DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: ESTUDIO COMPARATIVO DE ALGORITMOS DE FIRMA DIGITAL UTILIZANDO CRIPTOGRAFÍA ASIMÉTRICA RSA Y CRIPTOGRAFÍA BASADA EN CURVAS ELÍPTICAS

AUTOR: AYALA JACHO, WENDY ARACELLY

DIRECTOR: ING. GALÁRRAGA, FERNANDO

CODIRECTOR: ING. CAIZAGUANO, CARLOS

SANGOLQUÍ

2016

Sangolquí, 26 de noviembre de 2015

CERTIFICADO

Certificamos que el presente proyecto titulado "Estudio comparativo de algoritmos de firma digital utilizando criptografía asimétrica RSA y criptografía basada en curvas elípticas", fue desarrollado en su totalidad por el la Srta. Wendy Aracelly Ayala Jacho, bajo nuestra dirección y cumple con los objetivos propuestos.

Ing. Fernando Galárraga

Ing. Carlos Caizaguano

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORÍA DE RESPONSABILIDAD

Yo, **AYALA JACHO WENDY ARACELLY**, con cédula de identidad N° 1718334889, declaro que este trabajo de titulación "**ESTUDIO COMPARATIVO DE ALGORITMOS DE FIRMA DIGITAL UTILIZANDO CRIPTOGRAFÍA ASIMÉTRICA RSA Y CRIPTOGRAFÍA BASADA EN CURVAS ELÍPTICAS**" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 26 de Mayo del 2016

WENDY ARACELLY AYALA JACHO
C.C 1718334889

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Yo, **AYALA JACHO WENDY ARACELLY**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "**ESTUDIO COMPARATIVO DE ALGORITMOS DE FIRMA DIGITAL UTILIZANDO CRIPTOGRAFÍA ASIMÉTRICA RSA Y CRIPTOGRAFÍA BASADA EN CURVAS ELÍPTICAS**" cuyo contenido, ideas, y criterios son de mi autoría y responsabilidad.

Sangolquí, 26 de Mayo del 2016

WENDY ARACELLY AYALA JACHO
C.C 1718334889

DEDICATORIA

Dedico este trabajo a mi Madre por el gran apoyo que ha significado su presencia en la cimentación de mi vida personal y profesional, por inculcarme valores de responsabilidad y deseos de superación que han hecho este trabajo posible. Gracias por ser un gran ejemplo de esfuerzo y permanecer conmigo en tantas noches de desvelos. Este triunfo es tuyo mamá.

A mis amigos y familiares quienes de alguna u otra manera supieron apoyarme en este proceso.

AGRADECIMIENTO

Agradezco a todos los Ingenieros a los que tuve el gusto de conocer desde los inicios de mi carrera, por su conocimiento impartido, en especial a mi Director de tesis el Ing. Fernando Galárraga y a mi codirector Ing. Carlos Caizaguano.

A la Universidad de las Fuerzas Armadas “ESPE”, por haberme acogido en sus aulas durante el transcurso de mi formación académica.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente trabajo se realiza el estudio comparativo de algoritmos de firma digital entre el esquema criptográfico asimétrico RSA (Rivest, Shamir y Adleman) y el esquema criptográfico basado en curvas elípticas ECDSA (Elliptic Curve DSA). La necesidad de establecer canales de comunicación más seguros en un mundo donde el desarrollo tecnológico avanza a pasos agigantados es indispensable, aún más con la aparición de las computadoras cuánticas donde la capacidad de cálculo ha reducido los tiempos de descifrado de años a días; es por ello que la implementación de algoritmos criptográficos, cuyo descifrado sea más complejo, mejora las posibilidades del resguardo de la información. Por lo tanto, en este proyecto, se analiza los fundamentos matemáticos de los algoritmos de firma digital asimétricos como ElGamal, Rabin, Fiat Shamir, Shor y de curvas elípticas como Diffie-Hellman, ElGamal, Menezes-Vanstone, dando un enfoque especial a los algoritmos de firma electrónica RSA y ECDSA. Los algoritmos en mención, han sido codificados en el ambiente de desarrollo Java, con sus respectivas librerías, con el fin de establecer el algoritmo RSA o el algoritmo ECDSA posee las mejores características para el respaldo de información, considerando la arquitectura del lenguaje de programación. Se ha tomado en cuenta los estándares propios de los algoritmos de firma electrónica asimétricos y de curvas elípticas para que la comparación sea adecuada y arroje resultados que además de medibles sean sustentables. Los resultados obtenidos en este estudio se los visualiza a través de gráficos estadísticos, producto de la cuantificación de los tiempos de respuesta obtenidos en el proceso.

