BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

SISTEMA CRIPTOGRÁFICO BASADO EN CURVAS ELÍPTICAS SOBRE EL CAMPO FINITO GF(p)

TESIS PARA OBTENER EL TÍTULO DE:
LICENCIATURA EN INGENERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
EDUARDO SÁNCHEZ MENDOZA

DIRECTOR DE TESIS:
DR. HÉCTOR DAVID RAMÍREZ HERNÁNDEZ

ASESOR DE TESIS:
DR. ROBERTO CONTRERAS JUÁREZ

Enero 2020

Resumen

La criptografía incorpora las técnicas con las que se busca garantizar protección de la información, frente a personas no autorizadas. Esta manera de resguardar información ha existido desde tiempos remotos, en donde existían elementos que sólo ciertas personas eran capaces de entender e interpretar. En un principio la criptografía fue utilizada para efectos de guerra y poder, pero gracias a los grandes avances tecnológicos desarrollados a finales del siglo pasado, se ha visto la necesidad de resguardar la información que cada individuo maneja y comparte por medio del internet. Es por ello por lo que la criptografía toma una mayor importancia. La criptografía actual se basa en dos tipos de protocolos, uno el de la criptografía simétrica y el otro que corresponde a la criptografía asimétrica. En este trabajo analizaremos un protocolo del tipo asimétrico basado en curvas elípticas sobre el campo finito GF(p), proponiendo una librería desarrollada en PHP que permite cifrar y descifrar información, la cual pretende brindar los servicios de seguridad, autenticación, integridad y confidencialidad de la información.

Abstract

Cryptography incorporates the techniques with which it seeks to guarantee protection of information, in front of unauthorized persons. This way of protecting information has existed since ancient times, where there were elements that only certain people were able to understand and interpret. In the beginning, cryptography was used for the purposes of war and power, but thanks to the great technological advances developed at the end of the last century, we have seen the need to safeguard the information that everyone manages and shares through the internet. That is why cryptography takes on greater importance. Current cryptography is based on two types of protocols, one of symmetric cryptography and the other corresponding to asymmetric cryptography. In this paper we will analyze a protocol of the asymmetric type based on elliptic curves on the finite field GF(p), proposing a library developed in PHP that allows to encrypt and decrypt information, which aims to provide security services, authentication, integrity and confidentiality of the information.

Antecedentes

El origen de la criptografía se remonta sin duda a los orígenes del hombre, desde que aprendió a comunicarse. Fue así que tuvo que encontrar medios para asegurar la confidencialidad de una parte de sus comunicaciones.

La historia indica que desde el antiguo Egipto, la escritura jugó a veces ese papel. Aunque, el primer testimonio de uso deliberado de métodos técnicos que permitieron cifrar los mensajes viene de Grecia, sobre el siglo VI antes de Cristo, y se llama escítalo. Más tarde los ejércitos romanos utilizaron para comunicarse el cifrado de César, el cual consistía en crear un alfabeto encriptado a raíz del alfabeto en claro haciendo una asignación, pero desplazándose 3 posiciones a la derecha y así poder encriptar el mensaje. Para hacer el proceso de desencriptación se hace el proceso inverso, es decir, hacer un desplazamiento de 3 posiciones, pero a la izquierda. Después, durante cerca de 19 siglos, se asiste al desarrollo más o menos ingenioso de técnicas de cifrado experimentales donde la seguridad residía esencialmente en la confianza que quisieran darles los usuarios.

La criptografía en Europa data de la edad media, El primer libro europeo que describe el uso de la criptografía fue escrito en el siglo XIII por el monje franciscano Roger Bacon, titulado La Epístola sobre las obras de arte secretas y la nulidad de la magia, en él se describen siete métodos distintos para mantener en secreto los mensajes. En 1563 el físico Giambattista Della Porta crea un sistema con la particularidad de cifrar bloques de dos en dos letras, el cual ha sido utilizado con éxito durante más de tres siglos. Fue el inventor del primer sistema literal de clave doble, o sea, la primera cifra en la cual el alfabeto cifrado muda cada letra. Este sistema poli alfabético era extremadamente robusto para la época, de modo que muchos consideran Della Porta como el "Padre de la criptografía moderna".
A principios del siglo XIX Thomas Jefferson inventó una máquina constituida por 10 cilindros que estaban montados en un eje de forma independiente, en donde se colocaba el alfabeto y al girar

los cilindros, quedaba cifrado el mensaje. En el siglo XIX, Kerchoffs estableció los principios de la criptografía moderna. Los algoritmos modernos usan una clave para controlar el cifrado y descifrado de los mensajes. Generalmente el algoritmo de cifrado es públicamente conocido y sometido a pruebas por parte de expertos y usuarios. Se acepta, por lo tanto, la denominada hipótesis de Kerckhoffs, que establece que la seguridad del cifrado debe residir, exclusivamente, en el secreto de la clave y no en el del mecanismo de cifrado. En la primera Guerra Mundial los cifrados todavía eran hechos por diferentes asociaciones del alfabeto (cifrado por permutación) y los mensajes eran llevados por el hombre, usando medios de transporte muy lentos, de tal forma que había lugares a los cuales era imposible llevar el mensaje, como a barcos, aviones y submarinos, por lo que se comenzó a usar el teléfono, el telégrafo y la radio. Esto último era fácil de transportar, lo que cambió radicalmente las comunicaciones; sin embargo, de este modo los mensajes eran también fácilmente interceptados. Esto justificó incrementar el uso de la criptografía.

Hasta 1918 los cifrados se hacían manualmente, lo que causaba muchos errores en el proceso de cifrar y descifrar, de tal modo que los criptógrafos comenzaron a crear máquinas para tales fines. Por ejemplo, en Estados Unidos desde 1861 hasta 1980 se registraron 1769 patentes relacionadas con la criptografía. A principios de los años 20 ya había un gran número de estas máquinas, dando gran seguridad en la transmisión de información. Esta década fue la edad de oro para las máquinas de cifrado. Una de las más populares fue la Enigma creada por el ingeniero alemán Arthur Scherbius.

En 1926 la marina alemana decidió comprar la máquina ENIGMA, que fue patentada hasta 1928, fecha en que Scherbius murió. Irónicamente en 1929 su invento se vendió a gran escala en todo el mundo. A finales de la segunda Guerra Mundial se habían producido alrededor de 30,000 máquinas Enigma. No fue oficial, pero la fuerza aérea alemana era el más grande usuario con 20,000 del total de estas máquinas. Sin embargo, la seguridad de los alemanes no sólo dependía de Enigma. Después de la guerra se inició el desarrollo de la electrónica y las computadoras. Criptosistemas con algoritmos más sofisticados fueron implementados en la transmisión de la información, pero aún eran usadas máquinas del tipo Enigma como la M-209 Converter o C-36 inventada por Boris Hagelin, la cual fue usada hasta principios de los años 50 por la armada norteamericana. Como muchas otras actividades, la criptografía pasó a ser dominada predominantemente por quienes ganaron la guerra. Así se inició la nueva era de la criptografía electrónica.

A principios de los años 70, la criptografía estaba por iniciar la época de los circuitos integrados y el desarrollo en los algoritmos, concretamente, el uso de las matemáticas modernas. Por ejemplo, en 1975 se publica la creación de IBM, el sistema Data Encryption Standard (DES), que ha sido uno de los más usados hasta la fecha. Un año importante para la criptografía fue el de 1976, cuando W. Diffie y M. Hellman crean el concepto de Criptosistema de clave pública, es decir, un sistema donde la clave de cifrado se puede encontrar en un directorio público de usuarios; sin embargo, la clave de descifrado es diferente y no se obtiene fácilmente de la primera.

Poco más tarde, en 1978 se da a conocer el criptosistema de clave pública más seguro y usado hasta la fecha, el RSA. Sus inventores R. L. Rivest, A. Shamir y L. Adleman del MIT proponen la función de un sólo sentido que utiliza el exponente módulo un número entero n, producto de dos números primos y que tiene como seguridad la dificultad de factorizar a un número n de entre 100 y 200 dígitos. La necesidad de romper este criptosistema desarrolla la teoría de factorizar números grandes, cosa que después justifica la aparición de las curvas elípticas en criptografía. Otro sistema que se ha mantenido hasta hoy es el propuesto por T. ElGamal en 1984, que basa su seguridad en el problema del logaritmo discreto que aún no se ha podido resolver satisfactoriamente de manera rápida.

Muchas áreas de las matemáticas han podido ser usadas para crear criptosistemas, como los campos finitos y factorización de números enteros. Otros ejemplos son: en 1970 R. J. McEliece desarrolló un criptosistema de clave pública basado en códigos detectores - correctores de errores; en los años 80 V. Varadharajan propuso distintas estructuras de anillos que pueden ser aplicados en la generalización del sistema RSA; en 1984 Lidl y Müller proponen polinomios de permutación; en 1985, de forma independiente V. Miller y N. Koblitz usan la teoría de curvas elípticas, para crear criptosistemas, estas curvas fueron propuestas por Lenstra para factorizar números enteros; en 1988 J. Buchmann y H. Williams proponen usar campos cuadráticos reales e imaginarios, etc. Otro tipo de protocolo propuesto recientemente por el grupo de la IBM usa la teoría de incertidumbre y se le conoce como criptografía cuántica.

Concebir sistemas criptográficos requiere sistemas con unos cimientos matemáticos suficientes que permitan proveer utilidades para medir y cuantificar su resistencia a eventuales ataques y poder determinar un sistema incondicionalmente seguro.

Problema

En la Era de la Información en la que estamos inmersos hoy, en la que nuestra información viaja a través de canales no seguros, se ve en la necesidad de implementar protocolos criptográficos que permitan el intercambio de información de una forma más segura. La ciencia encargada de crear soluciones innovadoras y revolucionarias en los campos tan modernos como la informática, y tan antiguos como las matemáticas, es la criptografía. La criptografía hasta finales del siglo pasado fue que tomo más auge por el crecimiento tecnológico que existe, misma que se basa en dos tipos de protocolos, una la de la criptografía simétrica y la otra que corresponde a la criptografía asimétrica. En este proyecto se pretende analizar los avances que existen en la criptografía basada en curvas elípticas sobre un campo arbitrario $GF(p)$. Se introducirá la criptografía de llave pública con curvas elípticas y se tratarán cuestiones de seguridad y de implementación, además de ejemplos de esquemas sobre la criptografía.

Introducción

Históricamente, la criptografía surgió para permitir la comunicación desde distintos lugares manteniendo la privacidad de la información, incluso en la presencia de un atacante con acceso al canal de comunicación, convirtiéndose en una ciencia que abarca los campos de la teoría de números, la teoría de la complejidad computacional y la teoría de la probabilidad entre otros. Entre sus objetivos, se destaca construir y analizar protocolos que garanticen aspectos como la confidencialidad y la autenticidad de la información.

En está tesis se analizan los fundamentos matemáticos y principales primitivas criptográficas, además de llevar al laboratorio el uso de distintas técnicas y herramientas. El problema que se aborda involucra un estudio teórico-práctico sobre el uso de las curvas elípticas en criptografía y la de la implementación de diversos protocolos criptográficos basados en curvas elípticas. Cabe destacar que el estudio de las curvas elípticas no es un tema realmente nuevo. Las curvas elípticas han ocupado un papel central en matemáticas desde hace tiempo y sus notables propiedades aritméticas y geométricas han permitido encontrar aplicaciones en múltiples problemas y campos matemáticos. Como se mencionó anteriormente, su implementación en criptografía es relativamente reciente, pudiéndose situar su inicio en el año 1985, gracias a Miller y Koblitz. Actualmente, existen empresas que están realizando estudios de criptografía de curva elíptica en dispositivos de recursos limitados. La curvas elípticas también se utilizan en programas tan cotidianos como el Windows Media Player para proteger las claves de las licencias que autorizan a reproducir contenidos con DRM(Gestión digital de derechos) o en protocolos tan actuales como los bitcoins. Los reproductores de Blu-Ray y la Play Station 3 implementan una tecnología similar para evitar la copia de contenidos mientras que la Wii hace uso de las curvas elípticas para asegurar que nadie hace trampa cuando guardamos una partida on-line. Los smartphones también utilizan curva elíptica para cifrar la información que transmiten. Incluso los nuevos pasaportes alemanes usan las curvas elípticas para proteger los datos biomédicos que almacenan. Aunque quizá el ejemplo más representativo del uso de criptografía

con curvas elípticas es la Suite B del gobierno de los Estados Unidos. Dicha suite es un estándar federal de protección de documentos que actualmente usa algoritmos basados exclusivamente en curvas elípticas para cifrar documentos clasificados como críticos o confidenciales.

La finalidad de la criptografía es garantizar que el contenido del mensaje enviado no haya sido modificado en su tránsito, o que si se resguarda esta no pueda ser extraída [1]. Los criptosistemas utilizados son los denominados simétricos y los asimétricos. Los criptosistemas simétricos son aquellos que usan un método matemático para cifrar y descifrar un mensaje [2]. Este tipo de criptografía utiliza únicamente una llave para realizar el proceso, así el mensaje únicamente se descifra con la única llave existente. Este tipo de criptografía garantiza confidencialidad, pero al querer compartir el mensaje con otro usuario deberá crearse una nueva llave y el número de llaves aumenta conforme aumenten los usuarios con quienes se comparta el mensaje [3]. En 1976, Whitfield Diffie y Martin Hellman [4] revolucionan la criptografía al introducir el concepto de criptosistema asimétrico, que surge como una solución al problema de intercambiar claves privadas por canales inseguros. Los sistemas asimétricos son aquellos en los cuales tanto el emisor como el receptor poseen un par de claves: una de tipo pública y la otra de tipo privada y para enviar mensajes el emisor tiene que cifrar el mensaje con la clave pública del receptor para que así este sea el único que pueda descifrar el mensaje usando su clave privada [5].
Todo esto es parte de la motivación para realizar un estudio teórico-práctico de la criptografía con curvas elípticas. En el capítulo I daremos un recorrido historico de los criptosistemas utilizados, desde los mas sencillos hasta los mas complicados que se utilizan actualmente. En el capítulo II se analiza la parte más representativa de la teoría de curvas elípticas desde un punto de vista formal y riguroso que permita la aplicación en la criptografía. En el hacemos un estudio de las curvas elípticas sobre los números reales y posteriormente sobres los campos finitos $GF(p)$. En el capítulo III, se introduce la criptografía basada en curvas elípticas y se hace énfasis en cuestiones de seguridad y de implementación, explicando algunos esquemas criptográficos basados en curvas elípticas. En el capítulo IV se desarrolla una librería estandarizada computacional que implementa diversos protocolos criptográficos que brindan los servicios de seguridad, autenticación, integridad y confidencialidad, cifrando y descifrando la información haciendo uso de la criptografía basada en curvas elípticas sobre el campo $GF(p)$.

