph H. Silverman. *The Arithmetic of Elliptic Curves (Graduate Texts in Mathematics)*. Springer, 1986.

[14] Gustavus J. Simmons. *Contemporary Cryptology: The Science of Information Integrity*. IEEE Press, 1992.

[15] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.

[16] Michael J. Wiener. Cryptanalysis of short rsa secret exponents. *IEEE Transactions on Information Theory*, 36:553–558, 1990.