PALABRAS CLAVES:

- **CRIPTOGRAFÍA**
- **FIRMA DIGITAL**
- **CURVAS ELÍPTICAS**
- **RSA**
- **ECDSA**

ABSTRACT

This work is about a comparative study of digital signature algorithms between the asymmetric cryptographic scheme RSA (Rivest, Shamir and Adelman) and the cryptographic scheme based on elliptic curves ECDSA (Elliptic Curve DSA). It is necessary and indispensable to establish safer channels of communication around the world because of the big technological development, especially with the advent of quantum computers where capacity has reduced calculation time decryption from years to days; for that reason the implementation of cryptographic algorithms whose decryption is more complex improves the chances to keep the information safe. The content of this project analyses mathematical foundations of asymmetric digital signature algorithms such as ElGamal, Rabin, Shamir Fiat, Shor and elliptic curves as Diffie-Hellman, ElGamal, Menezes-Vanstone are analyzed, with special focus on RSA and ECDSA respectively. The algorithms have been codified into Java development environment with their respective libraries in order to establish which of them has the best features for data backup; for that reason the architecture of language was considered. Also as part of the study, it has taken in count the levels of security, number of bits according to the level of security and algorithm's own standards in order to have sustainable and measurable results. The form of analysis of the study is developed using Stat Graphs as a product of quantification of response time obtained along the whole process.

KEY WORDS:

- **CRIPTOGRAPHY**
- **DIGITAL SIGNATURE**
- **ELLIPTIC CURVES**
- **RSA**
- **ECDSA**

CAPÍTULO I

INTRODUCCIÓN

1.1. Antecedentes

Los sistemas criptográficos se han transformado en una ciencia de gran importancia, debido a la ne

cesidad de proteger la información, más aún en el medio actual donde se recurre constantemente a las redes de comunicación, las cuales incluyen desde el envío de mensajes de datos, hasta transacciones bancarias y donde el método utilizado con mayor frecuencia para proporcionar autenticidad e integridad es la firma digital. La intrusión en la información podría significar pérdidas de tipo económico, alteración del mensaje transmitido, o suplantación de la identidad del usuario.

La inseguridad de la información existe desde que se inventó la escritura, y se propagó por los diferentes canales de comunicación como lo son el papel, el telégrafo, el teléfono, la radio o las modernas redes de ordenadores. Años atrás la criptografía era casi exclusivamente de los gobiernos y mando militares, lo que ocasionó que esta herramienta sea conocida sólo por unos pocos. (Jesús J. Ortega Triguero, 2006). Sin embargo en la actualidad existe una lucha entre aquellos que intentan salvaguardar la información a través del desarrollo e implementación de métodos criptográficos y quienes quieren inutilizarlos.

La criptografía se basa en problemas matemáticos de difícil solución, donde hace uso de la aritmética modular trabajando con números enteros, (Tilborg, 2005) mas con el desarrollo de nuevas tecnologías estos problemas pueden ser resueltos de maneras más eficientes, por lo que se necesita de la búsqueda de nuevos algoritmos que resguarden la información a pesar de su avance.

Existen varios métodos de criptografía, uno de ellos es el algoritmo RSA (Rivest, Shamir y Adleman), mismo que al ser el más usado actualmente, es el más compatible. Para que sea segura la longitud de sus claves (una pública y otra privada) debe de ser de 1024 bits, es decir un número de un poco más de 300 dígitos. (William Raúl García Muñoz, 2005). Por otro lado, el estándar basado en curvas elípticas reduce hasta en 164 bits (aproximadamente 45 dígitos) la longitud de las claves, sin embargo es un algoritmo relativamente nuevo, cuya compatibilidad aún no se encuentra estudiada completamente.

Por lo expuesto, el estudio de ambos algoritmos, es necesario para determinar el mejor rendimiento en el uso de firmas digitales y cuáles serán sus beneficios al ser implementados.

1.2. Planteamiento del Problema

La información transmitida es de carácter sensible, por lo tanto, es preciso contar con un esquema criptográfico que garantice la confidencialidad de la información. En el caso de las firmas digitales se debe ser aún más riguroso ya que deben aseverar la identidad del firmante y verificar la integridad del mensaje de datos.