Objetivos

Objetivo General

Cifrar y descifrar información a través de la criptografía basada en curvas elípticas sobre el campo $GF(p)$, creando una librería estandarizada para integrar servicios de seguridad punto a punto en proyectos existentes como en nuevos desarrollo que brinde los servicios de seguridad, autenticación, integridad y confidencialidad de información.

Objetivo Específicos

- Realizar un análisis en profundidad de la teoría de las curvas elípticas sobre $GF(p)$, y la aritmética de sus puntos.
- Desarrollar e implementar los algoritmos para trabajar con la aritmética del grupo de una curva elíptica.
- Implementar los protocolos criptográficos basados en curvas elípticas sobre $GF(p)$.

Índice general

Índice de tablas

Índice de figuras

Índice de Algóritmos

Capítulo 1

Historia de la criptografía

La criptografía se puede definir como el arte o la ciencia de cifrar y descifrar información, utilizando técnicas matemáticas que hagan posible el intercambio de mensajes de manera que solo puedan ser leídos por las personas a quienes van dirigidos , la palabra, como tal, se forma a partir del término griego *'kryptós'* que significa 'oculto', y *'grafía'*, sufijo que quiere decir 'escritura'. La finalidad de la criptografía es garantizar el secreto en la comunicación y asegurar que la información que se envía sea autentica, es decir que el contenido del mensaje enviado no haya sido modificado en su tránsito o no pueda ser extraída, solo por aquel a quien se desea enviar el mensaje.

En la Universidad de Yale se encuentra un manuscrito de 235 páginas, donado en 1969, por H.P Kraus. El manuscrito, adquirido por Wilfrid M. Voynich en 1912, pertenecía a un colegio jesuita situado en Villa Mondragone, cerca de Roma. Escrito por un autor desconocido (aunque algunos lo atribuyen a Roger Bacon, uno de los grandes personajes del S.XIII), constituye en sí mismo todo un enigma, ya que está escrito con unos caracteres extraños, y que hasta la fecha nadie ha logrado descifrar. Se conoce como el El manuscrito Voynich. Es sin duda uno de los grandes retos del desciframiento.

Cuestiones militares, religiosas y comerciales impulsaron desde tiempos remotos el uso de escrituras secretas, listaremos algunos de los métodos de cifrado.

Figura 1.1: Página del manuscrito Voynich

1.1. Criptogramas egipcios

Ya los antiguos egipcios usaron métodos criptográficos. Por ejemplo, los sacerdotes egipcios utilizaron la escritura hierática (jeroglífica) que era claramente incomprensible para el resto de la población. Los antiguos babilonios también utilizaron métodos criptográficos en su escritura cuneiforme.

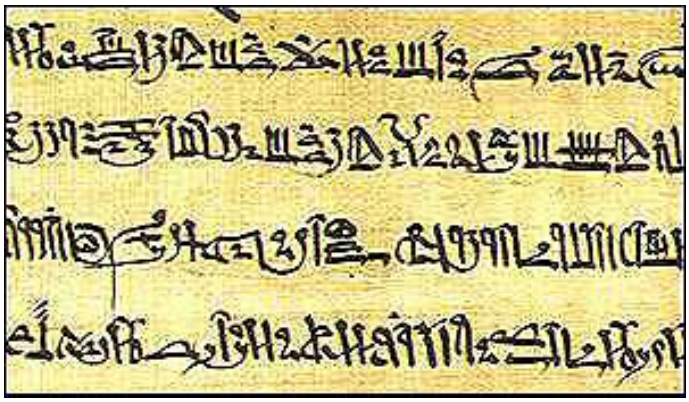

Figura 1.2: Criptogramas egipcios

1.2. Cifrado César

El cifrado César recibe su nombre en referencia a Julio César, que, según Suetonio, lo usó con un desplazamiento de tres para proteger sus mensajes de importancia militar. Si tenía que decir algo

confidencial, lo escribía usando el cifrado, esto es, cambiando el orden de las letras del alfabeto, para que ni una palabra pudiera entenderse. Si alguien quiere decodificarlo, y entender su significado, debe sustituir la cuarta letra del alfabeto, es decir, la D por la A, y así con las demás. Aunque César es la primera persona de la que se sabe que haya usado este sistema, anteriormente ya se utilizaron otros cifrados por sustitución. El sobrino de Julio César, Augusto, también empleó el cifrado pero con un desplazamiento de uno: Cuando escribía con cifrado, escribía la B por la A, la C por la B y el resto de las letras de ese mismo modo, usando AA por la X. No se sabe cuán efectivo resultaba realmente el cifrado César en esa época, pero debió ser razonablemente seguro, ya que pocos enemigos de César lo habrían entendido , y mucho menos podrían haber hecho el criptoanálisis necesario. Asumiendo que el atacante pudiera leer el mensaje, no existen pruebas de la existencia de técnicas para solucionar este tipo de codificación[**1**].

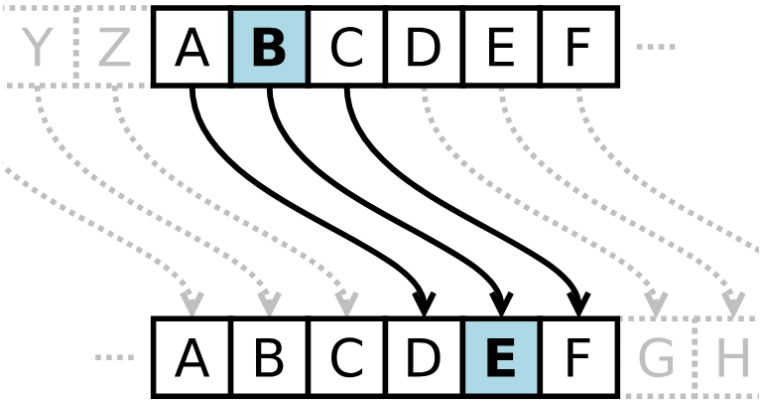

Figura 1.3: Cifrado César

1.3. Escítala

El primer caso claro de uso de métodos criptográficos se dio durante la guerra entre Atenas y Esparta. El historiador griego Plutarco, describe la escítala de la siguiente manera: "La escítala era un palo o bastón en el cual se enrollaba en espiral una tira de cuero. Sobre esa tira se escribía el mensaje en columnas paralelas al eje del palo. La tira desenrollada mostraba un texto sin relación aparente con el texto inicial, pero que podía leerse volviendo a enrollar la tira sobre un palo del mismo diámetro que el primero". Con este sistema los gobernantes de Espartana transmitieron, con eficacia, sus instrucciones secretas a los generales de su ejército, durante las campañas militares. Lógicamente, este procedimiento suponía que tanto el emisor como el receptor del mensaje dispusieran de un palo o bastón con las mismas características físicas: grosor y longitud.

Figura 1.4: Escítala

1.4. El atbash hebreo

El atbash se emplea en el libro de Jeremías 25,26 de la Biblia, donde la palabra Babilonia, en hebreo: Babel se convierte en Sheshash. Las letras del mensaje de origen se sustituyen una a una, de acuerdo con la norma siguiente: si la letra original se encuentra en la línea superior se sustituye por la letra correspondiente de la línea inferior, y a la inversa. De esta manera la a (aleph) se convierte en t (aw), y la letra b(eth) se convierte en sh(in). Por tanto la palabra Babel se convierte en Sheshash.

Normal:	א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ	ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת
Inverso:	ת	ש	ר	ק	צ	פ	ע	ס	נ	מ	ל	כ	י	ט	ח	ז	ו	ה	ד	ג	ב	א

Figura 1.5: Alfabeto atbash

1.5. Pigpen Cipher

El Pigpen Cipher es otro ejemplo de un cifrado de sustitución, pero en lugar de reemplazar cada letra con otra letra, las letras se reemplazan por símbolos. El cifrado tiene una historia interesante: aunque se desconocen sus verdaderos orígenes, ha sido utilizado por muchos grupos. Más notoriamente, fue la cifra elegida por los masones, una sociedad secreta del siglo XVIII. De hecho, lo usaron tanto que a menudo se lo conoce como el cifrado de los masones. Sin embargo, no fue utilizado exclusivamente por ellos, ya que los prisioneros de la Unión en los campos confederados lo usaron para comunicarse en la Guerra Civil estadounidense.

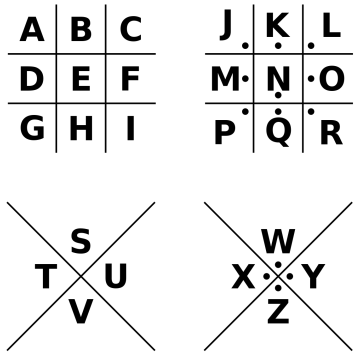

Figura 1.6: Pigpen Cipher

1.6. Cifrado Vigenère

Es un cifrado basado en diferentes series de caracteres o letras del cifrado César formando estos caracteres una tabla, llamada tabla de Vigenère, que se usa como clave. El cifrado de Vigenère es un cifrado polialfabético y de sustitución. El cifrado Vigenère se ha reinventado muchas veces. El método original fue descrito por Giovan Batista Belaso en su libro de 1553 "La cifra del Sig. Giovan Batista Belaso", quien construyó el cifrado basándose en la tabula recta de Trithemius, pero añadió una clave repetida para cambiar cada carácter entre los diferentes alfabetos. Sin embargo, fue incorrectamente atribuido en el siglo XIX a Blaise de Vigenère, a partir de un trabajo realizado en 1583, y por ello aún se le conoce como el çifrado Vigenère".

Este cifrado es conocido porque es fácil de entender e implementar, además parece irresoluble; esto le hizo valedor del apodo el código indescifrable (le chiffre indéchiffrable, en francés). El primer cifrado polialfabético fue creado por Leone Battista Alberti hacia 1467 y usaba un disco de metal para cambiar entre los diferentes alfabetos del cifrado.

El sistema de Alberti sólo cambiaba entre alfabetos después de muchas palabras, y los cambios se indicaban escribiendo la letra del correspondiente alfabeto en el mensaje cifrado. Más tarde, en 1508, Johannes Trithemius, en su trabajo Poligraphia, inventó la tabula recta, que es básicamente la tabla de Vigenère.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figura 1.7: Matriz Vigenère

Este ganó una gran reputación por ser excepcionalmente robusto. Incluso el escritor y matemático Charles Lutwidge Dodgson (Lewis Carroll) dijo que el cifrado Vigenère era irrompible en el artículo "The Alphabet Cipherpara una revista de niños. En 1917, “Scientific American"describió el cifrado Vigenère como imposible de romper. Esta reputación fue mantenida hasta que con el método Kasiski (1863) resolvió el cifrado. El cifrado Vigenère es lo suficientemente simple si se usa con discos de cifrado. Los Estados confederados de América, por ejemplo, utilizaron un disco de cifrado para implementar el cifrado Vigenère durante la Guerra Civil Americana. Los mensajes confederados fueron poco secretos, ya que los miembros de la Unión solían descifrar los mensajes.

1.7. Cifrado Playfair

Este cifrado fue inventado por Charles Wheatstone. No obstante, se le atribuye a su amigo el científico Lyon Playfair quien lo presenta a las autoridades inglesas de la época. Este sistema consiste en separar el texto en claro en digramas y proceder a su cifra de acuerdo a una matriz alfabética de dimensiones 5x5 en la cual se encuentran representadas 25 de las 26 letras del alfabeto inglés. Para que éste método de cifrado presente un mayor nivel de seguridad, se incluirá al comienzo de dicha matriz una clave que se escribe a partir de la primera fila omitiendo las letras repetidas. A continuación de dicha clave, se distribuyen las restantes letras del alfabeto hasta completar toda la matriz. Este método de cifrado debe cumplir las siguientes condiciones: Si M1M2 están en la misma fila, C1C2 son los dos caracteres de la derecha. Si M1M2 están en la misma columna, C1C2 son los dos caracteres de abajo. Si M1M2 están en filas y columnas distintas, C1 será la letra de la intersección horizontal del M1 respecto al M2 y C2 será la letra de la intersección vertical del M1 respecto al M2. El conteo de los caracteres en la matriz es circular.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 23 & 20 & 5 & 1 \\ 2 & 11 & 18 & 1 \\ 2 & 20 & 6 & 25 \\ 25 & 2 & 22 & 25 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Figura 1.8: Matriz Playfair

1.8. Cifrado ADFGVX

A finales del siglo XIX el italiano G. Marconi inventó una forma prodigiosa de comunicarse: la radio. En manos de los militares la radio fue un poderoso medio de transmisión, pero los mensajes podían caer también en manos enemigas, por lo que era necesario mandarlos cifrados. La Primera Guerra Mundial fue una guerra a gran escala, por lo que era necesario disponer de una codificación rápida y efectiva. Una de las cifras más famosas fue la llamada Cifra ADFGVX, introducida por los alemanes en el invierno de 1918. La cifra es una mezcla de métodos de sustitución y de trasposición,

esto hace que su desciframiento sea verdaderamente complicado.

Se empieza disponiendo las 26 letras del alfabeto anglosajón y los diez dígitos en una matriz 6x6. Las líneas y las columnas van encabezados por las letras A D F G V X. El modo de ordenar letras y números en la cuadrícula forma parte de la clave y necesita ser comunicada al receptor del mensaje. Su ordenación es aleatoria.

