CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Matemáticas e Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Análisis e implementación del algoritmo de Firma Digital de Curvas Elípticas Múltiples (MECDSA) para Blockchain.

Autora: Andrea del Nido García

Director: Vicente Jara Vera

MADRID, JUNIO 2019

A mis padres, por apoyarme siempre, por ayudarme cuando lo he necesitado, por confiar en mí y por enseñarme que siempre podré lograr todo lo que me proponga.

A mis hermanos y familia, por creer en mí y por darme fuerzas para no rendirme.

A mi novio, que me ha enseñado a ver la vida con una sonrisa y a ser valiente, al que le debo toda mi felicidad.

A mis profesores de universidad, especialmente a Vicente Jara Vera y Carmen Sánchez Ávila por descubrirme el maravilloso mundo de la criptografía y por dirigirme este proyecto.

"Es una locura odiar a todas las rosas porque una te pinchó.

Renunciar a todos tus sueños porque uno de ellos no se realizó."

ANTOINE DE SAINT-EXUPÉRY. *El principito.*

ÍNDICE

ÍNDICE 3
RESUMEN 5
ABSTRACT 6
1. INTRODUCCIÓN 7
1.1. CONTEXTO HISTÓRICO 7
1.2. DEFINICIONES PREVIAS 9
1.3. TIPOS DE CIFRADO CLÁSICOS 11
2. TRABAJOS PREVIOS 13
3. DESARROLLO 13
3.1. FUNCIONES HASH 14
3.2. TIPOS DE FUNCIONES DE CIFRADO 15
3.2.1. CIFRADO SIMÉTRICO 15
3.2.1.1. CRIPTOSISTEMA DES 16
3.2.1.2. CRIPTOSISTEMA AES 19
3.2.2. CIFRADO ASIMÉTRICO 20
3.2.2.1. PROTOCOLO DE INTERCAMBIO DE CLAVE DIFFIE-HELLMAN 20
3.2.2.2. CRIPTOSISTEMA RSA 21
3.2.2.3. CRIPTOSISTEMA ELGAMAL 23
3.2.3. FIRMA DIGITAL 24
3.2.4. FIRMA DIGITAL CON CURVAS ELÍPTICAS 25
3.2.4.1. DEFINICIÓN DE CURVA ELÍPTICA 26
3.2.4.2. OPERACIONES SOBRE LAS CURVAS ELÍPTICAS 28
3.2.4.3. CÁLCULO DEL ORDEN DE UNA CURVA ELÍPTICA 29
3.2.4.4. CÁLCULO Y REPRESENTACIÓN DE UN PUNTO DE UNA CURVA ELÍPTICA 30
3.2.4.5. CURVAS ELÍPTICAS RECOMENDADAS EN LAS ESPECIFICACIONES Y ESTÁNDARES 31
3.2.5. EL SISTEMA ECDSA 33
3.2.5.1. PROCESO DE FIRMA 33
3.2.5.2. PROCESO DE VERIFICACIÓN DE FIRMA 34

3.2.6. EL SISTEMA MECDSA 35
3.2.6.1. PROCESO DE FIRMA 35
3.2.6.2. PROCESO DE VERIFICACIÓN DE FIRMA 37
3.2.6.3. ANÁLISIS DE SEGURIDAD 37
4. IMPLEMENTACIÓN 38
4.1. SOFTWARE UTILIZADO 38
4.2. ESQUEMA DE LAS CLASES UTILIZADAS 38
4.2.1. PAQUETES Y CLASES DE JAVA 40
4.2.2. DESARROLLO DE LOS JFRAMES 41
4.2.3. DESARROLLO DE LOS JPANELS 42
4.3. CONEXIÓN ENTRE SAGE Y JAVA 44
4.4. PRUEBAS DE FUNCIONAMIENTO Y ERRORES 45
4.5. ANÁLISIS DEL RENDIMIENTO 48
5. RESULTADOS OBTENIDOS Y CONCLUSIONES 49
6. PROPUESTAS DE MEJORA 50
ANEXO 51
BIBLIOGRAFÍA 52

RESUMEN

El proyecto que se propone para este Trabajo Fin de Grado trata sobre diversos sistemas criptográficos, centrándose principalmente en el cifrado de curva elíptica para las firmas digitales. Este método de cifrado puede encontrarse en diferentes sistemas informáticos que requieran seguridad, pero este documento se centrará principalmente en el utilizado para los sistemas blockchain.

El principal algoritmo utilizado para las firmas digitales con curvas elípticas centrados en la blockchain es el algoritmo ECDSA (Elliptic Curve Digital Signature Algorithm). Éste se estudiará en el presente escrito, sin embargo, el trabajo realizado se desarrollará en torno al algoritmo propuesto en el artículo *"A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain"* de Wei Bi, Xiaoyun Jia y Maolin Zheng [1], el cual versa sobre el sistema MECDSA (Multiple Elliptic Curve Digital Signature Algorithm). Además de analizar esta propuesta, se implementará dicho sistema en el lenguaje Java de forma local (en una única máquina) con la ayuda del entorno de desarrollo SageMath para determinadas funciones matemáticas.

La principal motivación de este trabajo es desarrollar más a fondo el algoritmo MECDSA y llegar más allá tratando con la generación de curvas elípticas aleatorias para su desarrollo. Se analiza y se llega a la conclusión de que esta forma de firmar digitalmente en la blockchain es más seguro que utilizando curvas elípticas ya diseñadas por organismos gubernamentales o privados.

ABSTRACT

The proposed project for this Final Degree Project deals with the study of various cryptographic systems, focusing essentially on elliptic curve digital signatures for encryption. This encryption method can be found in different systems that require security, but this document centralizes mainly on used for blockchain systems.

The main algorithm used for digital signatures with elliptic curves centered in Blockchain is the ECDSA algorithm (Elliptic Curve Digital Signature Algorithm), this will be studied in the present report, however, the task will take place around the algorithm proposed in the article *"A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain"* de Wei Bi, Xiaoyun Jia y Maolin Zheng [1], which is about MECDSA system (Multiple Elliptic Curve Digital Signature Algorithm). In addition to analysing this proposal, MECDSA will be implemented locally with the Java language (on a single machine) with the help of the development environment Sage Math for certain mathematical functions.

The main motivation of this work is to develop more thoroughly the MECDSA algorithm and to go further dealing with the generation of random elliptic curves for its development. It is analysed and it is concluded that this way of signing digitally in the blockchain is safer than using elliptic curves already designed by governmental or private organizations.

1. INTRODUCCIÓN

En este Trabajo Fin de Grado se ha profundizado en el conocimiento, aplicación y desarrollo de algunos esquemas criptográficos actuales, en concreto, la propuesta del artículo *"A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain"* de Wei Bi, Xiaoyun Jia y Maolin Zheng [1] que versa sobre el sistema MECDSA para la seguridad en blockchain y criptomonedas.

La definición que nos ofrece la Real Academia de la Lengua Española sobre la "criptografía" es: "Arte de escribir con clave secreta o de un modo enigmático." Palabra que proviene del griego *kryptós* 'oculto' y *grafé* 'escritura' [2].

Desde el inicio de las antiguas civilizaciones, como la egipcia, el ser humano ha estado ocultando mensajes por distintos motivos (bélicos, personales, etc.). Aunque no existía un término definido para ello, la criptografía ha existido desde tiempos inmemorables. Fuimos evolucionando hasta el desarrollo de una tecnología mucho más compleja que hoy en día hace necesarios sistemas de seguridad más avanzados que consigan confidencialidad, autenticidad e integridad, que son los pilares de la seguridad actual.

A lo largo de este documento nos centraremos en los tipos de cifrado simétrico y asimétrico, pero especialmente en el asimétrico de curvas elípticas aplicado a la firma digital que se utiliza en el entorno de las criptomonedas, concretamente para la seguridad de las transacciones que se aplican en blockchain.

1.1. CONTEXTO HISTÓRICO

La historia de la criptografía se divide en las siguientes etapas: criptografía clásica, criptografía medieval, criptografía renacentista y criptografía desde el siglo XVII hasta la Segunda Guerra Mundial, criptografía de la Segunda Guerra Mundial y criptografía contemporánea.

En la criptografía clásica encontramos distintas formas de ocultar la información en varias civilizaciones. Los dos principios básicos utilizados en esta época son la sustitución, el cual establece relaciones entre las letras del alfabeto que constituye el mensaje en claro y los elementos de otro conjunto, y el principio de transposición; éste consiste en realizar permutaciones de tal forma que se modifiquen las posiciones del mensaje original consiguiendo que el mensaje cifrado resulte incomprensible.

En Egipto con los jeroglíficos, más tarde con los hebreos aparece el cifrado Atbash, utilizado en la Biblia hebrea, también aparecen fragmentos de textos cifrados en el libro del Apocalipsis del Nuevo Testamento cristiano.

En la época clásica de Grecia aparece una nueva forma de criptografía mediante el uso del escítalo (siglo V a.C.): este método consiste en envolver un bastón de un determinado diámetro con una tira de cuero que contenía el mensaje utilizado con fines bélicos por los espartanos. Además, el historiador griego Polybios (siglo II a.C.) utiliza otro sistema diferente de encriptación utilizando el principio de sustitución. También cabe destacar los métodos de cifrados romanos, principalmente, el cifrado de sustitución Julio César (100 a.C.-44 a.C.), el cifrado Augusto (27 a.C.-14 d.C.) y el cifrado Julio César con clave (es una variante no clásica).

En la etapa de la criptografía medieval y renacentista cabe destacar el método de sustitución de Leon Battista Alberti (1404-1472). Ingenió un dispositivo que consistía en utilizar dos discos concéntricos los cuales contienen abecedarios, el disco interior corresponde a las letras del texto en claro y el exterior son las letras por las que se cifrará el mensaje original, además, se necesita una clave. Destacamos también el método de Tritemius (1462-1516) ya que el cifrado es de sustitución polialfabética mediante el uso de una clave que se repite hasta cubrir el mensaje original. Este cifrado se conoce también como el cifrado Vigenère-Belasso, especialistas en cifrado del siglo XVI que difundieron este sistema.

Un cifrado de transposición de este periodo es la rejilla de Cardano (1501-1576): se construye una rejilla cuadrada que pueda contener el texto en claro, y se dejan vacías algunas casillas, se realizan giros de 90º para ir descifrando el mensaje. Otro cifrado basado en el principio de transposición es el de Richelieu (1585-1642) en el que la clave es numérica e indica el orden en el que se deben barajar las letras del mensaje original.

Desde el siglo XIX hasta la Segunda Guerra Mundial destacan Charles Wheatstone (1802-1875) mejorando el sistema que empleó Alberti y el tercer presidente de Estados Unidos, Thomas Jefferson (1743-1825). Asimismo, encontramos ejemplos literarios de cifrado en obras de Edgar Allan Poe, Julio Verne y Arthur Conan Doyle. Debido a la Primera Guerra Mundial los alemanes desarrollan el cifrado ADFGX y ADFGVX, basado en el principio de sustitución.

En la Segunda Guerra Mundial aparecen las máquinas de rotores, como la que los alemanes utilizaron para uso militar durante la guerra, la máquina Enigma. No fue la única que apareció, los japoneses utilizaron un sistema parecido denominado Purple y los americanos diseñaron las suyas, la más común es la máquina Red.

Finalmente, llegamos a la época moderna en la que se comercializa la criptografía y deja de tratarse exclusivamente de uso militar. Aparece el cifrado DES (Data

Encryption Standard), este sistema de cifrado está basado en el cifrado LUCIFER el cuál fue ganador del concurso público organizado por la agencia de los Estados Unidos NSA (National Security Agency) para buscar un algoritmo criptográfico estándar. Aparece también la condición de secreto perfecto, introducida por Claude Elwood Shannon (1916-2001) y surgirá en estos siguientes años una nueva división de la criptografía, la simétrica o de clave secreta y la asimétrica o de clave pública.

Aproximadamente durante la misma época, y en relación con los sistemas asimétricos, donde ya existían RSA o ElGamal, el matemático y criptógrafo Neal Koblitz (1948-) propuso en el año 1985 utilizar curvas elípticas para desarrollar ciertos criptosistemas que ya existían y se utilizaban. Además, describió cómo realizar el protocolo de intercambio de claves de Diffie-Hellman, usando curvas elípticas.

1.2. DEFINICIONES PREVIAS

En esta sección incluimos algunas descripciones sobre los conceptos que más se utilizan a lo largo del documento, con el fin de facilitar la lectura del mismo.

Criptología: es la ciencia que comprende las disciplinas de la criptografía y del criptoanálisis. Estudia y analiza la transmisión y creación de lenguajes y mensajes ocultos, así como el intento de desciframiento sin el uso de las claves cifradoras.

Criptografía: es una rama de la criptología que mediante diferentes técnicas y métodos matemáticos consigue cifrar y descifrar aplicando algoritmos. De esta manera, se consigue proteger la información que se pretende transmitir, mantener la integridad del mensaje y confirmar la identidad de los participantes de la comunicación.

Criptoanálisis: esta ciencia derivada de la criptología, trata de resolver los sistemas y métodos utilizados para cifrar sin tener apenas conocimiento sobre los algoritmos o sistemas de cifrado, o de las claves utilizadas en la generación de los mensajes cifrados que se utilizaron para ello.

Criptograma: es la quíntupla (M, C, K, E, D), en donde M representa el conjunto de los mensajes que se pueden cifrar, C es el conjunto de los mensajes cifrados, K es el conjunto de claves posibles que se utilizan para cifrar los mensajes del conjunto M y para descifrar los elementos del conjunto C, E es el conjunto de funciones criptográficas y D contiene las funciones de descifrado que nos dan los elementos de M aplicadas a los de C.

Criptosistema: son los procedimientos, algoritmos y claves con los que se consigue ocultar la información mediante el cifrado utilizando técnicas criptográficas.

Texto en claro: también llamado mensaje original es el mensaje que queremos cifrar y ocultar.

Texto cifrado: es el texto obtenido como resultado de aplicar algoritmos criptográficos sobre el texto en claro.

Algoritmo criptográfico: son una serie de pasos o secuencias de funciones matemáticas que pueden ser sencillas o complejas dependiendo del algoritmo, las cuales se utilizan para cifrar el texto en claro y a partir de ellas conseguir el texto cifrado.

Función hash: es una función matemática que transforma una cadena de bits (del mensaje en claro) de una longitud variable en una nueva cadena diferente cuya longitud es constante. A esta cadena se le llama valor hash y estas funciones matemáticas son utilizadas para firmas digitales y generación de claves entre otras aplicaciones. Podemos encontrar más información sobre las funciones hash en [3].

Firma digital: es el resultado de aplicar el algoritmo matemático de la firma y la función hash correspondiente junto a la clave privada calculada previamente y además realizar la verificación de la firma aplicando otro algoritmo matemático con la misma función hash y con la clave pública ya calculada. Esta definición está basada en [4].

Blockchain: consiste en una cadena de bloques entrelazados que son esencialmente una base de datos distribuida la cual contiene todas las transacciones públicas de los integrantes del sistema y que son compartidos entre ellos sin intermediarios [5].

Bitcoin: es una unidad monetaria cuyo funcionamiento no está centralizado y donde el uso criptográfico es central para su funcionamiento. Se utilizan métodos de intercambio comercial mediante transacciones sin que intervengan bancos o entidades similares. Esta moneda virtual se almacena en "monederos" que pueden llevarse en el móvil o dispositivos similares y se pueden intercambiar a través de la Red.

1.3. TIPOS DE CIFRADO CLÁSICOS

Principalmente los cifrados clásicos los dividimos en cifrados de transposición y de sustitución. El esquema que se muestra a continuación, representa los distintos tipos de cifrado clásico.

