Universidad Autónoma de Querétaro
Facultad de Informática

Desarrollo de Sistema Criptográfico basado en Curvas Elípticas para Firmas Digitales

Tesis

Que como parte de los requisitos para
obtener el Grado de
Maestra en Ciencias de la Computación

Presenta

Guadalupe Hernández Salmerón

Dirigido por:

Dr. Fidel González Gutiérrez

Co-Director:

M.C. Diego Abraham Olvera Mendoza

Querétaro, Qro. a 14 de Agosto del 2024.

Universidad Autónoma de Querétaro

Facultad de Informática

Maestría en Ciencias de la Computación

Desarrollo de Sistema Criptográfico basado en Curvas Elípticas para Firmas Digitales

Tesis

Que como parte de los requisitos para obtener el Grado de Maestra en Ciencias de la Computación

Presenta:

Guadalupe Hernández Salmerón

Dirigido por:

Dr. Fidel González Gutiérrez

Co-dirigido por:

M.C. Diego Abraham Olvera Mendoza

SINODALES

Dr. Fidel González Gutiérrez
Presidente

M.C. Diego Abraham Olvera Mendoza
Secretario

Dr. Arturo González Gutiérrez
Vocal

Dr. José Manuel Álvarez Alvarado
Suplente

Dr. Mauricio Arturo Ibarra Corona
Suplente

Centro Universitario Querétaro, Qro.
Agosto 2024
México.

Dedicatorias

En primer lugar, quiero expresar mi más profundo y sincero agradecimiento a mi querida familia. A mi mamá, por su amor incondicional, por ser mi refugio y mi fortaleza en cada paso de este camino. Gracias, mamá, por tus sacrificios, tu cariño inagotable y por creer en mí incluso en los momentos más difíciles. Tus sacrificios y tu dedicación me han mostrado el verdadero significado del amor. A mi Padre, quien aunque ya no está con nosotros, sigue siendo una fuente de inspiración y guía en mi vida. Te extraño cada día y tus enseñanzas y amor me acompañan siempre.

A mi querida Mony tu sensatez y sabios consejos siempre han sido una guía para mí. Gracias por cuidarme, por estar siempre ahí para escucharme y por brindarme tu apoyo incondicional. Tu equilibrio y tu fortaleza me han inspirado a ser mejor cada día. Agradezco cada momento que hemos compartido y todo el amor que me has dado. Te amo profundamente y este logro también es tuyo. A mi Vero, mi confidente, gracias por traer tanta alegría y diversión a mi vida. Tus palabras de aliento, tu amor, han sido fundamentales para llegar hasta aquí. Agradezco cada risa, cada conversación y cada aventura que hemos vivido juntas. Te amo profundamente y este logro no sería posible sin ti.

Dr. Fidel, no tengo palabras suficientes para expresar mi gratitud por su guía y apoyo a lo largo de mi maestría y licenciatura. Su sabiduría, paciencia y dedicación han sido esenciales para mi desarrollo académico y personal. Gracias por creer en mí y por brindarme las herramientas necesarias para superar cada desafío. Su compromiso con mi formación ha dejado una huella en mi vida. Le

agradezco de todo corazón por ser no solo un mentor, sino también una fuente de inspiración.

A mi mejor amigo Rogelio, por ser un pilar de apoyo y comprensión en los momentos de dificultad. Gracias por tu lealtad incondicional, tu comprensión, tus consejos y tu amistad han sido invaluables. Y, por supuesto, a Fernando, el amor de mi vida. Gracias por tu paciencia infinita, por tu amor y por estar siempre a mi lado, incluso en los momentos más oscuros. Tu apoyo ha sido mi ancla y tu fe en mí ha sido mi mayor motivación. Te amo con todo mi corazón.

Reconocimientos

Quiero expresar mi más sincero agradecimiento al CONAHCYT por la beca que hizo posible la realización de mis estudios. Su apoyo financiero fue crucial para alcanzar este objetivo, y estoy profundamente agradecida por la oportunidad brindada. Su compromiso con la educación y la investigación en México es admirable y ha tenido un impacto significativo en mi vida.

Extiendo mi más profundo agradecimiento a la Facultad de Informática de la Universidad Autónoma de Querétaro. Gracias por proporcionar los recursos, el entorno y el apoyo necesarios para desarrollarme académica y profesionalmente. Su compromiso con la excelencia educativa ha sido fundamental para mi crecimiento y éxito.

Agradezco sinceramente al sínodo por su evaluación y retroalimentación en este proceso. Su rigurosidad y profesionalismo han sido clave para el perfeccionamiento de mi trabajo. Su dedicación y compromiso con la calidad académica son verdaderamente apreciados.

Resumen

El presente trabajo aborda el diseño e implementación de un sistema criptográfico asimétrico basado en criptografía de curvas elípticas (ECC) y su comparación con el algoritmo RSA en términos de generación de llaves, cifrado, descifrado, y firmas digitales. Se implementaron ambos algoritmos utilizando Mathematica®, y se llevaron a cabo pruebas exhaustivas para evaluar el desempeño en cuanto a tiempo de ejecución y requisitos de almacenamiento. Los resultados demostraron que ECC supera significativamente a RSA en eficiencia, logrando una reducción del tiempo de ejecución de más del $70\,\%$ y una disminución de hasta el $75\,\%$ en los requisitos de almacenamiento. Estas mejoras se deben a la menor longitud de las llaves en ECC, lo que reduce el tiempo de procesamiento y el espacio necesario para almacenamiento. La investigación también destaca la relevancia de ECC en aplicaciones industriales, especialmente en el Servicio de Administración Tributaria (SAT) de México, donde la adopción de ECC podría traducirse en ahorros significativos en recursos y costos operativos, así como en una mejora en la capacidad de manejo de transacciones y la seguridad de los datos.

(**Palabras Clave: Criptografía, Curvas Elípticas, RSA, Firmas Digitales, Seguridad, Almacenamiento, Tiempo de Ejecución, SAT.**)

Abstract

This study addresses the design and implementation of an asymmetric cryptographic system based on elliptic curve cryptography (ECC) and its comparison with the RSA algorithm in terms of key generation, encryption, decryption, and digital signatures. Both algorithms were implemented using Mathematica®, and extensive tests were conducted to evaluate execution time and storage requirements. The results demonstrated that ECC significantly outperforms RSA in efficiency, achieving a reduction in execution time of over $70\,\%$ and a decrease in storage requirements of up to $75\,\%$. These improvements are attributed to the smaller key sizes in ECC, which reduce processing time and the required storage space. The research also highlights the relevance of ECC in industrial applications, particularly in Mexico's Tax Administration Service (SAT), where adopting ECC could lead to significant savings in resources and operational costs, as well as improved transaction handling capacity and data security.

(**Keywords: Cryptography, Elliptic Curves, RSA, Digital Signatures, Security, Storage, Execution Time, SAT.**)

Índice general

Índice de Figuras

Índice de Tablas

”Lo que sabemos es una gota, lo que ignoramos, un inmenso océano. La admirable disposición y armonía del universo no ha podido sino salir del plan de un Ser omnisciente y omnipotente”

Isaac Newton (1643-1727)
Físico, teólogo, inventor, alquimista y matemático inglés

CAPÍTULO 1

Introducción.

1.1. Prólogo.

La criptografía es el arte y una técnica de crear mensajes codificados usando llaves secretas con la finalidad de que solamente el destinatario final pueda descifrar el mensaje enviado a través de un canal de comunicación seguro o no seguro (Biddle et al., 2021). La palabra se forma a través del término griego $\kappa\rho\upsilon\pi\tau\sigma$ (kryptós) que significa oculto y el sufijo -graphos que quiere decir escritura, su definición etimológica sería escritura oculta (Rouse et al., 2019).

Las técnicas empleadas en la criptografía se relacionan estrechamente con el criptoanálisis, que estudia métodos para descifrar códigos y acceder a la información cifrada sin conocer la llave secreta. El objetivo principal de la criptografía es la protección de la información y la autenticación.

En la criptografía se encuentran dos formas de realizar el proceso de encriptación (Rouse et al., 2019):

a) El tipo simétrico el cual utiliza una única llave para encriptar y desencriptar mensajes; por ejemplo, el Estándar de Cifrado Avanzado (AES por sus siglas en inglés, Advanced Encryption Standard) el cual se estableció en el 2001 por el Instituto Nacional de Estándares y Tecnología (NIST por sus siglas en

inglés, National Institute of Standards and Technology).

b) La asimétrica utiliza dos llaves, una pública y una privada que se complementan entre sí ya que sirve una para encriptar y la otra para desencriptar mensajes o información; por ejemplo, RSA (nombrado así en honor a sus creadores Rivest, Shamir y Adleman), ECDSA (Elliptic Curve Digital Signature Algorithm), DSA (Digital Signature Algorithm), entre otros.

La criptografía ha evolucionado a lo largo de los años gracias a la aplicación de diversos procesos matemáticos como la factorización de números compuestos, el problema del logaritmo discreto (PLD) y la teoría de curvas elípticas. Estos procesos matemáticos han permitido la creación de sistemas criptográficos que aseguran la confidencialidad, integridad y autenticidad de la información, mediante la utilización de llaves públicas y privadas para encriptar y desencriptar mensajes.

El desarrollo del problema del logaritmo discreto (PLD) tuvo lugar en el año de 1976 por Whitfield Diffie y Martin Hellman donde su importancia radica en la obtención de la inversa a la exponenciación en un grupo. Diffie y Hellman proponían la aplicación de la llave pública y privada para cifrar y descifrar mensajes, la llave pública se comparte con el receptor y la llave privada se mantiene en resguardo sin compartir. Sin embargo, la llave privada no puede ser generada desde una llave pública (Diffie and Hellman, 1976).

En 1978 Ronald Linn Rivest, Adi Shamir y Leonard Adleman proponen el sistema criptográfico de llave pública (RSA), su funcionamiento se basa en un cifrado asimétrico (Rivest et al., 1978).

Posteriormente se presenta Taher Elgamal en 1985 describiendo el esquema del cifrado ElGamal, el cual está basado en el problema del logaritmo

discreto, este cifrado es aplicado principalmente tanto para generar llaves digitales como para cifrar y descifrar (Elgamal, 1985).

Víctor Miller y Neal Koblitz propusieron de forma independiente el uso de curvas elípticas en la criptografía. Argumentaron que este método es más eficaz y proporciona una seguridad equivalente a otros métodos.

Además, el tamaño de las llaves en este sistema es hasta tres veces menor que en los criptosistemas RSA(Miller, 1986; Koblitz, 1987).

La criptografía de curva elíptica o por sus siglas en inglés ECC (Elliptic Curve Cryptography) forma parte de la criptografía asimétrica, sus propiedades matemáticas se tornan sencillas de realizar, pero revertirlo es complejo, por lo que su eficiencia en seguridad es superior a otros métodos.

1.2. Descripción del problema.

Uno de los problemas a los que se enfrenta la generación de firmas digitales es respecto al tamaño de las llaves, como se detalla en la tabla 1.1. Es importante destacar que la generación de firmas digitales con RSA y ECC tiene diferentes características en cuanto al tamaño de llaves y tiempo de ejecución. Se muestra una comparación detallada para ambos algoritmos, donde se puede observar que ECC requiere llaves más pequeñas para generar una llave segura que RSA, lo que implica ventajas significativas en términos de almacenamiento y transmisión de datos. Según los resultados presentados en la tabla 1.1, el sistema ECC tiene una reducción en el tamaño de las llaves entre un $84\,\%$ y $96\,\%$ con respecto a RSA.

Es importante destacar que la generación de firmas digitales con RSA y

ECC tiene diferentes características en cuanto al tamaño de llaves y tiempo de ejecución. Uno de los problemas a los que se enfrenta la generación de firmas digitales es respecto al tamaño de las llaves, como se detalla en la tabla 1.1. Una llave más grande generalmente proporciona una mayor seguridad, ya que es más difícil de descifrar, pero también implica mayor almacenamiento y tiempo de procesamiento.

Tabla 1.1:
Longitudes de llave de RSA y ECC

Algoritmo	**80 bits**	**112 bits**	**128 bits**	**192 bits**	**256 bits**
Simétrico	80	112	128	192	256
ECC	163	133	283	409	571
RSA	1024	2240	3072	7680	15360

Nota: (Saho et al., 2020)

Otra problemática se presenta en la tabla 1.2 y está relacionado con el tiempo de ejecución computacional para la generación de llaves en ambos algoritmos. Se puede observar que ECC es más eficiente en términos de tiempo que RSA en todos los casos evaluados.

Tabla 1.2:
Tiempo de ejecución para la generación de llaves

Tamaño de llaves (bits)		**Tiempo de ejecución computacional (seg)**	
ECC	**RSA**	**ECC**	**RSA**
163	1024	0.219	0.958
233	2240	0.257	1.615
283	3072	0.197	6.122
409	7680	0.224	162.357
571	15360	0.156	2029.909

Nota: (Saho et al., 2020)

Teniendo en cuenta las tablas presentadas en el estudio comparativo de criptografía RSA y ECC, se puede observar que la generación de llaves RSA re-

quiere significativamente un tiempo mayor de ejecución que ECC, especialmente para tamaños de llave. Además, se necesita mayor almacenamiento para guardar las llaves RSA que para las llaves ECC. Esto puede ser una desventaja en el contexto de aplicaciones que requieren un procesamiento rápido y una gestión eficiente del almacenamiento.

Dentro del sistema de criptografía de curva elíptica (ECC) se encuentran dos algoritmos específicos utilizados para la generación de firmas digitales: Elliptic Curve Digital Signature Algorithm (ECDSA) y Elliptic Curve Nyberg-Rueppel (ECNR). El primero de ellos, denominado ECDSA, es un mecanismo de firma digital que utiliza la mencionada técnica de criptografía de curva elíptica. ECDSA se emplea en aplicaciones de seguridad para garantizar la autenticidad y la integridad de los datos. Este algoritmo es ampliamente utilizado en diversos protocolos de seguridad, incluyendo SSL/TLS, SSH, PGP y S/MIME, y se basa en la dificultad del problema del logaritmo discreto en una curva elíptica.

En contraste, ECNR es otro algoritmo de firma digital basado en curvas elípticas, que se desarrolló específicamente para mejorar el rendimiento y la seguridad de ECDSA en entornos con restricciones de recursos, como dispositivos móviles y sistemas embebidos. A diferencia de ECDSA, ECNR no emplea la función de hash y proporciona una mayor flexibilidad en cuanto a la elección de la curva elíptica.

Los resultados de la Tabla 1.3 indican que el tiempo de ejecución de la generación de la firma digital utilizando RSA y ECC para diferentes tamaños de llave. Los resultados indican que el tiempo de ejecución de los algoritmos ECC (ECDSA y ECNR) es significativamente menor que el tiempo de ejecución de RSA en cada tamaño de llave y el tiempo de ejecución en la generación de la firma digital.

Tabla 1.3:
Tiempo de ejecución de la generación de la Firma Digital RSA y ECC

Tamaño de llave (bits)		Tiempo de ejecución computacional (seg)		
ECC	**RSA**	**ECDSA**	**ECNR**	**RSA**
163	1024	0.009	0.057	0.036
233	2240	0.010	0.061	0.037
283	3072	0.010	0.060	0.067
409	7680	0.014	0.067	0.479
571	15360	0.018	0.083	3.015

Nota: (Saho et al., 2020)

Como puede observarse en las tablas presentadas anteriormente, los algoritmos basados en ECC ofrecen ventajas significativas, como obtener mejores tamaños de llaves y mejorar el tiempo de ejecución en la generación de llaves. Sin embargo, a pesar de estas ventajas, existen desafíos en su implementación práctica que deben ser abordados. Uno de estos desafíos es la complejidad en el proceso de adopción y adaptación de ECC en sistemas ya establecidos, lo cual puede requerir cambios significativos en la infraestructura tecnológica y en los protocolos de seguridad actuales.

Actualmente, el Servicio de Administración Tributaria de México (SAT) utiliza tecnologías de cifrado y firma digital para proteger la confidencialidad, integridad y autenticidad de la información fiscal. Entre las tecnologías utilizadas, destaca RSA como uno de los algoritmos de cifrado y firma digital empleados por el SAT. Lo cual conlleva una serie de desventajas y problemáticas.

Para la generación de firmas digitales, el SAT utiliza RSA con una longitud de llave de 2048 bits y un algoritmo de hash criptográfico seguro, SHA-256. Además, el SAT utiliza un certificado digital que contiene una llave pública y una llave privada, ambas generadas mediante el algoritmo RSA. Es importante destacar que el tamaño de las llaves pública y privada utilizadas por el SAT es de

1024 bits respectivamente. La llave privada es mantenida en secreto por el contribuyente, mientras que la llave pública es compartida con cualquiera. (DOF, 2017).

Los sellos digitales son un elemento fundamental en la operación del Servicio de Administración Tributaria (SAT) de México, estos utilizan dos tipos de sellos digitales: el sello digital del contribuyente y el sello digital del SAT. El sello digital del contribuyente es generado y resguardado por el propio contribuyente, y se utiliza para firmar y sellar los documentos fiscales que emite. Por otro lado, el sello digital del SAT es generado y resguardado por la propia autoridad fiscal, y se utiliza para validar y autenticar los documentos fiscales que recibe.

Para obtener el sello digital del SAT, los contribuyentes deben seguir un proceso de registro y validación ante la autoridad fiscal, que implica la presentación de cierta documentación y la verificación de la identidad y la situación fiscal del contribuyente. Una vez obtenido el sello digital como se muestra el ejemplo en la figura 1.1, el contribuyente debe utilizarlo para firmar y sellar los documentos fiscales que emite, de manera que puedan ser validados y autenticados por la autoridad fiscal(DOF, 2017).

Figura 1.1:
Ejemplo de Sello Digital generado con un certificado de 2048 bits

Nota: (Mendoza, 2021)

La firma digital basada en RSA es empleada por el Servicio de Adminis-

tración Tributaria (SAT) en México para asegurar la autenticidad e integridad de los documentos. No obstante, el empleo de este algoritmo conlleva ciertas limitaciones que inciden en su eficiencia y en los costos computacionales, término que se refiere a la demanda de recursos como tiempo de procesamiento, capacidad de cómputo y memoria. El tamaño de la llave requerida para generar firmas digitales con RSA es mayor que el de otros algoritmos criptográficos, lo que implica un incremento en estos costos computacionales durante la verificación de firmas.

1.3. Justificación.

Los sistemas criptográficos asimétricos RSA y ECC se utilizan para asegurar la confidencialidad e integridad de los datos. La generación de firmas digitales en ambos sistemas es una práctica común en la seguridad de la información.

Las firmas digitales son una herramienta crucial, permiten a los destinatarios de los datos verificar que el remitente es quien dice ser y que los datos no han sido modificados durante la transmisión. Además, las firmas digitales facilitan que los remitentes firmen electrónicamente los documentos y transacciones, lo que ahorra tiempo y reduce los costos asociados con la firma de documentos en papel.

Estas se utilizan en diversas aplicaciones de la vida cotidiana, desde la banca en línea hasta la autenticación de identidad en sitios web. Actualmente, es común que la mayoría de las transacciones financieras en línea se aseguren mediante el uso de firmas digitales, lo que garantiza la autenticidad de la información y previene el fraude (Chávez and Henríquez, 2021).

En México, las empresas deben generar facturas electrónicas que con-

tengan una firma digital válida para poder ser reconocidas por el Servicio de Administración Tributaria (SAT) y cumplir con las regulaciones fiscales. Las firmas digitales también se utilizan en la emisión de documentos oficiales, como el CURP, actas de nacimiento y certificados, para garantizar la autenticidad y proteger la información personal de los ciudadanos.

Estas firmas digitales se generan a través de algoritmos con un componente matemático robusto para la generación de llaves, una pública y otra privada. La llave pública se comparte con otras personas para que puedan enviar mensajes cifrados al propietario de la llave privada, mientras que la llave privada se mantiene en secreto (Seo, 2020).

Para la generación de llaves en los sistemas de criptografía RSA y ECC se utilizan técnicas diferentes. En el caso de RSA, se necesita generar dos números primos grandes que son utilizados para calcular la llave pública y privada. Estos números son escogidos aleatoriamente y se mantienen en secreto para evitar que terceros puedan acceder a ellos. Por otro lado, en ECC, se emplea una curva elíptica que tiene un número primo grande de puntos. No se utilizan números primos; en su lugar, se emplea una curva elíptica y un punto base para generar las llaves (Saho et al., 2020).

El uso de las firmas digitales en el SAT es de gran importancia debido a que estas son utilizadas como medio de autenticación y garantía de integridad en las transacciones electrónicas que se realizan con esta institución. Por lo tanto, es fundamental garantizar que las firmas digitales sean seguras, eficientes y confiables.

En este sentido, es relevante comparar el rendimiento de los algoritmos de generación de firmas basados en RSA y ECC ya que esto permitiría identificar

las fortalezas y debilidades de cada uno, y así tomar una decisión informada sobre cuál de ellos utilizar.

De acuerdo con la literatura existente, el uso de un algoritmo de generación de firmas basado en ECC podría ofrecer beneficios sobre el sistema RSA. Por ejemplo, se ha demostrado que este tipo de algoritmos son más eficientes en términos computacionales, lo que podría resultar en un tiempo reducido de procesamiento y transmisión de las firmas digitales. Además, las llaves utilizadas en el algoritmo ECC tiene una longitud reducida que las utilizadas en RSA para un nivel de seguridad equivalente, lo que implica una reducción en el tamaño de las firmas digitales y, por lo tanto, en el ancho de banda utilizado en la transmisión de estas (Saho et al., 2020).