	A	D	F	G	V	X
A	T	I	L	6	B	2
D	U	G	D	M	4	R
F	Q	V	K	F	Y	5
G	N	S	P	C	8	Z
V	3	7	O	H	J	9
X	W	A	1	E	X	0

Figura 1.9: Cifrado ADFGVX

Capítulo 2

Fundamentos Matemáticos

La críptografia de curvas elíticas utiliza conceptos matematicos que le dan el soporte necesario para que el criptoanálisis que se pueda llevar acabo sea difícil de implementar. Es por ello que este tipo dde criptografía está fuertemente ligada al empleo de herramientas matemáticas, mismas que son analizadas en este capítulo para una mejor comprensión de los protocolos criptográficos que se presentan más adelante.

2.1. Aritmética Modular

La aritmética modular es la rama de las matemáticas que sustenta la mayoría de los protocolos criptográficos. Su empleo permite realizar cálculos complejos manteniendo siempre una representación numérica compacta y permite trabajar con un conjunto finito de números enteros. A continuación, marcamos los elementos fundamentales necesarios para nuestro trabajo.

Definición 1 *El conjunto de los enteros módulo n, denotado por* $\mathbb{Z}_n$ *es el conjunto formado por los números enteros* $\{0, 1, 2, \cdots, n\}$. *La suma, resta y multiplicación en el conjunto son realizadas en módulo n.*

Definición 2 *Dados los números enteros a, b y n , se dice que a es congruente con b módulo n, denotado por* $a \equiv bmod(n)$, *si y sólo si* $n|(a - b)$. *Esto es, si y sólo si existe algún entero k que divida de forma exacta la diferencia* $a - b$, *es decir,* $a - b = kn$.

Propiedades

La relación de congruencia verifica las siguientes propiedades:

- Reflexiva: $a \equiv a \ (mod\ p)$, para todo $a\ \varepsilon\ \mathbb{Z}$.
- Simétrica: $a \equiv b \ (mod\ p) \Longrightarrow b \equiv a \ (mod\ p)$.
- Transitiva: $a \equiv b \ (mod\ p) \wedge b \equiv b \ (mod\ p) \Longrightarrow a \equiv c \ (mod\ p)$

Las operaciones de suma y producto sobre la aritmética modular se definen como:

- Suma: $a+\ b \equiv c \ (mod\ n)$, si y solo si $a+\ b = c+\ kn$ donde $k\ \varepsilon\ \mathbb{Z}$.
- Producto: $ab \equiv c \ (mod\ n)$, si y solo si $ab = c+\ kn$ donde $k\ \varepsilon\ \mathbb{Z}$.

Las operaciones de suma y multiplicación cumplen las propiedades asociativa y conmutativa, además que se poseen inversos aditivo y multiplicativo. Para el caso de la suma, se cuenta con elementos simétricos, y en la multiplicación no necesariamente se cuenta con simétricos.

Definición 3 *Sea $a \in \mathbb{Z}_n$. El inverso multiplicativo de a módulo n, denotado por a^{-1}, es el entero $a^{-1} \in \mathbb{Z}_n$ tal que $1 \equiv aa^{-1} mod\ p$. Si a^{-1} existe es único y se dice que a es invertible.*

Definición 4 *Sea un entero $h \in \mathbb{Z}_n$. Se considera un residuo cuadrático h si tiene una raíz cuadrada módulo n, esto es, si existe un valor de $x \in \mathbb{Z}_n$ tal que $x^2 = h\ mod\ p$, de lo contrario h no será residuo cuadrático.*

Dado un número primo impar p y un entero cualquiera h, se define el símbolo de Legendre como

$$\left(\frac{h}{p}\right) = \begin{cases} 1, \text{si } h \text{ es un residuo cuadrático } mod(p), \\ -1, \text{si } h \text{ no es un residuo cuadrático } mod(p), \\ 0, \text{si } h \text{ es divisible por } p \end{cases} \tag{2.1}$$

Se pueden determinar los residuos cuadráticos utilizando el siguiente criterio.

Criterio de Euler

Sea p un número primo impar y h un entero cualquiera coprimo con p, es decir, $mcd(h,p) = 1$. Entonces,

$$\left(\frac{h}{p}\right) = h^{\frac{p-1}{2}} mod(p) \tag{2.2}$$

Donde al resolver la ecuación 2.2 se obtendrán los valores de 1, -1 y 0, para tomar la decisión de que si es o no residuo cuadrático se utiliza la expresión 2.1.

2.2. Teoría de grupos

Hoy en día, la teoría de grupos es una de las áreas de matemáticas que más aplicaciones tiene. Estas van desde las ciencias exactas hasta la música. En las ciencias exactas, las aplicaciones incluyen áreas tales como geometría algebraica, teoría de números y topología algebraica; en física y química su aplicación tiene lugar en el estudio de simetrías de las estructuras moleculares, mientras que en la música, una fuente que da cuenta de su aplicación son las técnicas de composición musical.
Un grupo $(G, *)$ es una estructura algebraica formada por un conjunto G no vacío y una operación binaria * en él, tal que satisface las siguientes condiciones::

- La operación de grupo es asociativa. Es decir:

$$a * (b * c) = (a * b) * c, \quad \forall a, b, c \in G$$

- Existe un elemento $e \in G$ único llamado elemento neutro tal que:

$$e * a = a * e = a \quad \forall a \in G$$

- Para todo $a \in G$ existe un elemento a^{-1} único, llamado inverso de a tal que:

$$a* \ a^{-1} = a^{-1} * a = e$$

Definición 5 *Un grupo G es finito si la cantidad de elementos que contiene, denotado por $|G|$, es finita. El número de elementos de un grupo finito es llamado, el orden del grupo.*

Definición 6 *Un grupo G con orden n se dice cíclico, si existe un elemento $g \in G$ tal que para todo $h \in G$, existe algún entero $k \in \{0, 1, \cdots, k-1\}$ que satisface $h = kg$. El elemento g es llamado generador del grupo y se denota por $< g >= G$.*

Los grupos cíclicos son las estructuras algebraicas sobre las que se definen una parte importante de los criptosistemas asimétricos existentes.

2.3. Curva Elíptica sobre $\mathbb{R}$

Definición 7 *Una curva elíptica sobre* $\mathbb{R}$ *es el conjunto de puntos del plano* (x, y) *que cumplen la siguiente ecuación*

$$y^2 = x^3 + ax + b \tag{2.3}$$

Los coeficientes a y b caracterizan unívocamente cada curva. Las siguientes ecuaciones son ejemplos de curvas elípticas.

Si $x^3 + ax + b$ no tiene raíces múltiples, entonces la curva correspondiente, aunado con un punto especial ∞, llamado punto en el infinito, más la operación suma que se definirá más adelante, es lo que vamos a denominar grupo de curva elíptica $E(\mathbb{R})$. Hay que recalcar que ∞ es un punto imaginario situado por encima del eje de abscisas a una distancia infinita, y que por lo tanto no tiene un valor concreto. Veamos ahora que si $x^3 + ax + b$ no tiene raíces múltiples, entonces se cumple que $\Delta = 4a^3 + 27b^2 \neq 0$. A Δ se le conoce como el discriminante de la curva elíptica.

Proposición 1 *Si la expresión* $x^3 + ax + b$ *tiene raíces múltiples, entonces* $\Delta = 4a^3 + 27b^2 = 0$.

Supongamos que la expresión $x^3 + ax + b$ tiene raíces múltiples. Esto implica que

$$x^3 + ax + b = (x - q)^2(x - r).$$

Desarrollando el segundo término:

$$(x - q)^2(x - r) = (x^2 - 2xq + q^2)(x - r) = x^3 - 2qx^2 - rx^2 + q^2x + 2qrx - q^2r.$$

Igualando los coeficientes del mismo grado se tienen las siguientes relaciones:

$$-2q - r = 0$$

$$q^2 + 2qr = a$$

$$-q^2r = b$$

Despejando r en la primera igualdad y sustituyendo su valor en las dos restantes se obtiene:

$$r = -2q$$

$$a = q^2 + 2q(-2q) = q^2 - 4q^2 = -3q^2$$

$$b = -q^2(-2q) = 2q^3$$

Elevando al cubo la segunda expresión y al cuadrado la segunda expresión obtenemos:

$$a^3 = -27q^6$$

$$b^2 = 4q^6$$

Despejando q^6 de ambas expresiones e igualando tenemos:

$$q^6 = \frac{a^3}{-27} = \frac{b^2}{4}.$$

Lo cual implica que:

$$4a^3 = -27b^2,$$

esto es,

$$4a^3 + 27b^2 = 0$$

Sean $P(x_p, y_p)$ y $Q(x_q, y_q)$ puntos de una curva elíptica definida sobre $\mathbb{R}$. Se define la suma entre P y Q de la siguiente manera:

• Si P y Q son puntos distintos los cuales no son negativos uno del otro, se tiene la inclinación de la línea a través de P y Q, de la forma

$$m = \frac{y_p - y_q}{x_p - x_q},$$

Y se puede expresar la suma $P + Q = R$ como sigue:

$$x_R = m^2 - x_p - x_q,$$

$$y_R = -y_p + m(x_p - x_q).$$

Definimos de manera general a las curvas elípticas de la siguiente manera.

Sea K un campo. Una curva elíptica sobre K, es la curva plana sobre K definida por la ecuación de Weierstrass:

$$y^2 = x^3 + ax + b \tag{2.4}$$

Donde $x, y, a, b \in K$

Para poder definir una estructura algebraica de grupo abeliano es necesario incluir un punto, denotado por ∞ y llamado punto en el infinito, que no se encuentra en la curva. Este punto se encuentra situado por encima del eje de las abscisas a una distancia infinita, y que por lo tanto no tiene un valor en concreto.

Si $x^3 + ax + b$ no tiene raíces múltiples, entonces la curva correspondiente, aunado con el punto ∞, más la operación suma definida más adelante, es lo que se denomina grupo de la curva elíptica sobre K, denotado por $E(K)$. Esto es, el conjunto:

$$E(K) = \{(x, y) : x, y \in K, y^2 = x^3 + ax + b\} \cup \{\infty\}$$

forma un grupo abeliano, en donde, ∞ es el elemento identidad del grupo de curva elíptica.

Las siguientes gráficas son ejemplos de curvas elípticas definidas sobre $\mathbb{R}$.

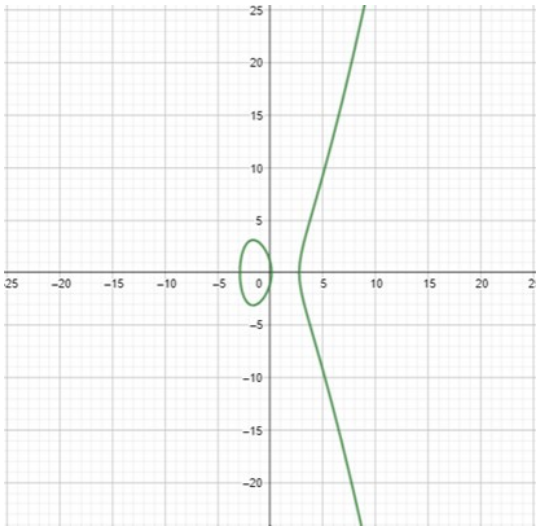

Figura 2.1: Curva elíptica $y^2 = x^3 - 8x + 1$

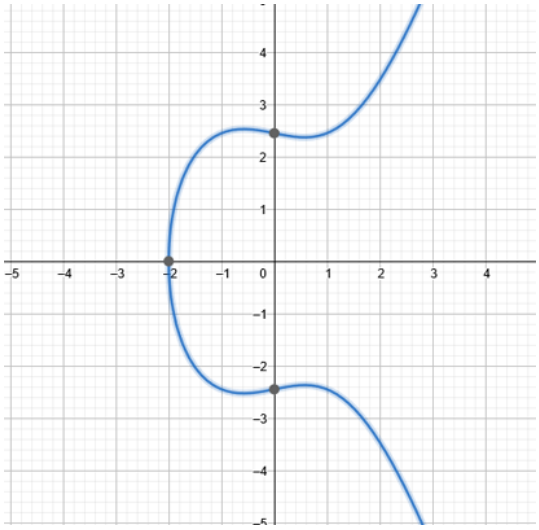

Figura 2.2: Curva elíptica $y^2 = x^3 - x + 6$

A la expresión $\Delta = 4a^3 + 27b^2$ se le conoce como el discriminante de la curva elíptica. Se verifica que para que la curva elíptica no tenga raíces múltiples es necesario que $\Delta \neq 0$. Esta condición permite excluir las curvas elípticas que tengan un punto doble o un pico como lo muestran las siguientes figuras.

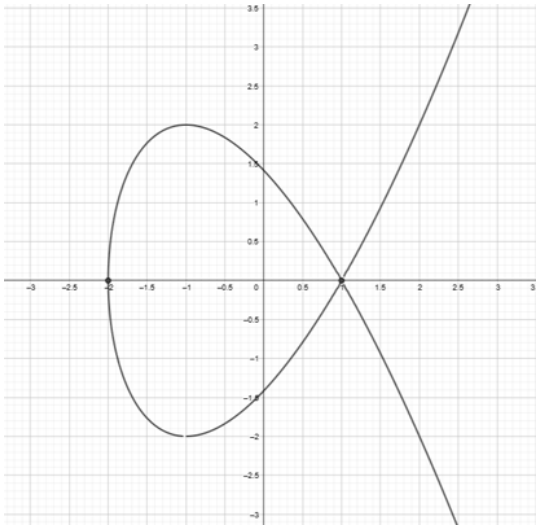

Figura 2.3: Curva elíptica $y^2 = x^3 - 3x + 2$ sobre $\mathbb{R}$

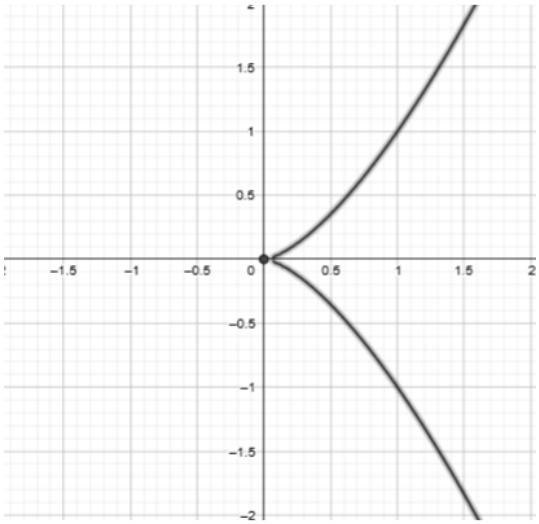

Figura 2.4: Curva elíptica $y^2 = x^3$ sobre $\mathbb{R}$

Trabajar criptografía con curvas elípticas sobre los números reales se puede volver lento e inexacto, debido a los errores de redondeo que puedan existir. En la práctica, el trabajo de criptografía se lleva a cabo con curvas elípticas sobre el campo finito $GF(p)$ pertenecientes a los campos primos y $GF(2^m)$ pertenecientes a los campos de binarios. En este trabajo nos centraremos en las curvas elípticas sobre el campo finito $GF(p)$, con p un número primo.