La seguridad del algoritmo asimétrico de clave pública RSA (Rivest, Shamir y Adleman) se encuentra en el problema de factorización en números enteros primos. (Koblintz, 1994), sin embargo con el avance de nuevas tecnologías y el desarrollo de la computación cuántica, se generan formas más rápidas de descomponer un número grande en producto de primos, lo cual representa una falla en la seguridad para este tipo de algoritmo.

Al momento, los criptosistemas de curvas elípticas están siendo estudiados de manera rigurosa, sin embargo, no se presentan argumentos

formales que puedan rebatir las principales características que los convierten en la opción asimétrica más deseada en la actualidad. (Trujillo, 2011)

1.3. Justificación

Para entender cómo funciona una firma digital, se plantea el siguiente ejemplo: David quiere enviar un mensaje a Sara, pero existe una tercera persona llamada Marco que quiere alterar la comunicación de las personas antes mencionadas, entones se necesita de un método que permita que a pesar de que la información sea sustraída, sólo sea entendible para Sara y David. Entonces, se resume el mensaje de David mediante función hash, se cifra este resultado con su clave privada para obtener su firma digital y enviar el mensaje original con la firma digital a Sara. Una vez que Sara haya recibido el mensaje junto con la fina digital, deberá comprobar su validez para saber que el mensaje es el correcto y que efectivamente pertenece a David (integridad y autenticación). El proceso correspondiente consiste en descifrar el resumen del mensaje mediante la clave pública de Ana, también aplicar al mensaje la función hash para obtener un segundo resumen, y así comparar el resumen recibido son el obtenido a partir de la función hash. Si son iguales, Sara podrá asegurar que la persona que envió el mensaje se trata de David y que éste no ha sido alterado.

La criptografía intenta resolver el problema anterior basándose en la aplicación de criptosistemas entre los cuales se encuentran el algoritmo RSA y ECDSA, mismos que serán analizados en el presente documento.

Y es por ello que, es necesario el desarrollo de nuevos algoritmos que permitan garantizar la integridad de los mensajes de datos, como lo es el algoritmo basado en curvas elípticas (ECC), mas su desventaja radica en que no se sabe qué curva es óptima para cierto valor.

Siendo así, se comprueba que tanto el algoritmo RSA como ECC, poseen fortalezas y debilidades; por ello el presente estudio se enfocará en

determinar cuál de ellos es más eficiente en lo que respecta a firmas digitales y cuál responderá de mejor manera ante el avance de las nuevas tecnologías.

1.4. Objetivos

1.4.1. Objetivo General

Realizar el estudio comparativo de algoritmos de firma digital utilizando criptografía asimétrica RSA y criptografía basada en curvas elípticas.

1.4.2. Objetivos Específicos

- Analizar la teoría de la complejidad de los criptosistemas asimétricos y de curvas elípticas.
- Describir las propiedades necesarias para el uso de funciones hash que garanticen una criptografía útil.
- Formalizar los fundamentos matemáticos soportados en los algoritmos de firma digital RSA y ECDSA.
- Implementar los algoritmos que permitan la generación de firmas digitales.

1.5. Alcance

El presente proyecto tendrá como finalidad establecer los resultantes del estudio y análisis comparativo de algoritmos de firma electrónica, utilizando criptografía RSA y ECDSA; para lo que se realizará una investigación acerca del marco teórico y práctico de los mismos, con el fin de aclarar los conocimientos y establecer los puntos de comparación a efectuarse, además de conocer, los elementos matemáticos que posibiliten estructurar un criptosistema.

El proyecto incluirá el desarrollo de un aplicativo que permita implementar los algoritmos RSA y ECDSA, que vayan de acuerdo a los principales estándares internacionales relacionados con la telemática; cuyo fin será, realizar las debidas comparaciones entre los algoritmos

Parte de la medición y análisis a realizarse consta de, estadísticas que ratifiquen, cuál de los algoritmos implementados es el más adecuado y cuál se adaptará de mejor manera ante los avances de las nuevas tecnologías.

CAPÍTULO II

MARCO TEÓRICO

2.1. Criptografía Teórica

Criptografía es un proceso de diseño de sistemas que se basa en métodos matemáticos que permiten mantener una comunicación fehaciente en canales que no lo son.