La complejidad, tanto en almacenamiento como en tiempo de ejecución, es fundamental en el estudio de algoritmos de curvas elípticas. Se utiliza la notación Big O para describir el rendimiento de un algoritmo basándose en el tamaño de la entrada. En los algoritmos de firma digital, la complejidad temporal puede ser subexponencial, lo que indica un rendimiento potencialmente ventajoso, representada como:

$$O\left(\exp(c+O(1))\left(\ln(p)\right)^{\frac{2}{3}}\left(\ln(\ln(p))\right)^{\frac{2}{3}}\right)$$

El almacenamiento eficiente y el tiempo de ejecución reducido son consideraciones críticas para la aplicación de los esquemas criptográficos basados en curvas elípticas. Los objetivos deseados están orientados hacia la mejora de estas áreas, buscando algoritmos que equilibren efectivamente estas demandas de complejidad (Johnson et al., 2021).

La emisión de documentos con firmas digitales garantiza la autenticidad

de la información y previene el fraude, por lo que llevar a cabo una investigación en este campo de las ciencias de computación tiene un gran impacto en el robo de identidades. En consecuencia, la investigación está orientada al desarrollo de una aplicación para la generación y validación de firmas digitales contribuyendo a mejorar la eficiencia y seguridad de las transacciones electrónicas; como las realizadas en el SAT.

1.4. Hipótesis.

El desarrollo de una aplicación usando sistemas criptográficos asimétricos basado en algoritmos con ECC para la generación y validación de firmas digitales tiene características computacionales superiores en rendimiento que un algoritmo basado en RSA, reduciendo el tiempo de ejecución de al menos un 50 %.

1.5. Objetivos.

1.5.1. Objetivo general.

Desarrollar, en el lenguaje de programación de alto nivel Mathematica®, una aplicación de cifrado asimétrico basada en ECC para generar y validar firmas digitales con el fin de incrementar la eficiencia computacional y disminuir los costos asociados al sistema actual basado en RSA, utilizado por el Servicio de Administración Tributaria (SAT) en México.

1.5.2. Objetivos específicos.

1. Elaborar un algoritmo basado en curvas elípticas en Mathematica® para generar llaves públicas y privadas empleadas en sistemas criptográficos asimétricos con fines de encriptación y desencriptación.

2. Implementar un algoritmo basado en curvas elípticas y RSA para encriptar y desencriptar información en el lenguaje de programación Mathematica®.

3. Implementar un algoritmo para la generación de firmas digitales basado en ECC y RSA en el lenguaje de programación Mathematica®.

4. Implementar un algoritmo para la validación de firmas digitales basado en ECC y RSA en el lenguaje de programación Mathematica®.

”La única manera de hacer un gran trabajo es amar lo que haces”

Steve Jobs (1955-2011)
Empresario y magnate de los negocios en el sector informático y de la tecnología

CAPÍTULO 2

Marco de Referencia

El desarrollo de la investigación se basó en una metodología cuantitativa siguiendo un proceso estructurado. Se aplicó una metodología de diseño la cual consiste en la implementación de la solución propuesta en un ambiente controlado.

Además, se utilizó el modelo V para el diseño de software basado en una estructura en forma de V. Cada etapa del desarrollo se asocia con una etapa de pruebas correspondiente en la parte opuesta de la V. Esto permitirá detectar errores y problemas en cada etapa del proceso y corregirlos antes de avanzar a la siguiente etapa.

El lenguaje de programación de alto nivel Mathematica® fue utilizado en esta investigación debido a sus características para la realización de cálculos y gráficos necesarios para el uso de curvas elípticas y criptografía. Asimismo, brinda una ventaja para el análisis de datos y grafos, redes neuronales, visualizaciones dinámicas y modelado matemático, entre otras funciones.

El algoritmo que se implementó en la investigación es de tipo criptográfico asimétrico, también conocido como criptografía de llave pública. Este algoritmo utiliza un par de llaves, una pública y una privada.

2.1. Metodología de Diseño

La metodología de diseño para esta investigación siguió las siguientes etapas:

2.1.1. Investigación

En esta etapa, se realizó una investigación detallada sobre la criptografía asimétrica, específicamente sobre los algoritmos RSA y ECC, y su aplicación para la generación de firmas digitales. Se identificaron las necesidades y requerimientos para el uso de firmas digitales, de acuerdo al proceso realizado por el Servicio de Administración Tributaria (SAT) de México.

2.1.2. Definición del problema

En esta etapa, se definió claramente el problema, que es la ineficiencia en la generación de firmas digitales en el SAT, y se establecieron los objetivos y metas del proyecto, como la implementación de un algoritmo de generación de firmas digitales más eficiente basado en ECC.

2.1.3. Generación de ideas

En esta etapa, se utilizaron diversas técnicas para generar ideas y soluciones creativas para el problema identificado. Se consideraron las ventajas y desventajas de los algoritmos RSA y ECC y se seleccionó la curva elíptica más adecuada para la implementación del algoritmo de generación de firmas digita-

les basado en ECC. También se tomaron medidas de seguridad necesarias para garantizar la integridad y autenticidad de las firmas digitales.

2.1.4. Prototipado

En esta etapa, se desarrollaron prototipos o modelos preliminares de la solución propuesta utilizando el modelo V para la realización de la aplicación. Se diseñó e implementó el algoritmo de generación de firmas digitales basado en ECC en el lenguaje de programación de alto nivel Mathematica®. Se ajustó a las características de la curva elíptica seleccionada y se realizaron pruebas preliminares para identificar posibles problemas y mejorar la solución.

2.1.5. Pruebas y evaluación

En esta etapa, se evaluaron los prototipos y se realizaron pruebas para analizar el tamaño de las llaves en bits, así como el tiempo de ejecución para la generación de las llaves, generación de la firma digital y validación de la firma digital. Se identificaron posibles problemas y se mejoraron las soluciones.

2.1.6. Implementación

En esta etapa, se implementó la solución final, el algoritmo de generación de firmas digitales basado en ECC, y se realizó el seguimiento y evaluación continua de su desempeño. Se analizaron los resultados obtenidos en las pruebas y evaluaciones para determinar si el algoritmo basado en ECC es una solución viable para mejorar la eficiencia computacional en la generación de firmas digitales en el SAT. Se compararon los resultados obtenidos con los objetivos planteados

en la investigación y se presentaron las conclusiones.

2.1.7. Materiales

Para el desarrollo del proyecto y en particular para la implementación de los algoritmos se contó con el siguiente equipo y lenguaje de programación:

1. Equipo de cómputo Dell Inspiron 14-3467 con procesador Intel(R) Core (TM) i5-7200U a 2.50GHz, 16GB de RAM y sistema operativo Windows 10.
2. Licencia para estudiante Mathematica® versión 14.

2.2. Modelo V

Para el desarrollo de software se utilizó el modelo V. Las etapas del modelo son:

2.2.1. Análisis y especificación de requerimientos

En esta etapa se llevó a cabo el análisis de los requerimientos para el desarrollo del software y se definieron las especificaciones esenciales. Se identificaron las características y propiedades requeridas para el sistema criptográfico basado en RSA y ECC, y se definieron las especificaciones para el algoritmo de generación de firmas digitales basado en ECC.

Requisitos del Sistema

Funcionales:

- **Generación de llaves:**
  - Debe generar pares de llaves públicas y privadas para los algoritmos RSA y ECC.
  - Las llaves deben tener un tamaño adecuado (1024 bits para RSA y 160 bits para ECC).
- **Firma Digital:**
  - Debe permitir la generación de firmas digitales usando RSA y ECC.
  - Debe poder verificar firmas digitales para asegurar la integridad y autenticidad del mensaje.
- **Cifrado y Descifrado:**
  - Debe cifrar y descifrar mensajes usando RSA.
  - Debe cifrar y descifrar mensajes usando ECC.
- **Almacenamiento de llaves:**
  - Debe almacenar llaves públicas y privadas para poder medir el almacenamiento.

2.2.2. Diseño de sistema

En esta etapa se diseñó el algoritmo de generación de firmas digitales basado en ECC en términos de su arquitectura y su diseño de alto nivel. Se definieron parámetros y la lógica de programación necesarias para el algoritmo.

Arquitectura del Sistema

El sistema está dividido en varios módulos principales:

- **Módulo de Generación de llaves:**
  - Este módulo es responsable de la generación de pares de llaves públicas y privadas para RSA y ECC.
  - Usa funciones matemáticas para generar números primos grandes (para RSA) y puntos en la curva elíptica (para ECC).
- **Módulo de Firma Digital:**
  - Este módulo proporciona funciones para firmar digitalmente mensajes y verificar dichas firmas.
  - Implementa los algoritmos RSA y ECDSA para la firma y verificación de mensajes.
- **Módulo de Cifrado y Descifrado:**
  - Este módulo maneja el cifrado y descifrado de mensajes usando RSA y ECC.
  - Asegura que los mensajes cifrados sean seguros y puedan ser descifrados únicamente por los destinatarios previstos.

En la Figura 2.1 se presenta el esquema de autenticación de Firma Digital por ECC:

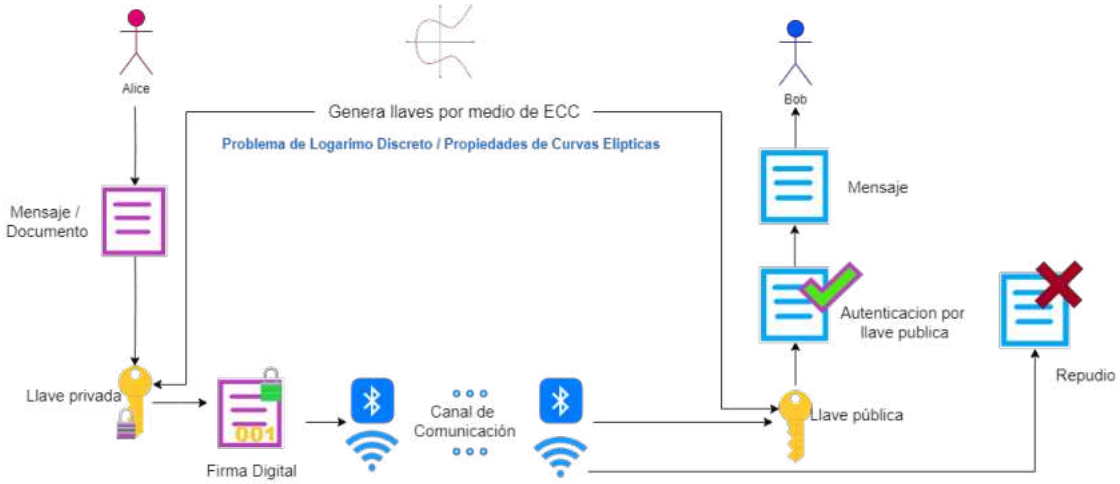
1. Alice tiene un mensaje para enviar a Bob
2. Alice firma digitalmente con su llave privada

3. Alice envía el mensaje por un canal de comunicación (inseguro)

4. Bob recibe el mensaje de Alice

5. Bob comprueba con la llave pública de Alice su autenticidad

Figura 2.1:
Ejemplo de Sello Digital generado con un certificado de 2048 bits



Nota: Elaboración propia.

2.2.3. Implementación

Posteriormente, se procedió a implementar el algoritmo del sistema criptográfico con el objetivo de garantizar la integridad y autenticidad de las firmas digitales. Su implementación se llevó a cabo en el lenguaje de programación de alto nivel Mathematica®.

2.2.4. Pruebas unitarias

Una vez implementado, se realizaron pruebas unitarias para verificar la funcionalidad del programa y se ajustaron las características de la curva elíptica

seleccionada. Asimismo, se garantizó que se cumplan todas las especificaciones definidas en la etapa de análisis y especificación de requerimientos.

2.2.5. Pruebas de integración

En esta fase, se llevaron a cabo pruebas de integración para comprobar que el programa interactúa adecuadamente con el resto del sistema y que el sistema completo cumple con los requisitos previamente definidos.

2.2.6. Pruebas de aceptación

Se procedió a realizar pruebas de aceptación para verificar que la aplicación cumple con los requerimientos previamente definidos. Además, se llevaron a cabo pruebas y evaluaciones del algoritmo implementado para analizar el tamaño de las llaves en bits, así como el tiempo de ejecución para la generación de las llaves, generación de la firma digital y validación de la firma digital. Se garantizó que el software esté libre de errores y funcione de manera adecuada.

2.2.7. Validación

Una vez finalizadas las pruebas y evaluaciones, se procedió a validar la aplicación, comprobando que se cumplen las especificaciones definidas en la etapa de análisis y especificación de requerimientos. Se analizaron los resultados obtenidos y se verificó que el programa funciona adecuadamente.

2.2.8. Mantenimiento

Una vez finalizadas las pruebas y evaluaciones, se realizaron tareas de mantenimiento del software, tales como la corrección de errores y la actualización de funcionalidades para garantizar su correcto funcionamiento.

En la figura 2.2 se detalla cada una de las fases del modelo V para el desarrollo de la aplicación:

Figura 2.2:
Modelo V

Nota: Elaboración propia.

2.3. Fundamentos del Álgebra Abstracta

2.3.1. Grupos

Los grupos son fundamentales en el ámbito de las estructuras algebraicas y son aplicables en diversos campos como la criptografía, la teoría de códigos y técnicas de conteo. Estos sistemas matemáticos comparten propiedades específicas que se definen a continuación.

Un grupo se define como un par ordenado $(G, \circ)$, donde G es un conjunto no vacío y $\circ$ es una operación binaria aplicada sobre los elementos de G. Para que $(G, \circ)$ constituya un grupo, debe cumplir con las siguientes condiciones:

- Cerradura: Para todo a, b en G, el resultado de la operación $a \circ b$ también debe pertenecer a G.
- Asociatividad: La operación debe ser asociativa, es decir, para cualquier a, b, c en G, se cumple que $(a \circ b) \circ c = a \circ (b \circ c)$.
- Elemento identidad: Debe existir un elemento e en G tal que, para cualquier elemento a en G, la operación de a con e resulte en a ($a \circ e = e \circ a = a$).
- Elemento inverso: Para cada elemento a en G, debe existir un elemento b en G tal que $a \circ b$ y $b \circ a$ resulten en el elemento identidad e.

Si se añade la condición de que la operación es conmutativa, es decir, $a \circ b = b \circ a$ para todos los a y b en G, entonces el grupo se denomina Abeliano o conmutativo. Este término honra al matemático noruego Niels Henrik Abel (1802-1829) (Trappe and Washington, 2006).

2.3.2. Campos

Un campo es una estructura algebraica que consta de un conjunto $\mathbb{F}$, dotado de dos operaciones binarias: adición y multiplicación. Además, este conjunto incluye dos elementos distintivos:

- Elemento 0 tal que $a + 0 = a$ para todo $a \in \mathbb{F}$.
- Elemento 1 tal que $a1 = a$.

Dentro de esta estructura, el conjunto $\mathbb{F}$ con la operación de adición forma un grupo Abeliano, es decir, es un grupo conmutativo donde la operación es asociativa, existe un elemento identidad (el 0), y cada elemento tiene un inverso aditivo. Por otro lado, el conjunto $\mathbb{F}$ sin el elemento 0, bajo la operación de multiplicación, también constituye un grupo Abeliano. Esto implica que la multiplicación es conmutativa y cada elemento no cero tiene un inverso multiplicativo (Trappe and Washington, 2006).

2.3.3. Campos Finitos

Los campos finitos, son tipos de campos que contienen un número finito de elementos. Se caracterizan por tener un tamaño $q = p^n$ donde p es un número primo y n es un entero positivo. En estos campos, se utilizan dos operaciones binarias: la adición y la multiplicación, ambas efectuadas bajo la modalidad de módulo p. Esto se denota como $\mathbb{F}_q$.

Por ejemplo, si consideramos s como el conjunto de elementos bajo el módulo 7, es decir, $s = \{\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$, podemos observar que este conjunto forma un grupo aditivo donde el elemento 0 actúa como el neutro

de la adición. Además, el elemento 1 funciona como un generador para un grupo multiplicativo excluyendo el 0, estableciendo así las bases para un campo finito, específicamente el campo $\mathbb{F}_7$. En este campo, las operaciones de suma y multiplicación están definidas de tal manera que cualquier operación entre elementos del conjunto vuelve a resultar en un elemento del mismo conjunto, cumpliendo con las propiedades de cerradura y las otras características de un campo.

2.3.4. Problema de Logaritmo Discreto (PLD)

El problema del logaritmo discreto es una cuestión fundamental en la criptografía que implica trabajar con números en un contexto modular. En este problema, se parte de un número primo p y dos enteros no nulos α y β que cumplen con la relación $\beta \equiv \alpha^x (mod\ p)$, donde x es el exponente desconocido que se desea determinar. Si n es el entero positivo más pequeño tal que $\alpha^n \equiv 1 (mod\ p)$, se asume que x se encuentra en el intervalo $0 \leq x < n$, y se denota por $x = L_\alpha(\beta)$ representando el logaritmo discreto de β respecto a α (omitiendo el primo p de la notación), reflejando así la búsqueda de un exponente que satisface una ecuación exponencial en un sistema finito (Trappe and Washington, 2006).

2.3.5. Curvas Elípticas

Una curva elíptica E sobre un campo $\mathbb{F}$ está definida como el conjunto de puntos (x, y) que satisfacen la ecuación de Weierstrass:

$$E : y^2 = x^3 + ax^2 + b \tag{2.1}$$

junto con un punto especial denominado “punto al infinito”, denotado por $\mathcal{O}$. En esta ecuación, tanto a, b como x son elementos del campo $\mathbb{F}$. Para asegurar que la curva sea no singular, los coeficientes deben satisfacer la condición del discriminante que se expresa como:

$$4a^3 + 27b^2 \neq 0 \tag{2.2}$$

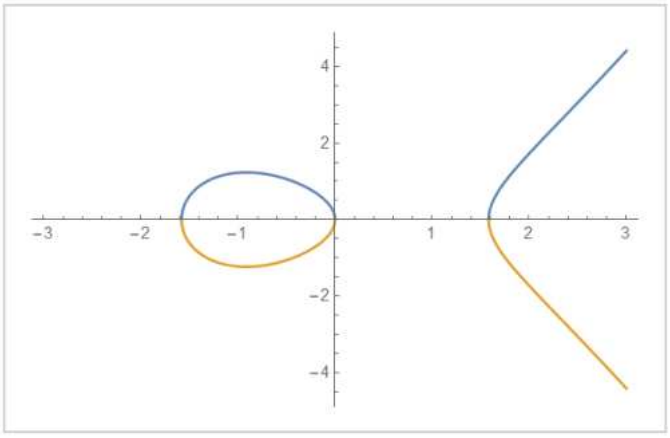
Se consideran dos curvas elípticas características que presentan una estructura regular, descritas por las siguientes ecuaciones:

$$E : y^2 = x^3 - x \tag{2.3}$$

$$E : y^2 = x^3 - x + 1 \tag{2.4}$$

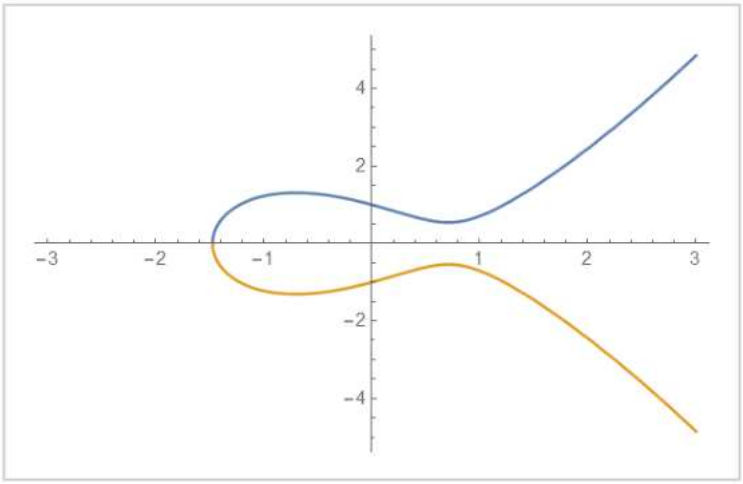
Las Figuras **??** y 2.4 ilustran las curvas elípticas correspondientes a las ecuaciones 2.3 y 2.4 respectivamente.

Figura 2.3:
Curva Elíptica $y^2 = x^3 - x$



Nota: Elaboración propia.

Figura 2.4:
Curva Elíptica $y^2 = x^3 - x + 1$



Nota:Elaboración propia.