Recordando que el campo de $GF(p)$ usa los números del 0 al $p-1$ y en cómputo final se obtiene el módulo de p.

Una curva elíptica definida sobre el campo de $GF(p)$, denotada por $E(GF(p))$, esta formada por las variables a y b dentro del campo de $GF(p)$. Las curvas elípticas incluyen todos los puntos de (x, y) que satisface la ecuación de una curva elíptica módulo p. Esto es, una curva elíptica sobre $GF(p)$ tiene por ecuación:

$$y^2 \mod p = (x^3 + ax + b) \mod p \tag{2.5}$$

donde $a, b, x, y \in GF(p)$.

De manera análoga, si $x^3 + ax + b$ contiene factores no repetidos, o equivalentemente si

$$4a^3 + 27b^2 \neq 0 \mod p \tag{2.6}$$

entonces la curva elíptica se puede utilizar para la criptografía. Una curva elíptica sobre el campo de $GF(p)$ tiene los puntos correspondientes en la curva elíptica, junto con un punto especial ∞ , el cual se le llama punto en infinito o punto cero. Son limitados los muchos puntos en las curvas elípticas. La cardinalidad de puntos de una curva elíptica se denota por $\#E(GF(p))$. El número de puntos de una curva elíptica es llamado el orden de la curva.

Ejemplo: 2.3.1 *Consideremos la curva elíptica sobre GF(11). Con a=6 y b=10, la ecuación de la curva elíptica es* $y^2 = x^3 + 6x + 10$. *Los puntos que pertenecen a esta curva son: (0,3), (0,5), (6,3), (6,8), (8,3), (8,8), (9,1), (9,10), (10,5), (10,6) incluyendo a* ∞. *Esto es,* $\#E(GF(p)) = 11$.

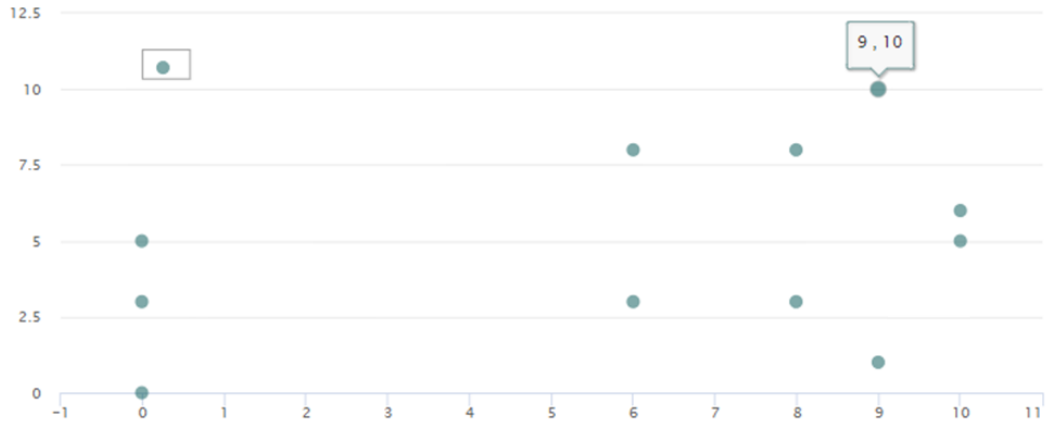

Figura 2.5: $\#E(GF(p))$

Desde el punto de vista algebraico la ley de grupo para una Curva Elíptica representada por la ecuación de Weierstrass (1), se define de acuerdo con las siguientes propiedades:

1. $P_1 + \infty = P_1$
2. Si $P_1 = (x_1, y_1)$, entonces $-P_1 = (x_1, -y_1)$.
3. Sean $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$ puntos de la curva elíptica con $P_1, P_2 \neq \infty$. Entonces si $x_1 = x_2$ pero $y_1 \neq y_2$ o $P_1 = P_2$ y $y_1 = 0$ entonces $P_1 + P_2 = \infty$. ∞. En otro caso $P_1 + P_2 = P_3 = (x_3, y_3)$ con

$$x_3 = m^2 - x_1 - x_2, y_3 = m(x_1 - x_3) - y_1 \tag{2.7}$$

$$m = \begin{cases} (\frac{3x_1^2+a}{2y_1}) & x_1 = x_2 \\ (\frac{y_2-y_1}{x_2-x_1}) & x_1 \neq x_2 \end{cases}$$

La curva elíptica $E(GF(p))$ dotada de la operación suma definida anteriormente forma un grupo abeliano.

Sea P un punto de la curva elíptica $E(GF(p))$. Entonces, kP significa la suma del punto P consigo mismo k-veces, con la suma definida anteriormente.

Es conocido el hecho de que si $\#E(GF(p))$ es un número primo, entonces el grupo $E(GF(p))$ es un grupo cíclico. Este proceso facilita el hecho de que, si una curva elíptica cumple con esta condición, entonces cualquiera de sus puntos diferente del ∞ será un punto generador del grupo.

Capítulo 3

Algoritmos del sistema

Hoy en día se vive en un ambiente donde la interacción con las nuevas tecnologías es tan cotidiana como tener que ir al colegio. No hay cosa que no manejemos a través de un ordenador, ya sea para mandar un mensaje a un ser querido, hacer un trabajo escolar o hasta nuestro mismo trabajo necesita de un ordenador, es claro que la mayoría de las veces es a través de diferentes páginas web.

Estas tecnologías han permitido un sin fin de cosas, como las aplicaciones móviles y aplicaciones web. Es más que un hecho que cualquier documento importante, transferencia bancaria, correos confidenciales hasta la documentación de un caso penal no sea enviada por internet, es por ello por lo que es importante proteger esa información que en términos computacionales llamamos encriptar y la herramienta que nos permite trabajar en el lado del servidor para proteger este tipo de datos es PHP.

Para llevar a cabo la encriptación y desencriptación usando curvas elípticas, es necesario realizar múltiples operaciones que nos permiten establecer los mecanismos de seguridad que se necesitan en el resguardo de la información. Para ello detallamos las operaciones utilizadas para llevar a cabo el objetivo de este trabajo. El desarrollo de este sistema fue utilizando PHP, listaremos con ejemplos los algoritmos utilizados en el sistema.

3.1. Algoritmo de euclides extendido

El algoritmo de euclides, además de encontrar el *mcd* entre dos números, sirve para expresar la combinación lineal de estos, esto es útil pues nos ayuda a calcular el inverso modular de cualquier número.

El resultado que se busca son los coeficientes, los cuales satisfacen dicha ecuación.

$Cx_2 + Dy_2 = 1$ donde x_2, y_2 son los coeficientes que multiplican a C, D para satisfacer la expresión.

Algoritmo 1 Euclides extendido

Entrada: Un número CyD

Salida: Combinación lineal $Cx_2 + Dy_2 = 1$

1: $x1 \leftarrow 0, x2 \leftarrow 1, y1 \leftarrow 1, y2 \leftarrow 0, q \leftarrow 0, r \leftarrow 0$
2: **while** $(D > 0$ OR $D < 0)$ **do**
3: $q \leftarrow floor(\frac{C}{D})$
4: $r \leftarrow (C \text{ MOD } D)$
5: $x \leftarrow x2 - (q \cdot x1)$
6: $y \leftarrow y2 - (q - y1)$
7: $C \leftarrow D$
8: $D \leftarrow r$
9: $C \leftarrow D$
10: $x2 \leftarrow x1$
11: $x1 \leftarrow x$
12: $y2 \leftarrow y1$
13: $y1 \leftarrow y$
14: **EndWhile**
15: **return** $Cx_2 + Dy_2 = 1$

Ejemplo: 3.1.1 *Supongamos que trabajamos sobre* $GF(1087)$ *y deseamos encontrar el inverso de 845, como p es un número primo, todos sus elementos tendrán un inverso.*
Obtenemos la ecuación $845(-274)+1087(213) = 1$, *pero en este punto nos interesa saber el inverso de 845 en este caso es -274, lo cual no nos sirve, lo único que tenemos que hacer es restar -274 a 1087,* $(1087 - 274 = 813)$. *Por lo tanto el inverso multiplicativo de* 845 *en* $GF(1087) = 813$.
Si en la combinación lineal el resultado es positivo se toma el dato obtenido.

Ejemplo: 3.1.2 *El inverso de* 456 *en* $GF(1087)$, *primeramente la ecuación nos da* $456(441) + 1087(-185) = 1$, *por tanto al ser positivo el resultado de* x_2, *el inverso de* 456 *es* 441.

3.2. Adición de dos puntos

El algoritmo que se presenta a continuación nos ayuda a calcular la suma de dos puntos 2.7, los cuales pertenecen a $GF(p)$.

Algoritmo 2 Suma de dos puntos

Entrada: Puntos P(x1,y1) y Q(x2,y2) con $P, Q \in GF(p)$
Salida: P + Q = (x3,y3)

1: **if** (x1 <>x2 AND y1 <>y2) **then**
2: $m1 \leftarrow (y2 - y1)$
3: $m2 \leftarrow eucExt(x2 - x1)$ //referencia al algoritmo de euclides extendido
4: $m \leftarrow (m1 \cdot m2)$
5: $x3 \leftarrow (m^2) - x1 - x2$
6: $y3 \leftarrow (m \cdot (x1 - x3) - y1)$
7: **EndIf**
8: **return** $(x3, y3)$

Cabe mencionar que en la ecuación 2.7 tenemos una parte m^2, la cual no puede ser efectuada directamente ya que existirían casos en los que la división no es exacta dando como resultado un número con decimales, recordemos que estamos trabajando sobre un módulo p en el cual solo existen enteros.

Ejemplo: 3.2.1 *Trabajaremos con* $GF(1087)$, $a = 5$ *y* $b = 19$, *valores que tomaremos para los siguientes ejemplos.*
Tenemos el punto $P = (18, 638)$ *y* $Q = (141, 220)$, *los cuales pertenecen a* $GF(1087)$, *al hacer adición entre estos puntos nos da como resultado,* $P + Q = (406, 168)$.

3.3. Doblado de puntos

Como se observó en el algoritmo 2, si alguno de los puntos P o Q coinciden, es decir, si $x1 = x2$ o $y1 = y2$, entonces estos puntos nos producen ∞.
Pero, supongamos que deseamos sumar un punto con si mismo, si se pretendiera usar el algoritmo 2 no podríamos realizarlo para ello se hace uso de suma de $2P$, que no es otra que sumar $P + P$.

Ejemplo: 3.3.1 *Al utilizar el algoritmo anterior con el punto* $(436, 653)$ *se obtiene como resultado* $(217, 52)$.

Algoritmo 3 Doblado de puntos

Entrada: Punto P(x_1, y_1) con $P \in GF(p)$
Salida: P + P = 2P
$m1 \leftarrow 3(x_1^2 + a)$
$m2 \leftarrow eucExt(2 \cdot y_1)$
$m \leftarrow (m1 \cdot m2)$
$x3 \leftarrow (m^2) - (2 \cdot x_1)$
$y3 \leftarrow m \cdot (x_1 - x_3) - y_1$
return (x_3, y_3)

Como sabemos lo operación principal de las curvas elipticas es la multiplicación de un entero k por un punto P.

Imaginemos que deseamos multiplicar un punto k veces, se haría de la siguiente forma $kP = P + P + P + P + P + ... + k$ veces, pero esto resulta no óptimo, si se tiene un k lo suficientemente grande.

Ejemplo: 3.3.2 *Multiplicaremos el punto* $P(436, 653)$ *con* $k = 15$.

1. $k = 2(436, 653) = (436, 653) + (436, 653) = (217, 52)$
2. $k = 3(436, 653) = (436, 653) + (217, 52) = (371, 528)$
3. $k = 4(436, 653) = (436, 653) + (371, 528) = (586, 480)$
4. $k = 5(436, 653) = (436, 653) + (586, 480) = (266, 93)$
5. $k = 6(436, 653) = (436, 653) + (266, 93) = (362, 422)$
6. $k = 7(436, 653) = (436, 653) + (362, 422) = (530, 640)$
7. $k = 8(436, 653) = (436, 653) + (530, 640) = (689, 18)$
8. $k = 9(436, 653) = (436, 653) + (689, 18) = (278, 205)$
9. $k = 10(436, 653) = (436, 653) + (278, 205) = (883, 171)$
10. $k = 11(436, 653) = (436, 653) + (883, 171) = (201, 594)$
11. $k = 12(436, 653) = (436, 653) + (201, 594) = (531, 202)$
12. $k = 13(436, 653) = (436, 653) + (531, 202) = (203, 289)$
13. $k = 14(436, 653) = (436, 653) + (203, 289) = (147, 293)$
14. $k = 15(436, 653) = (436, 653) + (147, 293) = (582, 459)$

3.4. Multiplicación NAF

Multiplicar un punto k veces de la manera del algoritmo (3), es una solución "óptima" cuando el número entero k no es muy grande. Pero, cuando el número es lo suficientemente grande existe un algoritmo llamado multiplicación NAF, el cual en descomponer el número entero k en $0, 1, -1$, y dependiendo del número binario que retorne se realiza el doblado de puntos o la adición de puntos de los algoritmos (2) y (3). Consta de dos algoritmos el primero que se encarga de retornar en un arreglo la representación binaria del entero k.