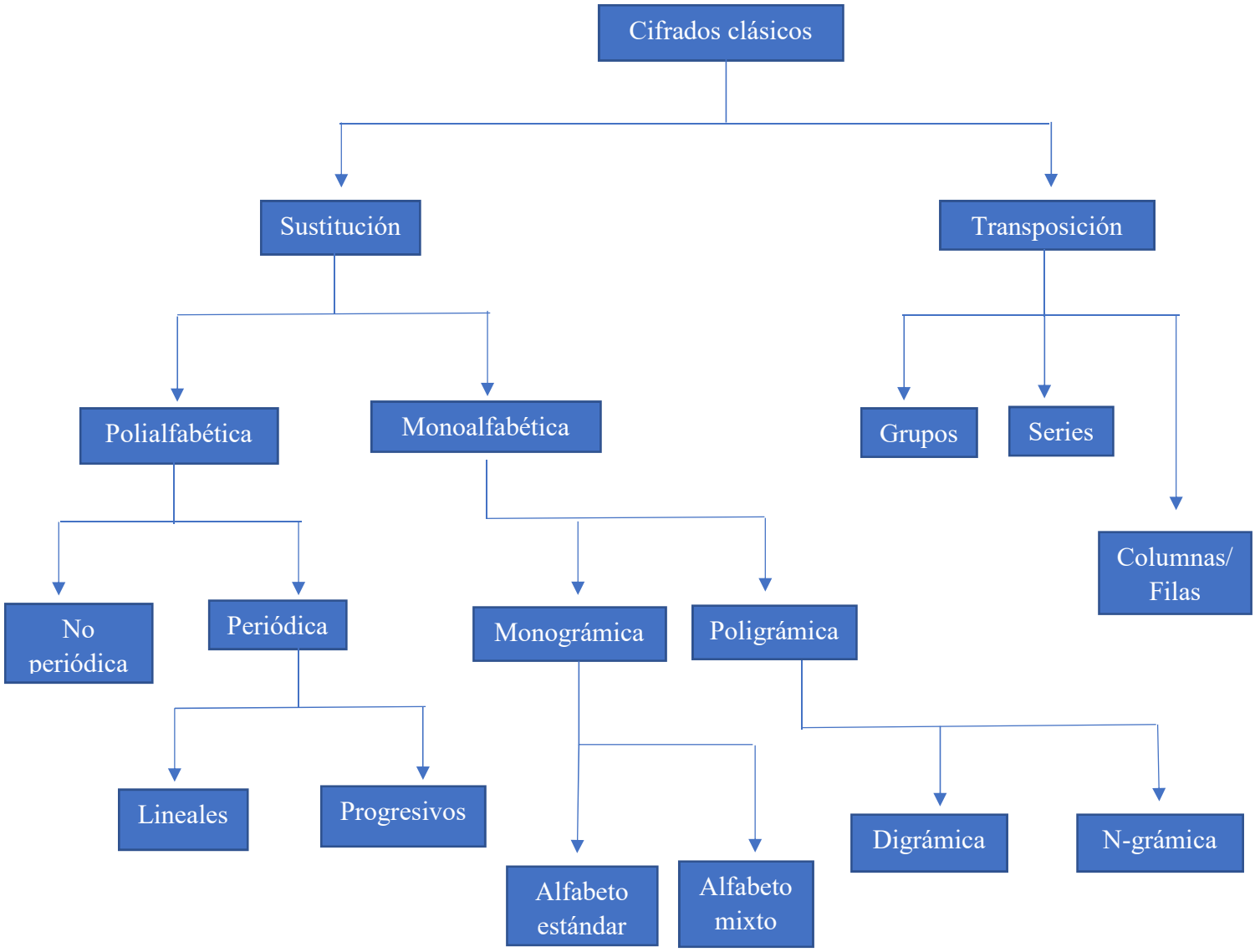

Figura 1. Esquema de los distintos cifrados clásicos.

Como se puede apreciar en el esquema de la Figura 1, los cifrados clásicos se dividen en cifrados de transposición y de sustitución. Los de transposición, consisten en alterar las

posiciones de los símbolos del texto en claro de tal forma que el texto cifrado contenga las mismas letras que el original, pero sea ininteligible. Los cifrados de sustitución se basan en remplazar las letras del mensaje original por otras diferentes.

El cifrado de sustitución polialfabético utiliza más de un alfabeto para cifrar la información de tal forma que una letra del mensaje original es sustituida por distintas letras del alfabeto utilizado para cifrar. El monoalfabético, por el contrario, utiliza un único alfabeto para sustituir las letras del texto en claro. Si el cifrado es periódico quiere decir que la clave que se utiliza para cifrar se repite a lo largo del mensaje; esto provoca que sea más fácil de realizar un criptoanálisis. Sin embargo, si el cifrado es no periódico la clave no se repite a lo largo del algoritmo de cifrado.

El cifrado de sustitución monoalfabético puede ser monográmico: esto se refiere a que cambiamos una letra del mensaje original por otra para cifrar o, poligrámica, si sustituimos una letra del texto en claro por varias del alfabeto con el que se cifra.

A continuación, veremos algunos ejemplos sencillos de estos cifrados clásicos.

El cifrado Julio César es un cifrado de sustitución, monoalfabético, monográmico y con alfabeto estándar. Podemos expresarlo matemáticamente como $C_i = M_i + 3 \bmod 27$, siendo C_i la letra cifrada y m_i el símbolo del mensaje original que queremos encriptar. En la fórmula aparece mod 27 debido al conjunto de letras del alfabeto español.

M = “Esto es un ejemplo de César”

C = HVWRHVXPHMHOSÑRGHFHVDU

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

Figura 2. Ejemplo de cifrado Julio César.

La rejilla de Cardano es un cifrado de transposición consistente en situar un cuadrado con agujeros que cubre el texto original y rotando dicho cuadrado se cifra el mensaje.

2. TRABAJOS PREVIOS

Previo al desarrollo del proyecto se ha buscado información en trabajos relacionados con el tema, utilizando principalmente el contenido del artículo *"A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain"* de Wei Bi, Xiaoyun Jia y Maolin Zheng [18].

En su trabajo explican por qué es necesario usar otro sistema diferente al ECDSA, normalmente utilizado en la blockchain; y proponen cómo mejorarlo mediante el desarrollo de un nuevo algoritmo que exponen y someten a un análisis de desempeño llamado Multiple Elliptic Curves Digital Signature Algorithm (MECDSA).

El artículo profundiza en el desarrollo de los dos sistemas mencionados, tanto en sus aspectos de generación como de verificación de la firma para cada uno. También incluye un análisis de seguridad y de eficiencia del sistema MECDSA. Al final del artículo sus autores recogen algunas curvas de los principales estándares, como una del estadounidense National Institute of Standards and Technology (NIST) y otra de la administración china, o las propuestas del consorcio Standards for Efficient Cryptography Group (SECG), entre las que está la curva utilizada en Bitcoin, la curva secp256k1.

Con este trabajo conseguiremos encontrar alternativas al sistema ECDSA que es el actualmente más utilizado como algoritmo de firma digital en blockchain como es la descrita en el artículo anteriormente mencionado, el sistema MECDSA, debido a que utiliza un mayor número de curvas elípticas, lo que hace que sea más seguro. Además, buscaremos mejorar la eficiencia y seguridad del algoritmo incluyendo curvas generadas de forma aleatoria a partir de los requerimientos y necesidades del usuario.

3. DESARROLLO

En esta sección realizaremos un análisis en profundidad sobre distintas funciones matemáticas, algoritmos y cifrados que se han estudiado para realizar el proyecto. Principalmente, se centra en el estudio de las curvas elípticas y, de los sistemas empleados para las firmas digitales, el sistema ECDSA y el MECDSA.

3.1. FUNCIONES HASH

Las funciones hash o también denominadas funciones resumen aplican un algoritmo de forma que tienen como entrada un conjunto de elementos, o texto de entrada, en general binario, y por medio de una serie de operaciones y transformaciones obtiene como resultado una cadena distinta de una longitud fija.

Su principal propiedad es lograr comprobar la integridad de un mensaje, es decir, poder verificar si un mensaje ha sido o no alterado o modificado en forma alguna de las funciones hash es que tiene un bajo costo, es decir, no necesitan muchos recursos ni consumen mucha memoria al aplicarlos. Además, las funciones hash comprimen datos que tengan una longitud demasiado grande, lo que lleva a llamarlas funciones resumen. Por otro lado, son funciones de bajo coste computacional, no consumiendo demasiada memoria ni recursos de procesamiento, en general.

Una buena función hash tiene que evitar las colisiones, es decir, impedir que dadas dos entradas diferentes se obtenga la misma salida.

Hay muchas funciones hash, pero las principales son: MD5, SHA-1 o RIPEMD-160. Una versión mejorada y actual de SHA-1 es SHA-2, que es la función que se ha utilizado en el desarrollo de la aplicación que más adelante se explicará para las firmas digitales con curvas elípticas.

La función MD5 fue desarrollada por Ron Rivest en el año 1992. Utiliza un resumen de 128 bits, pero actualmente está obsoleta pero la base de este algoritmo se utiliza para implementar otros.

El algoritmo SHA-1 fue diseñado por el NIST (National Institute of Standards and Technology), con un resumen de 160 bits, habiendo otras propuestas como SHA-2.

Otros ejemplos de algoritmos específicos que implementan diferentes funciones hash son: Snefru, N-Hash, Tiger y Haval entre otros.

La manera más habitual en la que se construyen las funciones hash es siguiendo la estructura de Ivan B. Damgård (1956-) y Ralph C. Merkle (1952-) que desarrollaron en 1989. El esquema es el que se presenta a continuación en la Figura 3.

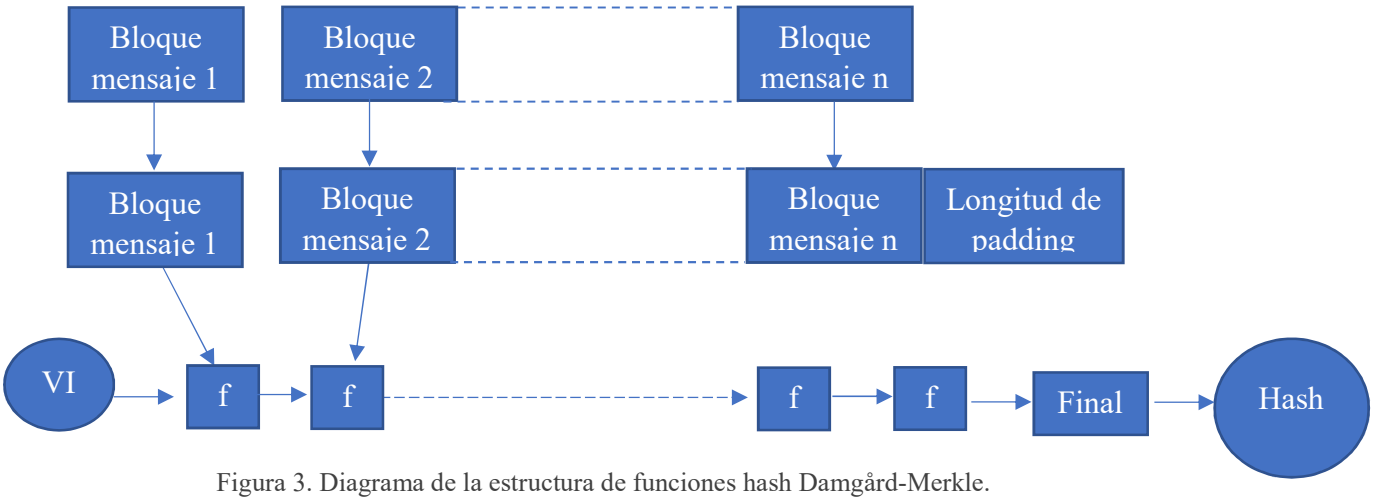

Figura 3. Diagrama de la estructura de funciones hash Damgård-Merkle.

Al comienzo de la estructura tenemos los bloques del mensaje divididos en tamaños, además obtenemos otro bloque correspondiente al "padding". El "padding" consiste en añadir al mensaje en binario un 1, a continuación, tantos ceros como sean necesarios para tener un bloque de 512 bits y finalmente se le añade la longitud en bits del mensaje.

La función hash comienza con un vector inicial que dependiendo de la función cambiará sus valores, pero siempre son fijos. Con todo ello y por medio de la función f aplicamos diferentes operaciones de confusión y difusión de la información y tras una etapa final obtenemos el hash o resumen del mensaje de entrada.

3.2. TIPOS DE FUNCIONES DE CIFRADO

El cifrado de mensajes actualmente se ha vuelto muy complejo, debido al gran avance tecnológico. Es por esto por lo que en esta sección veremos cómo ha evolucionado la criptografía, y cuáles son los distintos tipos de cifrado. Además, estudiaremos principalmente las curvas elípticas y las firmas digitales.

3.2.1. CIFRADO SIMÉTRICO

El cifrado o la criptografía simétrica, también denominado criptografía de clave secreta consiste en transformar un texto en claro en otro cifrado para así, lograr la confidencialidad del mensaje. Para ello, se utilizan lo principios básicos de la

criptografía clásica (transposición y sustitución) aplicados en algoritmos matemáticos complejos junto con una clave que deben tener tanto el emisor del mensaje como el receptor.

La clave que se utiliza para encriptar el mensaje puede ser de un tamaño fijo o variable, pero, es necesario dividir el mensaje original en bloques para poder cifrarlo. A esto se le denomina cifrado en boque y es característico de los criptosistemas de clave secreta.

Los criptosistemas de clave simétrica más destacados son: el criptosistema DES (Data Encryption Standard) y el criptosistema AES (Advanced Encryption Standard).

3.2.1.1. CRIPTOSISTEMA DES

El sistema DES surge debido a un concurso público organizado por la NBS (National Bureau of Standards) en 1973. El ganador de este concurso fue el algoritmo LUCIFER, pero se modificó, se redujo el tamaño de la clave que necesitaba y pasó a denominarse algoritmo DES en el año 1976.

Actualmente este criptosistema se ha conseguido romper mediante algoritmos de fuerza bruta debido a que la clave no es demasiado larga, pero existe una implementación alternativa llamada Triple DES. Este criptosistema sigue una estructura EDE (Encryption Decryption Encryption), basado en cifrar y descifrar con varias claves y repetidas veces.

El criptosistema DES es un criptosistema de tipo Feistel por bloques, es decir, originalmente tenemos un bloque de N bits (del mensaje en claro), se divide en dos mitades A y B, cada una de ellas tiene N/2 bits, al bloque A se le aplica una función matemática unidireccional durante un número finito de iteraciones, en la cual, se ve implicada la clave. A continuación, se realiza un or-exclusivo con el bloque anterior y el B y, finalmente, se intercambian para la siguiente iteración. Se puede observar un esquema sobre el desarrollo de los criptosistemas tipo Feistel en la Figura 4.

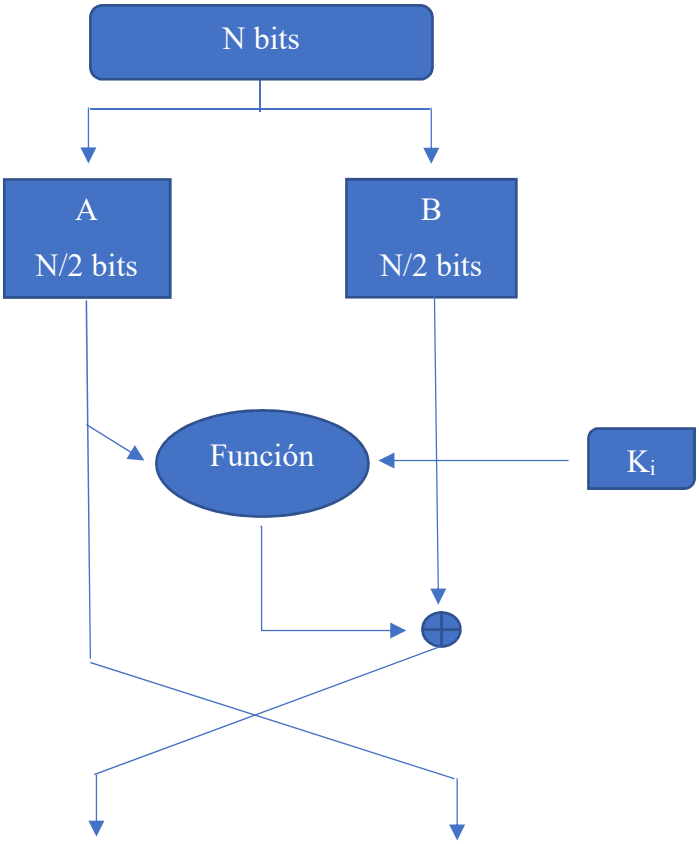

Figura 4. Esquema de cifrado tipo Feistel.

Utiliza una clave con longitud 56 bits, aunque no se tienen en cuenta los bits de paridad, utilizados para la detección de errores, teniéndolos en cuenta, la clave es de longitud 64 bits. El espacio de claves, es decir, todas las claves que podemos utilizar son de 2^{56} posibles claves. Además, se realizan 16 iteraciones aplicando la función para cifrar y descifrar.

A continuación, describiremos brevemente el funcionamiento del algoritmo de cifrado. Como entrada tenemos un bloque de 64 bits del texto en claro, además, tenemos la clave K, de 64 bits, a la cual le quitamos los 8 últimos bits de paridad (estos bits se utilizan para la corrección y detección de errores). En la Figura 5 observamos que interviene cada una de las subclaves de cada iteración i en la función, representado como K_i. Estas subclaves se obtienen a partir de la clave K. La primera fase del sistema, consiste en realizar una permutación inicial mediante la tabla IP (Initial Permutation) ya predefinida. Seguidamente necesitamos tener calculadas las subclaves a partir de la clave K, para ello, realizamos una permutación y dividimos la clave en dos bloques de 28 bits cada uno; llevamos a efecto un desplazamiento hacia la izquierda en cada bloque, permutamos nuevamente los bits y juntamos los bloques consiguiendo así la subclave de cada iteración.