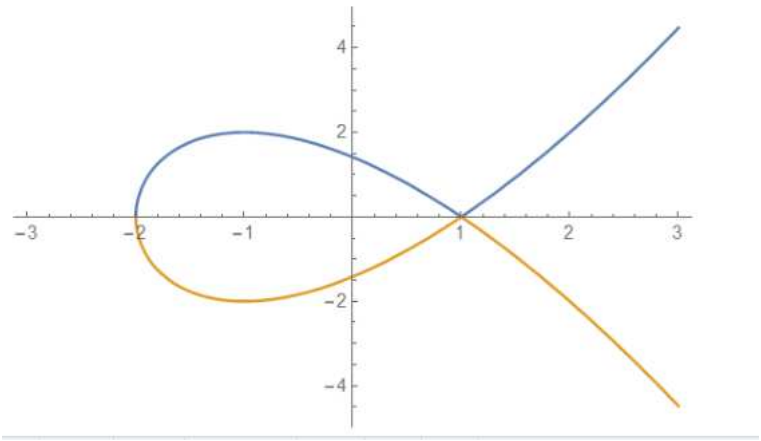
Las curvas elípticas representadas en las Figuras **??** y 2.4 exhiben características geométricas específicas que las definen como curvas elípticas en el sentido clásico:

1. Las curvas no pueden cruzarse sobre sí mismas.
2. Las curvas no presentan picos.

Estas propiedades aseguran que las curvas sean suaves y sin singularidades, lo cual es esencial para las aplicaciones criptográficas de las curvas elípticas.

Por otro lado, la curva mostrada en la Figura 2.5 difiere de las anteriores en que presenta un cruce sobre sí misma, indicativo de una curva elíptica no racional. Esta característica es atípica para las curvas elípticas usadas en criptografía, donde la no-singularidad es una condición necesaria para su funcionamiento adecuado.

Figura 2.5:
Curva Elíptica no racional



Nota:Elaboración propia.

Para asegurar la no-singularidad de una curva elíptica, es fundamental que el determinante $4a^3 + 27b^2 \neq 0$ se cumpla. Esta condición garantiza que la curva sea suave y sin puntos de auto-intersección o picos, permitiendo definir todos los valores racionales dentro de la curva de manera consistente.

En las curvas elípticas, se definen dos operación especial de suma de puntos. Primeramente se abordará la suma de dos puntos P y Q sobre la curva elíptica y posteriormente la suma de k veces el punto P; o también, el producto escalar de k veces el punto P sobre la curva elíptica.

Considere que se tienen dos puntos P y Q en la curva, la operación de suma de los dos puntos P y Q sigue un procedimiento geométrico específico, que puede describirse de la siguiente manera:

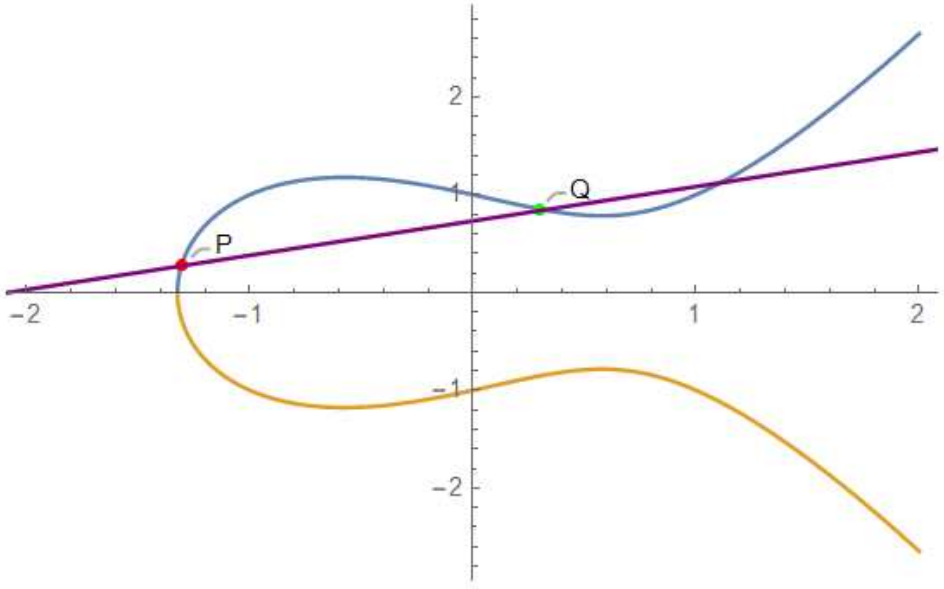
1. Se traza una línea recta que pasa por los puntos P y Q.
2. Esta línea continúa hasta intersectar nuevamente la curva en un tercer punto, que denominaremos S.

3. Desde S, se traza una línea vertical hasta que corte la curva en otro punto.

4. El punto donde esta línea vertical corta la curva por segunda vez se define como R, y este punto R es el resultado de la operación $P + Q$, que es un punto racional en la curva.

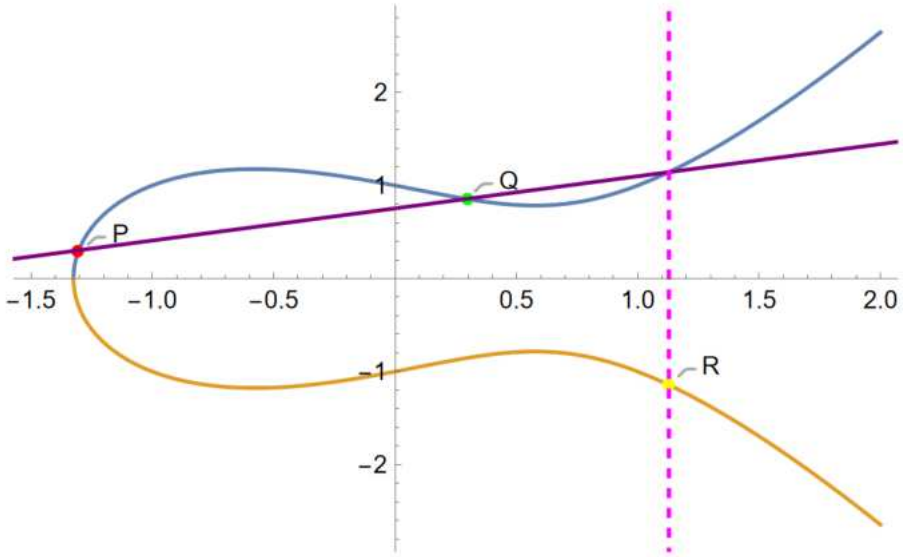
Este método asegura que la suma de dos puntos en la curva resulte en otro punto que también pertenece a la curva, manteniendo la estructura algebraica cerrada y consistente. Las Figuras 2.6 y 2.7 muestra gráficamente este proceso.

Figura 2.6:
Recta que une punto P y Q



Nota: Elaboración propia.

Figura 2.7:
$P + Q = R$



Nota: Elaboración propia.

La suma de k veces el punto P o multiplicación escalar de un punto P en una curva elíptica por un valor k es un concepto central en la criptografía basada en curvas elípticas. Se define como:

$$kP = \underbrace{P + P + \cdots + P}_{k \text{ veces}}$$

Este proceso implica sumar el punto P a sí mismo k veces. A continuación, se describe cómo se realiza esta operación paso a paso:

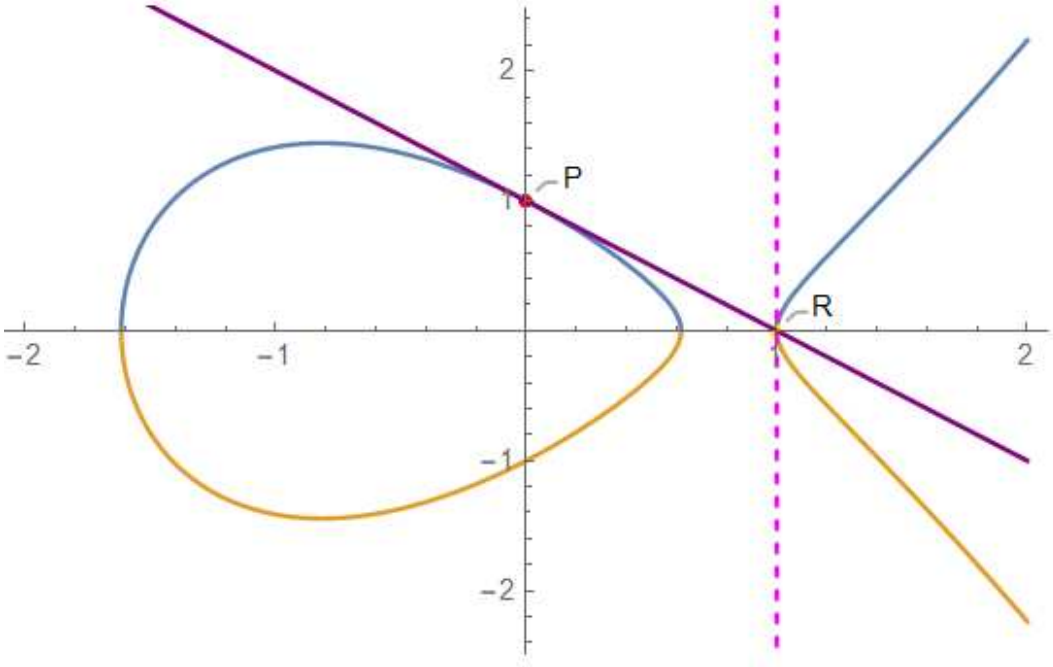
1. Comenzamos con el punto P.
2. Sumamos P a P para obtener $2P$. Este proceso involucra trazar una línea tangente a la curva en P, encontrar el otro punto de intersección con la curva, y luego reflejar ese punto a través del eje x para encontrar $2P$.

3. Para calcular $3P$, sumamos $2P$ a P utilizando el método descrito anteriormente para la suma de dos puntos diferentes.

4. Repetimos este proceso hasta que hayamos sumado P a sí mismo un total de k veces.

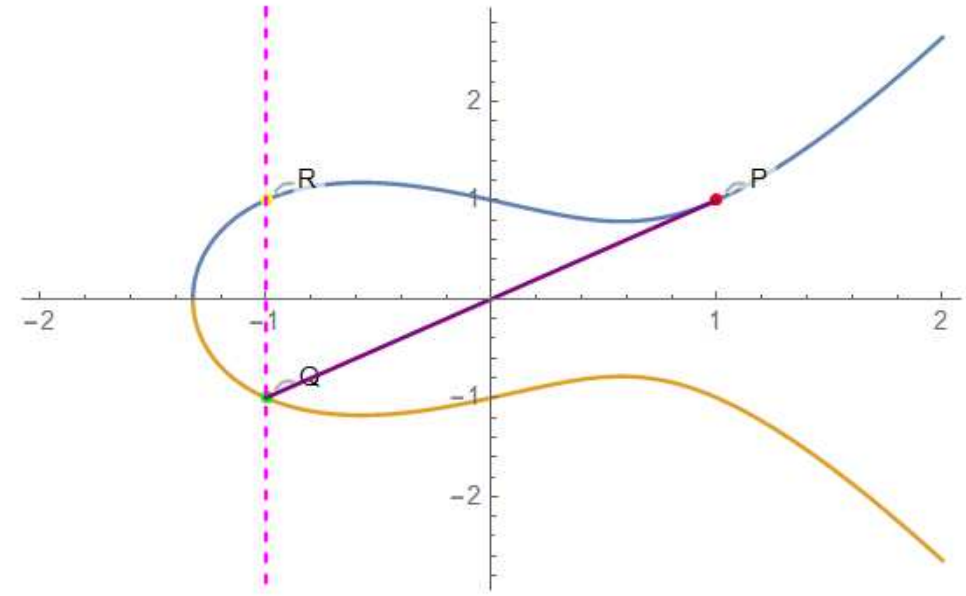
Este método se ilustra en la siguientes figuras 2.8, 2.9 y 2.10, donde mostramos cómo se suman puntos sucesivamente sobre la curva.

Figura 2.8:
Suma del mismo punto P



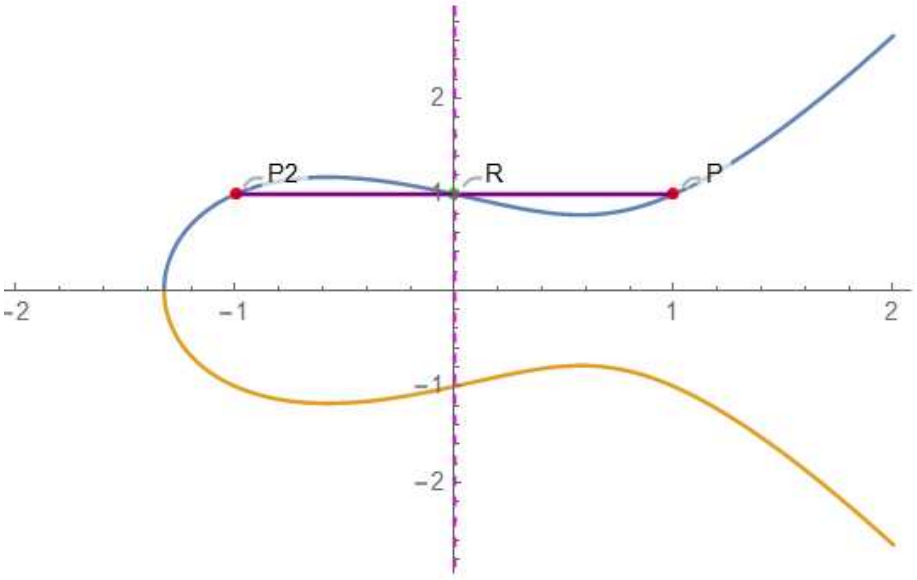
Nota: Elaboración propia.

Figura 2.9:
Suma de puntos en curva elíptica $k = 2$



Nota: Elaboración propia.

Figura 2.10:
Suma de puntos en curva elíptica $k = 2$



Nota: Elaboración propia.

Este proceso es fundamental para la implementación de muchos algoritmos de criptografía, como los de firma digital y los protocolos de intercambio de claves.

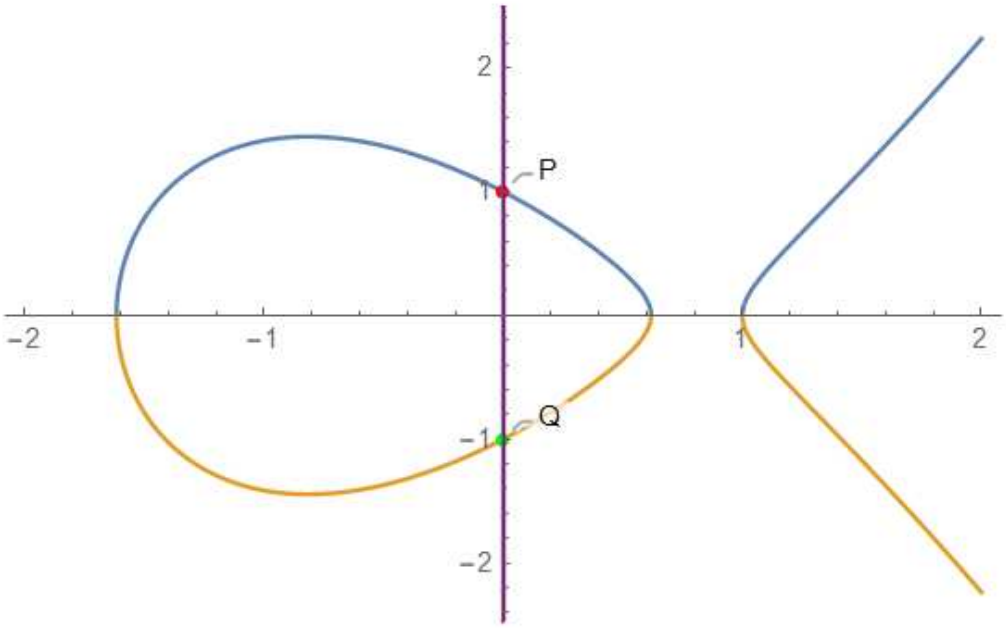
Cuando dos puntos P y Q sobre una curva elíptica están alineados verticalmente, la suma de estos puntos presenta un caso especial en la aritmética de curvas elípticas. Este alineamiento vertical significa que la línea que pasa por P y Q es vertical y no cortará la curva en ningún otro punto. Matemáticamente, este escenario se maneja de la siguiente manera:

1. Si P y Q son simétricos a través del eje x, es decir, si $P = (x, y)$ y $Q = (x, -y)$, entonces la línea vertical entre ellos no intersecta la curva en otro punto racional.

2. En este caso, la suma $P + Q$ se define como el punto al infinito $\mathcal{O}$, que actúa como el elemento neutro en la adición sobre la curva elíptica Como lo podemos observar en la Figura 2.11.

El punto al infinito $\mathcal{O}$ es un concepto abstracto que facilita la operación de suma en la curva, asegurando que el conjunto de puntos en la curva junto con $\mathcal{O}$ forme un grupo abeliano bajo la adición. La introducción de $\mathcal{O}$ permite que la suma de cualquier par de puntos siempre resulte en otro punto definido en la curva, manteniendo la estructura cerrada del grupo.

Figura 2.11:
Representación de la suma de dos puntos alineados verticalmente resultando en el punto al infinito $\mathcal{O}$.



Nota: Elaboración propia.

Este caso es crucial para entender completamente la aritmética sobre curvas elípticas y su estructura algebraica.

Sumar cualquier punto P en una curva elíptica con el punto al infinito $\mathcal{O}$ no cambia el punto P, mostrando que $\mathcal{O}$ actúa como el elemento neutro del grupo. Aunque algunas curvas elípticas contienen un número finito de soluciones racionales, otras pueden tener un número infinito de puntos racionales. La estructura de estos puntos en la curva se describe por la ecuación:

$$E : y^2 = x^3 + ax + b \quad a, b \in \mathbb{Q}$$

Esta estructura de puntos racionales forma un grupo abeliano bajo la operación de suma definida, proporcionando una rica conexión entre la geometría algebrai-

ca y la teoría de grupos.

2.4. Generación de Llaves

2.4.1. Generación de Llaves con RSA

El algoritmo RSA se basa en una idea matemática muy antigua: cada número mayor que uno es el resultado de multiplicar números primos (números que solo se pueden dividir por uno y por sí mismos) de una manera única. Imagina que estás intentando deshacer un producto de dos números grandes para encontrar exactamente cuáles fueron esos números originales. Aunque multiplicar dos números es fácil, encontrar esos números originales solo a partir de su producto es extremadamente difícil sin pistas. Esta dificultad es el pilar sobre el que se construye la seguridad del RSA (Lizy and Raj, 2021).

Proceso de Selección de Llaves

En el proceso de creación de un par de llaves en RSA, se utilizan cuidadosamente dos llaves distintas: una pública y una privada, que son fundamentales para la seguridad y eficiencia del sistema criptográfico. Este método se conoce como criptografía asimétrica debido a que las llaves, aunque matemáticamente relacionadas, no son intercambiables en sus funciones. Por ejemplo, lo que una llave pública cifra, solo la llave privada correspondiente puede descifrarlo, garantizando así la confidencialidad (Fagin, 2018). Este diseño asegura que únicamente el destinatario intencionado, el poseedor de la llave privada, pueda leer el mensaje cifrado.

Además, la llave privada permite firmar digitalmente documentos o mensajes, facilitando que cualquier persona con la llave pública pueda verificar estas firmas. Esta funcionalidad no solo autentica la fuente del mensaje sino que también asegura que el contenido no ha sido alterado, evitando que el firmante niegue su autoría o envío.

Las ventajas de usar llaves asimétricas incluyen una mayor seguridad, ya que la llave pública puede distribuirse libremente sin comprometer la seguridad del sistema, simplificación en la gestión de llaves porque las privadas nunca necesitan ser compartidas, y una mayor interoperabilidad, lo que permite interacciones seguras en redes abiertas como Internet sin necesidad de intercambiar llaves de forma segura previamente. Estas características hacen que RSA sea un sistema robusto y adaptable a diversas necesidades de seguridad digital(Imam et al., 2021).

El meticuloso proceso de selección de llaves, que consta de varios pasos esenciales, garantiza que el sistema sea tanto seguro como eficiente. A continuación, se detallan los pasos involucrados:

1. Elección de Números Primos: Se selecciona dos números primos grandes y diferentes entre sí, llamémoslos p y q. Usamos métodos especiales para asegurarnos de que estos números son realmente primos, lo cual es crucial porque si no lo fueran, la seguridad del sistema se podría comprometer.

2. Creación del Módulo n: Multiplicamos p y q para obtener un nuevo número n, este se usa como parte de la llave pública y privada. El valor de n se interpreta como una caja fuerte que solo se puede abrir con el conocimiento correcto de p y q.

3. Cálculo de la Función de Euler $\phi(n)$: Calculamos $\phi(n) = (p-1)(q-1)$. Este

paso es como preparar un sistema para hacer más difícil que alguien adivine la combinación correcta que abre la caja fuerte.

4. Selección del Exponente Público e: Elegimos un número e, que trabaja bien matemáticamente con $\phi(n)$ (deben ser coprimos, lo que significa que no comparten ningún factor además de 1). Este e es parte de la llave pública y se puede compartir de manera segura con otros.

5. Determinación del Exponente Privado d: Finalmente, encontramos un número d que funciona de manera especial con e y $\phi(n)$ para que d y e puedan desbloquear nuestra caja fuerte. d es la llave privada y debe mantenerse en secreto.

Es importante destacar que el tamaño de la llave desempeña un papel fundamental en la seguridad del sistema. A medida que el tamaño de la llave aumenta, también lo hace la seguridad ofrecida, ya que se vuelve más difícil para los atacantes descifrar la información cifrada. Sin embargo, un aumento en el tamaño de la llave implica que las operaciones criptográficas consuman más tiempo, dado que se manejan números más grandes. Actualmente, se recomienda utilizar llaves de al menos 2048 bits de longitud para proporcionar una defensa robusta contra los tipos de ataques criptográficos más modernos y sofisticados (Imam et al., 2021).