Algoritmo 4 NAF

Entrada: Entero k
Salida: Representación binaria de entero k

```
i ← 0
bin ← decbin(k) // función para transformar de decimal a binario
binario ← split(bin) // función para separar por letras en un arreglo
while (k >= 1) do
    if (binario[i] MOD 2 = 1) then
        binario[i] ← 2 - (k MOD 4)
        k ← k − binario[i]
     Else
    binario[i] ← 0
    EndIf
    k ← k/2
    i ← i + 1
 EndWhile
return binario
```

Algoritmo 5 Multiplicación NAF

Entrada: Punto P(x_1, y_1) con $P \in GF(p)$ y un entero k
Salida: $kP(x_1, y_1)$

```
1:  binarioNAF ← NAF(k) // hace referencia al algotimo (4)
2:  Q ← ∞
3:  for (binarioNAF to 0) do
4:      Q ← Q + Q
5:      if binarioNAF[i] = 1 then
6:          Q ← Q + P
7:      EndIf
8:      if binarioNAF[i] = −1 then
9:          Q ← Q − P
10:     EndIf
11: EndFor
12: return Q
```

Ejemplo: 3.4.1 *Tomaremos como ejemplo el mismo punto del algoritmo* (3)*, es decir, el punto* $15(436, 653)$.

1. $binarioNAF = -1, 0, 0, 0, 1$
2. $Q = 0$
3. $binarioNAF[4] = 1$
   - *3.1* $Q = 0$
   - *3.2* $Q = (0, 0) + (436, 653) = (436, 653)$
4. $binarioNAF[3] = 0$
   - *4.1* $Q = (436, 653) + (436, 653) = (217, 52)$
5. $binarioNAF[2] = 0$
   - *5.1* $Q = (217, 52) + (217, 52) = (586, 480)$
6. $binarioNAF[1] = 0$
   - *6.1* $Q = (586, 480) + (586, 480) = (689, 18)$
7. $binarioNAF[0] = -1$
   - *7.1* $Q = (689, 18) + (689, 18) = (11, 450)$
   - *7.2* $Q = (11, 450) + (4356, -653) = (582, 459)$

A diferencia del algoritmo (3) que realiza el doblado tantas veces sea el numero entero k, es decir, en este caso como $k = 15$, entonces el punto se suma 15 veces, pero con la multiplicación NAF estos calculos se reducen considerablemente, pues al representar el número k en su forma binaria realizamos menos operaciones.

Para una mejor visualización veamoslo de la siguiente manera:

De manera tradicional tendriamos que sumar 15 veces el punto $(436, 653)$.

$P + P + P + ... + P = 15P$

Con la multiplicación NAF hacemos lo siguiente:

$(436, 653) = P$

$(217, 52) = 2P$

$(586, 480) = 4P$

$(689, 18) = 8P$

Si sumamos esos puntos obtenemos $P + 2P + 4P + 8P = 15$P.

3.5. Símbolo de Legendre

Para hablar del símbolo de Legendre debemos mencionar a los residuos cuadráticos, supongamos que tenemos un primo p, y se considera un residuo cuadrático h si tiene raíz cuadrada módulo p . Existen 3 casos:

$$(\tfrac{h}{p}) = \begin{cases} 1 & \text{si } h \text{ es un residuo cuadrático módulo} \\ 0 & \text{si } h \text{ es divisible por p} \\ -1 & \text{si } h \text{ no es un residuo cuadrático módulo p} \end{cases}$$

Algoritmo 6 Símbolo de Legendre

Entrada: Número entero impar $n \geq 3$
Número entero a tal que $0 \leq a < n$
Salida: Símbolo de Legendre
if $(a = 0)$ **then**
return 0
EndIf
if $(a = 1)$ **then**
return 1
EndIf
$a1 \leftarrow \frac{a}{2^e}$ // calcular a1, donde e = funcExp(a)
if (e es par) **then**
$s \leftarrow 1$
Else
if $(n \equiv 1(MOD8)$ || $n \equiv 7(MOD8))$ **then**
$s \leftarrow 1$
Else
if $(n \equiv 3(MOD8)$ || $n \equiv 5(MOD8))$ **then**
$s \leftarrow -1$
EndIf
if $(n \equiv 3(MOD4) AND a1 \equiv 3(MOD4))$ **then**
$s \leftarrow -s$
EndIf
$n1 \leftarrow n\ MOD\ a1$
if $(a1 = 1)$ **then**
return s
Else
return $s \cdot$ legenndre$(n1, a1)$
EndIf

Se observa que el símbolo de Legendre necesita de otro algoritmo ($funExp$), la cual nos ayuda a calcular la variable e, además se debe mencionar que el algoritmo (6) es recursivo.

3.6. Función Exponencial

Algoritmo 7 Función Exponencial

Entrada: Número entero a
Salida: Número entero e

```
e ← 0
while (a MOD 2 = 0) do
    a ← a/2
    e ← e + 1
EndWhile
return e
```

Ejemplo: 3.6.1 *Definiremos a x como a todos los números menores a p, es decir, $x \leq p-1$ y r como el residuo. Para no saturar las páginas de información solo mostraremos los primeros y los últimos datos.*

Lista de residuos $p = 1087$

x	r	13	-1	26	-1	1064	1
1	1	14	-1	27	-1	1065	1
2	1	15	1	28	-1	1066	-1
3	-1	16	1	29	-1	1067	1
4	1	17	1	30	1	1068	1
5	-1	18	1	1056	1	1069	-1
6	-1	19	-1	1057	-1	1070	-1
7	-1	20	-1	1058	1	1071	-1
8	1	21	1	1059	1	1072	-1
9	1	22	-1	1060	1	1073	1
10	-1	23	-1	1061	1	1074	1
11	-1	24	-1	1062	-1	1075	1
12	-1	25	1	1063	1	1076	1

1077	1	1080	1	1083	-1	1086	-1
1078	-1	1081	1	1084	1		
1079	-1	1082	1	1085	-1		

3.7. Raíz cuadrada

En las operaciones de la aritmética modular se encuentra la raíz cuadrada, la cual solo puede ser calculada en un módulo p, si este número tiene residuo cuadrático.

Algoritmo 8 Raíz cuadrada

Entrada: Número entero a tal que $1 \leq a \leq p-1$ y un primo p
Salida: Raíces de a

1: **if** $(legendre(a,p) = -1)$ **then** //legendre() hace referencial al algoritmo 6
2: **return** No tiene raíz cuadrada
3: **Else**
4: **while** (aux <>-1) **do**
5: $temp \leftarrow temp + 1$
6: aux $\leftarrow legendre(temp, p)$
7: b $\leftarrow temp$
8: **EndWhile**
9: s $\leftarrow funcExp(p-1)$
10: $auxExp \leftarrow 2^s$
11: t $(\frac{(p-1)}{auxExp})$
12: Ia $\leftarrow eucExt(a,p)$
13: c $\leftarrow calcularExp(b,t,p)$// calcularExp() hace referencia al algoritmo (9)
14: r $\leftarrow CalcularExp(a, (\frac{(t+1)}{2}), p)$
15: **for** (i = 0 to s-1) **do**
16: base $\leftarrow r^2 \cdot Ia$
17: exp $\leftarrow 2^{s-i-1}$
18: d $\leftarrow CalcularExp(base, exp, p)$
19: **if** $((d+1)MODp) = 0$ **then**
20: mul $\leftarrow r + c$
21: r $\leftarrow mul$
22: **EndIf**
23: **EndFor**
24: raiz1 $\leftarrow r$
25: raiz2 $\leftarrow r \cdot (-1)$
26: **if** $(raiz1 < raiz2) = 0$ **then**
27: res $\leftarrow raiz1$
28: **Else**
29: res $\leftarrow raiz2$
30: **EndIf**
31: **return** *res*

3.8. Exponenciación modular

Algoritmo 9 Exponenciación modular

Entrada: Número entero a, Número entero k, Número entero primo p
Salida: Exponencial del número a

```
1: exp = 1
2: xp ← a MOD z
3: while (k >0 ) do
4:     a ← a/2
5:     e ← e + 1
6:     if (k MOD 2) <>0 then
7:         exp ← (exp · xp MOD p)
8:         r ← mul
9:     EndIf
10:    k ← k/2
11: EndWhile
12: return exp
```

Al resolver la raíz cuadrada nos retorna dos soluciones, las cuales son correctas pero en términos de eficiencia tomaremos la raíz más pequeña.

Ejemplo: 3.8.1 *Si deseamos calcular la raíz de* $a = 133$ *con* $p = 1087$*, obtendremos dos soluciones* $r1 = 868$ *y* $r2 = 219$*, donde* $r1$ *y* $r2$ *son las raíces de a. Así que la solución que tomaremos en este caso es* $r2$*.*

Cabe recordar que solo existe la raíz cuadráda, si el número tiene símbolo de Legendre igual a 1. El siguiente ejemplo calcula las raíces que tienen como residuo 1, tomando como r a los residuos y como R a las raíces cuadrádas.(De igual manera se calculan las primeras y últimas raíces).

Ejemplo: 3.8.2 *Lista de raíces cuadrádas* $p = 1087$

r	R														
1	*(1,1086)*	*16*	*(4,1083)*	*33*	*(318,769)*	*43*	*(206,881)*	*1036*	*(615,472)*
2	*(33,1054)*	*17*	*(136,951)*	*34*	*(140,947)*	*49*	*(1080,7)*	*1039*	*(114,973)*
4	*(2,1085)*	*18*	*(988,99)*	*35*	*(1040,47)*	*50*	*(922,165)*	*1040*	*(93,994)*
8	*(66,1021)*	*21*	*(783,304)*	*36*	*(1081,6)*	*55*	*(514,573)*	*1041*	*(211,876)*
9	*(1084,3)*	*25*	*(1082,5)*	*39*	*(81,1006)*	*57*	*(738,349)*	*1042*	*(836,251)*
15	*(754,333)*	*30*	*(968,119)*	*41*	*(1013,74)*	*60*	*(421,666)*	*1043*	*(480,607)*
		32	*(132,955)*	*42*	*(838,249)*	*64*	*(8,1079)*	*1047*	*(87,1000)*

1049	*(168,919)*	*1060*	*(458,629)*	*1067*	*(892,195)*	*1076*	*(240,847)*	*1084*	*(572,515)*
1050	*(158,929)*	*1061*	*(151,936)*	*1068*	*(598,489)*	*1077*	*(587,500)*		
1056	*(670,417)*	*1063*	*(794,293)*	*1073*	*(735,352)*	*1080*	*(714,373)*		
1058	*(328,759)*	*1064*	*(764,323)*	*1074*	*(861,226)*	*1081*	*(397,690)*		
1059	*(341,746)*	*1065*	*(311,776)*	*1075*	*(57,1030)*	*1082*	*(446,641)*		

Ahora tenemos los algoritmos principales de nuestro sistema, ahora procederemos con un ejemplo detallado, el problema principal en está altura es, calcular el número de puntos de la curva veremos dos formas. La primera es calculada con el símbolo de Legendre (Algoritmo 6), y la segunda forma será usando el algoritmo Baby-step giant-step el cual consiste en un algoritmo que reduce considerablemente las operaciones, que con ayuda del teorema de Hasse limita el rango donde buscaremos el punto generador de la curva.

3.9. Punto generador con residuos cuadráticos

El primer paso consiste en proporcionar un número primo p y los coeficientes a y b, para formar la curva elíptica de la forma 2.5 cumpliendo la condición (2.6). Para mostrar este paso, consideremos los coeficientes $a = 9$, $b = 13$ y el número primo $p = 19$. Con estos parámetros se obtiene la curva:

$$y^2 = (x^3 + 9x + 13) \quad \text{mód } (19) \tag{3.1}$$

cumpliendo con la condición (2.6).

Para el siguiente paso se necesita determinar el número de puntos con los que cuenta la curva elíptica, $\#E(GF(p))$. Utilizamos el cálculo de los residuos cuadráticos para determinar $\#E(GF(p))$. Para ello, el número de Legendre nos apoya para determinar los puntos que contienen residuos cuadráticos(Algoritmo 6). Una alternativa para el cálculo del símbolo de Legendre es mediante el criterio de Euler utilizando la siguiente fórmula:

$$\frac{x}{p} = x^{\frac{(p-1)}{2}} \quad \text{mód } (p) \tag{3.2}$$

en el que, al resolver (Euler) se obtienen los valores 1, -1 y 0, y para determinar si es o no un residuo cuadrático se utiliza (Algoritmo 6). A continuación, se muestra el algoritmo, del número de Legendre. Tomando a x como un número entero positivo cualquiera menor a p, z representa el resultado de evaluar en la ecuación $y^2 = (x^3 + 9x + 13)$ mód (19) y r es el residuo cuadrático

aplicando el símbolo de Legendre, se obtiene la tabla 3.1.

x	z	residuo
0	13	-1
1	4	1
2	1	1
3	10	-1
4	18	-1
5	12	-1
6	17	1
7	1	1
8	8	-1
9	6	1
10	1	1
11	18	-1
12	6	1
13	9	1
14	14	-1
15	8	-1
16	16	1
17	6	1
18	3	-1

Tabla 3.1: Residuos Cuadráticos.

Una vez calculado el número de Legendre, se conoce la cantidad de puntos que tiene la curva, que en nuestro caso es $\#E(GF(p)) = 21$. Ahora para calcular todos los puntos de la curva se toman los valores mostrados en la tabla 1 y se utiliza la fórmula $H = (x - p)^2$ dando como resultado la tabla 3.2.

x	H
0	0
1	1
2	4
3	9
4	16
5	6
6	17
7	11
8	7
9	5
10	5
11	7
12	11
13	17
14	6
15	16
16	9
17	4
18	1

Tabla 3.2: coordenadas y.

Cuando obtenemos los valores de x, z, y H en las tablas (3.1) y (3.2) respectivamente, se aplica la fórmula 3.2 para encontrar todos los puntos que pertenecen a la curva $E(GF(p))$. Para esto, se reciben dos arreglos donde tenemos almacenados los datos de las tablas anteriores, primeramente, se toma el primer valor de la tabla 3.1, es decir, el valor z, y si algún valor de z coincide con un valor de H en la tabla 3.2, entonces el punto está formado por el valor de x de la tabla 3.1 y el valor de y de la tabla 3.2.