En cada vuelta, además de obtener la subclave K_i correspondiente, tenemos que dividir el bloque inicial (ya permutado) en dos mitades, de 32 bits cada una. Al bloque derecho, R_i, se le aplica una permutación con expansión para obtener 48 bits y así, poder hacer un XOR con la subclave de la iteración i, ya que está tiene una longitud de 48 bits. Posteriormente se introduce el resultado en las cajas-S, de tal forma que pasamos de 48 bits a 36. De nuevo, permutamos la salida obtenida de la caja-S, realizamos un or-exclusivo con la mitad izquierda inicial, L_i, y esto será el nuevo bloque derecho de la siguiente iteración. El bloque izquierdo de la siguiente iteración L_{i+1} será el bloque derecho inicial R_i. Después de realizar todas las vueltas, se aplica la permutación inversa o final y obtenemos el mensaje cifrado de 64 bits.

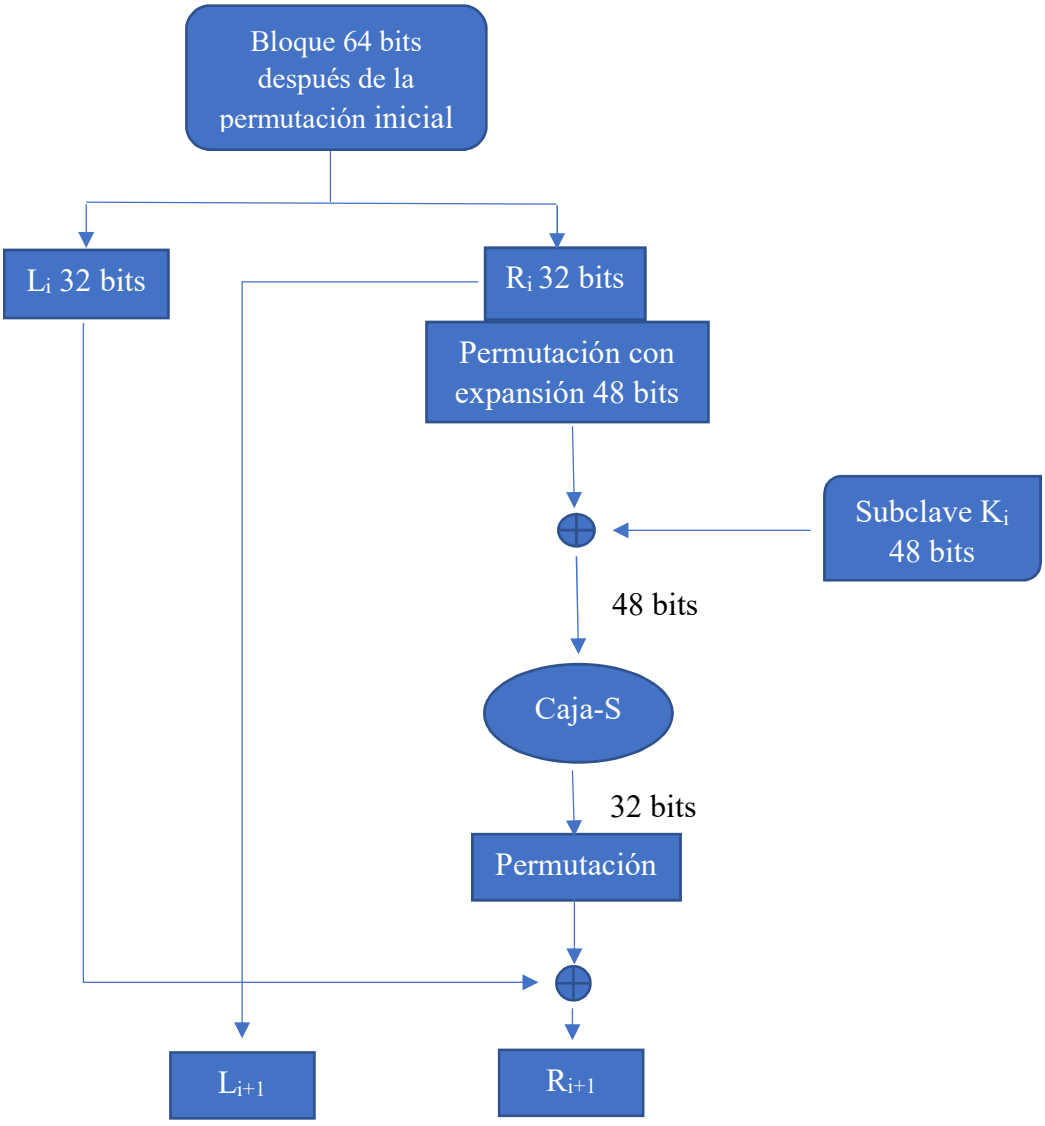

Figura 5. Esquema iterativo del algoritmo de cifrado DES.

3.2.1.2. CRIPTOSISTEMA AES

Debido a la necesidad de conseguir mayor seguridad, el NIST (National Institute for Standards and Technology) propuso en el año 1997 un programa para desarrollar un nuevo estándar de cifrado más potente que el criptosistema DES. El ganador fue el algoritmo Rijndael, que pasó a ser el nuevo estándar de cifrado avanzado, AES.

Este criptosistema es un cifrado por bloques al igual que el DES, pero no es tipo Feistel. Utiliza longitudes de bloque y clave variables, tienen que ser múltiplos de 4 bytes y además estar en el rango de 128 y 256 bits. Este sistema es seguro debido a que opera a nivel de bytes y los elementos pertenecen al cuerpo de Galois GF (2^8).

La estructura del AES se divide en tres capas que son: capa de mezcla lineal, capa no lineal y capa de adición de clave. Cada una de estas capas contiene funciones matemáticas. Inicialmente se parte de una matriz que tiene cuatro filas y N_b columnas, siendo N_b el tamaño de bloque/32 bits. De la misma manera ocurre con la clave. Además, el AES consta de un número determinado de rondas o iteraciones. Las funciones que se utilizan son *ShiftRow,* la cual se utiliza para intercambiar las filas y conseguir así mayor dispersión. *MixColumn,* en la que se mezcla las columnas de la matriz para conseguir dispersión. *SubByte* aplica cajas-S las cuales dependen del tamaño del bloque y que tienen propiedades de no linealidad; y *AddRoundKey,* que aplica un or-exclusivo en cada iteración con la subclave de esa ronda y con la matriz de estado intermedio. Para obtener las subclaves se utiliza el algoritmo de expansión de clave. En este algoritmo se aplican otras funciones como *SubWord*, *RotByte* y *Rc.*

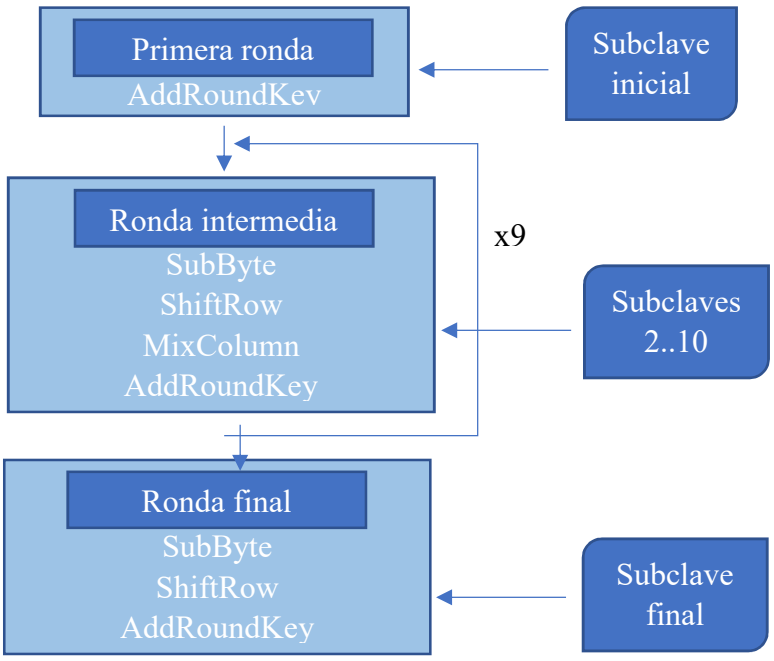

Figura 6. Esquema iterativo del algoritmo de cifrado AES.

3.2.2. CIFRADO ASIMÉTRICO

La criptografía asimétrica o de clave pública surge debido a la necesidad de suplir las restricciones e inconvenientes que tenía la criptografía simétrica, por ejemplo, el problema de la distribución de las claves por un medio seguro, cómo almacenar dichas claves y de qué manera poder identificar a los usuarios que participan en la comunicación, así como la enorme cantidad de claves simétricas necesarias que se precisarían, una por cada interlocutor diferente. La principal desventaja de estos cifrados es que son demasiado lentos y no son eficientes con muchos datos; además, la longitud de clave para alguno de ellos es excesivamente larga. La seguridad de estos criptosistemas para los más conocidos y extendidos reside en la gran dificultada de resolver los problemas matemáticos de la factorización de números enteros y el problema del logaritmo discreto.

3.2.2.1. PROTOCOLO DE INTERCAMBIO DE CLAVE DIFFIE-HELLMAN

Este algoritmo fue el primero de clave pública desarrollado en el año 1976 por Whitfield Diffie (1944-) y Martin Hellman (1945-). Se utiliza únicamente para intercambiar información de tamaño pequeño, por ejemplo, para intercambiar claves. La seguridad de este algoritmo reside en la utilización de los logaritmos discretos y la dificultad de su cálculo inverso.

El protocolo que sigue el algoritmo es el mostrado en la Figura 7. En este esquema participan dos usuarios A y B. Escogen un grupo cíclico finito de orden n y un generador del grupo llamado g, estos datos son públicos. En primer lugar, A genera un número entero aleatorio a y calcula el valor de g^a en G, el usuario A envía este resultado a B. Por otro lado, B genera un número entero aleatorio b, y calcula g^b en G y manda este valor a A. Una vez que A recibe el valor de B, realiza la operación $(g^b)^a$ en G; B en cambio obtiene el siguiente valor $(g^a)^b$ en G. Finalmente A y B tienen el mismo elemento del grupo G ya que $(g^b)^a = (g^a)^b = g^{ba}$ y además es secreto.

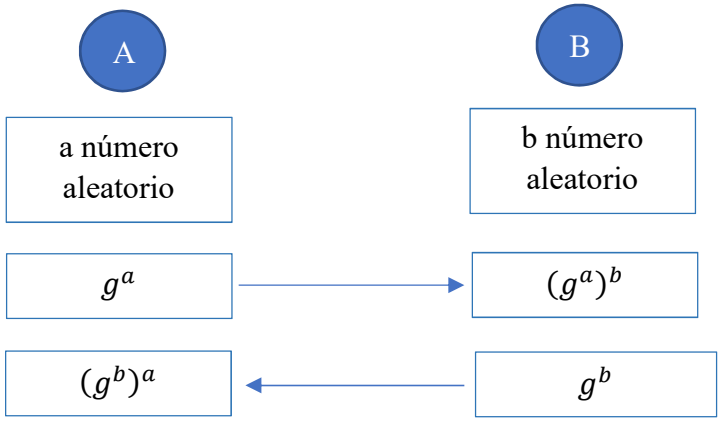

Figura 7. Esquema del protocolo de intercambio de clave Diffie-Hellman.

3.2.2.2. CRIPTOSISTEMA RSA

Este criptosistema fue desarrollado por R. Rivest (1947-), A. Shamir (1952-) y L. Adleman (1945-) en el año 1978. Este sistema se utiliza para cifrar y su seguridad consiste en el problema matemático de la factorización de números enteros.

Distinguimos tres fases en el esquema RSA: generación de claves, en la que los usuarios crean las claves que se aplicarán en el algoritmo; cifrado de mensajes; y finalmente el descifrado de los mensajes.

Generación de claves: cada usuario que participa en la comunicación debe elegir su clave privada y su clave pública para posteriormente realizar las funciones de cifrado y descifrado. Para generar las claves hay que seguir estos pasos:

1. El usuario escoge dos números primos que tengan entre 116 y 155 dígitos decimales a los que llamaremos p y q, es necesario que sean primos grandes de ese tamaño ya que de menor tamaño se pueden factorizar en un tiempo razonable y provoca que se pueda romper el criptosistema. Calcula $n = p \cdot q$ y, a continuación, se calcula la función ϕ de Euler, $\phi(n) = \phi(p \cdot q) = (p-1)(q-1)$.
2. Para obtener la clave pública se escoge un entero positivo e de forma que se verifique que $2 < e < \phi(n)$ y que $mcd(e, \phi(n)) = 1$. Finalmente, la clave pública es (n, e).
3. La clave privada d se calcula teniendo en cuenta que $1 < d < \phi(n)$ y que d tiene que ser inverso de e en $\mathbb{Z}^*_{\phi(n)}$ de forma que $e \cdot d \equiv 1 \, mod \, \phi(n)$.

Cifrado de mensajes: si el usuario A quiere mandar un mensaje m al usuario B el protocolo que debe seguir para cifrar el mensaje es el siguiente:

1. El usuario A consigue la clave pública de B que es (n_B, e_B) de un directorio de claves públicas en el que se guardan.
2. A debe representar el mensaje que quiere enviar como elementos del cuerpo en el que estamos, es decir, $\mathbb{Z}_{n_B}^*$.
3. El usuario emisor A calcula el criptograma $c = m^{e_B} \, mod \, n_B$ y se lo envía a B.

Descifrado de mensajes: el usuario receptor B recibe el criptograma c y recupera el texto en claro siguiendo los siguientes pasos.

1. B utiliza su clave privada d_B para descifrar el mensaje y obtener el texto en claro calculando $c^{d_B} = (m^{e_B})^{d_B} \bmod n_B = m^{e_B d_B} \bmod n_B \equiv m \bmod n_B$.

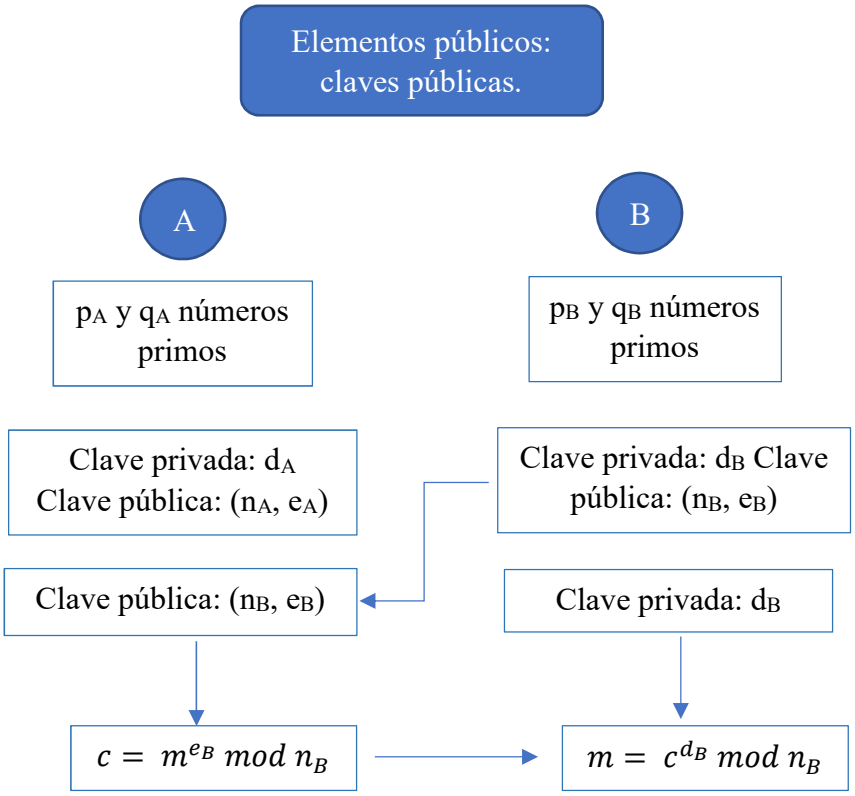

Figura 8. Esquema del criptosistema RSA.

Los ataques contra este sistema son tanto pasivos como activos. En los ataques activos el atacante se centra principalmente en modificar la transmisión como por ejemplo el ataque en mitad del camino; además influyen poniendo en entredicho la integridad de mensaje, la autenticidad de los participantes de la comunicación y la confidencialidad de los datos transmitidos. En los ataques pasivos, sin embargo, el adversario únicamente

observa el canal de comunicación y solamente se compromete la confidencialidad del mensaje transmitido, por ejemplo, el ataque al texto cifrado.

3.2.2.3. CRIPTOSISTEMA ELGAMAL

Este criptosistema fue desarrollado por Taher Elgamal (1955-) en 1984 siendo su uso libre. Al igual que el protocolo de intercambio de clave Diffie-Hellman, la seguridad de este sistema se basa en la dificultad de resolver el problema del logaritmo discreto.

Consideramos el caso particular en el que el grupo G es el grupo multiplicativo del cuerpo $\mathbb{Z}_p$ con p primo. El esquema se puede dividir en tres fases: generación de claves que se utilizarán en el cifrado y descifrado del mensaje, cifrado de mensajes y descifrado de mensajes.

Generación de claves: los elementos en común son el cuerpo $\mathbb{Z}_p$ y un generador del mismo α. Cada integrante de la comunicación debe seguir los siguientes pasos para generar las claves.