2.4.2. Generación de Llaves con ECC

La criptografía de Curvas Elípticas (ECC) utiliza las propiedades de las curvas elípticas en campos finitos para generar llaves de cifrado. Este método ofrece una seguridad comparable a otros sistemas criptográficos tradicionales

como RSA, pero con llaves más pequeñas, lo que mejora la eficiencia y reduce el uso de recursos computacionales (Ismail and Misro, 2022).

En ECC, comenzamos seleccionando una curva elíptica que cumpla con la ecuación $y^2 = x^3 + ax + b$ Los coeficientes a y b definen la forma de la curva, y se eligen de manera que se cumpla el discriminante $4a^3 + 27b^2 \neq 0$, asegurando así que la curva no tenga puntos singulares, es decir, puntos donde la curva se interseca a sí misma o forma un bucle (Trappe and Washington, 2006).

El siguiente paso en la generación de llaves ECC es elegir un campo finito. Este campo puede ser un conjunto de números primos de gran tamaño, conocido como campo primo, o puede basarse en números binarios, conocido como campo binario. La elección del campo afecta directamente tanto la seguridad del sistema como su rendimiento.

Una vez que la curva y el campo están definidos, seleccionamos un punto base G en la curva. Este punto base debe tener un orden elevado. El orden de un punto es el número de veces que se debe sumar a sí mismo antes de volver al punto de inicio, y un orden alto hace que sea difícil resolver ciertos problemas matemáticos relacionados con el cifrado(Trappe and Washington, 2006).

La llave privada en ECC es simplemente un número entero seleccionado al azar, llamado d, que es menor que el orden del punto base G. La llave pública, por otro lado, se calcula multiplicando el punto base G por la llave privada d a través de un proceso llamado multiplicación escalar de puntos. La llave pública se expresa como $Q = dG$, donde Q representa un punto en la curva (Alkhudhayr et al., 2021).

Este proceso de multiplicación escalar para calcular la llave pública a partir de la llave privada es relativamente eficiente. Sin embargo, el intento de deducir la

llave privada a partir de la llave pública, conocido como el problema del logaritmo discreto en curvas elípticas, es computacionalmente inviable con la tecnología actual, lo que otorga al sistema una robusta seguridad.

2.5. Cifrado y Descifrado con RSA

2.5.1. Cifrado con RSA

El cifrado con RSA es un método de criptografía asimétrica, esta técnica permite que los mensajes sean cifrados por cualquier persona que tenga acceso a la llave pública del receptor, pero solo pueden ser descifrados por el poseedor de la llave privada correspondiente, asegurando la confidencialidad de la información transmitida (Mohamad et al., 2021).

En el cifrado RSA, el mensaje original, que debe ser transformado a un número menor que n (el producto de los dos números primos utilizados para generar las llaves), se eleva a la potencia del exponente público e y luego se toma el residuo de esa operación respecto a n. Matemáticamente, si M es el mensaje original, el mensaje cifrado C se calcula como: $C = M^e \ mod(n)$ (Fagin, 2018).

Esta operación transforma el mensaje en una forma que solo puede ser revertida por alguien que conozca la llave privada adecuada.

2.5.2. Descifrado con RSA

El descifrado con RSA es el proceso que permite a un destinatario autorizado recuperar el mensaje original a partir de su forma cifrada, la cual ha sido encriptada utilizando la llave pública RSA. Este proceso se basa en el uso exclusivo de la llave privada del destinatario.

Cuando se recibe un mensaje cifrado C, este ha sido transformado previamente elevando el mensaje original M a la potencia del exponente público e y tomando el residuo respecto al módulo n, que es el producto de dos números primos seleccionados durante la generación de la llave. Para descifrar C y recuperar M, el destinatario debe aplicar su llave privada d al mensaje cifrado. Esto se realiza elevando C a la potencia de d y luego tomando el residuo de esta operación respecto a n, es decir, $M = C^d \, mod(n)$ (Fagin, 2018).

Esta operación matemática revierte el cifrado inicial, permitiendo que el mensaje vuelva a su estado original y legible. La llave privada d se ha generado de tal manera que es matemáticamente viable realizar esta operación, pero computacionalmente impracticable invertir el proceso sin conocer d. Esto se debe a la complejidad de deducir d a partir de la llave pública, que involucra la difícil tarea de factorizar el número compuesto n (Gidney and Ekerå, 2021).

El descifrado con RSA, por lo tanto, asegura que solo el poseedor de la llave privada adecuada pueda acceder al contenido del mensaje cifrado, proporcionando una capa robusta de seguridad y privacidad.

2.6. Cifrado y Descifrado con ECC

2.6.1. Cifrado con ECC

El cifrado con criptografía de Curvas Elípticas (ECC) emplea la estructura algebraica de las curvas elípticas para asegurar la comunicación digital de manera eficiente y segura. A continuación se explica el proceso y los métodos matemáticos involucrados en el cifrado de mensajes utilizando ECC, una vez que las llaves pública y privada ya han sido generadas.

Antes de que el cifrado pueda comenzar, el mensaje inicial debe ser transformado en un punto sobre la curva elíptica. Esta transformación debe ser reversible para permitir la recuperación del mensaje original una vez que sea descifrado.

El proceso para transformar el mensaje M en el punto P sobre la curva elíptica es el siguiente:

1. Convertir el mensaje M a su representación en ASCII.
2. Mapear esta representación numérica a un punto en la curva elíptica. Este mapeo se realiza asegurando que el punto resultante pertenezca a la curva elíptica definida.

Considerese que el emisor selecciona un número aleatorio k, que se utiliza para garantizar que cada instancia de cifrado sea única, incluso si el mensaje original es el mismo en múltiples ocasiones. Este valor k se utiliza para escalar el punto base G y la llave pública del receptor durante el proceso de cifrado (Trappe and Washington, 2006).

Utilizando k y la llave pública del destinatario Q, que también es un punto en la curva, el emisor calcula dos nuevos puntos. Primero, $C_1 = kG$, este punto actúa como parte del vector de cifrado y depende de k y del punto base G de la curva. Segundo, $C_2 = P + kQ$, aquí P es el punto que representa el mensaje, y kQ es la llave pública del destinatario escalada por k (Trappe and Washington, 2006). Este segundo punto cifra efectivamente el mensaje mediante la adición de un punto derivado de la llave pública.

El emisor envía el par de puntos (C_1, C_2) al destinatario. Estos dos puntos contienen toda la información necesaria para que el destinatario pueda recuperar el mensaje original, pero no son suficientes para que un observador sin la llave privada correcta los descifre.

2.6.2. Descifrado con ECC

El descifrado utilizando criptografía de Curvas Elípticas (ECC) es un proceso para recuperar un mensaje original cifrado, asegurando que solo el destinatario intencionado pueda leerlo. Este método utiliza operaciones matemáticas sobre los puntos de una curva elíptica, pero se puede explicar de manera que sea comprensible incluso para quienes no están familiarizados con el tema.

El destinatario comienza el proceso de descifrado utilizando su llave privada d. El primer paso es calcular el producto de d y el punto C_1 recibido $dC_1 = d(kG)$. Aquí, k es el número aleatorio usado por el emisor para cifrar el mensaje y G es el punto base. Como kG es exactamente lo que el emisor usó para crear C_1, este cálculo da como resultado el punto kQ (porque $Q = dG$). El siguiente paso es usar el resultado anterior para obtener el punto original P que representa el mensaje codificado $P = C_2 - dC_1$ Aquí, C_2 es el segundo punto del mensaje

cifrado, y el resultado de dC_1 (que es kQ) se resta de C_2. Esta operación cancela exactamente la parte del cifrado que involucró la llave pública, dejando solo el punto P que codifica el mensaje original.(Alkhudhayr et al., 2021)

2.7. Firmas Digitales

El uso de curvas elípticas en la criptografía digital, especialmente para firmas digitales y cifrado de mensajes, optimiza el espacio necesario para la generación de firmas y claves en comparación con otros métodos. Las siguientes tablas 2.1 y 2.2, ilustran esta optimización:

Tabla 2.1:
Comparación del tamaño de firmas en RSA, DSA, ECC

Método	Tamaño de la firma (bits)
RSA	1024
DSA	320
ECC	320

Nota: (Huguet Rotger. L. and Tena Ayuso, 2017)

Tabla 2.2:
Comparación del tamaño de mensajes cifrados en RSA, ElGamal y ECC

Método	Mensaje cifrado (bits)
RSA	1024
ElGamal	2048
ECC	321

Nota: (Huguet Rotger. L. and Tena Ayuso, 2017)

Las tablas 2.1 2.2 muestran cómo las curvas elípticas proporcionan una optimización significativa en términos de espacio y seguridad, convirtiéndolas en una opción preferida para protocolos de seguridad digital modernos.

2.7.1. Firma Digital con RSA

Las firmas digitales con RSA son una técnica fundamental en la criptografía que permite la autenticación de la identidad del remitente y la integridad del mensaje o documento digital. Utilizando el algoritmo RSA, la llave privada del remitente se emplea para generar una firma única que depende directamente del contenido del mensaje. Simultáneamente, la llave pública asociada permite a cualquier receptor verificar esa firma (Maletsky, 2015).

El proceso inicia cuando el mensaje original se transforma mediante una función hash criptográfica, como SHA-256, que convierte el texto de cualquier longitud en un resumen digital de tamaño fijo, denominado hash. Este hash actúa como una huella digital del mensaje, es único para cada mensaje y cualquier cambio mínimo en el mensaje cambiará este hash. Esta función hash permite comprimir el mensaje original a un tamaño manejable y constante que facilita los siguientes pasos del proceso de firma (Rojas and Meneses, 2000).

Después de generar el hash, el siguiente paso es cifrar este hash utilizando la llave privada del remitente, no el mensaje completo. Este cifrado produce lo que se conoce como firma digital. La llave privada garantiza que solo el remitente pueda producir una firma válida para ese mensaje específico, certificando su identidad. Aquí, si denotamos el hash como h, y la llave privada como d, la firma s se calcula como: $s = h^d \, mod(n)$ donde n es el producto de dos números primos grandes seleccionados durante la generación de la llave RSA (Seo, 2020).

El remitente envía tanto el mensaje original como su firma digital al receptor. Al recibirlos, para verificar la firma, el receptor aplica la llave pública del remitente e, a la firma s recibida para descifrarla y extraer el hash del mensaje

que el remitente calculó originalmente. Esto se realiza mediante la operación: $h' = s^e \ mod(n)$ donde h' debería ser igual al hash h del mensaje original si la firma es válida.

El receptor también genera un nuevo hash h'' del mensaje recibido usando la misma función hash que el remitente. Finalmente, compara este nuevo hash h'' con el hash h' descifrado de la firma. Si h' y h'' son idénticos, entonces el mensaje no ha sido modificado desde que fue firmado y proviene auténticamente del titular de la llave privada asociada a la llave pública usada (Schneider and Vaira, 2019).

2.7.2. Firma Digital con ECC

Las firmas digitales utilizando criptografía de Curvas Elípticas (ECC) son un método avanzado y eficiente para garantizar la autenticidad y la integridad de los mensajes o documentos digitales. Este proceso comienza con la creación de un hash del mensaje original usando una función hash criptográfica robusta. Este hash actúa como una representación condensada del contenido del mensaje, asegurando que cualquier alteración en el mensaje sea detectable(Aljebreen, 2022).

Una vez que el hash del mensaje ha sido generado, el firmante procede a seleccionar un número aleatorio k, que no solo debe ser único para cada firma para garantizar la seguridad, sino también temporal y nunca reutilizado para evitar ataques criptográficos específicos como el ataque de repetición. Este número k se utiliza para calcular el punto R en la curva elíptica multiplicándolo por el punto base G. El punto $R = kG$ resultante tiene coordenadas (x, y), pero solo la coordenada x se utiliza en los cálculos siguientes.

La firma se forma entonces a partir de dos componentes, r y s. El valor de

r se calcula como el residuo de la coordenada x del punto R módulo n, es decir, $r = x \ mod(n)$, donde n es el orden del punto base G. El segundo componente de la firma s, se calcula usando la fórmula $s = k^{-1} \ (h + dr) \ mod(n)$. Aquí, h es el hash del mensaje, d es la llave privada del firmante y r es el valor que se acaba de calcular. El término k^{-1} representa el inverso multiplicativo de k módulo n, lo cual es necesario para mantener la estructura algebraica de la curva y garantizar la validez matemática de la operación.

El receptor del mensaje y su firma digital verifica la autenticidad del mensaje recalculando el hash del mensaje y utilizando la llave pública del remitente Q para evaluar la firma. El receptor calcula dos coeficientes, u_1 y u_2, donde $u_1 = hs^{-1} \ mod(n)$ y $u_2 = rs^{-1} \ mod(n)$. Estos valores se usan para encontrar un punto X en la curva elíptica mediante la fórmula $X = u_1G + u_2Q$. Si todo es correcto, la coordenada x de X debe ser igual a $x \ mod(n) = r$, confirmando así que la firma es válida y que el mensaje no ha sido modificado (Wang et al., 2020).

”La matemática es la reina de las ciencias y la teoría de números es la reina de las matemáticas. Ella con frecuencia conduce a los descubrimientos más inesperados y, más que cualquier otra rama, estimula la facultad de invención.”

Carl Friedrich Gauss (1777-1855)
Matemático, físico y astrónomo alemán

CAPÍTULO 3

Algoritmos de Cifrado y Descifrado

3.1. Generación de Llaves con RSA

3.1.1. Algoritmo Generación de un número primo grande

El Algoritmo 1 es necesario en la generación de llaves para el algoritmo de RSA, ya que proporciona los números primos grandes necesarios para crear las llaves pública y privada. Este algoritmo genera un número primo aleatorio dentro del rango definido por $2^{tamanoBits-1}$ y $2^{tamanoBits} - 1$. Este rango asegura que el número primo tenga exactamente el tamaño especificado en bits, lo cual es fundamental para la seguridad del algoritmo RSA. Los números primos generados por esta función son utilizados posteriormente para calcular los módulos $n = p \cdot q$, donde p y q son números primos.

Algoritmo 1 Generación de un número primo grande

```
1: procedure GENERARPRIMOGRANDE(tamanoBits)
2:     L ← 2^(tamanoBits−1)          ▷ Define el límite inferior del rango
3:     U ← 2^(tamanoBits) − 1        ▷ Define el límite superior del rango
4:     primo ← RandomPrime(L, U)     ▷ Generar número primo aleatorio
5:     return primo
6: end procedure
```

3.1.2. Algoritmo Cálculo del inverso modular de a módulo m

El Algoritmo 2 determina el inverso modular de un entero a respecto a un módulo m. El inverso modular es el número x tal que $ax \equiv 1 \mod m$. Este procedimiento utiliza el método de extensión del algoritmo de Euclides para encontrar el inverso modular. Si a y m no son coprimos, el inverso modular no existe.

Algoritmo 2 Cálculo del inverso modular de a módulo m

```
1: procedure INVERSOMODULAR(a, m)
2:     x_0 ← 1
3:     x_1 ← 0
4:     m_0 ← m
5:     while a > 1 do
6:         q ← a|m                          ▷ Divide a por m y toma el cociente entero
7:         (a, m) ← (m, a mód m)            ▷ Actualiza a y m
8:         (x_0, x_1) ← (x_1, x_0 − q × x_1)   ▷ Actualiza x_0 y x_1
9:     end while
10:    if x_0 < 0 then
11:        x_0 ← x_0 + m_0                  ▷ Si x_0 es negativo, ajusta agregando m_0
12:    end if
13:    return x_0
14: end procedure
```

3.1.3. Algoritmo Generación de llaves para RSA

El Algoritmo 3 implementa la generación de llaves RSA, el cual se basa en la dificultad de factorizar el producto de dos números primos grandes. Este algoritmo asegura que las llaves generadas sean seguras al seleccionar dos primos distintos y calcular un exponente z que sea coprimo con $\varphi(n)$, permitiendo así determinar un inverso modular d para z bajo $\varphi(n)$, que es crucial para la operación de descifrado en RSA.

Algoritmo 3 Generación de llaves para RSA

procedure GENERARLLAVESRSA($tamanoPrimo$)
    $p \leftarrow$ generarPrimoGrande($tamanoPrimo$)
    $q \leftarrow$ generarPrimoGrande($tamanoPrimo$)
    while $p = q$ **do**
        $q \leftarrow$ generarPrimoGrande($tamanoPrimo$) ▷ Asegura que $p \neq q$
    end while
    $n \leftarrow p \times q$
    $\varphi(n) \leftarrow (p-1) \times (q-1)$ ▷ Calcula la función de Euler para n
    $z \leftarrow$ RandomInteger($\{2^{tamanoPrimo-2}, \varphi(n)\}$)
    while GCD$(z, \varphi(n)) \neq 1$ **do**
        $z \leftarrow$ RandomInteger($\{2^{tamanoPrimo-2}, \varphi(n)\}$)
    end while
    $d \leftarrow$ inversoModular$(z, \varphi(n))$
    llavePublica $\leftarrow \{n, z\}$
    llavePrivada $\leftarrow \{n, d\}$
    return {llavePublica, llavePrivada}
end procedure

3.2. Generación de Llaves con ECC

3.2.1. Algoritmo Validación de parámetros de Curvas Elípticas

El Algoritmo 4 determina si un conjunto de parámetros a y b puede definir una curva elíptica racional. El criterio se basa en el cálculo del discriminante de la ecuación de la curva elíptica $y^2 = x^3 + ax + b$. El algoritmo evalúa el discriminante $4a^3 + 27b^2$ si este es diferente de cero, la curva es no singular y se considera una curva elíptica racional. La salida incluye una visualización gráfica de la curva si los parámetros son válidos.

Algoritmo 4 Validación de parámetros de Curvas Elípticas

Require: Coeficientes a, b
Ensure: Devuelve *True* si los parámetros definen una curva elíptica racional, *False* de lo contrario.

```
1: s ← 4a^3 + 27b^2
2: if s ≠ 0 then
3:     Mostrar gráfico de la curva
4:     return True                                  ▷ La curva es racional
5: else
6:     return False                  ▷ No pasa la validación del discriminante
7: end if
```

3.2.2. Algoritmo Suma de dos puntos sobre la Curva Elíptica

El Algoritmo 5 describe cómo sumar dos puntos P y Q sobre una curva elíptica definida por la ecuación $y^2 = x^3 + ax + b$. El resultado es un nuevo punto R que también pertenece a la curva. El procedimiento comienza calculando la pendiente de la línea que une los puntos P y Q. A continuación, se utilizan las fórmulas de suma de puntos en curvas elípticas para hallar las coordenadas del nuevo punto R. Finalmente, el punto R se visualiza gráficamente junto con P y Q para demostrar su relación en la curva.

Algoritmo 5 Suma de dos puntos sobre la Curva Elíptica

Require: Coeficientes a, b y puntos $P = (x_1, y_1)$, $Q = (x_2, y_2)$
Ensure: El punto $R = (x_3, y_3)$ y su representación gráfica
1: Calcular la pendiente $m \leftarrow \frac{y_2 - y_1}{x_2 - x_1}$
2: Determinar $x_3 \leftarrow m^2 - x_1 - x_2$
3: Calcular $y_3 \leftarrow m(x_1 - x_3) - y_1$
4: $R \leftarrow (x_3, y_3)$
5: Mostrar gráfico con los puntos P, Q, y R

3.2.3. Algoritmo Suma de k veces el punto P sobre una curva elíptica

El Algoritmo 6 determina cómo sumar k veces un punto P sobre una curva elíptica definida por la ecuación $y^2 = x^3 + ax + b$, donde $k \geq 2$. Este algoritmo inicia calculando la tangente en el punto inicial P y luego suma P consigo mismo para obtener $2P$. Continúa sumando P hasta alcanzar k veces, actualizando el punto resultante en cada iteración. El resultado final (x_c, y_c) representa kP sobre la curva.

Algoritmo 6 Suma de k veces el punto P sobre una curva elíptica

Require: Coeficientes a, b y punto $P = (x_1, y_1)$, $k \geq 2$
Ensure: El punto $R = (x_k, y_k)$ resultante y su representación gráfica

$(x_c, y_c) \leftarrow (x_1, y_1)$
$m \leftarrow \frac{3x_c^2 + a}{2y_c}$ ▷ Calcula la pendiente en P
$x_3 \leftarrow m^2 - 2x_c$
$y_3 \leftarrow m(x_c - x_3) - y_c$
$(x_c, y_c) \leftarrow (x_3, y_3)$ ▷ Actualiza P para $k = 2$
$i \leftarrow 3$
while $i \leq k$ **do**
    $m \leftarrow \frac{y_c - y_1}{x_c - x_1}$ ▷ Pendiente entre (x_1, y_1) y (x_c, y_c)
    $x_3 \leftarrow m^2 - x_1 - x_c$
    $y_3 \leftarrow m(x_1 - x_3) - y_1$
    $(x_c, y_c) \leftarrow (x_3, y_3)$
    $i \leftarrow i + 1$
end while
output (x_c, y_c)
Mostrar gráfico de la suma de k veces P

3.2.4. Algoritmo Raíz Cuadrada Módulo p

El Algoritmo 7 explica el procedimiento para calcular una raíz cuadrada de un número y bajo el módulo p, es decir, encontrar un x tal que $x^2 \equiv y \ mod \ p$. Este determina la raíz cuadrada de y módulo p usando la operación de extracción de raíz cuadrada módulo p. Si una raíz válida es encontrada, la lista de raíces puede incluir hasta dos valores, puesto que tanto $raiz$ como $p - raiz$ pueden ser soluciones válidas. Si no se encuentra ninguna raíz válida, el algoritmo devuelve nulo, indicando que no existe tal raíz bajo el módulo especificado.