Algoritmo 10 Cálculo de puntos de GF(p)

Entrada: ArrayXZ,ArrayYH
Salida: Puntos de generador

```
1:  count ← 0
2:  for (i = 0 to length ArrayXZ) do
3:      z ← ArrayXZ[i]->getZ()
4:      for (j = 1 to length ArrayYH) do
5:          if (z = arrayYH[j]->getH()) then
6:              arrayPoint[count] ← (arrayXZ[i]->getX(),arrayYH[j]->getY())
7:              count ← count + 1
8:              if ( isGen(arrayPoint[count]) = 0 ) then
9:                  return arrayPoint[count]
10:             EndIf
11:         EndIf
12:     EndFor
13: EndFor
```

Al hacer una modificación al algoritmo anterior se puede calcular el orden de todos los puntos de la curva. Para fines de ilustración en nuestro ejemplo se calcularon los órdenes de todos los puntos de la curva obteniendo la tabla 3. En la práctica esto no es necesario, puesto que sería demasiado costoso, ya que al ser un método que necesita calcular el símbolo de Legendre desde 0 hasta $p-1$ este se vuelve inviable a medida que el p crece.

x	z	residuo
1	2	21
1	17	21
2	1	7
2	18	7
6	6	21
6	13	21
7	1	21
7	18	21
9	5	21
9	14	21
10	1	3
10	18	3
12	5	7
12	14	7
13	3	21
13	16	21
16	4	7
16	15	7
17	5	21
17	14	21

Tabla 3.3: Orden de cada punto de la curva elíptica.

El segundo método para calcular el número de puntos de una curva, es con el algoritmo baby-step giant-step, como se habia mencionado este hace uso del Teorema de Hasse, el cual nos permite calcular el rango en el que se encuentra en número de puntos de la curva. El intervalo queda expresado de la siguiente forma:

$$[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}] \tag{3.3}$$

Esto reduce considerablemente los cálculos pues nos dice que el número buscado se encuentra en a lo más $4\sqrt{p}$.

Algoritmo 11 Baby-steps giant-steps

Entrada: a,b, p

Salida: Número de puntos de la curva

1: Generar un punto aleatorio $P \in GF(p)$
2: Calcular m que pertenece a Hasse tal que $mP = 0$, si m es el único número de ese tipo en el intervalo, entonces $m = \#E(Fp)$
3: Calcular Baby steps : calcular lista desde 0 hasta s = $\sqrt[4]{p}$, multiplicando el punto P hasta s. Tomar en cuenta los inversos de cada número.
4: Calcular Giant steps: Calcular $Q = [2s]P$ luego $R, R \pm Q, R \pm 2Q ... R \pm tQ$ donde t = $\sqrt{p}/(2s)$
5: Buscar hasta que donde i, $R - iQ = jP$ j son las varibles que estan en los ciclos.
6: Calculamos $m = p + 1 - (2s)i - j$
7: Entonces obtenemos que mP = 0.
8: return m

Como nota adicional para calcular el número de puntos de una curva se puede implementar el algoritmo de Schoff que es de un tiempo polinómico. Sin embargo, su teoría e implementación queda fuera del alcance de este trabajo, para obtener detalles de este algoritmo se puede consultar [6].

Capítulo 4

Aplicación del sistema

La implementación de la librería esta basada en los conceptos matemáticos asi como en los algoritmos de los capítulos 1 y 2 respectivamente. En este último capítulo llevaremos un ejemplo donde un usario ingresa su usario y contraseña que, aunque para él solo es ingresar sus datos en este apartado se mostrará todo el proceso que es llevado acabo para cifrar y descifrar la información.
Para realizar el cifrado de los datos, se debe contar con los siguientes parámetros: el número primo p, los coeficientes a y b, la cardinalidad de puntos de la curva elíptica $\#E(GF(p))$, un punto generador G de la curva, valores M y h con $Mh <p$, $llaveSA$ (llave secreta del usuario A), $llaveSB$ (llave secreta del usuario B) en donde mcd(llaveSA,llaveSB)=1. Estos parámetros son utilizados por el cifrado de El Gamal [7]. Entonces tenemos los parámetros:

4.1. Parámetros de la curva

$p = 500009$

$a = 15567$

$b = 7896$

$\#E(GF(p)) = 499879$ $G = (241479, 71146)$

$M = 456$

$h = 123$

$LlaveSA = 24528$

$LlaveSA = 11923$

Obtenemos la ecuación:

$$y^2 = x^3 + 15567x + 7896 \quad \text{mód } 500009 \tag{4.1}$$

Supongamos que tenemos un usuario que se va a registrar en una plataforma y lo que interesa cifrar es la contraseña. Para nuestro caso la contraseña es "carlos123",el usuario ingresa sus datos en un formulario, Figura 4.1.

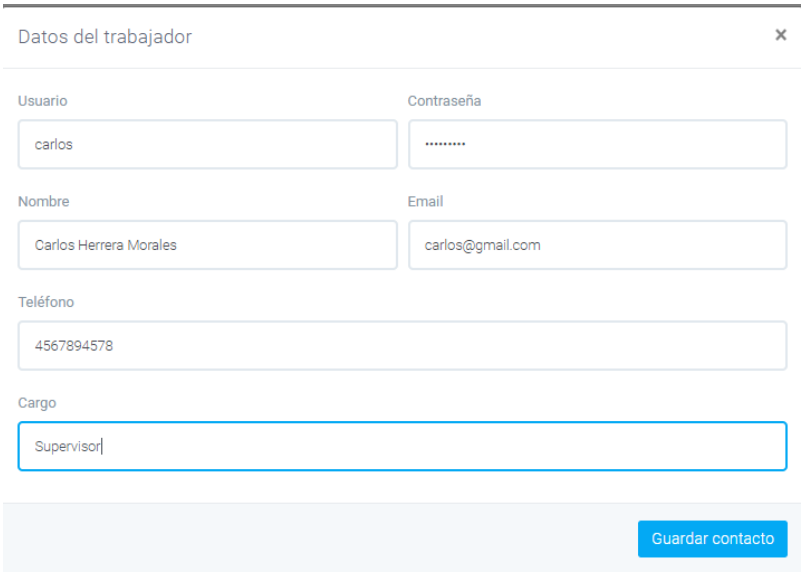

Figura 4.1: Registro del usuario

Podemos cifrar la información con el algoritmo del Gamal, que explicaremos a detalle.

4.2. Codificación del mensaje (Paso 1)

Debemos codificar el mensaje separando cada letra del mensaje en un arreglo, pues cada una se cifrará individualmente, usaremos la siguiente formula $x = ascii(c)(h) + j$, nos apoyaremos del código ascii que comprende del 0 al 255. Entonces tenemos que la letra c es igual 123 y sustituyendo $x = ascii(c)(h) + j = 99(123) + 1 = 12178$ dicho resultado es sustituido en la ecuación 4.1, esto es, $y^2 = (x^3 + 15567x + 7896) = (12178)^3 + 15567(12178) + 7896 = 334992$, luego debemos resolver la ecuación calculando la raíz cuadra, pero dado que no existe valor de y que cumpla con esta ecuación, aumentamos $j = 2$.

4.2.1. Paso 1.1

$x = ascii(c)h + j = 99(123) + 2 = 12179$, y se sustituye este resultado en la ecuación de la curva elíptica, esto es, $y^2 = (x^3 + 15567x + 7896) = (12179)^3 + 15567(12179) + 7896 = 290136$ este número si tiene raíz cuadrada que es igual a 161435. Por tanto, la codificación de la letra c es (12179 ,161435). En las siguientes imágenes muestran los códigos de la raíz cuadrada (figura 4.2) y el de codificación (figura 4.3), respectivamente.

```
function raiz($a,$p){
    $aux=0;$auxc=0;$b = 0;$s=0;$t;$Ia=0;$c=0;$r=0;$d=0;$base=0;$exponente=0;$aux_exp;$ban=0;
    $ban=$this->Simjacobi($a,$p);
    $res=0;
    if ($ban == -1) {
        return -1;
    }else {
          while($aux != -1){
             $auxc++;
             $aux=$this->Simjacobi($auxc, $p);
             $b=$auxc;
          }
          $s=$this->funcion_exponencial($p-1);
          $aux_exp=pow(2,$s);
          $t= (($p-1)/$aux_exp);
          $Ia=$this->euc_ext($a,$p);
          $c=$this->CalcularExp($b, $t, $p);
          $r=$this->CalcularExp($a, (($t+1)/2), $p);
          for($i=1;$i<=$s-1;$i++){
             $base=$r*$r*$Ia;
             $exponente= pow(2,$s-$i-1);
             $d=$this->CalcularExp($base,$exponente,$p);
             if(fmod(($d+1),$p)==0){
                $mul=$r*$c;
                $r=$this->comprobarMod($mul);
             }

          }
          $raiz1 = $this->comprobarMod($r);
          $raiz2 = $this->comprobarMod(-1*$r);
          $c = $this->CalcularExp($c, 2, $p);
          if ($raiz1>$raiz2) {
             $res = $raiz2;
          }else{
             $res=$raiz1;
          }
    }
    return $res;
}
```

Figura 4.2: Raíz cuadrada

```
function codificar($mensaje){
    $puntos = array();
    $puntosEncriptados = array();
    $conP=0;
    $letras;

    $x;$y;$j;$YR;
    $letras =  str_split($mensaje);
    for ($i = 0; $i < count($letras) ; $i++) {
            $j=1;
             while(true){
                $x = $this->comprobarMod((ord($letras[$i]) * $this->h) + $j);
                $y = $this->comprobarMod( pow($x, 3)+($this->a*$x)+$this->b);
                $YR=$this->raiz($y,$this->p);
                if ($YR == -1) {
                  $j++;
                }else{
                  $puntos[$conP]=$x;
                  $puntos[$conP+1] = $YR;
                  $conP+=2;
                  break;
                }
             }
      }
      return $puntos;
  }
```

Figura 4.3: Codificar

Repetimos este proceso para cada una de las letras y números obtenemos:

$c = (12179, 161435)$

$a = (11936, 218659)$

$r = (14023, 101746)$

$l = (13285, 115296)$

$o = (13656, 29398)$

$s = (14147, 176240)$

$1 = (6031, 183341)$

$2 = (6152, 153375)$

$3 = (6277, 52620)$

4.3. Calcular llave pública de A y B (Paso 2)

Para calcular la llave pública de A. se multiplica la llave secreta de esta por el punto generador G, dando como resultado dos puntos, el mismo proceso se hace para calcular la llave pública de B.
Llave pública de A:
$llavePA = LlaveSA * G = 24528(241479, 71146) = (253513, 78497)$, entonces la pareja es igual: $A = (24528, (253513, 78497))$.
Llave pública de B:
$llavePB = LlaveSB * G = 11923(241479, 71146) = (339894, 358573)$, entonces la pareja es igual $B = (4562, (339894, 358573))$.

4.4. Cifrado de datos (Paso 3)

. Ciframos la letra c eligiendo un entero aleatorio k = 205887 y multiplicamos ese número por el punto G, esto es: $kG = 205887(241479, 71146) = (45235, 155942)$, ahora bien, sumando la codificación de la letra $c + k(llavePB)$ se obtiene: $(12179, 161435) + 205887(339894, 358573) = (12179, 161435) + (237547, 189319) = (493092, 311065)$.
Una vez que obtenemos este resultado basta con unir los resultados para que la pareja de coordenadas quede como: $((45235, 155942), (493092, 311065))$.Repetimos estos pasos para cada uno de los caracteres restantes, se obtuvieron los siguientes resultados:

Letra	coordenada	k
c	(45235,155942,493092,311065)	205887
a	(464947,454208,261541,436656)	425387
r	(1424,194779,211919,45374)	294652
l	(91384,133847,124363,495034)	258306
o	(22940,216821,391383,182458)	99447
s	(30055,298941,323275,69773)	340580
1	(220188,341581,147194,45165)	162281
2	(254288,121557,358833,451256)	470133
3	(310143,142529,222099,59930)	92723

Tabla 4.1: Puntos encriptados

Ahora veremos el código que se uso para encriptar los datos, como se menciona primero calcularemos la llave pública tanto de A y B con ayuda de la multiplicación NAF (Algoritmo 5), a continuación generamos enteros k aleatorios para multiplicarlos y asi obtener los puntos correpondientes de la tabla 4.1.

```
function encriptacion($puntos){
  $PmaskPb = array();
  $mulA = $this->mulNaf($this->puntoGenerador->getX(),$this->puntoGenerador->getY(),$this->llaveSA);
  $parejaA=array($this->llaveSA,$mulA[0],$mulA[1]);

  $mulB = $this->mulNaf($this->puntoGenerador->getX(),$this->puntoGenerador->getY(),$this->llaveSB);
  $parejaB=array($this->llaveSB,$mulB[0],$mulB[1]);


 $puntosCifrados=array();
  for ($i = 0; $i <count($puntos) ; $i+=2) {
    $k = rand(1,  $this->p-1);
    $kLlaveP = $this->mulNaf($this->puntoGenerador->getX(),$this->puntoGenerador->getY(),$k);
    $kPb = $this->mulNaf( $parejaB[1], $parejaB[2],$k);
    $Q=array($puntos[$i],$puntos[$i+1]);
    $PmaskPb = $this->sumaDosP($Q,$kPb);
    array_push($puntosCifrados,$kLlaveP[0],$kLlaveP[1],$PmaskPb[0],$PmaskPb[1] );


  }
    return $puntosCifrados;
}
```

Figura 4.4: Encriptación de datos

Al tener todos los puntos cifrados solo bastará con mandarlos a una base de datos para que el usuario quede registrado, como se muestra en la Figura 4.5.