1. Clave privada del usuario A es el número aleatorio a.
2. La clave pública del usuario A es α^a en $\mathbb{Z}_p$.

Cifrado de mensajes: el usuario A quiere enviar un mensaje m ∈ $\mathbb{Z}_p$ a B y para ello debe seguir la siguiente secuencia.

1. El emisor A genera un número aleatorio v y obtiene el valor α^v en $\mathbb{Z}_p$.
2. A consigue la clave pública de B, es decir, α^b y calcula los valores $(\alpha^b)^v \ mod \ p$ y $m \cdot \alpha^{bv} \ mod \ p$.
3. El criptograma resultante es $(\alpha^v, m \cdot \alpha^{bv})$ y se envía al usuario B.

Descifrado de mensajes: el usuario receptor debe seguir las indicaciones siguientes para poder recuperar el mensaje original a partir del criptograma $(\alpha^v, m \cdot \alpha^{bv})$ enviado por el usuario A.

1. A partir de la primera parte del criptograma, α^v, B calcula $(\alpha^v)^b \ mod \ p$, siendo b su clave privada.
2. B obtiene el texto en claro realizando la siguiente operación $m \cdot \alpha^{bv} \cdot (\alpha^{vb})^{-1}$ en $\mathbb{Z}_p$.

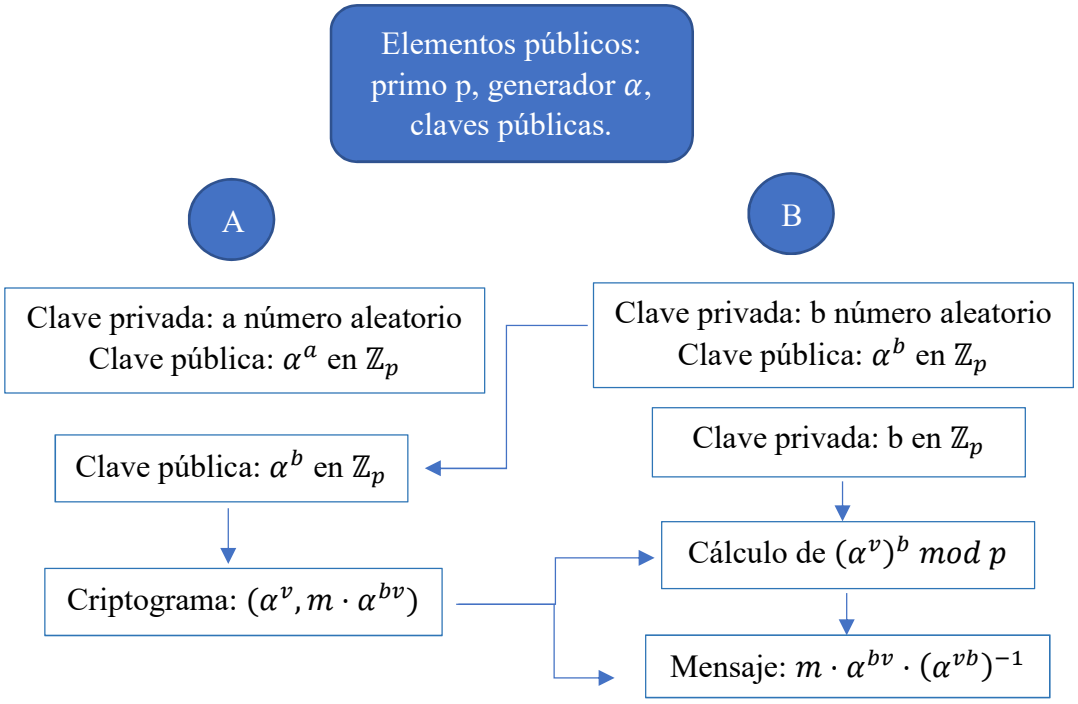

Figura 9. Esquema del criptosistema ElGamal.

3.2.3. FIRMA DIGITAL

La firma digital se utiliza para la autenticación de mensajes debiendo en general poder verificar el autor y en ocasiones la fecha, y también a veces el resto del contenido teniendo que poder verificarse por un tercero.

El protocolo general para elaborar la firma digital y la verificación es el mostrado en la siguiente secuencia de pasos: supongamos que el usuario A quiere firmar el mensaje M que envía a B.

Generación de la firma:

1. El usuario A calcula la rúbrica cifrando M o su hash dependiendo del algoritmo con su clave privada.
2. A cifra con la clave pública de B la rúbrica que acaba de obtener.

Verificación de la firma:

1. El usuario B recupera el mensaje.
2. B descifra con su clave privada la rúbrica que ha recibido de A.

3. Comprueba si la rúbrica que ha obtenido coincide con la enviada por A descifrando con la clave pública de A y realizando el hash del mensaje si fuera el caso.
4. Si coinciden las dos rúbricas entonces se verifica la firma.

Los principales sistemas de firma digital son la firma digital RSA y la firma digital ElGamal, aunque posteriormente en el año 1994 se establecerá como estándar de firma digital el DSA, que es una variante del algoritmo de ElGamal. En los siguientes pasos mostramos cómo generar la firma y cómo verificarla según DSA.

Elaboración de la firma:

1. El emisor A escoge un número aleatorio k de forma que $0 < k < q$.
2. A calcula $r = (g^k \bmod p) \bmod q$, siendo g el generador de $\mathbb{Z}_p$ de orden q. Con p primo que cumpla que $2^{511} < p < 2^{512}$.
3. Obtiene la función hash aplicada al mensaje m y resulta $H(m)$, a continuación, calcula $s = k^{-1}(H(m) + ar) \bmod q$, siendo a la clave privada de A.
4. A envía al receptor B la firma siendo ésta el par (r, s).

Verificación de la firma:

1. El usuario B recibe el par (r, s) y obtiene el valor de $w = s^{-1} \bmod q$.
2. Calcula $u_1 = (H(m)w) \bmod q$ y $u_2 = (rw) \bmod q$.
3. B calcula el valor de $v = [g^{u_1}(g^a)^{u_2} \; mod \; p] \; mod \; q$.
4. Finalmente, si se cumple que $v = r$ entonces se verifica la firma.

3.2.4. FIRMA DIGITAL CON CURVAS ELÍPTICAS

La criptografía y la firma digital con curvas elípticas surgen debido a la necesidad de suplir la longitud tan grande de claves que se utilizan en los algoritmos criptográficos asimétricos como es el caso de RSA, ElGamal y la firma basada en ambos DSA. El gran beneficio de la utilización de las curvas elípticas es que disminuyen considerablemente la longitud de las claves manteniendo el mismo nivel de seguridad o incluso mayor. El principal inconveniente es el mayor coste computacional. Se puede observar las mejoras de longitud de clave entre RSA y ECC (Elliptic Curve Cryptography) en la tabla comparativa de la Figura 10 obtenida de [6].

Longitud clave RSA (b)	Longitud clave ECC (b)
1024	160-223
2048	224-255
3072	256-383
7680	384-511
15360	512-571

Figura 10. Tabla comparativa de la longitud claves RSA y ECC.

En cuanto al uso de la firma digital en las criptomonedas y en la blockchain, objeto de nuestro trabajo, en concreto, en Bitcoin, base del resto de monedas criptográficas, la firma digital tiene tres propósitos: a) la firma prueba que el poseedor de la clave privada, quien es además el poseedor de los fondos monetarios, ha autorizado el gasto de una determinada cantidad de fondos en una transacción; b) imposibilidad de negar la transacción monetaria realizada por él; c) la firma prueba que la transacción ni ha sido ni ha podido ser modificada por nadie una vez que ha sido firmada.

A continuación, se estudiarán y analizarán las curvas elípticas y la firma digital de manera más profunda que en los anteriores apartados debido a que el proyecto y la implementación desarrollada se basa en estos conceptos.

3.2.4.1. DEFINICIÓN DE CURVA ELÍPTICA

Tal y como se explica en [6], una curva elíptica es una curva plana cuya ecuación tiene la siguiente expresión:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

Las Figuras 11 y 12 muestran ejemplos gráficos de curvas elípticas implementadas con la herramienta Maple.

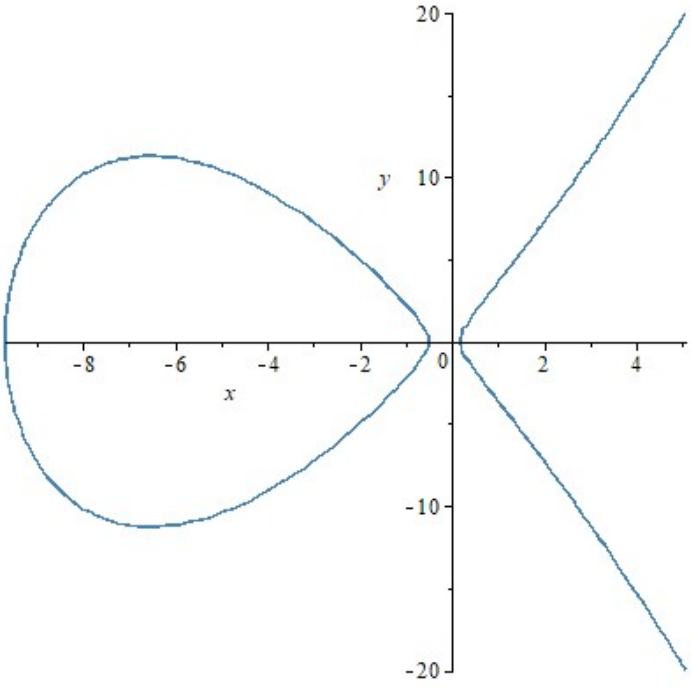

Figura 11. Curva elíptica E_1: $y^2 + 6 = x^3 + 10\,x^2 + 3x + 5$.

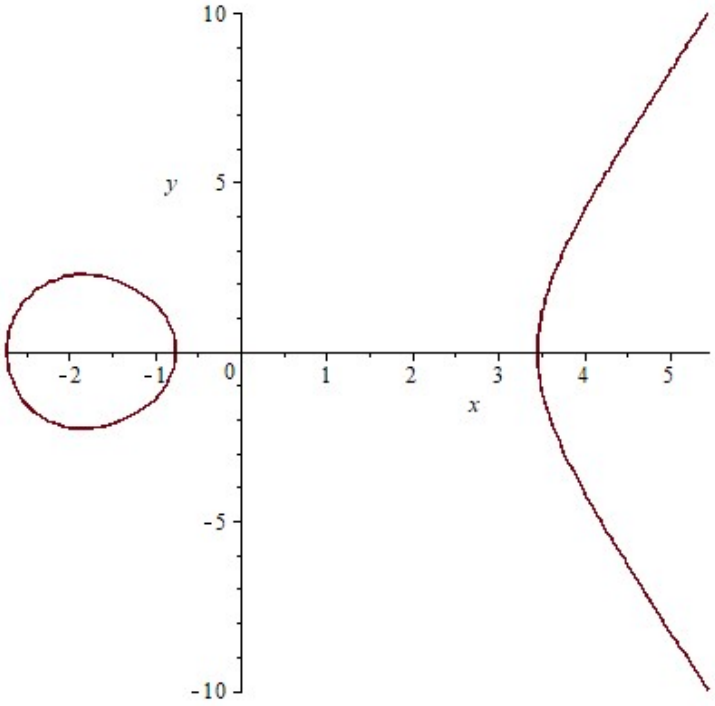

Figura 12. Curva elíptica E_1: $y^2 = x^3 - 10\,x + 7$.

Además, según [6], podemos afirmar que una curva elíptica E con la ecuación definida anteriormente, sobre un cuerpo F es una curva regular proyectiva que tiene al menos un punto racional y es de género 1. La nomenclatura para denominar a dicha curva elíptica es, E(F).

El número de puntos que tiene una curva elíptica se denota por #E(F) y se llama cardinal u orden.

Cualquier curva elíptica cumple una ecuación canónica, llamada ecuación de Weierstrass. La expresión de esta ecuación en coordenadas homogéneas es:

$$E: Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

Una característica importante que hay que tener en cuenta es que el discriminante de la ecuación tiene que cumplir que $\Delta \neq 0$, lo cual nos asegura que la curva sea regular.

Las curvas elípticas sobre un cuerpo F tienen estructura de grupo, por lo que podemos definir operaciones sobre ella. El punto en el infinito se denomina O su representación en coordenadas homogéneas es $O = [0 : 1 : 0]$. Además, dados dos puntos de la curva P y Q, podemos ver que se cumplen las siguientes propiedades, lo que lleva a que los puntos de la curva tengan estructura de grupo abeliano.

1. Se cumple la asociatividad, es decir, dados Q, P y R ∈ E, siendo E la curva elíptica, tenemos que $(Q + P) + R = Q + (P + R)$.
2. El elemento neutro existe, este es O, además, tiene que verificar que $O + P = P$ siendo P un punto de la curva elíptica E.
3. El elemento opuesto existe: dado un punto Q ∈ E, curva elíptica, existe el elemento Q' de tal forma que se tiene que verificar Q + Q' = O, donde Q' = -Q. Siendo Q' el elemento opuesto de Q y O el elemento neutro.
4. Se verifica la conmutatividad de tal forma que dados Q y P ∈ E, curva elíptica, Q + P = P + Q.

Los cuerpos sobre los que se definen las curvas elípticas son: cuerpos finitos, cuerpos primos y cuerpos binarios. Solamente existirán cuerpos finitos sobre los que se puedan representar curvas elípticas si el cardinal es de la forma p^m, donde p es un número primo y m es cualquier entero positivo. Podemos distinguir dos casos principalmente de cuerpos finitos, F_p y $F_2{}^m$.

En el caso de que el cuerpo finito sea F_p es porque p es un número primo y m = 1. A este cuerpo finito se le conoce como cuerpo finito primo. Los elementos que pertenecen a este cuerpo se encuentran en el siguiente conjunto {0, 1, …, p - 1}.

Si por el contrario p = 2 y m es cualquier entero positivo, entonces tenemos que el cuerpo finito es $F_2{}^m$, a este caso particular de cuerpo finito se denomina cuerpo finito binario. Los elementos de este cuerpo se representan mediante polinomios cuyos coeficientes están en F_2, es decir, en el conjunto {0, 1}.

3.2.4.2. OPERACIONES SOBRE LAS CURVAS ELÍPTICAS

Como ya se mencionó en el apartado anterior, dada una curva elíptica definida sobre un cuerpo tiene estructura de grupo. Debido a esta estructura podemos realizar operaciones sobre la curva elíptica. Nos centraremos en los dos cuerpos finitos principales, cuerpos primos y cuerpos binarios, para determinar las operaciones que se pueden realizar sobre las curvas.

La ecuación de una curva elíptica definida en un cuerpo finito primo se reduce a partir de la ecuación (1) mediante un cambio de variables, por lo que, en definitiva, utilizamos la siguiente ecuación.

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

Definimos las siguientes operaciones para los puntos pertenecientes a una curva elíptica sobre un cuerpo finito primo:

1. Sea Q un punto de E podemos obtener el mismo punto realizando la suma con el elemento neutro, curva elíptica en F_p, $Q + O = O + Q = Q$.
2. Si tenemos el punto Q, para obtener el elemento neutro O mediante la operación suma, utilizamos el elemento opuesto de la siguiente forma, $Q + Q' = O$. Siendo Q' el elemento opuesto y sabiendo que Q' = -Q,
3. La suma de dos puntos $Q = (x_Q, y_Q)$ y $P = (x_P, y_P)$ siendo $Q \neq P$, tenemos que el resultado es otro punto $S = (x_S, y_S)$ y se define la operación de la siguiente forma: $x_S = \lambda^2 - x_Q - x_P$; $y_S = \lambda(x_Q - x_S) - y_Q$; $\lambda = \frac{y_P - y_Q}{x_P - x_Q}$.
4. La operación de duplicar un punto se define como $Q + Q = 2Q = S$, siendo $Q = (x_Q, y_Q)$ y $S = (x_S, y_S)$, cuyos valores se obtienen con las siguientes operaciones: $x_S = \lambda^2 - 2x_Q$, $y_s = \lambda(x_Q - x_S) - y_Q$, $\lambda = \frac{3x_Q^2 + a}{2y_Q}$.

En el caso de que el cuerpo sobre el que esté nuestra curva sea el cuerpo finito binario F_2^m, aplicando un cambio de variables a (1), obtenemos la ecuación que define de forma más concreta a las curvas elípticas de este cuerpo finito.

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

Las operaciones en este cuerpo son:

1. Para todo punto $P \in E$, curva elíptica en F_2^m, $P + O = O + P = P$.
2. Sea P y P' el elemento opuesto, entonces, $P + P' = P + (-P) = O$.
3. Dados los puntos de la curva elíptica $P = (x_P, y_P)$ y $Q = (x_Q, y_Q)$ siendo $P \neq Q$, la suma de estos dos puntos nos da otro punto $S = (x_S, y_S)$ se calcula como $x_S = \lambda^2 + \lambda + x_P + x_Q + a$, siendo a el parámetro de la curva; $y_S = \lambda(x_P + x_S) + x_S + y_Q$; $\lambda = \frac{y_Q + y_P}{x_Q + x_P}$.
4. La duplicación de un punto de la curva $P = (x_P, y_P)$ da como resultado la suma de ese punto dos veces, es decir, el punto $S = (x_S, y_S)$. Los valores de esas coordenadas se calculan de la siguiente forma $x_S = \lambda^2 + \lambda + a$; $y_S = \lambda(x_P + x_S) + x_S + y_Q$; $\lambda = x_P + \frac{y_P}{x_P}$.