La criptografía posee diversos tipos de sistemas criptográficos con los que se custodia ciertos principios básicos de la seguridad informática, conocidos por otros autores como propiedades criptográficas, los mismos que permiten asegurar a confidencialidad, integridad, autentificación y no repudio de la información.

a) Confidencialidad.

El sistema será accedido únicamente por los usuarios que posean la autorización correspondiente. Manteniendo el secreto de la información para los que no la poseen. Si un mensaje cifrado fuera interceptado por un intruso como lo muestra la Figura 1, se cumpliría el principio de confidencialidad ya que el mensaje no podría ser entendido, más si el mensaje no fuese cifrado la información fuera fácilmente sustraída.

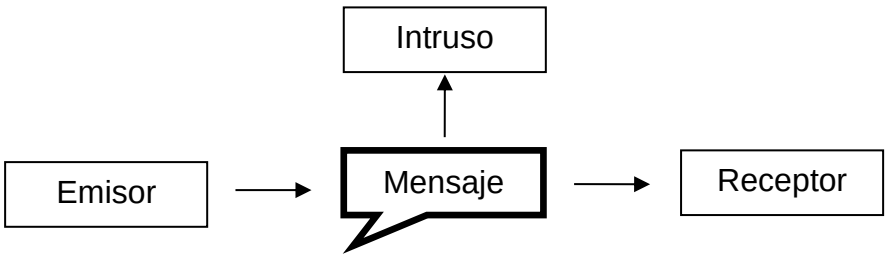

Figura 1 Interrupción del mensaje

b) Integridad.

Los componentes del sistema podrán ser instaurados o alterados por los usuarios que posean la autorización correspondiente. Si un intruso logra interceptar el mensaje y enviar uno diferente al receptor, como se observa en la Figura 2, este principio se vería afectado.

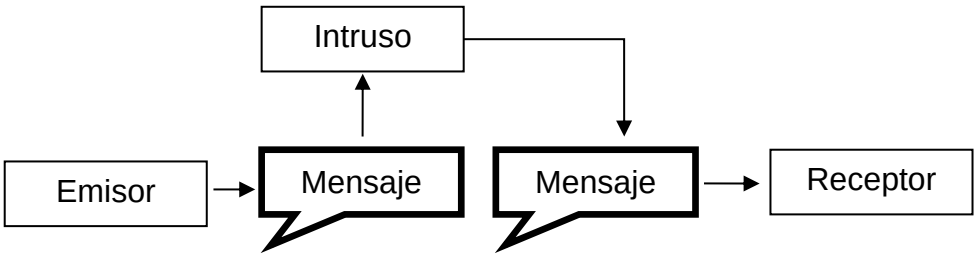

Figura 2 Sustitución del mensaje

c) Disponibilidad.

Los componentes del sistema deber poder ser accedidos cuando los usuarios así lo soliciten.

d) Autenticación.

La identidad de las partes implicadas en un proceso será verificada y asegurada. Si el intruso suplanta la identidad del emisor y envía información alterada como se observa en la Figura 2, este principio se vería afectado.

e) No repudio.

Hace referencia a la aceptación de los protocolos de comunicación entre el emisor y el receptor, es decir que, el emisor no debe negar la transmisión del mensaje.

En el proceso de comunicación cifrada existen dos tipos de actores el primero es el criptógrafo quien se encarga de generar algoritmos que protejan la información y el segundo que se vale de técnicas y metodologías conformando así el proceso denominado como criptoanálisis. Y a la

combinación de criptografía y criptoanálisis se le conoce como criptología. (Pino Caballero Gil)

El resguardo de los datos se consigue a través de procesos como el cifrado el cual consiste en transformar un texto original a un texto cifrado o criptograma, de forma inversa el proceso de transformar del texto cifrado al texto original se lo conoce como descifrado.

Existen diversos tipos de cifrado uno de ellos es el cifrado digital donde los textos se conforman de un compendio finito llamada alfabeto; el cifrado análogo por su parte genera una colección continua de valores y es por ello que dicho cifrado no es del todo seguro.

Para resguardar la seguridad del texto es de suma importancia mantener una serie de reglas, mismas que fueron dadas por Kerckhoffs en el siglo XIX en su obra titulada La criptografía Militar y se mencionan a continuación:

- No se debe recuperar el texto original mediante el criptograma.
  Si el proceso de cifrado se realizó adecuadamente esta regla será cumplida sin embargo se debe tomar en cuenta que ningún cifrado puede ser demasiado complejo para que no afecte al proceso de descifrado.
- En los sistemas criptográficos se incluirán información de tipo:
  i. Pública, como lo son los algoritmos que lo precisan.
  ii. Privada, como la clave que se utiliza para el cifrado.
- La forma de establecer la clave debe ser fácil de recordar y modificar.
- El criptograma debe poder enviarse a través de los diferentes canales de comunicación.
- El costo de descifrado debe ser equivalente al costo de su contenido.