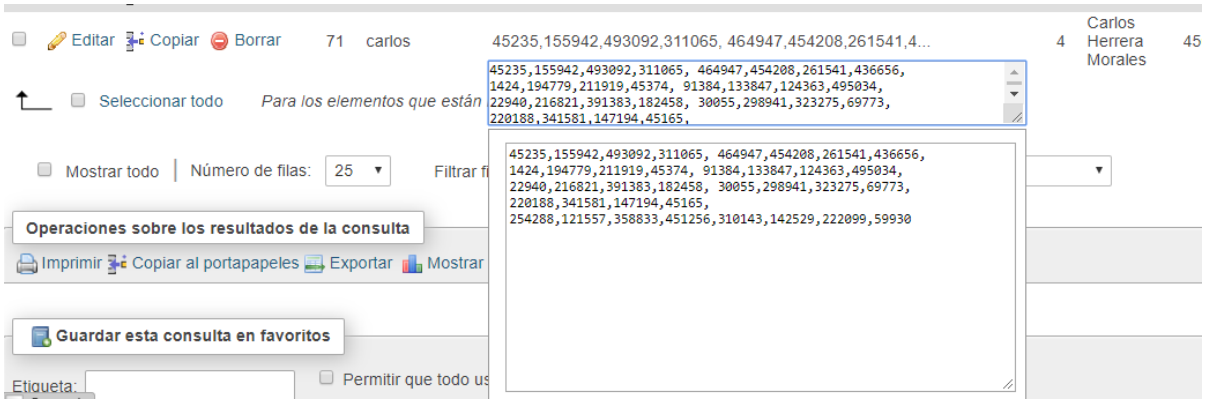

Figura 4.5: Puntos almacenados en una base de datos

El usuario una vez registrado desea entrar a la plataforma para ello debe proporcionar sus datos con los cuales se registro.

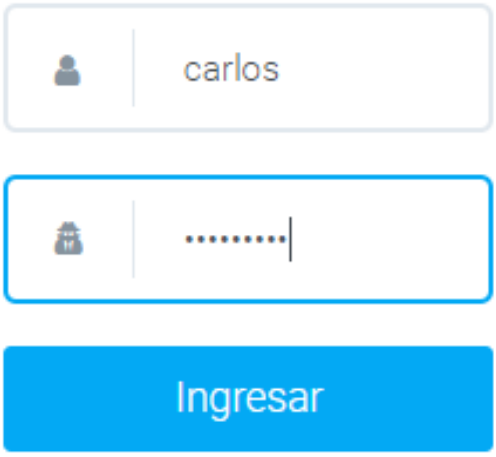

Figura 4.6: Encriptación de datos

Ahora debemos verificar que el usuario ingreso la contraseña correcta para ello debemos descifrar esa información.

4.5. Descifrar datos (Paso 1)

Como primer paso para descifrar la información tomaremos el primer punto que es $(45235, 155942, 493092, 311065)$, tomando la primera pareja y multiplicándola por la llave secreta de B, esto es, $11923(45235, 155942) = (237547, 189319)$

4.6. Resta de puntos (Paso 2)

Se resta el resultado del paso 1 con la segunda pareja, recordando que $(x1, y1)$ - $(x2, y2) = (x1, y1)+$ $(x2, -y2)$ Por lo que se obtiene:
$(493092, 311065) + (237547, -189319) = (12179, 161435)$.

4.7. Decodificar puntos(Paso 3)

Se realiza la decodificación usando la fórmula $\frac{(x+1)}{h}$. Sustituyendo los valores tenemos:
$\frac{(12179-1)}{123} = 99{,}00813$.
Redondeando el resultado para que este quede solo en 99, y como sabemos el número 99 en el código ascii corresponde a la letra c, se obtiene el primer descifrado. Procedemos así para cada uno de los puntos.

```
function descifrar($puntos){
  $resMulB=array();
  $resSuma=array();
  $puntosDec=array();
  $P = array();
     for ($i = 0; $i <count($puntos) ; $i+=4) {
          $resMulB = $this->mulNaf( $puntos[$i], $puntos[$i+1],$this->llaveSB);
          $P[0] = $puntos[$i+2];
          $P[1] = $puntos[$i+3];
          $Q = array($resMulB[0],$resMulB[1]*-1);
          $resSuma = $this->sumaDosP($P,$Q);
          array_push($puntosDec,$resSuma[0],$resSuma[1] );
       }
       echo "puntos decodificados <br>";
       echo var_dump($puntosDec);
       $palabra="";$res;
       for ($i = 0; $i <count($puntosDec)  ; $i++) {
         if (fmod($i, 2)==0) {
           $res = floor(($puntosDec[$i]-1)/ $this->h);
           $palabra .= chr($res);
          }
   }
   return $palabra;
   }
```

Figura 4.7: Descifrar datos

4.8. Conclusiones

Se muestran los conceptos matemáticos para la realización de los algoritmos de cifrado y descifrado usando criptografía en curvas elípticas. Dado que actualmente las plataformas móviles o web recaban mucha información confidencial, se observó la necesidad de aportar un método de cifrado de esta información para que no pueda ser utilizada de manera incorrecta. Al hacer uso de una programación en PHP, se propone una librería que puede ser implementada para estos propósitos. La dificultad que se presenta es la de calcular el número de puntos que contiene una curva elíptica, para ello será necesario establecer la programación adecuada para que logremos contar con este dato de una manera eficiente y poder trabajar con valores de números primos aún más grandes. Es por ello que, como trabajo a futuro se pueda realizar la implementación del algoritmo de Schoff para este propósito.

Apéndice A

Lista de puntos

Se listan algunos puntos de la ecuación $y^2 = x^3 + 15567x + 7896 \ mod \ 500009$

Punto	Orden
(2 , 39038)	499879
(6 , 101514)	499879
(7 , 117208)	499879
(9 , 148728)	499879
(13 , 212464)	499879
(14 , 228578)	499879
(16 , 261064)	499879
(17 , 277448)	499879
(21 , 344064)	499879
(22 , 361018)	499879
(23 , 378104)	499879
(26 , 430214)	499879
(32 , 38799)	499879
(34 , 76469)	499879
(35 , 95607)	499879
(37 , 134519)	499879
(38 , 154305)	499879
(45 , 299527)	499879
(52 , 457979)	499879
(53 , 481815)	499879
(54 , 5960)	499879
(58 , 105876)	499879
(60 , 157898)	499879
(64 , 266310)	499879
(65 , 294358)	499879
(70 , 440568)	499879
(71 , 471046)	499879
(74 , 65051)	499879
(75 , 97269)	499879
(86 , 482687)	499879
(87 , 20692)	499879
(90 , 137890)	499879
(91 , 178028)	499879
(93 , 259948)	499879
(94 , 301742)	499879
(97 , 430532)	499879
(98 , 474618)	499879
(99 , 19283)	499879
(100 , 64551)	499879
(102 , 156893)	499879
(104 , 251683)	499879
(108 , 448799)	499879
(109 , 499683)	499879
(110 , 51212)	499879
(111 , 103410)	499879
(114 , 264024)	499879
(115 , 318922)	499879
(118 , 487780)	499879
(119 , 45465)	499879
(121 , 163001)	499879
(122 , 222855)	499879
(123 , 283441)	499879
(124 , 344765)	499879
(125 , 406833)	499879
(126 , 469651)	499879
(127 , 33216)	499879

(130 , 228534) 499879	(190 , 324455) 499879	(259 , 413350) 499879
(131 , 295192) 499879	(195 , 458156) 499879	(261 , 350077) 499879
(133 , 430872) 499879	(196 , 88375) 499879	(262 , 70782) 499879
(135 , 69735) 499879	(197 , 219779) 499879	(264 , 16923) 499879
(137 , 211847) 499879	(198 , 352365) 499879	(272 , 365336) 499879
(142 , 81608) 499879	(199 , 486139) 499879	(273 , 103663) 499879
(143 , 158094) 499879	(200 , 121098) 499879	(275 , 85246) 499879
(145 , 313646) 499879	(202 , 394640) 499879	(277 , 73429) 499879
(146 , 392724) 499879	(203 , 33217) 499879	(278 , 320015) 499879
(150 , 217847) 499879	(209 , 390512) 499879	(280 , 318188) 499879
(151 , 301365) 499879	(212 , 336003) 499879	(282 , 323081) 499879
(154 , 57370) 499879	(215 , 292942) 499879	(283 , 78058) 499879
(155 , 144548) 499879	(216 , 447830) 499879	(290 , 410813) 499879
(157 , 321700) 499879	(218 , 261491) 499879	(294 , 496247) 499879
(160 , 94499) 499879	(220 , 80384) 499879	(298 , 109896) 499879
(162 , 281161) 499879	(221 , 241812) 499879	(300 , 177429) 499879
(164 , 471711) 499879	(223 , 68643) 499879	(311 , 428843) 499879
(165 , 68450) 499879	(224 , 234067) 499879	(312 , 235498) 499879
(167 , 264922) 499879	(225 , 400835) 499879	(313 , 44025) 499879
(168 , 364658) 499879	(227 , 238418) 499879	(314 , 354439) 499879
(169 , 465402) 499879	(231 , 429985) 499879	(315 , 166728) 499879
(175 , 91352) 499879	(232 , 106320) 499879	(316 , 480916) 499879
(176 , 199320) 499879	(233 , 284056) 499879	(318 , 114968) 499879
(177 , 308344) 499879	(234 , 463190) 499879	(320 , 256661) 499879
(178 , 418430) 499879	(237 , 9022) 499879	(324 , 63126) 499879
(181 , 255111) 499879	(240 , 67661) 499879	(327 , 63368) 499879
(183 , 484991) 499879	(245 , 27603) 499879	(328 , 400704) 499879
(184 , 101566) 499879	(247 , 421835) 499879	(332 , 269761) 499879
(185 , 219254) 499879	(248 , 121162) 499879	(333 , 116988) 499879
(187 , 457966) 499879	(252 , 433437) 499879	(334 , 466222) 499879
(188 , 78993) 499879	(257 , 482839) 499879	(339 , 242536) 499879
(189 , 201157) 499879	(258 , 197316) 499879	(340 , 103875) 499879

(342 , 332688) 499879
(344 , 69700) 499879
(346 , 314977) 499879
(349 , 448473) 499879
(350 , 330482) 499879
(356 , 166855) 499879
(360 , 267080) 499879
(361 , 172519) 499879
(364 , 401865) 499879
(365 , 316004) 499879
(367 , 150858) 499879
(368 , 71585) 499879
(371 , 347047) 499879
(372 , 276642) 499879
(373 , 208469) 499879
(374 , 142534) 499879
(378 , 401303) 499879
(384 , 107603) 499879
(386 , 28071) 499879
(387 , 491785) 499879
(388 , 457812) 499879
(389 , 426167) 499879
(390 , 396856) 499879
(392 , 345260) 499879
(397 , 257535) 499879
(399 , 239077) 499879
(400 , 233436) 499879
(402 , 229360) 499879
(403 , 230937) 499879
(404 , 234932) 499879
(405 , 241351) 499879
(407 , 261485) 499879
(408 , 275212) 499879
(410 , 310016) 499879
(411 , 331105) 499879
(415 , 440181) 499879
(420 , 132587) 499879
(423 , 278228) 499879
(425 , 388002) 499879
(426 , 446711) 499879
(427 , 7967) 499879
(432 , 352842) 499879
(433 , 429569) 499879
(437 , 262508) 499879
(438 , 352285) 499879
(440 , 39720) 499879
(441 , 137399) 499879
(442 , 237724) 499879
(443 , 340701) 499879
(444 , 446336) 499879
(445 , 54626) 499879
(447 , 279240) 499879
(448 , 395567) 499879
(450 , 136282) 499879
(451 , 260691) 499879
(452 , 387806) 499879
(453 , 17624) 499879
(456 , 423437) 499879
(461 , 154574) 499879
(462 , 309079) 499879
(465 , 289241) 499879
(467 , 123286) 499879
(478 , 162177) 499879
(484 , 420059) 499879
(486 , 362518) 499879
(488 , 316641) 499879
(489 , 48087) 499879
(493 , 3289) 499879
(494 , 249474) 499879
(497 , 5819) 499879
(499 , 24952) 499879
(500 , 289011) 499879
(501 , 56061) 499879
(503 , 99194) 499879
(504 , 375289) 499879
(506 , 436548) 499879
(507 , 221724) 499879
(508 , 9942) 499879
(509 , 301217) 499879
(510 , 95537) 499879
(511 , 392926) 499879
(512 , 193372) 499879
(514 , 303495) 499879
(515 , 113175) 499879
(517 , 241820) 499879
(521 , 36373) 499879
(522 , 367818) 499879
(526 , 224951) 499879
(527 , 72107) 499879
(528 , 422434) 499879
(534 , 91089) 499879
(536 , 339548) 499879
(539 , 486367) 499879
(540 , 375097) 499879
(541 , 267067) 499879
(546 , 275736) 499879

(547 , 187272) 499879
(548 , 102090) 499879
(549 , 20196) 499879
(551 , 366305) 499879
(552 , 294311) 499879
(553 , 225629) 499879
(554 , 160265) 499879
(555 , 98225) 499879
(557 , 484150) 499879
(558 , 432118) 499879
(559 , 383434) 499879
(562 , 257530) 499879
(565 , 161974) 499879
(567 , 115210) 499879
(568 , 96928) 499879
(572 , 57940) 499879
(574 , 59014) 499879
(575 , 64714) 499879
(580 , 145084) 499879
(581 , 171574) 499879
(585 , 312454) 499879
(587 , 403930) 499879
(591 , 129185) 499879
(592 , 194351) 499879
(597 , 73572) 499879
(598 , 160140) 499879
(599 , 250296) 499879
(601 , 441396) 499879
(603 , 146911) 499879
(608 , 224132) 499879
(610 , 480518) 499879
(613 , 392553) 499879
(614 , 37240) 499879
(617 , 493456) 499879
(618 , 152915) 499879
(621 , 153582) 499879
(623 , 5977) 499879
(627 , 255680) 499879
(632 , 277492) 499879
(634 , 212717) 499879
(636 , 163158) 499879
(642 , 106257) 499879
(643 , 360225) 499879
(644 , 118042) 499879
(645 , 379732) 499879
(646 , 145283) 499879
(649 , 465234) 499879
(652 , 320222) 499879
(657 , 323525) 499879
(658 , 135984) 499879
(660 , 272761) 499879
(661 , 97082) 499879
(665 , 434104) 499879
(666 , 278315) 499879
(667 , 126522) 499879
(668 , 478740) 499879
(670 , 195188) 499879
(672 , 427725) 499879
(673 , 300034) 499879
(675 , 56772) 499879
(678 , 222278) 499879
(679 , 118905) 499879
(682 , 333263) 499879
(683 , 246222) 499879
(686 , 9711) 499879
(688 , 372616) 499879
(689 , 310253) 499879
(690 , 252024) 499879
(691 , 197935) 499879
(694 , 60568) 499879
(695 , 23099) 499879
(696 , 489809) 499879
(699 , 414992) 499879
(700 , 398433) 499879
(701 , 386074) 499879
(702 , 377921) 499879
(707 , 400456) 499879
(708 , 417665) 499879
(709 , 439122) 499879
(710 , 464833) 499879
(712 , 29032) 499879
(713 , 67541) 499879
(714 , 110328) 499879
(716 , 208760) 499879
(717 , 264417) 499879
(718 , 324376) 499879
(719 , 388643) 499879
(721 , 30116) 499879
(722 , 107343) 499879
(723 , 188902) 499879
(726 , 459631) 499879
(729 , 269555) 499879
(733 , 244054) 499879
(734 , 373661) 499879
(736 , 146084) 499879
(737 , 288921) 499879