3.2.4.3. CÁLCULO DEL ORDEN DE UNA CURVA ELÍPTICA

En esta sección se estudiará cómo calcular el orden de una curva elíptica, es decir, #E, el número de puntos que tiene y; el orden de un punto que pertenezca a E.

Si la curva elíptica se define sobre un cuerpo finito F_q entonces, el orden de la curva será finito. Los puntos serán aquellos que satisfacen la ecuación de la curva más el punto del infinito O.

Tal y como podemos encontrar en [6] el teorema de Hasse otorga una aproximación respecto al orden de una curva con la expresión:

$$\#E(F_q) = q + 1 + t, \quad |t| \leq 2\sqrt{q}$$

en la que t representa la traza. Los algoritmos más destacados para poder obtener el número total de puntos que tiene una curva elíptica son el algoritmo de Schoof [7] y el algoritmo SEA (Schoof-Elkies-Atkin). Este último es más eficiente que el algoritmo de Schoof, ya que es una mejora del mismo. En la implementación llevada a cabo se utilizan librerías matemáticas SageMath que aplican este algoritmo para realizar de forma eficaz operaciones con las curvas elípticas.

Dado el punto Q de la curva elíptica E, el orden de este punto se expresa como n o como ord(Q) y, es el entero positivo más pequeño que hace que se cumpla la siguiente igualdad:

$nQ = O$, siendo O el punto del infinito.

Si nuestra curva elíptica E está definida sobre un cuerpo finito entonces el orden del punto Q ∈ E será un múltiplo del orden de la curva, es decir, de $\#E(F_q)$.

3.2.4.4. CÁLCULO Y REPRESENTACIÓN DE UN PUNTO DE UNA CURVA ELÍPTICA

Dada la curva elíptica E, definida sobre un cuerpo finito, concretamente nos centraremos en el caso de los cuerpos primos; podemos calcular un punto de la curva sustituyendo valores en su ecuación.

Con ayuda del género de la curva elíptica podemos determinar si tiene puntos racionales o no. Un punto de la curva se considera que es un punto racional cuando las coordenadas de dicho punto pertenecen al cuerpo sobre el que esta representada la curva elíptica. El género de una curva elíptica se calcula con la fórmula siguiente, siendo mP_i la multiplicidad de los puntos singulares P_i y con n siendo el grado del polinomio que define la curva elíptica E [6].

$$g = \frac{(n-1)}{2} - \sum_{P_i} \frac{mP_i(mP_i - 1)}{2}$$

Analizamos los posibles resultados que se pueden obtener del género de la curva.

- g = 0: la curva no tiene puntos racionales o tiene infinitos puntos racionales.
- g = 1: significaría que o no tiene puntos racionales o tiene un número finito de puntos o incluso que tiene infinitos puntos racionales.
- g ≥ 2: implicaría que la curva elíptica solamente puede tener un número finito de puntos racionales.

Para representar los puntos de una curva elíptica E sobre F_q se puede hacer de forma comprimida o descomprimida.

Si queremos representar el punto P = (x, y) en forma comprimida hay que tener en cuenta que el punto es la cadena K || H, siendo H la representación en hexadecimal de la coordenada x de P. El elemento K puede variar su valor y ser 0x02 o 0x03 según sea la coordenada y par o impar, respectivamente.

Para representar el punto P = (x, y) de la curva elíptica E en forma descomprimida, la representación es una cadena con la forma 04 || H1 || H2, siendo H1 y H2 las representaciones en hexadecimal de las coordenadas x e y respectivamente del punto P [6].

Es necesario tener en cuenta que si se quiere optimizar los recursos de memoria del computador y de almacenamiento es aconsejable utilizar el formato comprimido para representar los puntos de las curvas elípticas.

3.2.4.5. CURVAS ELÍPTICAS RECOMENDADAS EN LAS ESPECIFICACIONES Y ESTÁNDARES

Las curvas elípticas que se han recomendado aparecen en el artículo [1], siendo estas curvas son las más utilizadas para realizar la firma digital con ECDSA; por lo que también se han utilizado en el desarrollo de este proyecto para MECDSA. Las curvas utilizadas para este fin se muestran a continuación con sus parámetros: el primo p, el orden n, el coeficiente de la curva a y b y el punto base P en forma descomprimida o en sus dos coordenadas x e y.

- Curva P-256 del NIST (National Institute of Standards and Technology).

p=115792089210356248762697446949407573530086143415290314195533631308867097853951

n=1157920892103562487626974469494075735299969552241357603424222590610685120443 69
a= -3
b= 5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b
x= 6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296
y= 4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5

- Curva SM2 del Chinese State Cryptography Administration.

  p=fffffffeffffffffffffffffffffffffffffffff00000000ffffffffffffffff
  n=fffffffeffffffffffffffffffffffff7203df6b21c6052b53bbf40939d54123
  a=fffffffeffffffffffffffffffffffffffffffff00000000fffffffffffffffc
  b= 28e9fa9e9d9f5e344d5a9e4bcf6509a7f39789f515ab8f92ddbcbd414d940e93
  x=32C4AE2C1F1981195F9904466A39C9948FE30BBFF2660BE1715A4589334C74C7
  y=BC3736A2F4F6779C59BDCEE36B692153D0A9877CC62A474002DF32E52139F0A0

- Curva secp256r1 de estándares para la eficiencia criptográfica.

  p= ffffffff00000001000000000000000000000000ffffffffffffffffffffffff
  n= ffffffff00000000ffffffffffffffffbce6faada7179e84f3b9cac2fc632551
  a= ffffffff00000001000000000000000000000000fffffffffffffffffffffffc
  b= 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 7d2604b
  x=48439561293906451759052585252797914202762949526041747995844080717082404635286
  y=36134250956749795798585127919587881956611106672985015071877198253568414405109

- Curva secp256k1 de estándares para la eficiencia criptográfica.

  p=fffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f
  n=fffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
  a=0
  b=7
  x=79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
  y=483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

3.2.5. EL SISTEMA ECDSA

El sistema ECDSA (Elliptic Curve Digital Signature Algorithm) es como su nombre indica, un algoritmo que utiliza curvas elípticas para realizar firmas digitales y así, poder verificar la autenticidad del emisor por el receptor del mensaje que se ha enviado y asegurar la integridad del mensaje.

Este algoritmo matemático utiliza las operaciones descritas anteriormente sobre los puntos de las curvas elípticas, en vez de utilizar el problema del logaritmo discreto como originalmente se contemplaba en el DSA (Digital Signature Algorithm).

3.2.5.1. PROCESO DE FIRMA

Dada una curva elíptica E definida sobre F_p la cual tiene la siguiente ecuación:

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

Además, escogemos un punto de la curva P, al que llamaremos punto base, cuyo orden se denotará como n.

En primer lugar, será necesario calcular la clave pública y la privada. Escogemos un número aleatorio d ∈ [1, n] el cual será la clave privada. La clave pública se obtiene calculando Q = dP.

Para firmar el mensaje en claro m, hay que realizar los siguientes pasos:

1. Calculamos el valor e = H(m), siendo H una función hash segura.
2. Escoger un entero k aleatorio del intervalo [1, n-1] y calcular kP = (x, y)
3. Obtener el valor r = x mod n. Si r = 0, entonces tenemos que volver al paso 2 y volver a escoger un k distinto.
4. Calcular el valor s = $(\frac{e+d}{k})$ mod n. Si resulta que s = 0, entonces volvemos al paso 2 y escogemos un nuevo k.

El resultado es la firma del mensaje m siendo éste el par (r, s). En la Figura 13 se muestra el diagrama de flujo del método para generar la firma.

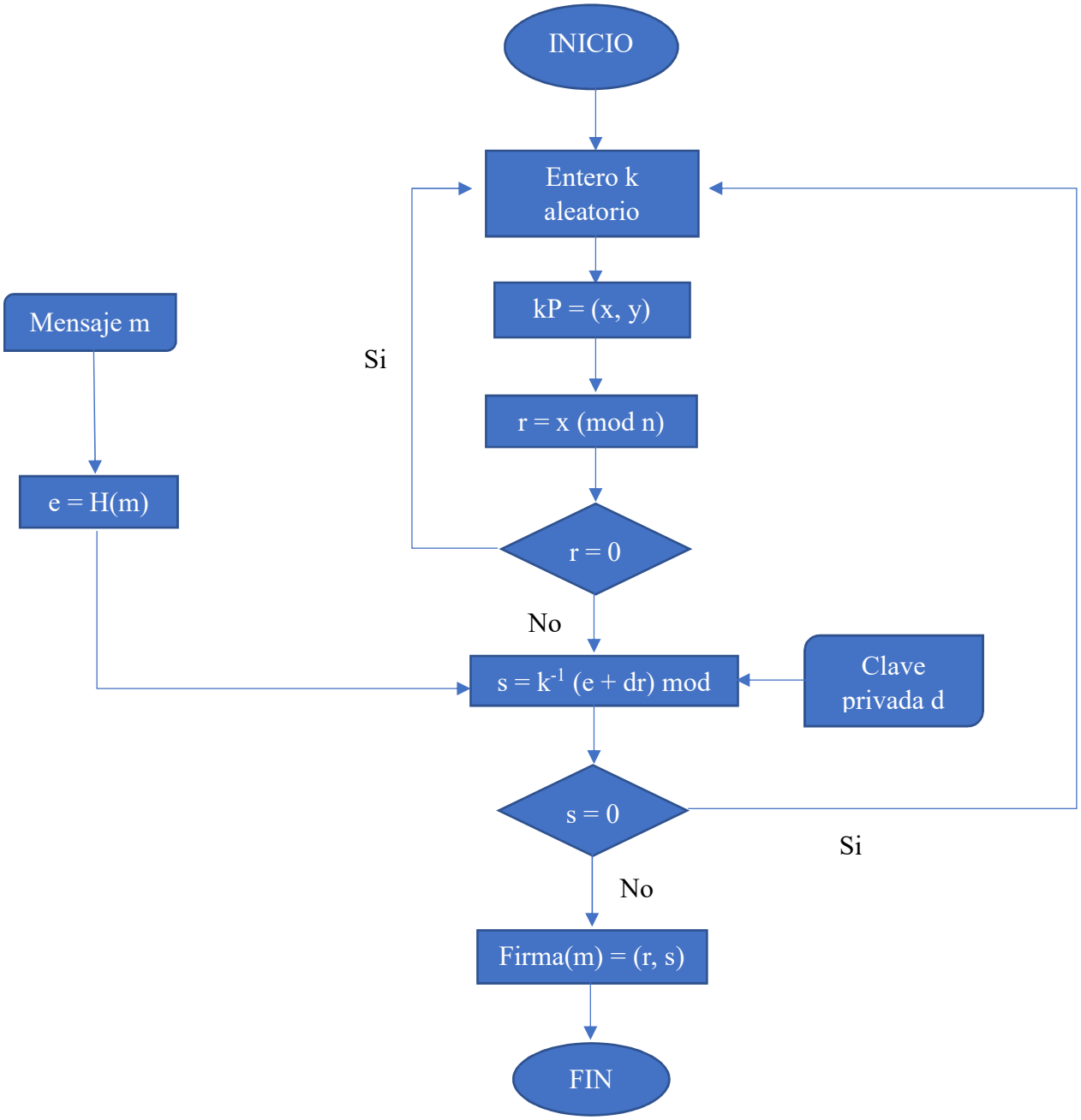

Figura 13. Diagrama de flujo del proceso de firma del sistema ECDSA.

3.2.5.2. PROCESO DE VERIFICACIÓN DE FIRMA

Para verificar si una firma, siendo esta el par (r, s), es válida para el mensaje m, hay que seguir estos pasos.

1. Comprobar que r y s son enteros y que están en [1, n-1]. Si no se cumple esto, la firma no es válida.
2. Calcular e = H(m) siendo H la misma función hash que se ha utilizado para firmar el mensaje m.
3. Calcular $w = s^{-1} \bmod n$.
4. Obtener los valores de u = (ew) mod n y v = (rw) mod n.
5. Calcular R = uP + vQ = (x, y).

La firma (r, s) será válida si r = x mod n, en cualquier otro caso la firma no es válida.

3.2.5. EL SISTEMA MECDSA

El sistema MECDSA (Multiple Elliptic Curve Digital Signature Algorithm) introducido en el artículo [1], es un algoritmo similar al sistema ECDSA. Realiza firmas digitales y además verifica si son válidas. La diferencia entre este algoritmo y el de ECDSA, es que utiliza más de una curva para realizar la firma. Cabe destacar que no es lo mismo realizar este algoritmo que repetir ECDSA k veces ya que el coste computacional es superior de esta última manera además de que, la longitud de la firma aumentaría considerablemente.

3.2.5.1. PROCESO DE FIRMA

Los parámetros iniciales de este algoritmo son por un lado el entero t que representa el número de curvas con las que se quiere realizar la firma, cuya ecuación es:

$$E_i: y^2 = x^3 + a_i x + b_i, \text{ en } F_{pi} \text{ con } i = 1, 2, \ldots, t.$$

Además, se necesita definir un punto base por cada curva elíptica, P_i además del orden de cada uno de los puntos definido como n_i, con i = 1, 2, …, t. En primer lugar, se tiene que generar la clave pública y la privada para cada una de las curvas. Para esto, escogemos un entero $d_i \in [1, n_i]$, siendo este número la clave privada. Obtenemos la clave pública a partir de la siguiente operación $Q_i = d_i P_i$, siendo i = 1, 2, …, t. Realizamos los siguientes pasos del algoritmo para firmar el mensaje m.

1. Calcular e = H(m) siendo H una función hash, en el caso de este proyecto se ha utilizado la función SHA-256.
2. Seleccionar un entero aleatorio k_i del conjunto $[1, n_i-1]$ y calculamos $k_i P_i = (x_i, y_i)$ con i = 1, 2, …, t.
3. Obtener el valor de la operación $r_i = x_i \bmod n_i$. Si $r_i = 0$, volvemos al paso 2 y escogemos otro k_i diferente para dicha curva.

4. Calculamos $r = r_1 + \ldots + r_t$. Si tenemos que $r = 0 \bmod n_i$ volvemos al paso 2 y repetimos los sucesivos pasos.
5. Calculamos $s_i = (k_i^{-1}[e + d_i r]) \bmod n_i$. Volver al paso 2 si resulta que $s_i = 0$.

La firma del mensaje m será $(r, s_1, s_2, \ldots, s_t)$.

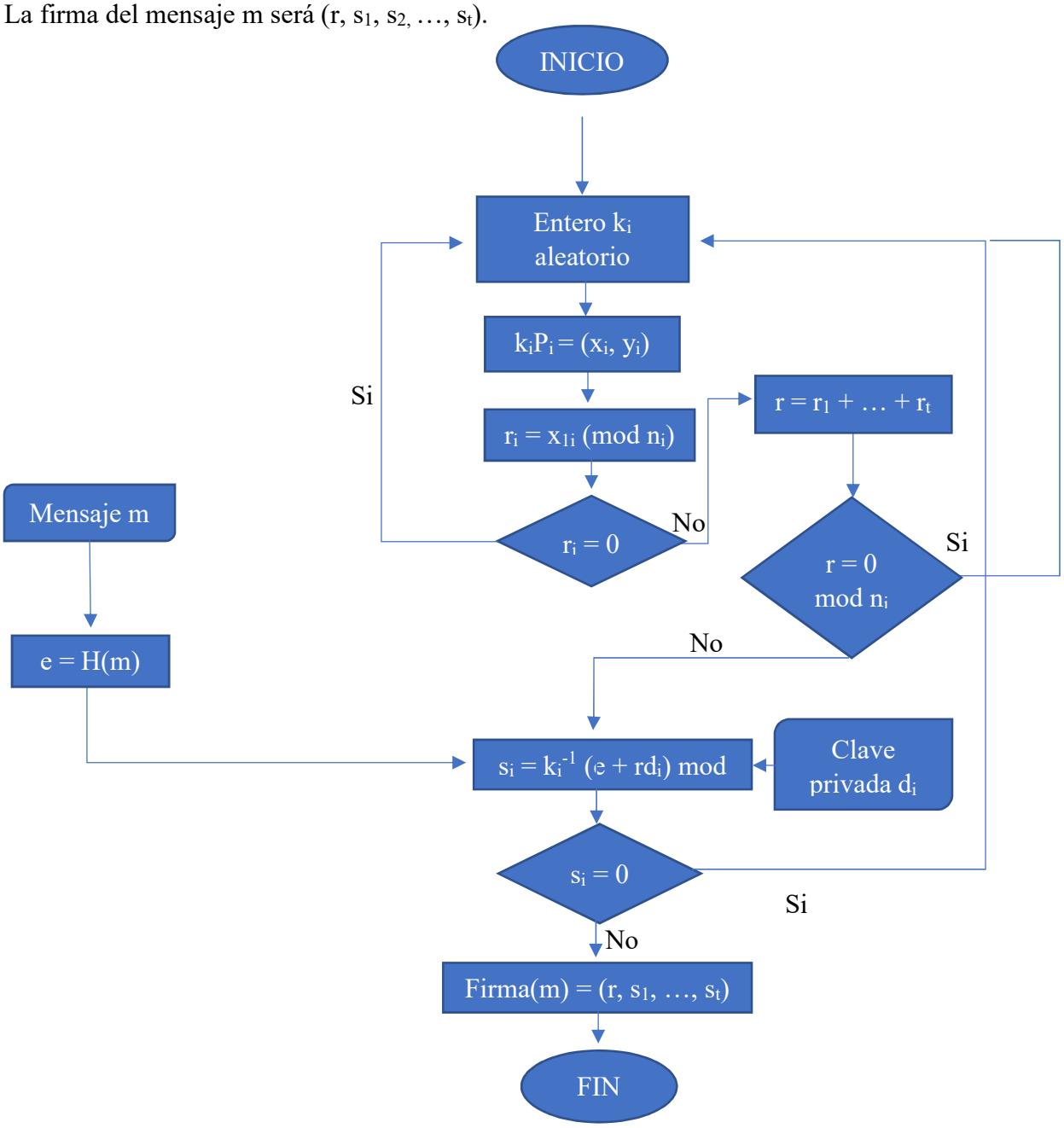

Figura 14. Diagrama de flujo del proceso de firma del sistema MECDSA.