Algoritmo 7 Raíz Cuadrada Módulo p

```
1: procedure RAIZCUADRADAMOD(y, p)
2:     Iniciar una lista vacía de raíces
3:     raiz ← (y^{1/2} mod p)
4:     if raiz ≠ Null and raiz^2 ≡ y mod (p) then
5:         Agregar raiz a la lista de raíces
6:         raiz ← p − raiz
7:         if raiz no está en la lista de raíces then
8:             Agregar raiz a la lista de raíces
9:         end if
10:    end if
11:    if la lista de raíces no está vacía then
12:        return la primera raíz en la lista
13:    else
14:        return Null
15:    end if
16: end procedure
```

3.2.5. Algoritmo Codificación de caracteres a coordenadas de puntos en una curva elíptica

El Algoritmo 8 describe cómo codificar caracteres $ASCII$ en puntos sobre una curva elíptica, utilizando un proceso iterativo para encontrar coordenadas válidas que satisfagan la ecuación de la curva.

El Algoritmo 8 toma como entrada un mensaje y los parámetros de una curva elíptica (a, b, p) y un multiplicador h.

Itera sobre cada carácter del mensaje, calcula un valor de x basado en el código ASCII del carácter y ajusta con un offset j hasta que encuentra un punto válido en la curva, j se inicializa en 1 y se incrementa en cada iteración hasta encontrar una coordenada válida.

Para asegurar una buena distribución de los puntos sobre la curva elíptica, el valor del multiplicador h debe ser elegido de manera que no cause colisiones (es decir, diferentes caracteres $ASCII$ resultando en el mismo punto).

Un valor adecuado para h suele ser un número grande y primo relativo con respecto al módulo p. Esto ayuda a evitar que diferentes caracteres se mapeen al mismo punto en la curva.

Algoritmo 8 Codificación de caracteres a coordenadas de puntos en una curva elíptica

```
1: procedure CODIFICARCARACTER(mensaje, a, b, p, h)
2:     i ← 1
3:     cc ← ∅                ▷ Inicializa un conjunto vacío para almacenar coordenadas
4:     while i ≤ longitud(mensaje) do
5:         j ← 1
6:         ascii ← ord(mensaje[i])          ▷ Obtiene el código ASCII del carácter
7:         repeat
8:             x ← ascii × h + j
9:             y^2 ← (x^3 + a × x + b) mod p
10:            y ← RaizCuadradaMod(y^2, p) ▷ Calcular la raíz cuadrada módulo p
11:            if y ≠ Null then
12:                cc ← cc ∪ {(x, y)}              ▷ Agrega el punto si y es válida
13:            end if
14:            j ← j + 1
15:        until y ≠ Null
16:        i ← i + 1
17:    end while
18:    return cc
19: end procedure
```

3.2.6. Algoritmo Duplicación de un punto en una curva elíptica

El siguiente Algoritmo 9 calcula el doble de un punto en una curva elíptica, un procedimiento clave en la criptografía de curvas elípticas. El algoritmo toma como entradas el módulo primo p, el coeficiente a de la ecuación de la curva elíptica, y las coordenadas (x, y) del punto P que se va a duplicar. El resultado es el punto $R = (x_3, y_3)$, que es el doble del punto P.

Algoritmo 9 Duplicación de un punto en una curva elíptica

1: **procedure** DUPLICARPUNTO($p, a,$ punto P)
2: $m \leftarrow \frac{3x^2+a}{2y} \ mod \ p$ ▷ Calcula la pendiente de la tangente en P
3: $x_3 \leftarrow (m^2 - 2x) \ mod \ p$ ▷ Calcula la abscisa x del punto resultante
4: $y_3 \leftarrow (m(x - x_3) - y) \ mod \ p$ ▷ Calcula la ordenada y del punto resultante
5: **return** (x_3, y_3) ▷ Devuelve las nuevas coordenadas del punto duplicado
6: **end procedure**

3.2.7. Algoritmo Generación de llave pública con Curvas Elípticas

El Algoritmo 10 describe cómo se genera una llave pública a partir de una llave privada en el contexto de criptografía de curvas elípticas. El proceso consiste en la multiplicación de un punto base G de la curva elíptica por un escalar k, que es la llave privada. El Algoritmo 10 emplea un método eficiente conocido como duplicación del Algoritmo 9 y suma del Algoritmo 5 basado en la representación binaria de k para calcular la multiplicación escalar.

Algoritmo 10 Generación de llave pública

1: **procedure** GENERACIONLLAVEPUBLICA($p, a, G, k,$ LlavePrivada)
2: llavePublica $\leftarrow G$
3: bits $\leftarrow$ binario(k) ▷ Convierte k en su representación binaria
4: **for** i from 1 to longitud$(bits)$ **do**
5: llavePublica $\leftarrow$ DuplicarPunto$(p, a,$ llavePublica$)$ ▷ Duplicar el punto
6: **if** bits$[i] = 1$ **then**
7: llavePublica $\leftarrow$ SumaPuntos$(p, G,$ llavePublica$)$ ▷ Sumar G y $llavePublica$
8: **end if**
9: **end for**
10: **return** llavePublica ▷ Devuelve la llave pública calculada
11: **end procedure**

3.3. Cifrado con RSA

3.3.1. Algoritmo Conversión de texto a número ASCII y de número ASCII a texto

El Algoritmo 11 convierte una cadena de texto en un número grande mediante la concatenación de sus códigos ASCII. Esta realiza la operación inversa, convirtiendo un número grande de nuevo en texto. La `textoANumerosASCII` toma un texto y convierte cada carácter en su correspondiente código ASCII, luego concatena estos códigos en un único número utilizando una base de 256 para preservar cada código como un dígito único en esa base. La `numerosASCIITexto` realiza la operación inversa, donde un número que representa una serie de códigos ASCII se descompone de nuevo en caracteres individuales, reconstruyendo el texto original.

Algoritmo 11 Conversión de texto a número ASCII y de número ASCII a texto

1: **function** TEXTOANUMEROSASCII($texto$)
2: $codigoASCII \leftarrow$ convertir cada carácter de $texto$ a su código ASCII
3: $numero \leftarrow$ concatenar $codigoASCII$ utilizando la base 256
4: **return** $numero$
5: **end function**
6: **function** NUMEROSASCIITEXTO($numero$)
7: $digitos \leftarrow$ convertir $numero$ a dígitos en base 256
8: $texto \leftarrow$ convertir cada código de $digitos$ por su carácter ASCII
9: **return** $texto$
10: **end function**

3.3.2. Algoritmo Exponenciación modular

El Algoritmo 12 calcula $b^e \ mod \ m$, donde b es la base, e es el exponente, y m es el módulo. Este algoritmo utiliza la técnica de exponenciación por cuadrados para calcular el resultado de manera eficiente, reduciendo significativamente el número de multiplicaciones requeridas al manejar exponentes grandes.

Algoritmo 12 Exponenciación modular

```
1: procedure EXPONENCIACIONMODULAR(base, exponente, modulo)
2:     resultado ← 1
3:     b ← base
4:     e ← exponente
5:     while e > 0 do
6:         if Mod(e, 2) = 1 then
7:             resultado ← Mod(resultado × b, modulo)
8:         end if
9:         e ← Quotient(e, 2)                    ▷ Divide el exponente por 2
10:        b ← Mod(b^2, modulo)                  ▷ Eleva b al cuadrado y toma módulo m
11:    end while
12:    return resultado
13: end procedure
```

3.3.3. Algoritmo Cifrado RSA de un número utilizando la llave pública

El Algoritmo 13 utiliza la exponenciación modular para cifrar un número, que es representativo de un mensaje de texto, utilizando la llave pública de RSA. Calcula $numero^e \ mod \ n$. Este paso eleva el número que representa el mensaje original a la potencia del exponente público e y luego toma el resultado módulo n, que es el producto de dos números primos grandes seleccionados durante la generación de llaves RSA. El resultado es el mensaje cifrado, que solo puede ser descifrado correctamente por el poseedor de la llave privada correspondiente.

Algoritmo 13 Cifrado RSA de un número utilizando la llave pública

```
1: function CIFRARRSA(numero, {n, e})
2:     mensajeCifrado ← exponenciacionModular(numero, e, n) ▷ Cifra número
3:     return mensajeCifrado
4: end function
```

3.4. Cifrado con ECC

3.4.1. Algoritmo Cifrado de un punto en la curva elíptica

El Algoritmo 14 describe el proceso de cifrado de un punto en una curva elíptica utilizando técnicas de cifrado basadas en la multiplicación escalar de puntos. Este toma como entrada un punto en la curva elíptica y utilizando una combinación de la multiplicación escalar y la suma de puntos, cifra este punto bajo la llave pública derivada de una llave privada dada. La salida es un conjunto de elementos que incluyen el punto cifrado y los elementos necesarios para su posterior descifrado. Este algoritmo realiza el cifrado de manera no iterativa.

Algoritmo 14 Cifrado de un punto en la curva elíptica

```
1: procedure CIFRADO(punto, p, G, k, LlavePrivada)
2:     msgC ← ∅                          ▷ Inicializa un conjunto vacío para el mensaje cifrado
3:     k ← Random(1, p − 1)              ▷ Genera un número aleatorio
4:     kG ← GeneracionLlavePublica(p, a, G, k, LlavePrivada)   ▷ Genera kG
5:     cifrado ← SumaPuntos(p, punto, kG)   ▷ Suma punto al punto kG
6:     msgC ← (kG, cifrado, k)
7:     return msgC                       ▷ Devuelve el mensaje cifrado
8: end procedure
```

La salida del algoritmo es un conjunto que contiene tres elementos: k, kG, y el punto cifrado. Estos elementos son los necesarios para el descifrado posterior y aseguran que cada instancia del cifrado es única.

3.5. Descifrado con RSA

3.5.1. Algoritmo Descifrado RSA de un número utilizando la llave privada

El Algoritmo 15 utiliza la exponenciación modular para descifrar un número, que ha sido cifrado previamente utilizando la llave pública correspondiente de RSA. Este método es fundamental para recuperar el mensaje original de forma segura. Este calcula $numeroCifrado^d \ mod \ n$. Este paso eleva el número cifrado a la potencia del exponente privado d y toma el resultado módulo n, que es el producto de dos números primos seleccionados durante la generación de llaves RSA. Una vez obtenido el número descifrado, se convierte de nuevo a texto utilizando la función `numerosASCIITexto`, reconstruyendo así el mensaje original.

Algoritmo 15 Descifrado RSA de un número utilizando la llave privada

1: **function** DESCIFRARRSA($numeroCifrado, \{n, d\}$)
2: $numeroDescifrado \leftarrow$ exponenciacionModular($numeroCifrado, d, n$) ▷ Descifra el número
3: **return** $numeroDescifrado$
4: **end function**
5: **function** NUMEROSASCIITEXTO($numeroDescifrado$)
6: $mensajeDescifrado \leftarrow$ convertir cada dígito de $numeroDescifrado$ a su carácter ASCII
7: **return** $mensajeDescifrado$
8: **end function**

3.6. Descifrado con ECC

3.6.1. Algoritmo Descifrado de un punto en la curva elíptica

El siguiente Algoritmo 16 demuestra cómo se puede descifrar un punto en una curva elíptica que ha sido previamente cifrado utilizando la llave pública de un receptor. Este proceso implica la utilización de la llave privada del receptor para revertir el cifrado aplicado. Este asume que el mensaje cifrado consiste en puntos que han sido transformados utilizando la llave pública y un valor aleatorio k. El proceso de descifrado utiliza la llave privada para calcular la operación inversa y recuperar el punto original en la curva.

Algoritmo 16 Descifrado de un punto en la curva elíptica

```
1: procedure DESCIFRAR(msgCifrado, k, LlavePrivada, p, a)
2:     msgD ← {}        ▷ Inicializa un conjunto vacío para los puntos descifrados
3:     while msgCifrado ≠ ∅ do
4:         punto ← msgCifrado[1]      ▷ Toma el primer punto del mensaje cifrado
5:         msgD ← msgD ∪ GeneracionLlavePublica(p, a, punto, LlavePrivada) ▷
   Aplica la función inversa de cifrado
6:         msgCifrado ← msgCifrado − {punto}      ▷ Elimina el punto procesado
7:     end while
8:     return msgD                              ▷ Devuelve los puntos descifrados
9: end procedure
```

”La vida no es fácil para ninguno de nosotros. Pero, ¿qué importa? Debemos tener perseverancia y, sobre todo, confianza en nosotros mismos. Debemos creer que estamos dotados para algo y que esto debe alcanzarse.”

Marie Curie (1867-1934)
Física y química polaco-francesa

CAPÍTULO 4

Algoritmos para Firmas Digitales

4.1. Firmas Digitales con RSA

El algoritmo 17 implementa la firma digital de un mensaje utilizando la llave privada de RSA. Este método garantiza la autenticidad e integridad del mensaje. Este algoritmo toma el mensaje que se hashiza utilizando SHA-256 para obtener un resumen que será firmado. Este hash se convierte de una cadena hexadecimal a un número entero y posteriormente la firma se realiza utilizando la llave privada mediante exponenciación modular, que cifra el hash del mensaje con el exponente privado d y el módulo n.

Algoritmo 17 Firma digital de un mensaje utilizando RSA

1: **function** FIRMARRSA($mensaje, llavePrivada$)
2: $(d, n) \leftarrow llavePrivada$ ▷ Asignar exponente y módulo de llave privada
3: $hashMensaje \leftarrow SHA-256(mensaje)$ ▷ Aplicar función hash al mensaje y convertir a entero
4: $firma \leftarrow$ PowerMod$(hashMensaje, d, n)$ ▷ Calcular la firma utilizando exponenciación modular
5: **return** $firma$ ▷ Devuelve la firma, que es el hash cifrado
6: **end function**

4.2. Firmas Digitales con ECC

El Algoritmo 18 es fundamental en la definición de parámetros para curvas elípticas, como $secp256k1$, utilizada en tecnologías de criptografía como Bitcoin. Este algoritmo genera un número primo grande adecuado para su uso como parte de los parámetros de la curva. Este procedimiento asegura que el número primo generado tenga exactamente el número de bits especificado, lo cual es vital para alcanzar el nivel de seguridad requerido en aplicaciones criptográficas basadas en curvas elípticas. El algoritmo selecciona aleatoriamente un número primo dentro del rango especificado, asegurando que tenga propiedades adecuadas para la seguridad criptográfica.

Algoritmo 18 Generación de un número primo grande adecuado para curvas elípticas

```
1: function CalcularPrimoGrande(bits)
2:     L ← 2^(bits−1)                ▷ Establece el límite inferior del rango
3:     U ← 2^bits − 1                ▷ Establece el límite superior del rango
4:     primo ← Random(L, U)          ▷ Generar número primo aleatoriamente
5:     return primo
6: end function
```

El Algoritmo 19 emplea operaciones sobre una curva elíptica para generar una firma digital de un mensaje, donde el proceso es el siguiente:

- El número k es generado aleatoriamente dentro del rango de $[1, n-1]$, siendo n el orden del punto base.
- El punto $(x1, y1)$ se calcula como k veces el punto base sobre la curva, reducido módulo el número primo que define el campo sobre el cual opera la curva.

- La componente r de la firma se deriva directamente de la abscisa x de la coordenada del punto calculado.
- El hash del mensaje se calcula usando SHA-256 para asegurar la integridad.
- El componente s de la firma se calcula mediante operaciones aritméticas que involucran el hash del mensaje, la llave privada, y el inverso de k.
- La variable puntoBase en el algoritmo corresponde al punto base P sobre la curva elíptica.

Algoritmo 19 Generación de firma digital usando ECDSA

function GENERACIONDEFIRMA($mensaje, puntoBase, llavePrivada, primo, n$)
    $k \leftarrow$ RandomInteger$(1, n-1)$ ▷ Selecciona aleatoriamente el entero k
    $(x1, y1) \leftarrow k \cdot puntoBase \ mod \ primo$ ▷ Calcula el punto resultante kP
    $r \leftarrow x1$ ▷ La abscisa x del punto es el componente r de la firma
    $e \leftarrow$ Hash$(mensaje, "SHA-256")$ ▷ Calcula el hash del mensaje
    $kInverso \leftarrow k^{-1} \ mod \ n$ ▷ Calcula el inverso modular de k
    $s \leftarrow (kInverso \cdot (e + llavePrivada \cdot r)) \ mod \ primo$ ▷ Calcula el componente s de la firma
    return (r, s) ▷ Devuelve la firma como un par de valores
end function

4.3. Validación de Firmas Digitales con RSA

El Algoritmo 20 examina la validez de una firma digital RSA, comparando el hash del mensaje original con el resultado de la exponenciación modular de la firma utilizando la llave pública siguiendo este proceso:

- La función hash SHA-256 se aplica al mensaje, y el resultado, normalmente una cadena hexadecimal, se convierte en un número entero interpretando cada carácter hexadecimal en base 16.
- La verificación se realiza exponenciando la firma al exponente público e, utilizando el módulo n, mediante el método de exponenciación por cuadrados. Este es un método eficiente para calcular potencias grandes bajo un módulo.
- Se compara el hash del mensaje con el resultado de la exponenciación modular para determinar si la firma es válida.

Algoritmo 20 Verificación de una firma digital RSA

```
1: function VERIFICARRSA(mensaje, firma, llavePublica)
2:     (e, n) ← llavePublica    ▷ Extracción exponente y módulo de llave pública
3:     hashMensaje ← SHA − 256(mensaje) ▷ Aplicar función hash al mensaje
4:     hashVerificacion ← 1
5:     base ← firma
6:     while e > 0 do
7:         if e mod 2 = 1 then
8:             hashVerificacion ← (hashVerificacion · base) mod n
9:         end if
10:        base ← (base · base) mod n
11:        e ← e | 2
12:    end while
13:    if hashMensaje = hashVerificacion then
14:        return ”Firma Válida”
15:    else
16:        return ”Firma Inválida”
17:    end if
18: end function
```

4.4. Validación de Firmas Digitales con ECC

El Algoritmo 21 comprueba la validez de una firma digital creada con ECDSA, utilizando la llave pública del firmante, un mensaje y la firma que se verifica. Este algoritmo verifica si el punto calculado a partir de la firma y la llave pública corresponde al punto base proyectado por los componentes de la firma. La validez de la firma se determina comparando la coordenada x del punto resultante con el componente r de la firma.

Algoritmo 21 Verificación de una firma digital usando ECDSA

function VERIFICACION($mensaje, firma, llavePublica, puntoBase, primo, n$)
    $(r, s) \leftarrow firma$ ▷ Desempaqueta la firma en sus componentes r y s
    $e \leftarrow$ Hash$(mensaje, "SHA-256")$ ▷ Calcula el hash del mensaje
    $w \leftarrow s^{-1}\ mod\ primo$ ▷ Calcula el inverso modular de s
    $v1 \leftarrow (e \cdot w)\ mod\ primo$ ▷ Calcula $v1$
    $v2 \leftarrow (r \cdot w)\ mod\ primo$ ▷ Calcula $v2$
    $(x1, y1) \leftarrow (v1 \cdot puntoBase + v2 \cdot llavePublica)\ mod\ primo$ ▷ Suma ponderada de puntos
    if $(x1, y1) = (0, 0)$ **then**
        return "Firma Inválida" ▷ El punto resultante es el punto en el infinito
    else
        $v \leftarrow x1$
        if $v = r$ **then**
            return "Firma Válida"
        else
            return "Firma Inválida"
        end if
    end if
end function

”Nuestra mayor debilidad radica en rendirnos. La forma más segura de tener éxito es intentarlo una vez más.”

Thomas Edison (1847-1931)
Inventor y empresario estadounidense

CAPÍTULO 5

Resultados

En esta sección, se presentan los resultados de los algoritmos implementados, así como un análisis de los parámetros de entrada empleados y los resultados de salida obtenidos.

5.1. Curvas Elípticas

5.1.1. Validación de Parámetros de una Curva Elíptica

Como parte de la verificación de los parámetros según el Algoritmo 4, los resultados están agrupados en la Tabla 5.1, donde se muestran los valores de los coeficientes a y b, el calculo del determinante $4a^3 + 27b^2 \neq \emptyset$, y la evaluación para determinar si la curva elíptica es racional.

Caso 1: $a = 1$**,** $b = 7$

- **Determinante**: 3559
- **Evaluación**: Curva Racional (Verdadero)

Este caso muestra que con un determinante significativamente positivo y no cero, la curva es suave y bien definida, asegurando así que sea racional.

Caso 2: $a = -1$**,** $b = 1$

- **Determinante**: 23
- **Evaluación**: Curva Racional (Verdadero)

Un determinante positivo y no cero confirma que la curva es racional. La curva resultante es suave, sin puntos de autointersección o discontinuidades.

Caso 3: $a = 0$**,** $b = 0$

- **Determinante**: 0
- **Evaluación**: Curva No Racional (Falso)

El determinante cero resulta en una curva con singularidades, lo que hace que la curva no sea racional.