(738 , 436180) 499879
(739 , 87858) 499879
(743 , 238997) 499879
(745 , 91286) 499879
(746 , 274137) 499879
(747 , 461464) 499879
(748 , 153264) 499879
(754 , 398558) 499879
(755 , 121900) 499879
(759 , 60646) 499879
(760 , 306707) 499879
(761 , 57319) 499879
(763 , 72256) 499879
(764 , 336593) 499879
(765 , 105505) 499879
(770 , 19053) 499879
(771 , 315604) 499879
(776 , 367842) 499879
(777 , 192230) 499879
(784 , 93836) 499879
(785 , 455697) 499879
(786 , 322259) 499879
(790 , 335736) 499879
(791 , 225938) 499879
(793 , 20586) 499879
(797 , 167035) 499879
(798 , 90585) 499879
(801 , 389996) 499879
(803 , 280284) 499879
(804 , 232652) 499879
(808 , 90424) 499879
(810 , 48374) 499879
(811 , 34636) 499879
(814 , 22642) 499879
(818 , 75054) 499879
(819 , 100412) 499879
(822 , 205994) 499879
(823 , 251044) 499879
(824 , 301032) 499879
(825 , 355964) 499879
(828 , 50475) 499879
(829 , 125243) 499879
(831 , 289707) 499879
(832 , 379415) 499879
(833 , 474115) 499879
(834 , 73804) 499879
(836 , 288218) 499879
(839 , 147465) 499879
(840 , 277277) 499879
(841 , 412129) 499879
(843 , 196968) 499879
(845 , 2039) 499879
(846 , 162181) 499879
(848 , 497699) 499879
(849 , 173078) 499879
(850 , 353560) 499879
(851 , 39133) 499879
(852 , 229821) 499879
(854 , 126530) 499879
(855 , 332572) 499879
(857 , 260043) 499879
(858 , 481493) 499879
(859 , 208082) 499879
(860 , 439834) 499879
(861 , 176737) 499879
(863 , 166056) 499879
(866 , 438889) 499879
(870 , 42090) 499879
(872 , 124991) 499879
(873 , 424291) 499879
(874 , 228820) 499879
(876 , 353625) 499879
(879 , 330263) 499879
(881 , 7709) 499879
(883 , 206308) 499879
(884 , 63547) 499879
(885 , 426099) 499879
(886 , 293952) 499879
(890 , 318593) 499879
(891 , 213086) 499879
(896 , 265870) 499879
(897 , 192529) 499879
(899 , 61999) 499879
(900 , 4822) 499879
(903 , 365724) 499879
(905 , 300066) 499879
(906 , 275379) 499879
(909 , 233958) 499879
(911 , 233604) 499879
(912 , 241623) 499879
(913 , 255114) 499879
(916 , 328479) 499879
(917 , 363918) 499879
(923 , 192295) 499879
(924 , 266374) 499879
(925 , 345997) 499879

(927 , 21890) 499879
(928 , 118181) 499879
(930 , 327473) 499879
(931 , 440486) 499879
(934 , 313056) 499879
(938 , 388303) 499879
(939 , 46163) 499879
(941 , 378809) 499879
(942 , 53589) 499879
(949 , 435686) 499879
(952 , 113293) 499879
(953 , 350584) 499879
(958 , 122902) 499879
(960 , 172025) 499879
(961 , 455228) 499879
(963 , 38920) 499879
(967 , 275706) 499879
(968 , 99388) 499879
(970 , 264191) 499879
(971 , 105315) 499879
(972 , 452274) 499879
(973 , 305056) 499879
(975 , 28140) 499879
(976 , 398463) 499879
(977 , 274633) 499879
(978 , 156665) 499879
(986 , 424691) 499879
(990 , 200487) 499879
(991 , 159271) 499879
(994 , 71323) 499879
(1000 , 56617) 499879

(470231 , 125052) 499879
(470232 , 191258) 499879
(470234 , 287699) 499879
(470235 , 317946) 499879
(470238 , 336865) 499879
(470240 , 289686) 499879
(470241 , 248177) 499879
(470243 , 129350) 499879
(470244 , 52044) 499879
(470246 , 361686) 499879
(470247 , 248637) 499879
(470250 , 338109) 499879
(470251 , 10817) 499879
(470255 , 416292) 499879
(470256 , 196338) 499879
(470257 , 297875) 499879
(470258 , 220900) 499879
(470259 , 465428) 499879
(470260 , 31447) 499879
(470262 , 128036) 499879
(470269 , 94324) 499879
(470270 , 375718) 499879
(470271 , 478678) 499879
(470272 , 403210) 499879
(470274 , 217023) 499879
(470276 , 317214) 499879
(470277 , 349714) 499879
(470278 , 203822) 499879
(470279 , 379553) 499879
(470280 , 376904) 499879
(470282 , 336499) 499879
(470285 , 188214) 499879

(470286 , 115429) 499879
(470287 , 364315) 499879
(470290 , 41005) 499879
(470292 , 433912) 499879
(470293 , 112905) 499879
(470294 , 113611) 499879
(470298 , 333631) 499879
(470300 , 374039) 499879
(470302 , 201440) 499879
(470303 , 97779) 499879
(470307 , 400880) 499879
(470311 , 352634) 499879
(470313 , 259377) 499879
(470315 , 453434) 499879
(470320 , 70852) 499879
(470321 , 259924) 499879
(470324 , 258405) 499879
(470327 , 153923) 499879
(470328 , 96243) 499879
(470329 , 360486) 499879
(470330 , 446649) 499879
(470331 , 354738) 499879
(470334 , 10639) 499879
(470335 , 206510) 499879
(470336 , 224337) 499879
(470344 , 458206) 499879
(470348 , 53256) 499879
(470350 , 282988) 499879
(470355 , 243273) 499879
(470356 , 201543) 499879
(470357 , 481904) 499879
(470360 , 255539) 499879

(470361 , 324297) 499879
(470363 , 428164) 499879
(470364 , 463285) 499879
(470367 , 1461) 499879
(470368 , 325156) 499879
(470370 , 439014) 499879
(470373 , 276079) 499879
(470374 , 32797) 499879
(470375 , 111714) 499879
(470376 , 12827) 499879
(470379 , 149429) 499879
(470384 , 321642) 499879
(470386 , 246255) 499879
(470388 , 459934) 499879
(470389 , 50177) 499879
(470391 , 197536) 499879
(470392 , 254655) 499879
(470394 , 335802) 499879
(470395 , 359842) 499879
(470396 , 206198) 499879
(470399 , 179249) 499879
(470402 , 53387) 499879
(470403 , 156150) 499879
(470404 , 81277) 499879
(470406 , 398665) 499879
(470408 , 5581) 499879
(470409 , 42636) 499879
(470414 , 64076) 499879
(470415 , 35639) 499879
(470418 , 384995) 499879
(470420 , 230168) 499879
(470423 , 416469) 499879
(470426 , 5144) 499879
(470429 , 496382) 499879
(470431 , 436473) 499879
(470433 , 166701) 499879
(470434 , 265637) 499879
(470435 , 187123) 499879
(470436 , 431174) 499879
(470438 , 386968) 499879
(470440 , 133067) 499879
(470441 , 490006) 499879
(470442 , 169528) 499879
(470444 , 496399) 499879
(470447 , 406309) 499879
(470449 , 459397) 499879
(470450 , 219898) 499879
(470452 , 208862) 499879
(470454 , 488476) 499879
(470455 , 362285) 499879
(470457 , 77946) 499879
(470458 , 419819) 499879
(470461 , 381626) 499879
(470462 , 14311) 499879
(470463 , 469732) 499879
(470466 , 272354) 499879
(470467 , 18707) 499879
(470468 , 87817) 499879
(470470 , 194314) 499879
(470474 , 280584) 499879
(470478 , 31539) 499879
(470479 , 151309) 499879
(470480 , 93899) 499879
(470482 , 447581) 499879
(470484 , 92615) 499879
(470486 , 29067) 499879
(470487 , 231592) 499879
(470488 , 256985) 499879
(470489 , 105252) 499879
(470490 , 276408) 499879
(470491 , 270450) 499879
(470492 , 87384) 499879
(470494 , 189970) 499879
(470498 , 270146) 499879
(470499 , 347519) 499879
(470500 , 247832) 499879
(470501 , 471100) 499879
(470503 , 386498) 499879
(470505 , 93761) 499879
(470507 , 92946) 499879
(470508 , 77022) 499879
(470512 , 243362) 499879
(470513 , 342486) 499879
(470515 , 9812) 499879
(470516 , 78035) 499879
(470521 , 264945) 499879
(470527 , 150778) 499879
(470528 , 95937) 499879
(470531 , 370149) 499879
(470532 , 107809) 499879
(470535 , 259677) 499879
(470537 , 143371) 499879
(470539 , 319755) 499879
(470543 , 50758) 499879
(470547 , 453088) 499879
(470550 , 148731) 499879

(470551 , 27102) 499879
(470553 , 253624) 499879
(470554 , 101778) 499879
(470557 , 85911) 499879
(470558 , 227199) 499879
(470560 , 479672) 499879
(470561 , 90860) 499879
(470567 , 47858) 499879
(470568 , 422392) 499879
(470573 , 145501) 499879
(470577 , 344929) 499879
(470579 , 385085) 499879
(470580 , 140290) 499879
(470583 , 346521) 499879
(470584 , 395484) 499879
(470586 , 463775) 499879
(470587 , 483115) 499879
(470589 , 492214) 499879
(470591 , 295242) 499879
(470596 , 214090) 499879
(470598 , 46261) 499879
(470600 , 172586) 499879
(470601 , 221069) 499879
(470602 , 93104) 499879
(470603 , 288706) 499879
(470606 , 316929) 499879
(470607 , 306832) 499879
(470609 , 257417) 499879
(470611 , 2411) 499879
(470612 , 110332) 499879
(470613 , 41871) 499879
(470618 , 54100) 499879
(470619 , 427500) 499879
(470620 , 124551) 499879
(470621 , 145277) 499879
(470622 , 489684) 499879
(470625 , 464997) 499879
(470626 , 104152) 499879
(470627 , 67018) 499879
(470632 , 237142) 499879
(470634 , 371339) 499879
(470635 , 424064) 499879
(470636 , 300545) 499879
(470637 , 788) 499879
(470638 , 24808) 499879
(470641 , 39554) 499879
(470642 , 358724) 499879
(470644 , 468464) 499879
(470645 , 259046) 499879
(470646 , 373453) 499879
(470648 , 73739) 499879
(470649 , 159639) 499879
(470650 , 69379) 499879
(470651 , 302974) 499879
(470652 , 360421) 499879
(470656 , 328876) 499879
(470658 , 6385) 499879
(470659 , 330991) 499879
(470661 , 451909) 499879
(470662 , 248233) 499879
(470663 , 368484) 499879
(470664 , 312659) 499879
(470665 , 80764) 499879
(470667 , 88806) 499879
(470669 , 392658) 499879
(470670 , 280521) 499879
(470672 , 28160) 499879
(470673 , 387957) 499879
(470674 , 71729) 499879
(470676 , 411276) 499879
(470677 , 67045) 499879
(470678 , 46831) 499879
(470682 , 206211) 499879
(470683 , 306145) 499879
(470685 , 478160) 499879
(470687 , 446408) 499879
(470689 , 210937) 499879
(470690 , 79323) 499879
(470692 , 288377) 499879
(470693 , 129048) 499879
(470696 , 95736) 499879
(470698 , 194146) 499879
(470701 , 22799) 499879
(470707 , 432470) 499879
(470709 , 162485) 499879
(470711 , 189318) 499879
(470714 , 161195) 499879
(470715 , 133612) 499879
(470716 , 430274) 499879
(470719 , 265736) 499879
(470720 , 359411) 499879
(470721 , 277352) 499879
(470722 , 19565) 499879
(470723 , 86065) 499879

Bibliografía

[1] Granados, G. (2006), *Introducción a la criptografía. Revista Digital Universitaria, Vol. 7, No. 7, 1-17.*

[2] Gómez, J. (2002), *Criptografía y curvas elípticas. La Gaceta de la RSME, Vol. 5, 737-777.*

[3] Amalraj, J., Raybin, J. (2016), *A survey paper on cryptography techniques. IJCSMC, Vol. 5, Issue 8, 55 – 59.*

[4] Diffie, W. and Hellman, M. (1976), *New directions in cryptography. IEEE transactions on Information Theory, 22(6), 644-654.*

[5] Hankerson, D., Menezes, A. and Vanstone, S. (2004), *Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc.*

[6] Schoff, N. (1995), *Counting points on elliptic curves over finite fields, Journal de Théorie des Nombres de Bordeaux 7, 219-254.*

[7] ElGamal, T. (1985), *A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information Theory, 31, 469-472*

[8] Gupta, V., Stebila, D. y Chang, S. (2004), *Integrating Elliptic Curve Cryptography into the Gupta, V., Stebila, D. y Chang, S. (2004)*

[9] Koblitz, N. (1987), *Elliptic curve in cryptography. American Mathematical Society J. Comput. Math., 207–209.*

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

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

[12] Katz, J., Lindell, Y. (2015), *) Introduction to Modern Cryptography. NY: Chapmann and Hall Book/CRC.*