3.2.5.2. PROCESO DE VERIFICACIÓN DE FIRMA

Dada la firma $(r, s_1, s_2, \ldots, s_t)$ del mensaje m, para verificarla basta con seguir los siguientes pasos.

1. Comprobar que r pertenece al conjunto $[t, n_1 + n_2 + \ldots + n_t - t]$ y es entero y que $s_i \in [1, n_i - 1]$, con i = 1, 2, …, t. Si no se cumplen estas condiciones, la firma no es válida.
2. Calculamos e = H(m) siendo m el mensaje y H la misma función hash utilizada para firmar, es decir, en este caso se ha utilizado SHA-256.
3. Calcular $w_i = s_i^{-1} \bmod n_i$.
4. Obtener el valor de $u_i = ew_i \bmod n_i$, $v_i = rw_i \bmod n_i$.
5. Calcular $R_i = u_iP_i + v_iQ_i = (x_i, y_i)$.
6. Calculamos $r'_i = x_i \bmod n_i$.

Si $r = r'_1 + \ldots + r'_t$ entonces la firma es válida, en cualquier otro caso no.

3.2.5.3. ANÁLISIS DE SEGURIDAD

La principal medida de seguridad eficaz que se puede encontrar en el sistema MECDSA es que utiliza curvas elípticas para el desarrollo de sus firmas por lo que hace que sea realmente complicado resolver el problema de invertir un logaritmo discreto en el grupo de puntos de dicha curva. Al igual que para el problema matemático del logaritmo discreto, no existe aún ningún algoritmo conocido que pueda resolver el problema que utiliza MECDSA en un tiempo razonable. Es por esto por lo que es realmente seguro el sistema múltiple de firma digital con curva elíptica.

La seguridad del algoritmo MECDSA propuesto [1] reside en la dificultad de resolver el problema ECDLP (Elliptic Curve Discrete Logarithm Problem). Analizamos los posibles ataques que puede sufrir el sistema.

Si el atacante quisiera obtener las claves privadas utilizando toda la información podría tratar de calcular $r_1, \ldots, r_t$, siendo parte de la firma del mensaje, conociendo la clave pública y la firma del mensaje, pero finalmente, el intruso tendría que resolver la siguiente ecuación $Q_i = d_iP_i$, en donde Q_i es la clave pública, d_i la clave privada y P_i es el punto base de la curva elíptica; pero como se puede observar el cálculo de esta operación es inviable debido a que no se puede resolver el problema del logaritmo discreto con curvas elípticas.

Ahora supongamos que el atacante quiere conseguir la clave privada de cada curva elíptica a partir de la clave pública y que todo el mundo puede conocerla. Al igual que en el caso anterior, es claro que tendría que resolver la ecuación $Q_i = d_iP_i$ o $K_i = k_iP_i$, lo que es inviable ya que tendría que resolver el ECDLP.

Por todo lo explicado anteriormente, el sistema MECDSA es realmente seguro y con longitud de clave menor en comparación con otros sistemas de firma digital.

4. IMPLEMENTACIÓN

A continuación, se explicará con detalle la implementación que se ha llevado a cabo. La aplicación desarrollada consiste en una interfaz a partir de la cual se realiza el protocolo MECDSA.

Existen tres modos diferentes. Modo predeterminado, donde el usuario escoge tres curvas elípticas estándares, introduce el mensaje que quiere firmar y finalmente el sistema aplica el algoritmo de cifrado y devuelve la firma. El modo avanzado, por el que se permite al usuario introducir los parámetros para generar su propia curva; el último modo, mezcla, que es una combinación del modo predeterminado y del avanzado.

Además, la aplicación incluye algunas funcionalidades más como ajustes, la cual incluye el cambio de aspecto, así como también se pueden descargar ficheros con el mensaje y la firma. Finalmente, también se ha incorporado una sección de ayuda.

4.1. SOFTWARE UTILIZADO

El lenguaje de programación que se ha utilizado para desarrollar la aplicación es Java, concretamente la versión 8, compilación 1.8.0_161-b12. Además, se ha utilizado el lenguaje Python en su última versión a través del entorno de desarrollo SageMath para algunas operaciones muy pesadas que no se podían implementar en Java.

El IDE (Entorno de Desarrollo Integrado) con el que se ha realizado la implementación ha sido NetBeans debido a la facilidad para implementar interfaces.

4.2. ESQUEMA DE LAS CLASES UTILIZADAS

El proyecto llamado “MECDSA_tfg”, se ha dividido en siete paquetes, estos son: andrea.tfg.dowloads, andrea.tfg.frames, andrea.tfg.panels, andrea.tfg.program, andrea.tfg.programConsola, andrea.tfg.resources, andrea.tfg.sage.

En cada paquete podemos encontrar las diferentes clases. A continuación, se muestra en la Figura 15 un esquema resumido de los paquetes en forma de rectángulo y las clases en forma de rectángulo con dos de sus vértices diagonales redondeados.

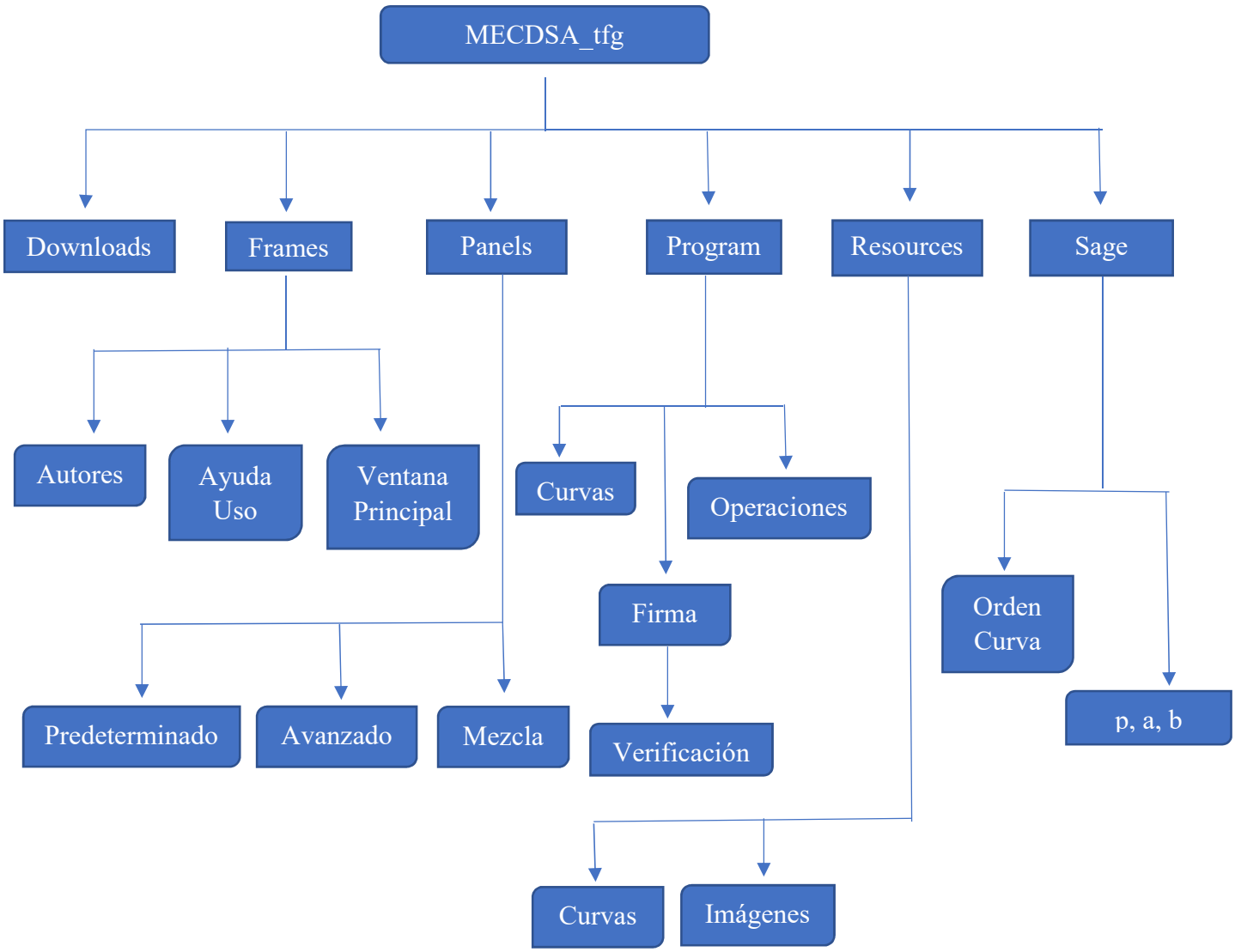

Figura 15. Esquema con paquetes y clases del proyecto MECDSA_tfg.

4.2.1. PAQUETES Y CLASES DE JAVA

A continuación, se explicará cada paquete del proyecto y las clases Java que contienen. Todos estos paquetes y clases han sido necesarios para poder llevar a cabo la aplicación que desarrolla el algoritmo MECDSA.

- Downloads: este paquete de trabajo tiene como principal función almacenar la firma digital y el mensaje que se quiere firmar. Se guardan en archivos de formato txt.
- Frames: paquete que contiene las ventanas o frames principales del sistema. Las clases que contiene son:
    - AutoresFrame.java: ventana que contiene información acerca de los autores del proyecto.
    - AyudaUsoFrame.java: ventana que contiene la ayuda al usuario sobre cómo utilizar la aplicación y para qué sirve.
    - VentanaPrincipalFrame.java: esta ventana contiene la imagen inicial de la aplicación y el menú con las opciones necesarias para el funcionamiento.
- Panels: paquete que contiene los paneles utilizados para poder transitar entre diferentes opciones.
    - PanelAvanzado.java: en este panel el usuario introduce valores para generar curvas aleatorias y un mensaje para realizar la firma.
    - PanelPredeterminado.java: el usuario escoge curvas ya definidas e introduce el mensaje que desea firmar.
    - PanelMezcla.java: este panel es una mezcla de los dos anteriores. El usuario escoge una curva ya definida y en la otra introduce los parámetros para generar una curva nueva.
- Program: incluye todas las clases del funcionamiento del algoritmo MECDSA, además de las operaciones de las curvas elípticas y la generación de las propias curvas.
- Resources: este paquete contiene los recursos tales como imágenes para el icono de las ventanas, el fondo de presentación de la aplicación y documentos que contienen los parámetros necesarios para formar las curvas de las especificaciones.
- Sage: contiene el archivo con extensión ipynb utilizado en Jupyter Sage para poder realizar ciertas operaciones como el cálculo del orden de una curva. Además, se crean archivos txt que contienen información sobre los parámetros de las curvas que genera el usuario de forma aleatoria.

4.2.2. DESARROLLO DE LOS JFRAMES

El primer JFrame o ventana que aparece nada más ejecutar la aplicación es el representado en la Figura 16, el cual corresponde con VentanaPrincipalFrame y como se puede apreciar es la ventana inicial de presentación de la aplicación que contiene una barra de menú con distintas opciones. Estas opciones son: Modo, Fichero, Ajustes y Ayuda.

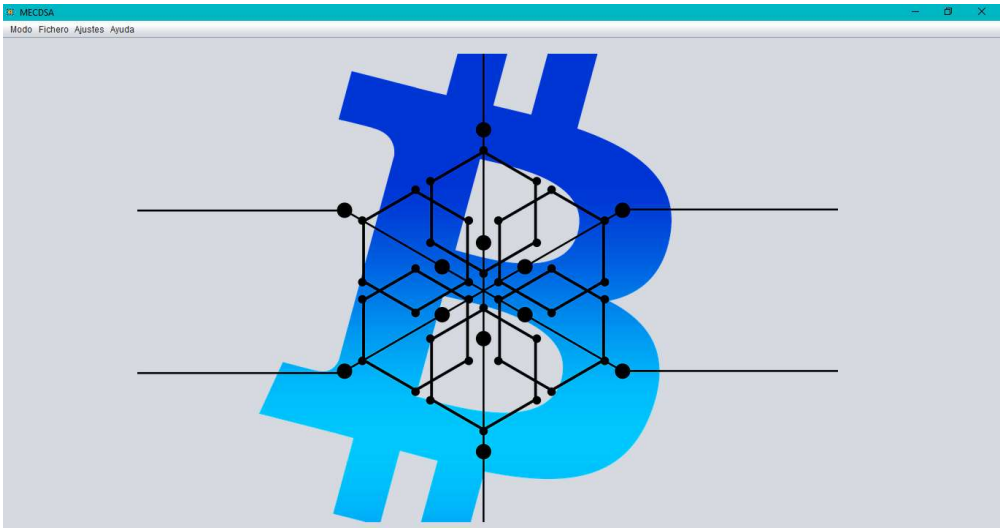

Figura 16. VentanaPrincipalFrame.

El siguiente frame que se ha desarrollado es AyudaUsoFrame, este se activa al presionar en cualquier momento la opción Manual del menú Ayuda. Aparece tal y como se muestra en la Figura 17.

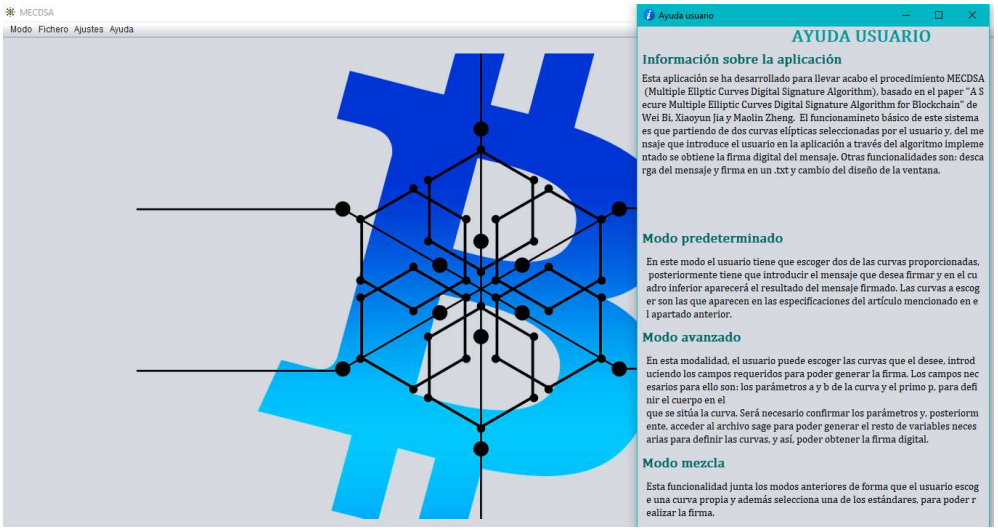

Figura 17. AyudaUsoFrame.

Y finalmente tenemos el frame AutoresFrame que incluye datos sobre lo autores de la aplicación, tal y como se muestra en la Figura 18.