Caso 4: $a = -3$**,** $b = 1$

- **Determinante**: -81
- **Evaluación**: Curva Racional (Verdadero)

A pesar de un determinante negativo, el hecho de que no sea cero es suficiente para mantener la curva libre de singularidades.

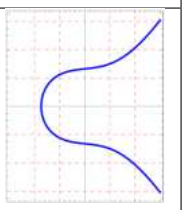
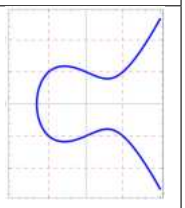
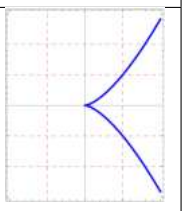
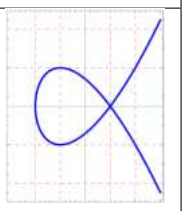
Caso 5: $a = -3$**,** $b = 2$

- **Determinante**: 0
- **Evaluación**: Curva No Racional (Falso)

Un determinante de cero indica la presencia de una singularidad, invalidando la curva como racional.

Los casos analizados demuestran claramente que el determinante $4a^3 + 27b^2$ juega un papel importante en la determinación de la racionalidad de las curvas elípticas. Un determinante no nulo es una condición necesaria para que la curva sea suave y continua, lo que es esencial para su racionalidad.

Tabla 5.1:
Verificación de parámetros Algoritmo 4

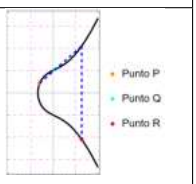
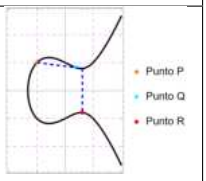
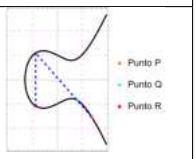
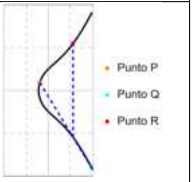
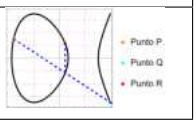
Caso	**a**	**b**	**Determinante** $4a^3 + 27b^2$	**Curva Racional**	**Gráfico**
1	1	7	3559	True	
2	-1	1	23	True	
3	0	0	0	False	
4	-3	1	-81	True	
5	-3	2	0	False	

Nota: Elaboración propia.

5.1.2. Suma de dos puntos sobre la Curva Elíptica

Los resultados obtenidos del Algoritmo 5 muestran la suma de dos puntos sobre distintas configuraciones de curvas elípticas, según se detalla en la Tabla 5.2. Cada conjunto de operaciones se ha evaluado no solo en su validez aritmética sino también en su consistencia geométrica, como se visualiza en los gráficos correspondientes. Cada caso demuestra cómo la elección de puntos P y Q, así como los coeficientes de la curva influyen en el punto resultante R, verificando geométricamente la operación de suma en el plano.

Tabla 5.2:
Suma de dos puntos sobre la Curva Elíptica $P + Q = R$

Caso	a	b	P	Q	R	Gráfico
1	1	1	{-0.6, 0.4289}	{0.10, 1.0492}	{1.2853, -2.0997}	
2	-1	1	{-1, 1}	{0.4, 0.815}	{2.28, 3.253}	
3	-1	1	{1.4, -1.531}	{0.8, -0.843}	{-0.88, -1.089}	
4	2	1	{-0.4, 0.368}	{2, -3.605}	{1.14, 2.18}	
5	-3	1	{-1.6, 1.305}	{0.3, 0.356}	{1.549, 0.267}	

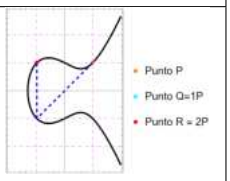
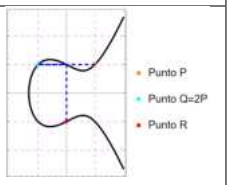
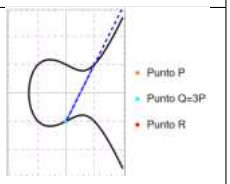
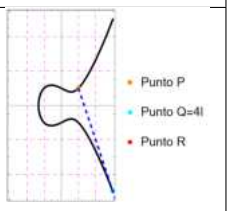
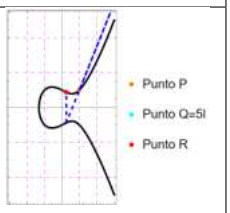
Nota: Elaboración propia.

5.1.3. Suma de k veces el punto P sobre la Curva Elíptica

La Tabla 5.3 y los gráficos ilustran los resultados del Algoritmo 6 de la multiplicación del punto $P = \{1, 1\}$ en una curva elíptica definida por la ecuación $y^2 = x^3 - x + 1$. A medida que incrementamos el escalar k, observamos cómo el resultado de kP cambia:

- En el **Caso 1**, con $k = 2$, el punto resultante es $\{$-1, 1$\}$, indicando que la duplicación de P se mapea simétricamente respecto al eje y.
- El **Caso 2** muestra que al triplicar el punto, $k = 3$, el resultado es $\{$0, -1$\}$. Esto ilustra que el punto triple se localiza en la parte inferior de la curva, demostrando cómo la adición iterativa altera las coordenadas.
- Para $k = 4$ en el **Caso 3**, el punto $\{$3, -5$\}$ sugiere un aumento significativo en la magnitud de las coordenadas, resaltando el efecto no lineal de la suma repetida en la curva.
- El **Caso 4**, donde $k = 5$, resulta en el punto $\{$5, 11$\}$, mostrando un incremento aún mayor, tanto en la coordenada x como y.
- Finalmente, el **Caso 5** para $k = 6$ da como resultado $\{$0.25, 0.875$\}$, un punto que parece devolver las coordenadas a una escala más cercana a P, cerrando el ciclo de transformaciones observadas a lo largo de la secuencia.

Tabla 5.3:
Suma de k veces el punto P sobre la Curva Elíptica

Caso	a	b	k	P	kP	Gráfico
1	-1	1	2	{1, 1}	{-1, 1}	
2	-1	1	3	{1, 1}	{0, -1}	
3	-1	1	4	{1, 1}	{3, -5}	
4	-1	1	5	{1, 1}	{5, 11}	
5	-1	1	6	{1, 1}	{0.25, 0.875}	

Nota: Elaboración propia.

5.1.4. Generación de llaves con Curvas Elípticas

Inicialmente, se establecieron las llaves públicas y privadas para Alice y Bob utilizando el Algoritmo 10, así como las coordenadas asignadas para diversas operaciones y el valor aleatorio k. La Figura 5.1 muestra las llaves asignadas.

El procedimiento de generación de llaves con curvas elípticas es un com-

Figura 5.1:
Llaves públicas y privadas de Alice y Bob en Mathematica®.

Out[255]=

	Private Key	Public Key
Alice	18940	{631177, 921574}
Bob	85980	{292424, 300396}

ponente fundamental en la criptografía moderna, especialmente en esquemas de cifrado y firma digital. A continuación, analizamos los resultados obtenidos para dos participantes, Alice y Bob.

La generación de llaves se realiza bajo los siguientes pasos:

1. Selección de un número primo grande p que define el campo finito. En este caso, $p = 948581$.

2. Elección de coeficientes a y b para la ecuación de la curva elíptica, asegurando que la discriminante $4a^3 + 27b^2 \neq 0$ para evitar singularidades. Aquí, $a = 85018$ y $b = 7858$.

3. Determinación de un punto base G en la curva, que será utilizado para todas las operaciones criptográficas. Para este ejemplo, $G = (906513, 590846)$.

4. Generación de un número aleatorio privado que sirve como llave privada. En nuestro caso, las llaves privadas son $LlaveSA = 18940$ y $LlaveSB = 85980$.

5. Cálculo de la llave pública como el producto escalar de la llave privada y el punto base G. Las llaves públicas resultantes son:

   - Llave pública de Alice: $llavePA = (631177, 921574)$.

- Llave pública de Bob: $llavePB = (292424, 300396)$.

5.1.5. Cifrado de mensaje con Curvas Elípticas

Considerando el mensaje ”Fernando123”, se transformó en una secuencia de puntos sobre la curva elíptica usando los códigos ASCII de los caracteres. Los puntos resultantes de esta codificación se detallan a continuación:

```
{{57263, 245925}, {82623, 421674}, {93254, 412653}, {89983,
  182930}, {79348, 33448}, {89983, 182930}, {81801, 144226}, {90799,
  375650}, {40083, 274758}, {40901, 448639}, {41720, 284598}}
```

La Figura 5.2 presenta la asignación de coordenadas junto con sus correspondientes valores k. La asignación de valores aleatorios k y su correspondiente

Figura 5.2:
Asignación de coordenadas y el valor aleatorio k en Mathematica®.

Out[356]=

Assigned Coordinates	k
{515 753, 347 815, 650 473, 661 421}	488 089
{69 924, 361 324, 579 798, 407 265}	801 429
{870 544, 44 539, 443 493, 649 997}	10 345
{731 680, 320 381, 923 334, 69 401}	864 327
{605 911, 471 103, 370 129, 457 489}	424 866
{273 865, 461 648, 491 982, 482 537}	600 841
{422 855, 635 147, 787 957, 651 211}	549 415
{431 541, 180 869, 647 225, 625 986}	21 596
{277 545, 110 540, 915 092, 87 814}	525 540
{97 045, 361 457, 272 092, 601 882}	716 155
{532 208, 553 699, 887 344, 790 419}	208 466

asignación de coordenadas validan la integridad y la no predictibilidad de las operaciones criptográficas implementadas.

5.1.6. Descifrado de mensaje con Curvas Elípticas

En la fase de descifrado, se utilizó un procedimiento de múltiples pasos para transformar los puntos cifrados de vuelta a su forma original, validando así la integridad y eficacia del cifrado basado en curvas elípticas. A continuación, se detalla el análisis de los tiempos de ejecución y la exactitud del texto descifrado. El proceso de descifrado se ejecutó utilizando una secuencia de pasos definidos en Mathematica®, donde cada paso es crucial para asegurar la recuperación precisa del mensaje original. Los pasos incluyen la multiplicación escalar de puntos, la negación de la coordenada y de puntos, y finalmente la decodificación ASCII del mensaje. Los tiempos de ejecución para cada paso se registraron para evaluar la eficiencia del proceso.

- La **multiplicación escalar** de puntos, que es esencial para recuperar el mensaje cifrado, mostró un tiempo de ejecución que varió desde 0.0008121 segundos para los primeros puntos hasta 0.0028986 segundos para otros, indicando una operación relativamente rápida.
- La **negación de puntos y sus coordenadas y** se completó con tiempos igualmente breves, lo cual es testimonio de la eficiencia computacional del algoritmo utilizado.
- El paso final, la **decodificación numérica y ASCII** del mensaje, confirmó la exactitud del proceso de descifrado, reconstruyendo el mensaje original $"Fernando123"$ en un tiempo de solo 0.001725 segundos.

Los tiempos totales de descifrado, sumando todas las operaciones, resultaron ser sumamente eficientes, lo que demuestra la viabilidad del uso de curvas elípticas

para aplicaciones criptográficas seguras y rápidas. La secuencia exacta de puntos obtenidos y su correspondiente transformación de vuelta a texto se muestran a continuación:

Figura 5.3:
Descifrado del mensaje con Curvas Elípticas en Mathematica®.

```
Mensaje Cifrado: {57 263, 82 623, 93 254, 89 983, 79 348, 89 983, 81 801, 90 799, 40 083, 40 901, 41 720}
Mensaje Codificado en ASCII: {70, 101, 114, 110, 97, 110, 100, 111, 49, 50, 51}
Mensaje Descifrado: Fernando123
```

5.1.7. Pruebas de rendimiento de Cifrado y Descifrado de mensaje con Curvas Elípticas

En esta sección, se discuten los resultados obtenidos de las pruebas de rendimiento en la generación de llaves y los procesos de cifrado y descifrado empleando curvas elípticas. Los procedimientos utilizados se apoyan en la configuración de parámetros esenciales que se describen a continuación:

- **Número Primo Grande** p: Se seleccionó un número primo grande, específicamente $p = 948581$, que define el campo sobre el cual se configura la curva elíptica.
- **Coeficientes** a **y** b: Para la ecuación de la curva $y^2 = x^3 + ax + b$, se establecieron los coeficientes $a = 85018$ y $b = 7858$, cumpliendo con la condición $4a^3 + 27b^2 \neq 0$ para asegurar una curva no singular.
- **Punto Generador** G: Se utilizó el punto $G = \{906513, 590846\}$ como generador.
- **Llaves Privadas**: Las llaves privadas se generaron aleatoriamente dentro del rango permitido por p, siendo $keySA = 18940$ para Alice y $keySB = 85980$ para Bob.

Para evaluar la eficacia de los algoritmos de cifrado y descifrado, se seleccionaron segmentos de texto de diferentes longitudes y se midió tanto el tiempo de ejecución como el almacenamiento requerido. La Tabla 5.4 resume estos resultados.

En los datos obtenidos de las pruebas de rendimiento para las operaciones de cifrado y descifrado, se presenta un resumen que refleja el tiempo

Tabla 5.4:
Resultados de Tiempo y Almacenamiento con Curva Elíptica

Núm. de caracteres	Tiempo Cifrado (seg)	Tiempo Descifrado (seg)	Almacen. Texto Cifrado (bytes)
100	0.111709	0.0637211	4,328
650	0.845711	0.367182	28,216
1300	1.51056	0.799223	56,432
1950	2.13254	1.0682	84,728
2600	2.69175	1.41516	113,000
3250	3.21832	1.65021	141,274
3900	3.82563	1.93796	169,312
4550	4.38426	2.25774	197,572
5200	5.1316	2.79604	225,795
5850	5.8079	2.97587	254,160

Nota: Elaboración propia.

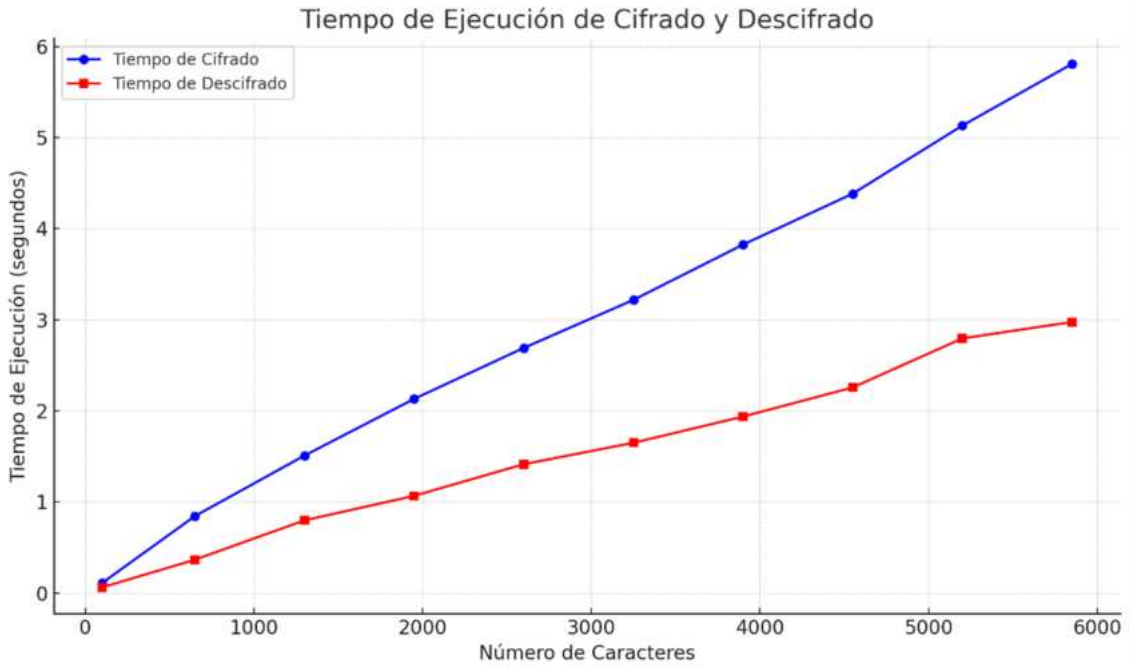
de ejecución en segundos para diferentes tamaños de fragmentos de texto. La complejidad de las operaciones criptográficas se evidencia a través de estos resultados, ofreciendo una visión del desempeño del algoritmo en relación con el número de caracteres procesados.

Al analizar el tiempo de ejecución para las operaciones de cifrado, se observa un crecimiento proporcional al tamaño del fragmento de texto en la Figura 5.4. Desde un modesto 0.111709 segundos para 100 caracteres, este valor aumenta gradualmente hasta 5.8079 segundos para 5850 caracteres. Este comportamiento sugiere una complejidad que puede describirse mediante la notación Big O, indicando una relación lineal entre el tiempo de ejecución y la cantidad de datos de entrada.

Similarmente, las operaciones de descifrado exhiben un patrón de crecimiento proporcional. Este fenómeno es coherente con la naturaleza de las operaciones criptográficas y la influencia directa del tamaño de los datos en el tiempo de procesamiento.

En cuanto al almacenamiento cifrado, se observa en la Figura 5.5 un aumento en el espacio necesario para almacenar la versión cifrada del texto (en

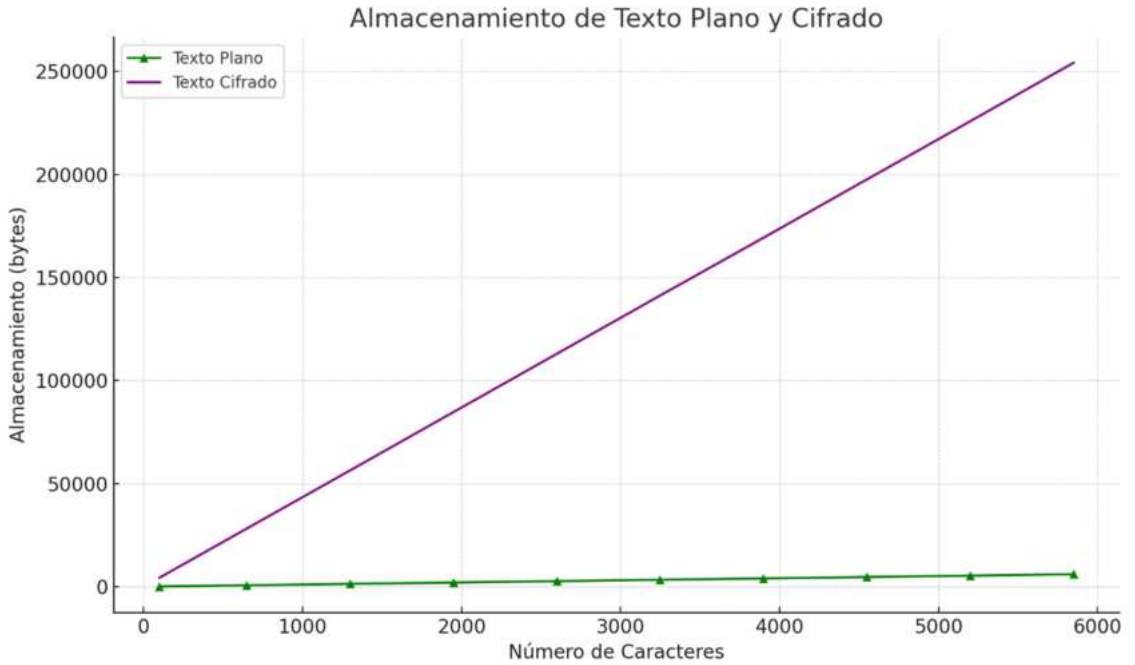
Figura 5.4:
Tiempo de Ejecución de Cifrado y Descifrado



Nota: Elaboración propia.

bytes) a medida que la longitud del texto plano aumenta. Este comportamiento es previsible y común en muchos algoritmos de cifrado, donde el tamaño de la salida cifrada tiende a ser mayor que el del texto plano. La ventaja distintiva de utilizar Curvas Elípticas Criptográficas (ECC, por sus siglas en inglés) radica en su capacidad para ofrecer niveles de seguridad robustos, aunque con el aumento gradual en el espacio requerido para almacenar la versión cifrada del texto. Este incremento en el almacenamiento cifrado se presenta como un indicador positivo desde la perspectiva de la seguridad en la Criptografía de Curvas Elípticas.

Figura 5.5:
Almacenamiento de Texto Plano y Cifrado



Nota: Elaboración propia.

5.1.8. Firmas Digitales con Curvas Elípticas

En el ámbito de la seguridad digital, las firmas digitales juegan un papel crucial al asegurar la autenticación, integridad de datos y no repudio. El algoritmo realizado implementa las firmas digitales usando curvas elípticas, incluyendo la generación de llaves, firma de mensajes y la verificación de firmas.

Generación de Firma Digital con Curvas Elípticas

La generación de firmas digitales mediante el Algoritmo 19 ECDSA (Elliptic Curve Digital Signature Algorithm) se basa en el uso eficiente de las propiedades matemáticas de las curvas elípticas y la teoría de números.

Tabla 5.5:
Parámetros definidos en Curva Elíptica secp256k1

Parámetro	Valor
Número Primo p	115792089237316195423570985008687907852837564279074904382605163141518161494337
Punto Base G_x	55066263022277343669578718895168534326250603453777594175500187360389116729240
Punto Base G_y	32670510020758816978083085130507043184471273380659243275938904335757337482424
Orden n	115792089237316195423570985008687907852837564279074904382605163141518161494337

Nota: Elaboración propia.

En la implementación, la firma se genera a partir de un mensaje utilizando un punto base predefinido sobre la curva, junto con la llave privada del usuario y parámetros criptográficos como un número primo p y el orden n del punto base. El procedimiento comienza con la selección de un entero aleatorio k, elegido dentro del rango $[1, n-1]$. Este valor aleatorio es esencial para garantizar la seguridad de la firma, ya que una elección predecible podría comprometer todo el sistema criptográfico. Para esta implementación, se utilizó la curva elíptica secp256k1, definida en la tabla 5.5

1. Se calcula el punto (x_1, y_1) como k veces el punto base en la curva, reducido módulo el número primo p. La coordenada x_1 de este punto se utiliza como la componente r de la firma. Si x_1 resultase ser cero, se debería seleccionar un nuevo k y repetir el proceso.

2. Se emplea la función hash SHA-256 para generar un resumen del mensaje, e.

3. La componente s de la firma se calcula con la fórmula: $s = k^{-1} \cdot (e + r \cdot d)$ $mod\ n$ donde d representa la llave privada y k^{-1} es el inverso multiplicativo de k módulo n. En donde podemos ver la firma generada en la Figura 5.6

La firma generada, compuesta por las componentes r y s, se almacena junto con el mensaje original para su futura verificación.

Figura 5.6:
Firma Digital con Curvas Elípticas Generada en Mathematica®

Validación de Firma Digital con Curvas Elípticas

Utilizando el algoritmo de validación basado en Curvas Elípticas Criptográficas (ECC), se verifica si la firma asociada a un mensaje es legítima y si el mensaje no ha sido alterado desde su firma.

El Algoritmo 20 de validación se inicia extrayendo y utilizando los componentes de la firma, r y s, y la llave pública del emisor. Estos elementos se emplean para verificar la correspondencia y autenticidad de la firma asociada al mensaje. Los pasos específicos son los siguientes:

1. El hash del mensaje es recalculado usando SHA-256, resultando en un valor e, que garantiza que el contenido del mensaje no ha sido modificado.

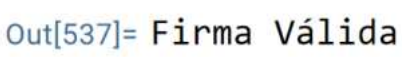
2. Se calculan dos valores intermedios, $v1$ y $v2$, usando el inverso del componente s de la firma: $v1 = e \cdot s^{-1} \ mod \ p$, $v2 = r \cdot s^{-1} \ mod \ p$ donde p es el número primo que define el campo finito sobre el que opera la curva.

3. Estos valores se utilizan para generar un punto $(x1, y1)$ a partir del punto base de la curva y la llave pública, mediante:

$$(x1, y1) = v1 \cdot \text{punto base} + v2 \cdot \text{llave pública}$$

4. La firma se considera válida mostrada en la figura 5.7 si la coordenada x de este punto, $x1$, es igual a r. Si no coincide, la firma es declarada inválida.

Figura 5.7:
Validación de Firma Digital con Curvas Elípticas Generada en Mathematica®



Los resultados obtenidos en la Tabla 5.6 de las pruebas de rendimiento para la firma y validación de mensajes utilizando criptografía de curvas elípticas (ECC) muestran un aumento progresivo en el tiempo de ejecución conforme el número de caracteres en el mensaje aumenta. Esto es esperable ya que un mayor volumen de datos implica cálculos de hash más extensos y operaciones criptográficas más complejas. Sin embargo, los tiempos registrados son notablemente bajos, destacando la eficiencia de ECC para procesar operaciones de firma y validación en tiempo real, lo cual es esencial en aplicaciones donde la seguridad no debe comprometer la rapidez, como en comunicaciones seguras y transacciones electrónicas.

Respecto al almacenamiento, se observa que la cantidad de espacio necesario para almacenar las firmas es constante a través de todas las pruebas. Esto indica que la longitud de la firma depende únicamente de las propiedades de la curva elíptica y de la implementación del algoritmo, no del tamaño del mensaje. Este comportamiento es una ventaja significativa, especialmente en sistemas donde el almacenamiento es limitado o en aplicaciones que demandan escalabilidad sin fluctuaciones en los requisitos de almacenamiento por cada mensaje.

Las llaves más pequeñas y las firmas de tamaño constante son particularmente útiles en entornos donde el rendimiento y la eficiencia del almacenamiento son críticos. En dispositivos con recursos limitados, como teléfonos móviles y otros dispositivos IoT, la capacidad para almacenar y gestionar llaves de forma

Tabla 5.6:
Tiempos de Ejecución y Almacenamiento de Firma Digital con Curvas Elípticas Generada en Mathematica®

Almacenamiento (*bytes*)			**Tiempo de Ejecución (*seg*)**		
Llave Pública	**Llave Privada**	**Firma**	**Número de caracteres**	**Firma**	**Validación**
157	77	156	100	0.0005187	0.0004531
			650	0.0004534	0.0004661
			1300	0.0005195	0.0005517
			1950	0.00055201	0.0005949
			2600	0.0005846	0.0005422
			3250	0.0005852	0.0005007
			3900	0.0006132	0.000651
			4550	0.0006716	0.0006756
			5200	0.000742	0.0007245
			5850	0.0007671	0.0007693

Nota: Elaboración propia.

eficiente sin sacrificar la seguridad es fundamental. La reducción en el tamaño de las llaves también facilita un intercambio más rápido y seguro de información criptográfica, lo cual es crucial en aplicaciones de comunicación en tiempo real.

Además, los tiempos de ejecución reducidos para operaciones de firma y validación significan que ECC puede soportar sistemas de alto rendimiento, permitiendo la autenticación y la verificación de integridad de mensajes en tiempo real sin una degradación perceptible del rendimiento del sistema. Esto es esencial para aplicaciones críticas, como sistemas financieros, comunicaciones de seguridad nacional y plataformas de comercio electrónico, donde la velocidad de las transacciones es tan importante como su seguridad.

5.2. RSA

5.2.1. Validación de Parámetros de RSA

El Algoritmo 1 de RSA requiere la selección de dos números primos grandes p y q, y el cálculo de n y $\phi(n)$, donde $n = p \times q$ y $\phi(n) = (p-1) \times (q-1)$.

La Tabla 5.7 presenta una evaluación detallada de los parámetros utilizados en el algoritmo RSA. Este análisis nos permite asegurar la validez de los números primos seleccionados y la correcta generación de las llaves criptográficas. Los casos válidos muestran la correcta selección de números primos p y q, resultando en valores de n y $\phi(n)$ que cumplen con los requisitos del algoritmo RSA. Por otro lado, los casos inválidos destacan errores comunes como la repetición de valores para p y q, o la selección de números no primos. La evaluación minuciosa de estos parámetros garantiza la seguridad y eficiencia del cifrado y descifrado en el esquema RSA. La correcta elección y verificación de p y q asegura la integridad del sistema criptográfico y previene vulnerabilidades potenciales.

La información presentada se basa en pruebas realizadas con números primos de 1024 bits, siguiendo los estándares recomendados para la implementación segura de algoritmos de criptografía asimétrica como RSA.

Véase los datos completos de la Tabla 5.7 en el Anexo **??**

Tabla 5.7:
Evaluación de Parámetros RSA

Caso	p (1024 bits)	q (1024 bits)	n	$\phi(n)$	Evaluación
1	120586792...	197802749...	238476648...	238476648...	Válido
2	174953476...	165843268...	290066654...	290066653...	Válido
3	198745290...	128736479...	255853298...	255853298...	Válido
4	113587290...	102983479...	116870153...	116870153...	Válido
5	173847290...	173847290...	301262812...	301262812...	Inválido
6	154983272...	0	0	0	Inválido
7	193847290...	193847290...	375825281...	375825281...	Válido
8	185372001...	100037261...	185372001...	185372000...	Válido
9	180290381...	0	0	0	Inválido
10	0	100271382...	0	0	Inválido

Nota: Elaboración propia.

5.2.2. Generación de Llaves RSA

Este sistema criptográfico asimétrico utiliza pares de llaves: una llave pública para cifrar los datos y una llave privada para descifrarlos. La longitud de estas llaves, medida en bits, es fundamental para la seguridad del sistema, pero también influye en el almacenamiento y la eficiencia del proceso criptográfico.

La generación de llaves RSA del Algoritmo 3 implica calcular el módulo n, el exponente público e y el exponente privado d. El exponente público e debe ser un entero pequeño que sea coprimo con $\phi(n)$, y el exponente privado d se calcula como el inverso multiplicativo de e módulo $\phi(n)$.

Para entender mejor el impacto del tamaño de las llaves RSA, se realizó una evaluación con llaves de 1024 bits para dos usuarios, Alice y Bob. Las llaves publicas y privadas para ambos usuarios se almacenaron en archivos de texto y se midió su tamaño tanto en términos de bytes como de espacio en disco. A continuación se presentan los datos obtenidos en la Tabla 5.8

El tamaño de las llaves RSA en el archivo de texto es de 618 bytes. Sin

Tabla 5.8:
Tamaño y Almacenamiento de Llaves RSA

Llave	Tamaño (bytes)	Tamaño en disco (KB)
Llave privada de Alice	618	4.00 (4096 bytes)
Llave pública de Alice	618	4.00 (4096 bytes)
Llave privada de Bob	618	4.00 (4096 bytes)
Llave pública de Bob	618	4.00 (4096 bytes)

Nota: Elaboración propia.

embargo, el tamaño en disco es significativamente mayor, de 4.00 KB. Esta diferencia se debe a la forma en que los sistemas de archivos gestionan el almacenamiento de datos. Cada archivo ocupa al menos un bloque de disco, cuyo tamaño puede variar según el sistema de archivos. En muchos sistemas de archivos modernos, el tamaño de un bloque es típicamente de 4 KB. Por lo tanto, aunque el contenido del archivo solo ocupe 618 bytes, el sistema de archivos asigna un bloque completo de 4 KB para almacenarlo, resultando en una utilización de disco de 4.00 KB.

El tamaño de las llaves RSA influye directamente en la seguridad del sistema. Llaves más largas proporcionan mayor seguridad pero requieren más espacio de almacenamiento y tiempo de procesamiento. Las llaves de 1024 bits, como las utilizadas en esta evaluación, son actualmente consideradas seguras para la mayoría de las aplicaciones, aunque a costa de un mayor uso de recursos comparado con llaves más cortas.

La optimización del tamaño de las llaves RSA es un equilibrio entre seguridad y eficiencia. Mientras que las llaves más largas ofrecen mayor seguridad, también aumentan los requisitos de almacenamiento y el tiempo de procesamiento. Los resultados obtenidos demuestran que, a pesar del contenido relativamente pequeño de las llaves en texto, el almacenamiento en disco es con-

siderablemente mayor debido a la gestión de bloques del sistema de archivos. Esta comprensión es esencial para la planificación y gestión de sistemas criptográficos seguros y eficientes. La información presentada se basa en pruebas realizadas y estándares recomendados para la implementación segura de algoritmos de criptografía asimétrica como RSA.

Figura 5.8:
Generación de Llaves RSA generado desde Mathematica®.

5.2.3. Cifrado y Descifrado RSA

En esta sección se presentan los resultados de las pruebas de rendimiento realizadas para evaluar el tiempo de ejecución y el almacenamiento necesario al cifrar y descifrar mensajes de diferentes longitudes utilizando el algoritmo RSA. La Tabla 5.9 muestra los tiempos de ejecución y los requisitos de almacenamiento tanto para el texto plano como para el texto cifrado.

El cifrado de un mensaje m se realiza mediante la fórmula $c = m^e \mod n$, y el descifrado se realiza mediante $m = c^d \mod n$.

Tabla 5.9:
Resultados de Tiempo y Almacenamiento con RSA

Número de caracteres	Tiempo de Ejecución (seg)		Almacenamiento (bytes)	
	Cifrado	Descifrado	Texto plano	Texto Cifrado
100	0.075466	0.076584	103	308
650	0.490529	0.497796	663	28,216
1300	0.981058	0.995592	1,327	52,499
1950	1.47158	1.493338	1,999	84,728
2600	1.962116	1.991184	2,662	113,000
3250	2.45264	2.48898	3,333	141,274
3900	2.94317	2.986776	3,997	169,312
4550	3.43370	3.484572	4,661	197,572
5200	3.924232	3.982368	5,333	225,795
5850	4.41476	4.480164	5,999	254,160

Nota: Elaboración propia.

La Tabla 5.9 proporciona una visión detallada de cómo el tiempo de ejecución y el almacenamiento requerido varían con el tamaño del mensaje. A continuación, se presentan los principales puntos observados en los resultados:

Tiempo de Ejecución

- **Cifrado:**

- El tiempo de cifrado aumenta de manera casi lineal con el número de caracteres en el mensaje. Para mensajes de 100 caracteres, el tiempo de cifrado es de aproximadamente 0.075466 segundos, mientras que para mensajes de 5850 caracteres, el tiempo de cifrado es de 4.41476 segundos.
- Esta tendencia lineal indica que el tiempo de cifrado depende directamente de la longitud del mensaje.

- **Descifrado:**

  - Similar al cifrado, el tiempo de descifrado también muestra un incremento casi lineal con el tamaño del mensaje. Para mensajes de 100 caracteres, el tiempo de descifrado es de aproximadamente 0.076584 segundos, y para mensajes de 5850 caracteres, es de 4.480164 segundos.
  - La similitud en los tiempos de cifrado y descifrado sugiere una eficiencia consistente del algoritmo RSA en ambos procesos.

Almacenamiento

- **Texto Plano:**

  - El almacenamiento del texto plano también aumenta proporcionalmente con el número de caracteres, como era de esperarse. Por ejemplo, un mensaje de 100 caracteres requiere 103 bytes, mientras que un mensaje de 5850 caracteres necesita 5,999 bytes.

- **Texto Cifrado:**

- El almacenamiento requerido para el texto cifrado es significativamente mayor que el del texto plano. Esto se debe a la naturaleza del algoritmo RSA, que expande los datos durante el cifrado.
- Para un mensaje de 100 caracteres, el texto cifrado ocupa 308 bytes. Para un mensaje de 5850 caracteres, el almacenamiento necesario se incrementa a 254,160 bytes.

- **Eficiencia del Almacenamiento:**
  - La diferencia considerable entre el almacenamiento del texto plano y el texto cifrado resalta la sobrecarga inherente en el cifrado RSA. Esta sobrecarga es una característica importante a considerar en aplicaciones prácticas, especialmente en sistemas con limitaciones de almacenamiento.

5.2.4. Firmas Digitales RSA

La firma digital RSA se genera cifrando un hash del mensaje con la llave privada del firmante, y se verifica descifrando la firma con la llave pública del firmante y comparando el resultado con el hash del mensaje.

En esta sección se presentan los resultados de las pruebas de rendimiento realizadas para evaluar el tiempo de ejecución y el almacenamiento necesario al firmar y validar mensajes de diferentes longitudes utilizando el Algoritmo 17.

La Tabla 5.10 muestra los tiempos de ejecución y los requisitos de almacenamiento tanto para la firma digital como para las llaves pública y privada.

Tabla 5.10:
Tiempos de Ejecución y Almacenamiento de Firma Digital con RSA

Almacenamiento (*bytes*)			**Tiempo de Ejecución (*seg*)**		
Llave Pública	**Llave Privada**	**Firma**	**Número de caracteres**	**Firma**	**Validación**
618	618	618	100	0.0020158	0.0021645
		618	650	0.0024899	0.0015624
		619	1300	0.001807	0.001762
		618	1950	0.0022154	0.0015554
		620	2600	0.0018279	0.0018739
		619	3250	0.0017259	0.0014527
		618	3900	0.0020981	0.0023099
		618	4550	0.0024051	0.0022389
		619	5200	0.0020075	0.0015553
		618	5850	0.0025382	0.0024926

Nota: Elaboración propia.

Tiempo de Ejecución

- **Firma:**
  - El tiempo de generación de la firma muestra variaciones leves pero en general se mantiene en el mismo rango para diferentes tamaños de mensajes. Para mensajes de 100 caracteres, el tiempo de firma es de aproximadamente 0.0020158 segundos, mientras que para mensajes de 5850 caracteres, el tiempo es de 0.0025382 segundos.
- **Validación:**
  - El tiempo de validación de la firma también presenta ligeras variaciones con el tamaño del mensaje. Para mensajes de 100 caracteres, el tiempo de validación es de aproximadamente 0.0021645 segundos, y para mensajes de 5850 caracteres, es de 0.0024926 segundos.

Almacenamiento

- **Firma:**
  - El almacenamiento requerido para las firmas varía ligeramente, oscilando entre 618 y 620 bytes. Este tamaño relativamente pequeño es una ventaja significativa en términos de eficiencia de almacenamiento.
- **Llave Pública y Llave Privada:**
  - El tamaño de las llaves pública y privada se mantiene constante en 618 bytes para todos los tamaños de mensajes. Este comportamiento es esperable dado que el tamaño de la llave RSA no depende del mensaje, sino de la longitud de la clave utilizada en la criptografía.

5.2.5. Comparativa entre RSA y ECC

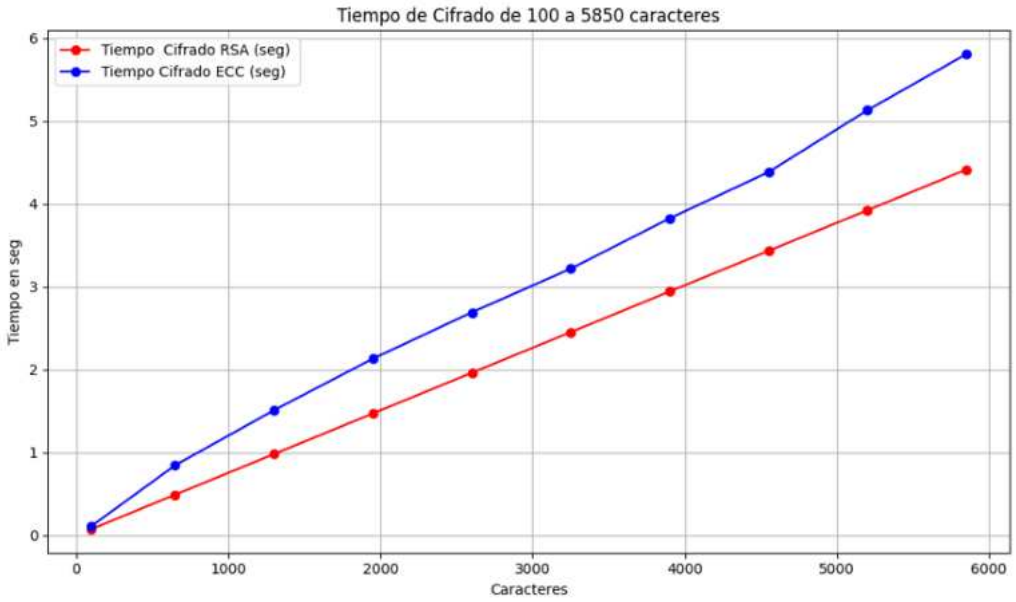
En esta redacción se realiza una comparativa entre los algoritmos RSA (Rivest-Shamir-Adleman) y ECC (Elliptic Curve Cryptography), centrándonos en la eficiencia, el tiempo de ejecución, el almacenamiento y la viabilidad en la industria, incluyendo su posible uso en el Sistema de Administración Tributaria (SAT) para firmas digitales, y tomando en cuenta las normas de seguridad aplicables.

RSA utiliza llaves que son generalmente largas, con tamaños típicos de 2048 bits o más para proporcionar una seguridad adecuada conforme a las recomendaciones del Instituto Nacional de Estándares y Tecnología (NIST) y otras organizaciones internacionales de estándares. Esto resulta en mayores requisitos de almacenamiento y más tiempo de procesamiento. En contraste, ECC emplea llaves significativamente más pequeñas, ofreciendo la misma seguridad que RSA pero con tamaños de llaves mucho menores (por ejemplo, una llave ECC de 256 bits es comparable en seguridad a una llave RSA de 3072 bits). Las normas de seguridad como FIPS 186-4 y el estándar ISO/IEC 15946 avalan el uso de ECC en aplicaciones de criptografía moderna. Esto implica menores requerimientos de almacenamiento y más rapidez en las operaciones.

Los tiempos de cifrado mostrados en la Figura 5.9 y descifrado en la Figura 5.10 reflejan diferencias importantes entre RSA y ECC. El cifrado con RSA es considerablemente más rápido, sin embargo, el descifrado con ECC es más rápido gracias a la eficiencia de las operaciones con curvas elípticas y el menor tamaño de las llaves. Por otro lado, las firmas digitales mostradas en las Figuras 5.11 y 5.12 indican que, aunque RSA es ampliamente utilizado para firmas digitales, el proceso de generación y validación de firmas puede ser más lento en comparación con ECC debido a los mayores tamaños de las llaves. ECC permite

una generación y validación de firmas digitales más rápida y eficiente, lo cual es especialmente beneficioso en entornos donde el rendimiento y la velocidad son esenciales.