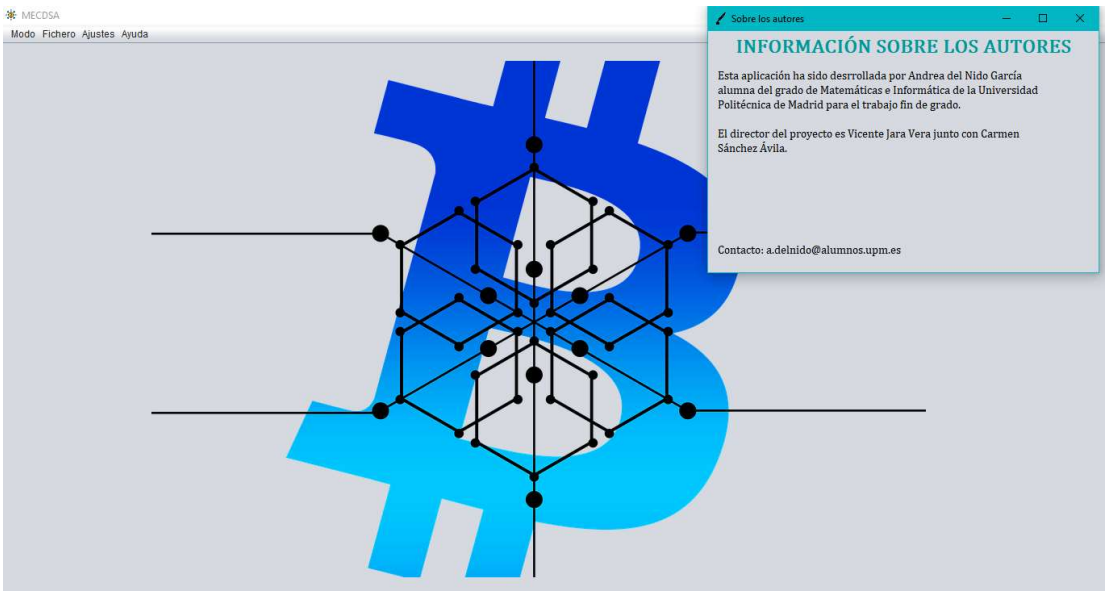

Figura 18. AutoresFrame.

4.2.3.DESARROLLO DE LOS JPANELS

La tarea principal de los paneles es contener los tres distintos modos de aplicar el algoritmo MECDSA. Para poder cambiar de modo basta con ir al menú Modo de la barra de menú superior.

El primer modo asociado a un panel es el Predeterminado. Este panel contiene las curvas ofrecidas en las especificaciones de [1], un cuadro de texto en el que el usuario debe introducir el mensaje que quiere firmar, y un cuadro de texto no editable en el que aparecerá la firma una vez introducidos todos los parámetros necesarios y habiendo pulsado el botón firmar como se aprecia en la Figura 19.

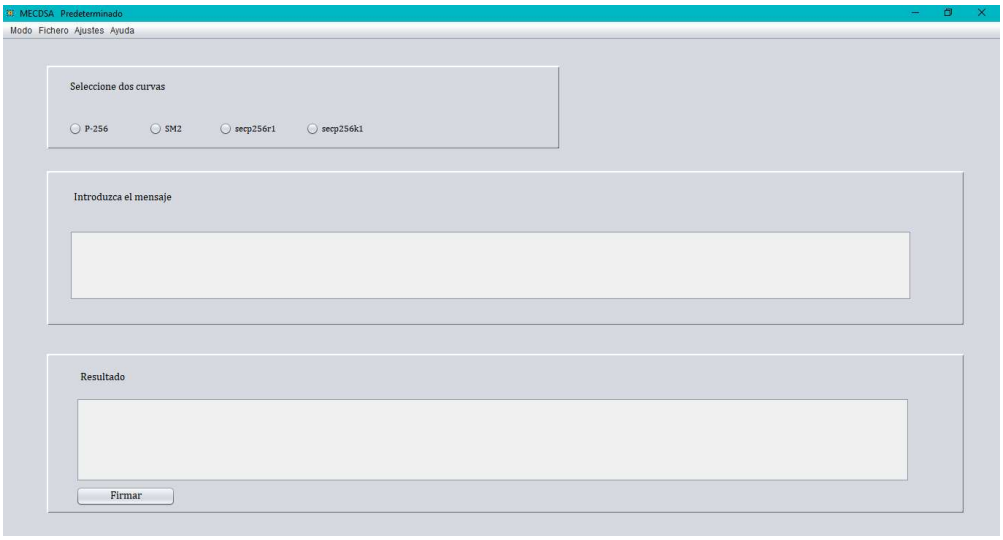

Figura 19. Modo Predeterminado.

El panel del modo Avanzado consiste en campos de texto en los que el usuario tiene que introducir los parámetros a, b y la longitud del primo en cantidad de dígitos binarios de cada una de las curvas con la que se realizará la firma, una vez introducido el mensaje. El programa calcula el primo más cercano posterior de la curva con esos parámetros aleatorios y, finalmente el orden para poder firmar el mensaje.

Para obtener el primo más cercano se ha implementado el símbolo de Lagrange y el test de Rabin Miller. En la Figura 20 podemos apreciar cómo es el panel Avanzado.

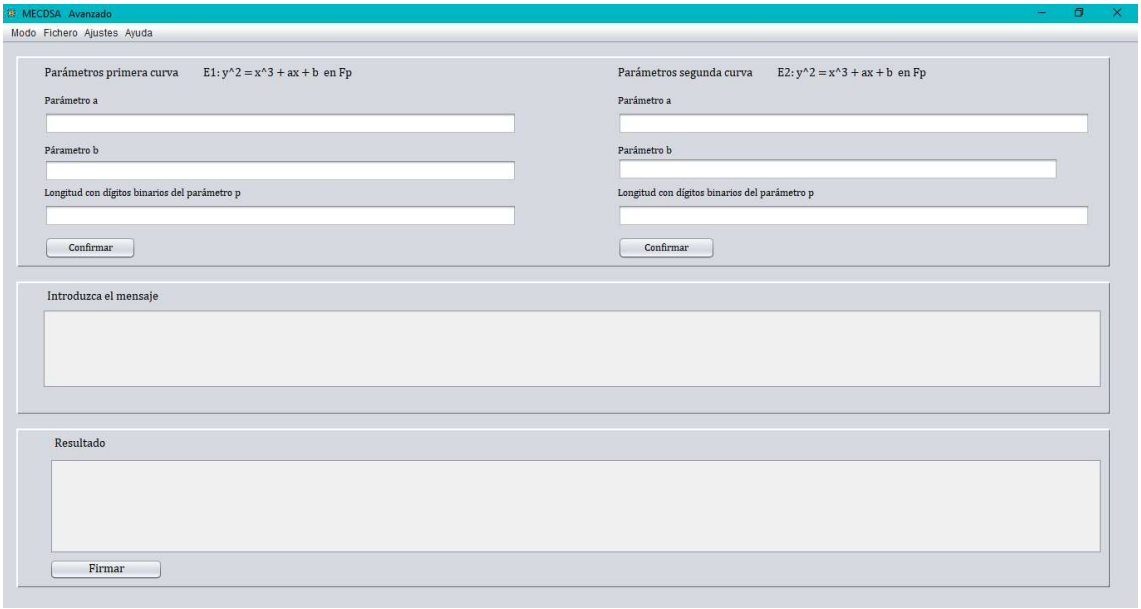

Figura 20. Modo Avanzado.

El último modo disponible en la aplicación es el modo Mezcla. Con este modelo lo que se pretende es realizar la firma del mensaje introducida por el usuario mediante una curva estándar, que selecciona el usuario entre varias opciones y, otra curva aleatoria generada por los parámetros introducidos por el usuario y por el programa. Podemos observar este modo en la Figura 21.

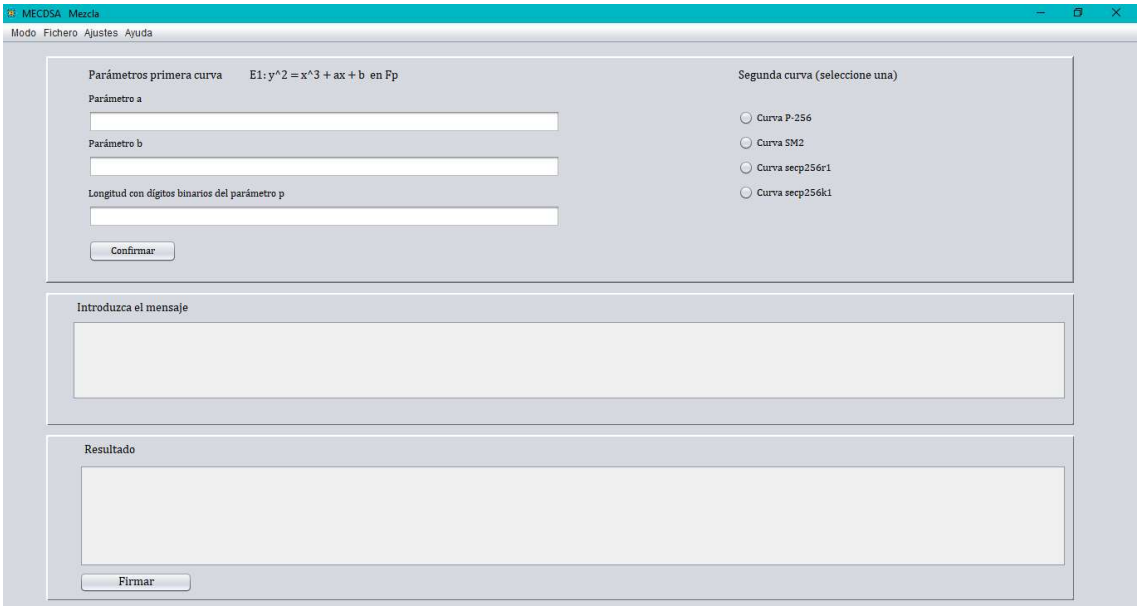

Figura 21. Modo Mezcla.

4.3. CONEXIÓN ENTRE SAGE Y JAVA

El proceso de la generación de curvas aleatorias es bastante costoso computacionalmente ya que es necesario calcular muchos parámetros de la curva como el primo p, el punto base, el orden del punto y el orden de la curva. Como estamos tratando con números grandes estás operaciones son complejas si queremos que se hagan en un tiempo razonable. Por todo esto, ha sido necesario realizar una conexión entre la aplicación y Notebook Sage, que realiza estas operaciones.

En Java no ha sido posible encontrar librerías que obtuvieran estos resultados de forma rápida por lo que se ha optado por Sage, que utiliza el lenguaje de programación Python y es una herramienta matemática mucho más fuerte y donde hay librerías que realizan estas operaciones.

La manera de proceder es la siguiente, el usuario introduce los parámetros de las curvas con las que realizará la firma. Estos datos se registran en un archivo el cuál leerá nuestro programa en Sage. Realizará las operaciones oportunas y, finalmente, devolverá a la aplicación un fichero con los parámetros que necesita el sistema para poder calcular la firma y mostrarla al usuario por pantalla.

4.4. PRUEBAS DE FUNCIONAMIENTO Y ERRORES

Se han realizado diferentes pruebas para verificar el correcto funcionamiento de la aplicación. Se han incluido mensajes de comprobación para poder determinar si las distintas operaciones realizadas por el programa funcionan correctamente. A continuación, mostramos algunas imágenes de las mismas.

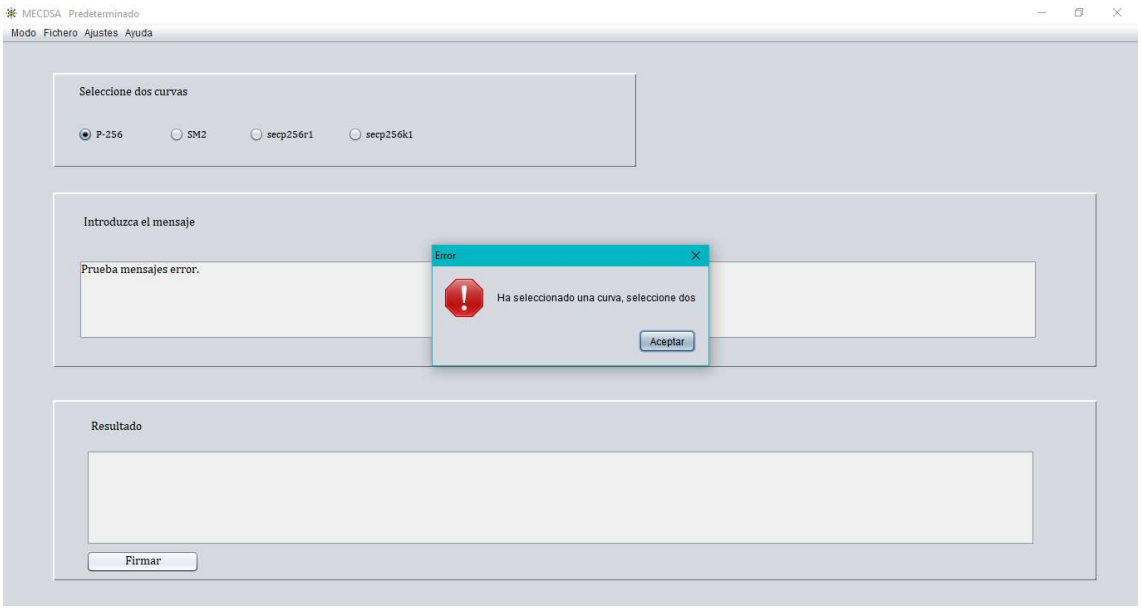

Figura 22. Mensaje de comprobación con el resultado de error.

Para todos los modos y para todos los posibles fallos por inserción incorrecta de datos del usuario se han establecido mensajes de error como el de la Figura 22. Además, se han realizado diferentes pruebas para analizar si el programa es capaz de realizar firmas digitales con caracteres especiales en el mensaje. Un ejemplo de ello es el mostrado en la Figura 23.

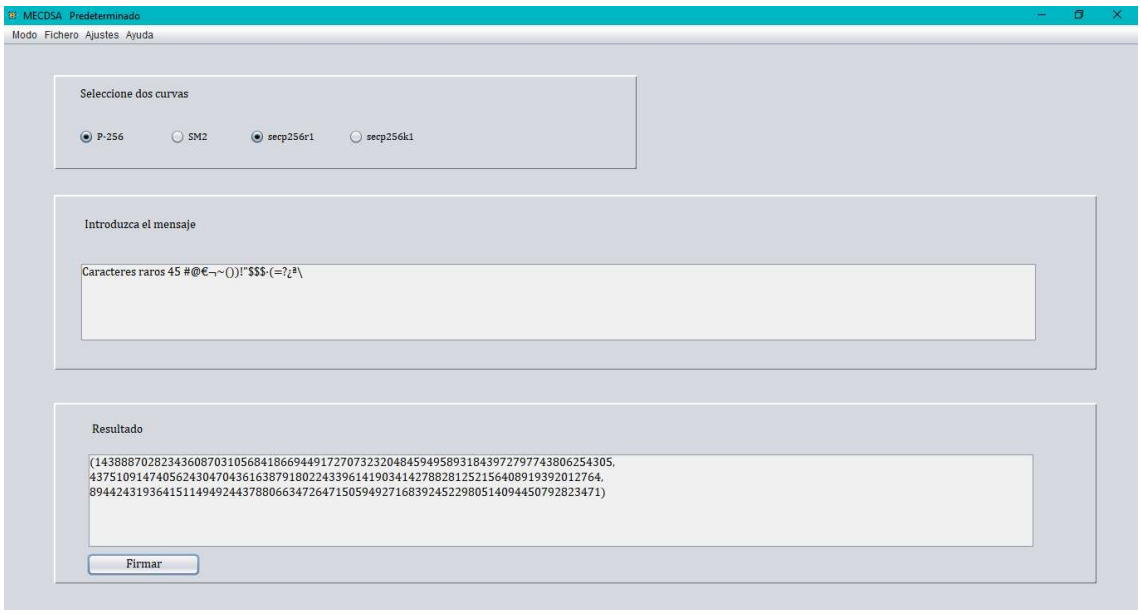

Figura 22. Análisis del resultado de la firma utilizando caracteres especiales.

Se han realizado pruebas para la verificación de las firmas digitales una vez obtenida y si no se verifica el resultado aparece un cuadro de diálogo mostrando el error de que no se ha confirmado.

También hay que tener en cuenta que se han realizado varias pruebas en la herramienta SageMath ya que se opera con la curva elíptica en dicho entorno de desarrollo. Si por ejemplo el parámetro p de la curva no es un número primo se muestra un mensaje de error indicándolo, aunque la aplicación está diseñada para que no ocurra ya que se utilizan algoritmos para obtener el número primo a partir de la longitud dada por el usuario y de los parámetros de la curva indicados por el usuario.

Se han realizado varias pruebas con diferentes parámetros para las curvas elípticas y obteniendo como resultado la firma digital en las modalidades Avanzado y Mezcla de la aplicación y en algunos casos es posible que se produzcan fallos en los cálculos como, por ejemplo, que no se pueda obtener el inverso al calcular la duplicación de un punto o al calcular un punto por un escalar; por esto mismo se ha incluido un mensaje de error advirtiendo al usuario para que cambie los parámetros. En la Figura 23 podemos observar un ejemplo de error de verificación de la firma. La Figura 24 muestra un ejemplo de la ventana de error que aparece en la aplicación al producirse un error interno en el cálculo de la firma debido a los parámetros introducidos por el usuario o por el uso de la herramienta SageMath.