Figura 5.9:
Tiempo de ejecución de Cifrado con ECC y RSA

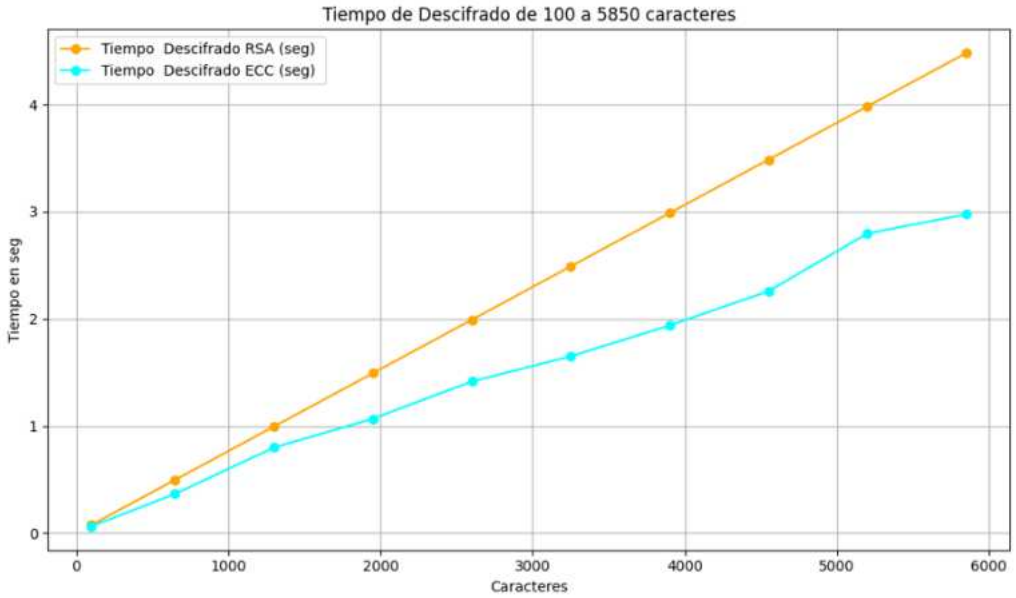


Nota: Elaboración propia.

En cuanto a los resultados obtenidos de las pruebas de tiempo y almacenamiento, se observan diferencias significativas. Para RSA, las llaves pública y privada tienen un tamaño constante de 618 bytes, mientras que en ECC, las llaves pública y privada son de 157 y 77 bytes respectivamente. Los tiempos de ejecución también reflejan esta diferencia en eficiencia. En la Tabla 5.11 se presentan los tiempos de ejecución y los requisitos de almacenamiento tanto para RSA como para ECC. Cabe aclarar que estas pruebas no comparan la seguridad de los algoritmos, sino que están enfocadas únicamente en medir la eficiencia en términos de tiempos de ejecución y requisitos de almacenamiento.

RSA sigue siendo muy utilizado en la industria debido a su robustez y la

Figura 5.10:
Tiempo de ejecución de Descifrado con ECC y RSA



Nota: Elaboración propia.

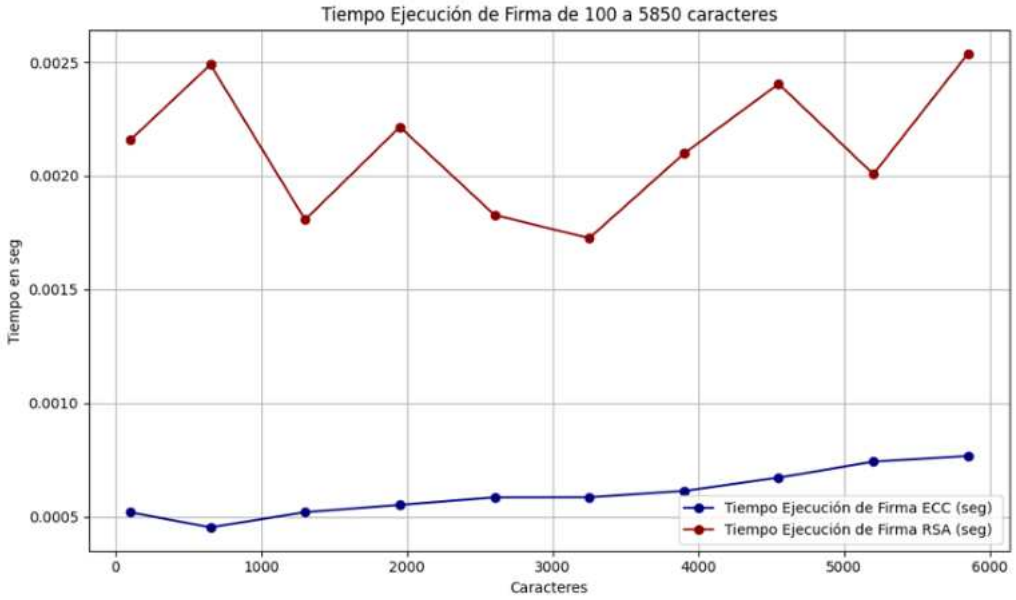
familiaridad que los desarrolladores y las infraestructuras tienen con este algoritmo. Sin embargo, su eficiencia disminuye a medida que las llaves deben ser más largas para mantener la seguridad. ECC está ganando popularidad rápidamente debido a sus ventajas en términos de eficiencia y menores requerimientos de almacenamiento. Las menores llaves y la rapidez en el procesamiento hacen que ECC sea una opción atractiva para dispositivos con recursos limitados, como dispositivos móviles y IoT.

Tabla 5.11:
Comparativa de Tiempos de Ejecución y Almacenamiento entre RSA y ECC

Almacenamiento (bytes)			**Tiempo de Ejecución (seg)**			
Llave Pública	**Llave Privada**	**Firma**	**Cifrado**	**Descifrado**	**Firma**	**Validación**
RSA						
618	618	618	0.076584	0.076584	0.0020158	0.0021645
ECC						
157	77	156	0.111709	0.0637211	0.0005187	0.0004531

Nota: Elaboración propia.

Figura 5.11:
Tiempo de ejecución de Firma Digital con ECC y RSA



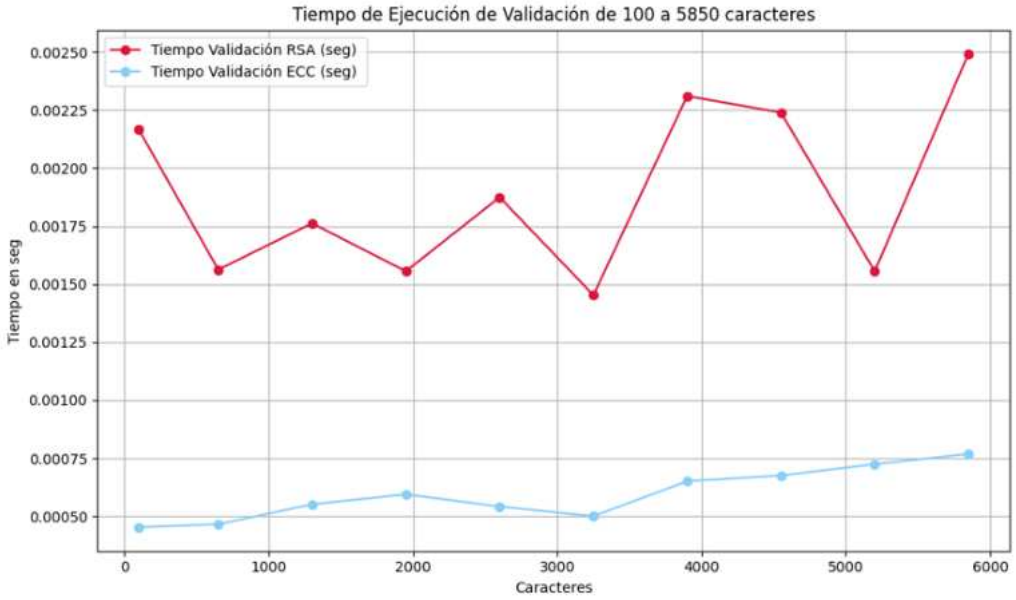
Nota: Elaboración propia.

El SAT actualmente utiliza RSA para sus firmas digitales, lo que garantiza seguridad robusta pero con ciertos costos en términos de tiempo de procesamiento y almacenamiento. Implementar ECC podría ofrecer varias ventajas:

1. **Reducción de Almacenamiento:** Al utilizar llaves más pequeñas, ECC puede reducir significativamente los requisitos de almacenamiento. 2. **Aumento en la Velocidad:** ECC permite tiempos de cifrado, descifrado, firma y validación más rápidos, lo que podría mejorar la eficiencia general de las operaciones del SAT. 3. **Escalabilidad:** La eficiencia de ECC facilita la escalabilidad en sistemas grandes, permitiendo manejar más transacciones sin comprometer el rendimiento.

En los resultados obtenidos se observa que los tiempos de ejecución de ECC para firmas y validación son significativamente menores. Por ejemplo, en la firma de 100 caracteres, ECC toma aproximadamente 0.0005187 segundos frente

Figura 5.12:
Tiempo de ejecución de Validación de Firma Digital con ECC y RSA



Nota: Elaboración propia.

a los 0.0020158 segundos de RSA, lo que representa una reducción en el tiempo de ejecución de alrededor del $74.3\,\%$. Similarmente, en la validación de firmas, ECC reduce el tiempo de 0.0021645 segundos a 0.0004531 segundos, una mejora del $79.1\,\%$.

Estas cifras demuestran que ECC no solo es más eficiente en términos de almacenamiento, sino también en rendimiento, proporcionando una alternativa viable y mejorada frente a RSA para aplicaciones que requieren alta eficiencia y rapidez.

”No tengo talentos especiales, pero sí soy profundamente curioso.”

Albert Einstein (1879-1955)
Físico teórico alemán

CAPÍTULO 6

Conclusiones y Trabajos Futuros

6.1. Conclusiones

La realización de este proyecto se originó a partir de una curiosidad inicial por entender y mejorar los sistemas criptográficos utilizados en diversas aplicaciones de seguridad, especialmente en el contexto del Servicio de Administración Tributaria (SAT) de México. La motivación principal fue abordar las limitaciones y desafíos que presentan los algoritmos de criptografía tradicionales, como RSA, en términos de eficiencia y seguridad. A través del estudio y la implementación de criptografía basada en curvas elípticas (ECC), se buscó no solo comprender mejor estos mecanismos, sino también proponer una solución más eficiente y segura.

La investigación se centró en la implementación de un sistema criptográfico basado en curvas elípticas para la generación y validación de firmas digitales, comparando su desempeño con el algoritmo RSA. Los resultados obtenidos a lo largo del desarrollo del proyecto demostraron que ECC ofrece ventajas significativas en términos de eficiencia computacional y seguridad. En particular, se observó que las llaves generadas con ECC son más pequeñas en tamaño comparadas con RSA, lo que implica una reducción en los costos de almacenamiento y una mejora en la velocidad de procesamiento.

El proceso de desarrollo incluyó la definición y análisis de requerimientos, diseño de sistema, implementación, pruebas unitarias e integrales, y finalmente la validación de la solución propuesta. Cada etapa del proyecto fue llevada a cabo con rigurosidad, aplicando metodologías de diseño y desarrollo de software que aseguraron la calidad y fiabilidad del sistema implementado. El uso del lenguaje de programación Mathematica® facilitó la implementación y análisis de los algoritmos criptográficos, permitiendo realizar cálculos y visualizaciones necesarios para la validación de los resultados.

En términos de desempeño, los algoritmos basados en ECC demostraron ser superiores a RSA en múltiples aspectos. La generación de llaves, cifrado, descifrado y la creación y validación de firmas digitales fueron procesos más rápidos y eficientes utilizando ECC. Estas mejoras son especialmente relevantes en entornos donde el tiempo de procesamiento y la capacidad de almacenamiento son críticos, como es el caso de los sistemas tributarios y financieros.

El proyecto logró cumplir con los objetivos planteados, que incluían demostrar la viabilidad de utilizar ECC para la generación y validación de firmas digitales, así como comparar su rendimiento con RSA. Los resultados mostraron que ECC puede reducir el tiempo de ejecución de al menos un $50\,\%$, tal como se planteó en la hipótesis. Sin embargo, las pruebas realizadas superaron esta expectativa, mostrando reducciones en el tiempo de ejecución superiores al $70\,\%$ en algunos casos. Por ejemplo, en la firma de 100 caracteres, ECC tomó aproximadamente 0.0005187 segundos frente a los 0.0020158 segundos de RSA, representando una reducción del $74.3\,\%$ en el tiempo de ejecución. En la validación de firmas, ECC redujo el tiempo de 0.0021645 segundos a 0.0004531 segundos, mejorando en un $79.1\,\%$.

Es importante destacar que en ambos casos, tanto para ECC como para

RSA, se utilizó una función hash para generar un resumen del mensaje antes de proceder con la firma. Para ambos sistemas se utilizó SHA-256, lo que garantiza la integridad del mensaje y la seguridad del proceso de firma.

En cuanto a los resultados obtenidos de las pruebas de tiempo y almacenamiento, se observan diferencias significativas. Para RSA, las llaves pública y privada tienen un tamaño constante de 618 bytes, mientras que en ECC, las llaves pública y privada son de 157 y 77 bytes respectivamente. Los tiempos de ejecución también reflejan esta diferencia en eficiencia. En la Tabla 5.11 se presentan los tiempos de ejecución y los requisitos de almacenamiento tanto para RSA como para ECC.

Cabe mencionar que las pruebas realizadas no se enfocan en comparar la seguridad de los algoritmos, sino únicamente en medir la eficiencia en términos de tiempos de ejecución y almacenamiento.

Además, las pruebas de almacenamiento confirmaron que las llaves de ECC, siendo significativamente más pequeñas que las de RSA, proporcionan una mayor eficiencia en términos de espacio utilizado. Las llaves públicas y privadas de ECC son de 157 y 77 bytes respectivamente, en contraste con las llaves RSA de 4 bytes cada una. Esto no solo reduce los requisitos de almacenamiento sino también facilita el manejo y la transmisión de las llaves. La reducción en el almacenamiento puede alcanzar hasta un $75\,\%$ en comparación con RSA, lo cual tiene un impacto significativo en la infraestructura de almacenamiento necesaria para manejar grandes volúmenes de datos.

Esta mejora en la eficiencia no solo es beneficiosa desde una perspectiva técnica, sino también económica. Para una institución como el SAT, que maneja un gran volumen de transacciones y datos diariamente, la adopción de ECC

podría traducirse en ahorros sustanciales en términos de recursos de hardware y costos operativos. La reducción del tiempo de procesamiento permite manejar más transacciones en menos tiempo, mejorando la productividad y la capacidad de respuesta del sistema. Además, la disminución en los requisitos de almacenamiento reduce los costos asociados con el mantenimiento de servidores y sistemas de almacenamiento.

La implementación de ECC en el SAT podría facilitar una mayor escalabilidad y robustez en sus operaciones, permitiendo gestionar un mayor volumen de transacciones sin comprometer la seguridad ni la eficiencia. Además, al adoptar estándares modernos y más eficientes, el SAT podría posicionarse como una institución pionera en el uso de tecnologías avanzadas de criptografía, mejorando la confianza y seguridad en sus servicios.

Tabla 6.1:
Comparativa del Estado del Arte con la Propuesta ECC

Método	Llave Pública (bytes)	Llave Privada (bytes)	Firma (bytes)	Cifrado (seg)	Descifrado (seg)	Firma (seg)	Validación (seg)
ECC	157	77	156	0.111709	0.0637211	0.0005187	0.0004531
RSA	618	618	618	0.076584	0.076584	0.0020158	0.0021645
ElGamal	256	256	256	0.090123	0.090456	0.003123	0.003567
DLP	512	512	512	0.120789	0.120456	0.004567	0.004890

En la Tabla 6.1 observamos una comparativa y ventajas en cuanto a otros criptosistemas mencionados en el estado del arte contra la propuesta de ECC. En la comparación entre ECC y otros métodos criptográficos en firmas digitales, se observan las siguientes ventajas:

Tamaño de Llaves Más Pequeño: Las llaves públicas y privadas de ECC son significativamente más pequeñas que las de RSA y otros métodos tradicionales como ElGamal y DLP. Esto reduce los requisitos de almacenamiento y facilita la transmisión de las llaves.

Eficiencia en el Tiempo de Ejecución: ECC demuestra ser más eficiente

en términos de tiempo de generación y validación de firmas en comparación con RSA. Los tiempos de ejecución más cortos permiten un procesamiento más rápido de transacciones y datos.

6.2. Trabajos Futuros

Dado el potencial de ECC demostrado en este estudio, se proponen las siguientes líneas de investigación y desarrollo para futuros trabajos:

- **Implementación en Sistemas Reales:** Desarrollar y probar implementaciones de ECC en sistemas reales, como plataformas de comercio electrónico y aplicaciones móviles, para evaluar su rendimiento y seguridad en entornos prácticos.
- **Optimización de Algoritmos:** Investigar y desarrollar técnicas avanzadas de optimización para los algoritmos de ECC que puedan mejorar aún más su eficiencia y reducir el tiempo de ejecución.
- **Comparativa con Otros Algoritmos:** Realizar estudios comparativos de ECC con otros algoritmos criptográficos modernos, como los basados en criptografía cuántica, para evaluar sus ventajas y desventajas relativas.
- **Estudio de la Seguridad Post-Cuántica:** Investigar la seguridad de ECC en el contexto de la criptografía post-cuántica, asegurando que las implementaciones actuales sean resistentes a ataques futuros posibles con computadoras cuánticas.
- **Aplicaciones en IoT y Dispositivos Móviles:** Explorar el uso de ECC en dispositivos con recursos limitados, como dispositivos IoT y móviles, donde

la eficiencia en el uso de recursos es crítica.

Bibliografía

Aljebreen, E. A. A. P. B. F. M. S. M. M. (2022). Secure sensitive data sharing using rsa and elgamal cryptographic algorithms with hash functions. *Information*.

Alkhudhayr, F., Moulahi, T., and Alabdulatif, A. (2021). Evaluation study of elliptic curve cryptography scalar multiplication on raspberry pi4. *IJACSA) International Journal of Advanced Computer Science and Applications*, 12:472–479.

Biddle, G., McGoldrick, L., and Halamek, J. (2021). Non-traditional encryption methods: Moving towardelectrochemical cryptography. *Electrochemical Science Advances*, pages 1–7.

Chávez, M. A. L. and Henríquez, F. R. (2021). Post-quantum digital signature for the mexican digital invoices by internet. *Computacion y Sistemas*, 25.

Diffie, W. and Hellman, M. E. (1976). New directions in cryptography invited paper. *IEEE Transactions on Information Theory*, IT-22:644–654.

DOF (2017). Anexo 20 de la segunda resolución de modificaciones a la resolución miscelánea fiscal para 2017.

Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *Advances in Cryptology*, pages 10–18.

Fagin, B. (2018). Composite numbers that give valid rsa key pairs for any coprime p. *Information (Switzerland)*, 9.

Gidney, C. and Ekerå, M. (2021). How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5.

Huguet Rotger. L., Rifà. Coma, J. and Tena Ayuso, J. (2017). riptografía con curvas eípticas. *FUOC. Fundació per a la Universitat Oberta de Catalunya, 68.*

Imam, R., Areeb, Q. M., Alturki, A., and Anwer, F. (2021). Systematic and critical review of rsa based public key cryptographic schemes: Past and present status.

Ismail, N. H. and Misro, M. Y. (2022). Bézier coefficients matrix for elgamal elliptic curve cryptosystem. *Malaysian Journal of Mathematical Sciences*, 16.

Johnson, D., Menezes, A., and Vanstone, S. (2021). The elliptic curve digital signature algorithm (ecdsa).

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics Of Computation*, 48:203–209.

Lizy, R. F. S. and Raj, V. J. (2021). Improvement of rsa algorithm using euclidean technique. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12.

Maletsky, K. (2015). Rsa vs ecc comparison for embedded systems. *Atmel-8951A-CryptoAuth-RSA-ECC-Comparison-Embedded-Systems-WhitePaper*$_0 72015.$

Mendoza, D. A. O. (2021). Criptografía con curvas elípticas aplicada a la firma digital.

Miller, V. S. (1986). Use of elliptic curves in cryptography. *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426.

Mohamad, M. S. A., Din, R., and Ahmad, J. I. (2021). Research trends review on rsa scheme of asymmetric cryptography techniques. *Bulletin of Electrical Engineering and Informatics*, 10.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). Programming techniques a method for obtaining digital signatures and public-key cryptosystems. *Comunications of the ACM*, 21:120–12.

Rojas, D. A. P. and Meneses, A. V. R. (2000). Método criptográfico rsa y algoritmo de shor.

Rouse, M., Rosencrance, L., and Cobb, M. (2019). What is asymmetric cryptography?

Saho, N. J. G., Ezin, E. C., Saho, N. G. J., Ezin, E. C., Watson, B., Badouel, E., and Niang, O. (2020). Comparative study on the performance of elliptic curve cryptography algorithms with cryptography through rsa algorithm.

Schneider, C. D. C. and Vaira, S. M. (2019). Protocolos criptográficos basados en los algoritmos de diffie-hellman y rsa.

Seo, J. H. (2020). Efficient digital signatures from rsa without random oracles. *Information Sciences*, 512.

Trappe, W. and Washington, L. C. (2006). *Introduction to Cryptography with Coding Theory*. 2 edition.

Wang, H., He, D., and Ji, Y. (2020). Designated-verifier proof of assets for bitcoin exchange using elliptic curve cryptography. *Future Generation Computer Systems*, 107:854–862.