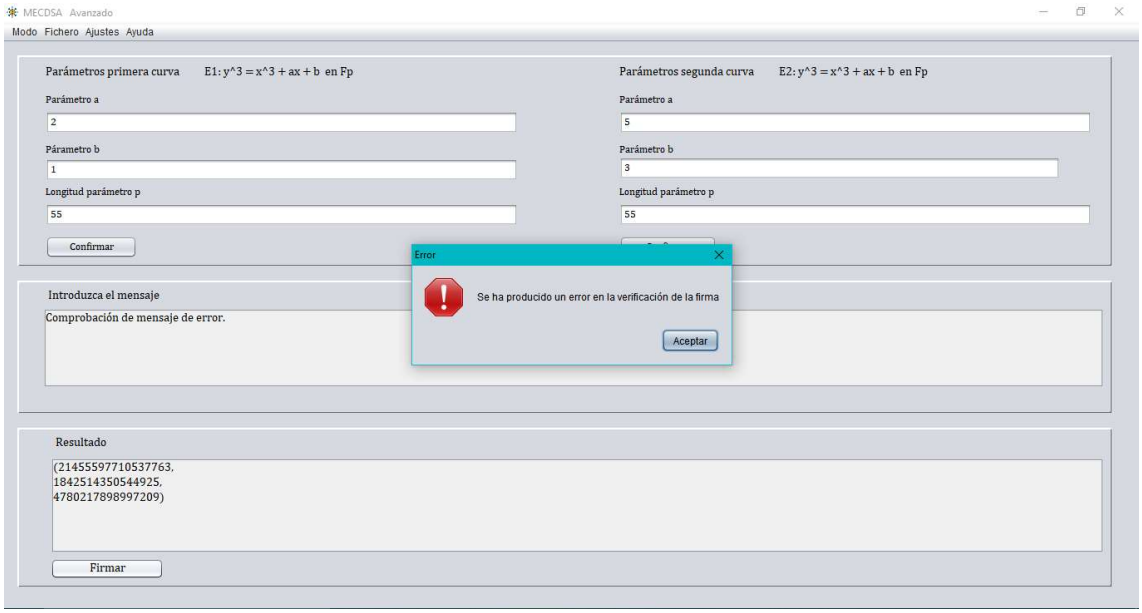

Figura 23. Ventana mostrando mensaje de error de verificación de la firma.

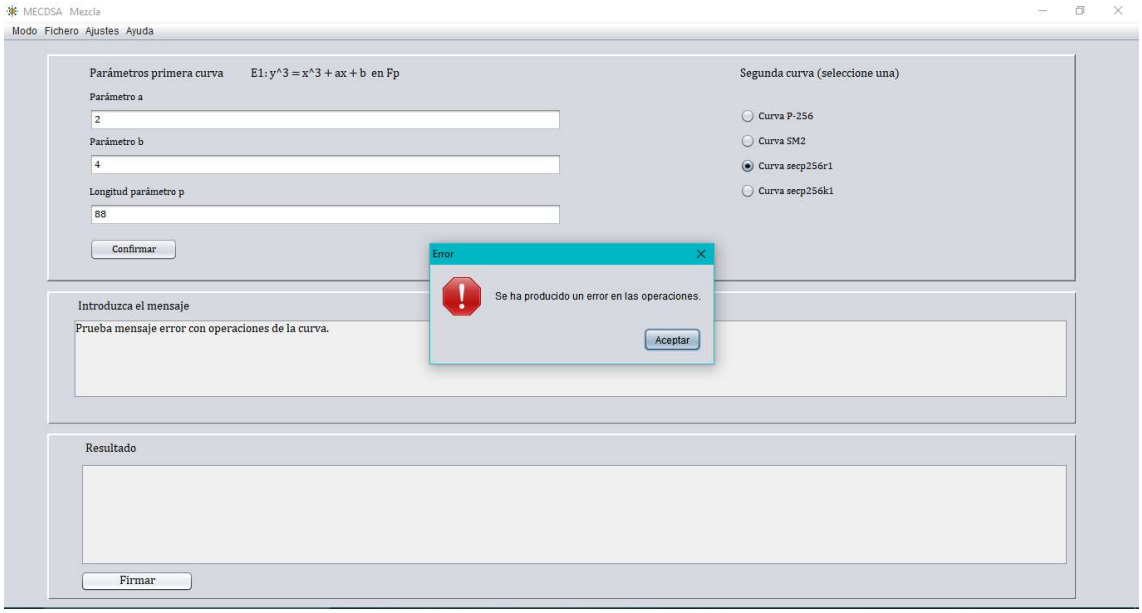

Figura 24. Ventana mostrando mensaje de error de operaciones.

Otro caso de error a tener en cuenta es la posibilidad de que el usuario no introduzca ningún mensaje por lo que no se podría firmar. En esta situación, el sistema mostrará una ventana emergente de aviso como la de la Figura 25.

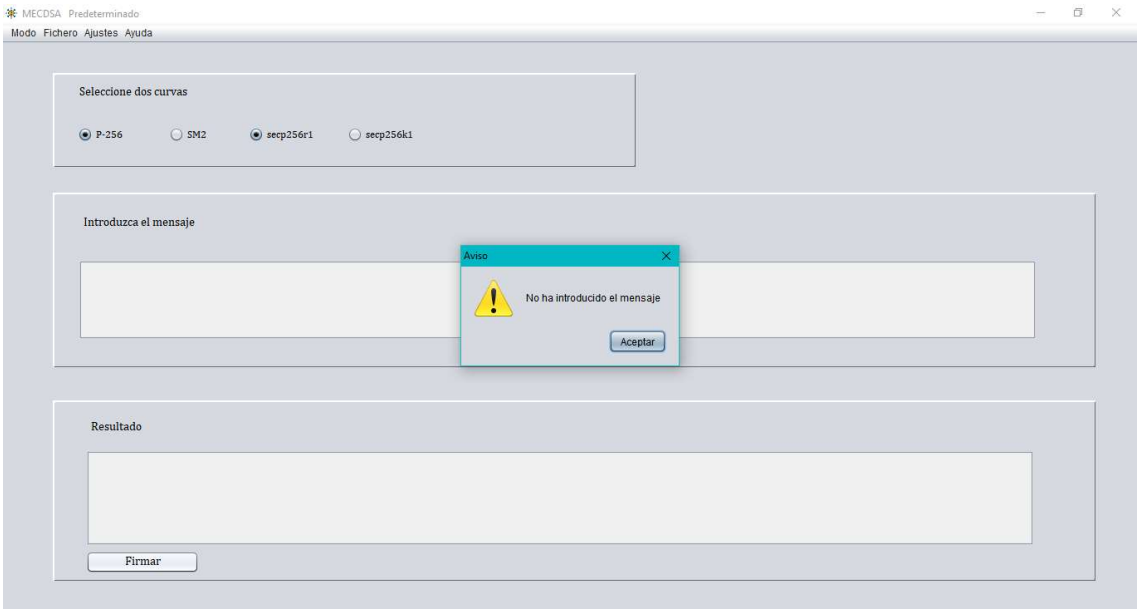

Figura 25. Ventana mostrando mensaje de aviso al no haber introducido el mensaje.

4.5. ANÁLISIS DEL RENDIMIENTO

En cuanto al rendimiento de la aplicación hay que tener en cuenta que ha sido necesario utilizar la herramienta SageMath para poder realizar operaciones con curvas elípticas ya que se han implementado desde cero para el modo Avanzado y el modo Mezcla. Al tratar con curvas elípticas las operaciones son muy costosas computacionalmente a la hora de generar los parámetros básicos (no en el curso habitual de la realización de las operaciones propias del cifrado y descifrado, o la firma digital) y por ello resulta complicado optimizar la implementación desarrollada para disminuir el tiempo de ejecución.

La máquina en la que se ha desarrollado la implementación de la aplicación y en la que se han llevado a cabo las pruebas tiene como sistema operativo Windows 10 y es de 64 bits por lo que no se pueden utilizar tamaños de variables demasiado grandes ya que una máquina convencional no podría realizar tantas operaciones. Si es cierto que se utilizan tamaños de clave y parámetros de curvas elípticas recomendados por los estándares y que son eficaces además de seguros.

A continuación, se analizan los tiempos de ejecución que tarda la máquina en diferentes situaciones. El programa tarda en ejecutarse aproximadamente entre 3 y 5 segundos en una máquina con las características detalladas. Respecto a la ejecución de los diferentes modos de la aplicación resulta que para el modo Predeterminado la ejecución tarda entre 2 y 4 segundos con parámetros máximos recomendados. Para las otras dos modalidades,

Avanzado y Mezcla el tiempo de ejecución es más complejo de obtener ya que es necesario ejecutar parte del código en SageMath.

5. RESULTADOS OBTENIDOS Y CONCLUSIONES

La principal conclusión a la que se ha llegado con el desarrollo de este proyecto es que, para mejorar la seguridad de las firmas digitales y conseguir así la autenticidad del emisor, es necesario generar curvas aleatorias de cierta cantidad de bits. El hecho de utilizar curvas aleatorias es más seguro que utilizar las curvas de los estándares para firmar ya que, es muy posible que, al conocer estas curvas elípticas y los puntos base de trabajo, terceras personas puedan identificar determinadas palabras comunes en el lenguaje, resultando ser más débiles.

Hay que tener en cuenta que el proceso de generación de curvas es costoso computacionalmente y debemos preguntarnos qué merece más la pena, la seguridad de la información o el rendimiento de los sistemas debiendo optarse por la seguridad siempre que los sistemas permitan unos tiempos de ejecución aceptables.

Gracias a este proyecto, se ha podido comprender mejor el funcionamiento matemático de los algoritmos de firma digital además de comprender los principales métodos de cifrado clásico, simétrico y asimétrico. También se ha podido profundizar en dos lenguajes de programación diferentes, Java y Python, en dos entornos de desarrollo completamente distintos, NetBeans y SageMath.

Se ha profundizado en el análisis y la comprensión del funcionamiento de la blockchain y las criptomonedas, concretamente de Bitcoin, así como la seguridad que requieren y las curvas elípticas que se utilizan en este sistema. Se ha llegado a la conclusión de que el sistema de cadena de bloques es realmente efectivo para poder realizar transacciones seguras que dependan de todos los participantes sin tener que depender de ningún sistema externo y que la seguridad más eficaz en estos modelos está basada en el empleo de las curvas elípticas para la autenticidad del emisor y la integridad del mensaje. Esto se debe a la menor longitud de clave que se necesita en el protocolo de firma.

6. PROPUESTAS DE MEJORA

Con el desarrollo de este proyecto hemos conseguido aplicar varias curvas elípticas para poder firmar un mensaje y verificarlo, sin embargo, a continuación, se han propuesto algunas posibles mejoras del proyecto de cara al futuro.

- Implementar el sistema para que pueda permitir el uso de más de dos curvas elípticas.
- Mejorar la conexión de Sage para que sea más fluida la aplicación. Implementar los algoritmos necesarios para poder generar curvas elípticas aleatorias, pero en Java.
- Cerciorarse de que los métodos aleatorios de Java realmente lo son, es decir, que no tengan ataques posibles de obtención de la pseudoaletoriedad, o sean fáciles de obtener las semillas de generación o no sean métodos altamente pseudoaleatorios.
- Incluir diferentes opciones de idioma en la aplicación.
- Implementar y desarrollar el algoritmo basado únicamente en curvas elípticas aleatorias.
- Integrar el sistema desarrollado de MECDSA con la blockchain.

ANEXO

Problema del logaritmo discreto.

Este problema consiste en obtener el valor de x en la ecuación modular $g^x = y$. No se conocen métodos clásicos que puedan obtener el resultado de un logaritmo discreto $\log_b y$. Existe una manera de obtener el resultado que consiste en elevar g en varias potencias hasta que finalmente se encuentre la potencia que necesitamos que es y.

Este problema no se puede resolver en un tiempo computacionalmente razonable, debido a la aritmética modular, razón por la que se utiliza este problema en criptografía, como ya hemos visto, por ejemplo, con el protocolo de intercambio de claves Diffie-Hellman y en el criptosistema ElGamal.

Problema de la factorización de números enteros.

Este problema utilizado en sistemas criptográficos como RSA se basa en que cada número entero se puede descomponer de manera única en números primos, si bien la principal dificultad de este problema es poder resolverlo en tiempo polinómico con números enteros muy grandes, habiéndose actualmente conseguido factorizar números de hasta 640 bits.

Ataques criptoanalíticos.

Estos ataques consisten en obtener los mensajes originales de los procesos criptográficos sin utilizar las claves, basándose en general en vulnerabilidades de los procedimientos matemáticos involucrados o incluso de las implementaciones software y hardware. Los ataques se pueden clasificar en ataques activos los cuales implican que se modifiquen los datos, y los ataques pasivos en los que el atacante únicamente escucha la información de la comunicación. Un ejemplo de ataque activo es la suplantación, es decir, el atacante se hace pasar por el receptor.

BIBLIOGRAFÍA

[1] Wei Bi, Xiaoyun Jia y Maolin Zheng, «A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain,» Seele Tech Corporation, San Francisco, USA, 2018.

[2] Real Academia Española, «Diccionario de la lengua española», Madrid: Espasa Calpe, S.A, 1997.

[3] Sandra Blain Escalona y Inclán Lázaro Vázquez, «Telemática. Revista digital de las tecnologías de la Información y de las Telecomunicaciones,» 2011. [En línea]. Disponible: http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/52/51. [Último acceso: 24 04 2019].

[4] Universidad-Politecnica-de-Valencia, «¿Qué es una Firma Electrónica?,» [En línea]. Disponible: https://www.upv.es/contenidos/CD/info/711250normalc.html. [Último acceso: 24 04 2019].

[5] Michael Crosby (Google), Nachiappan (Yahoo), Pradan Pattanayak (Yahoo), Sanjeev Verma (Samsung Research America) y Vignesh Kalyanaraman (Fairchild Semiconductor), «BlockChain Technology: Beyond Bitcoin,» *Applied Innovation Review,* nº 2, pp. 6-19, 2016.

[6] Víctor G. Martínez, Luis H. Encinas y Agustín M. Muñoz, «Criptografía con curvas elípticas», Madrid: CSIC, 2018.

[7] J. McGee, «René Schoof's Algorithm for Determining the Order of the Group of Points on an Elliptic Curve over a Finite Field», Blacksburg, Virginia: Virginia Polytechnic Institute and State University, 2006.

[8] Aqeel Khalique, Kuldip Singh y Sandeep Sood, «Implementation of Elliptic Curve Digital Signature Algorithm,» *International Journal of Computer Applications,* vol. 2, nº 2, pp. 0975-8887, 2010.

[9] Don Johnson, Alfred Menezes y Scott Vanstone, «The Elliptic Curve Digital Signature Algorithm (ECDSA),» *International Journal of Information Security,* vol. 1, nº 1, pp. 36-63, 2014.

[10] Hisham Dahshan, «An elliptic curve Key Management Scheme for Internet of Things,» *International Journal of Applied Research,* vol. 11, nº 20, pp. 10241-10246, 2016.

[11] Mehmet Adalier y Antara Teknik, «Efficient and Secure Elliptic Curve Cryptography Implementation of Curve P-256,» National Institute of Standards and Technology, LLC, 2015.

[12] Michael Nofer, Peter Gomber, Oliver Hinz y Dirk Schiereck, «Blockchain,» *Business & Information Systems Engineering,* vol. 59, nº 3, pp. 183-187, 2017.

[13] Moumita Roy, Nabamita Deb y Amar Jyoti Kumar, «Point Generation And Base Point Selection In ECC: An Overview,» *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 3, nº 5, pp. 6711-6713, 2014.

[14] Tetsuya Izu, Jun Kogure, Masayuki Noro y Kazuhiro Yokoyama, «Efficient Implementation of Schoof's Algorithm,» *International Conference on the Theory and Application of Cryptology and Information Security,* vol. 1514, pp. 66-79, 1998.

[15] Víctor Gayoso Martínez, Luis Hernández Encinas y Carmen Sánchez Ávila, «A Survey of the Elliptic Curve Integrated Encryption Scheme,» *Journal of computer science and engineering,* vol. 2, nº 2, 2010.

[16] Víctor Gayoso Martínez y Luis Hernández Encinas, «Implementing ECC with Java Standard Edition 7,» *International Journal of Computer Science and Artificial Intelligence,* vol. 3, nº 4, pp. 134-142, 2013.

[17] U.S. Department of Commerce y National Institute of Standards and Technology, «Secure Hash Standard (SHS), FIPS PUB 180-4, » Gaithersburg, MD, 2015.

[18] Carlos Dolader Retamal, Joan Bel Roig y Jose Luis Muñoz Tapia, «La blockchain: fundamentos , aplicaciones y relación con otras tecnologías disruptivas,» pp. 33-40.

[19] Joppe W. Bos, J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michale Naehring y Eric Wustrow, «Elliptic Curve Cryptography in Practice,» *Financial Cryptography and Data Security,* vol. 8437, pp. 157-175, 2014.

[20] Nigel P. Smart, «Cryptography Made Simple», Switzerland: Springer, 2016.

[21] Satoshi Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System,» 2008. [En línea]. Disponible: https://bitcoin.org.

[22] Shweta Lamba y Monika Sharma, «An Efficient Elliptic Curve Digital Signature Algorithm (ECDSA),» 2013.