2.2.1 Métodos de Transposición y sustitución

Existen algoritmos de criptografía clásicos, tales como el método de transposición y sustitución; los mismos que han sucumbido ante la velocidad de cálculo de la computación actual.

2.2.1.1. Transposición

Este método radica en alterar el orden de las unidades de mensaje original según una clave la cual debe ser secreta y tan solo compartida entre el emisor y receptor. En las claves de transposición el mensaje se escribe sin separación entre palabras, en filas de letras organizadas en tal forma que simula un bloque rectangular. Las letras se van transponiendo según un orden convenido previamente, por ejemplo, por columnas verticales, diagonales o espirales, o a través sistemas de mayor complejidad. El orden de las letras en el texto cifrado depende del tamaño del bloque utilizado y de la vía utilizada para apuntar y transponer las letras. Con motivos de incrementar la seguridad de la clave se puede usar un número o palabra clave.

Para entender de mejor manera se propone el siguiente ejemplo:

- Palabra clave: MUSICA
- Mensaje a ser cifrado: “la mañana de hoy es demasiado helada”

Tabla 1

Ejemplo de cifrado por Transposición

M	U	S	I	C	A
4	6	5	3	2	1
L	A	M	A	Ñ	A
N	A	D	E	H	O
Y	E	S	D	E	M
A	S	I	A	D	O
H	E	L	A	D	A

Para el caso el mensaje cifrado será:

AOMOANHEDDAEDAALNYAHMDSILAAESE

2.2.1.2 . Sustitución

Consiste en el reemplazamiento de las unidades del mensaje original según una clave que al igual que en el método de transposición deber ser compartida únicamente entre el emisor y el receptor.

Existen diversos tipos de sustitución uno de ellos es la sustitución mono alfabética que consiste en que cada una de las letras se sustituye por otra del mismo alfabeto. En este mismo caso existen dos subdivisiones adicionales las cuales son.

a) ***Sustitución de letras:*** El cifrado más común de este tipo de sustitución es el cifrado de César, está denominado de esa manera debido a que Julio César lo utilizaba en la época Romana para enviar mensajes secretos a sus legiones (Santiago Fernandez, 2004). Este tipo de cifrado consiste en sustituir una letra por la que le corresponde tres letra más allá en el alfabeto es decir que a la letra A le corresponde la D, a la B la E y así sucesivamente tal como se muestra en la siguiente tabla.

Tabla 2

Cifrado de César

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

Para el caso de la Tabla 2 que consta de 26 letras del alfabeto la fórmula matemática correspondiente es C= (M+3) (mod 26). Por ejemplo, si las letras son representadas por un número entero de la siguiente manera:

Tabla 3

Equivalente numérico del alfabeto

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Se puede cifrar la palabra: MUSICA como se ve a continuación.

a) M se reemplaza por el valor equivalente a la letra del alfabeto
b) Resolver la fórmula. C = (12+3) (mod 26) = 15
c) Reemplazar el valor por la letra correspondiente. 15 = P
d) Repetir el proceso con las letras restantes.

Tabla 4

Ejemplo de Cifrado de César

M	**C**
M	P
U	X
S	V
I	L
C	F
A	D

Otro tipo de clasificación corresponde a fuentes que generan bloques de letras llamado cifrado en bloque y aquellos que generan letras llamado cifrado en flujo.

2.1.1. Tipos de Ataque

En todo momento la información tiene un cierto grado de inseguridad y es por ello que si se encuentra almacenada puede ser sustraída o acceder a ella suplantando la identidad del encargado de la información, mientras que si se encuentra siendo transmitida pueden intervenir en el canal de información y alterar el mensaje o creando mensajes falsos.

Con lo descrito previamente se dice que existen dos tipos de amenazas contra un mensaje.

- **Pasiva:** Robo, acceso no autorizado, intercepción.
- **Activa:** Modificación, interrupción, falsificación.

En el primer caso el principio violentado es la confidencialidad, mientras que en el segundo lo son los principios de integridad y autenticidad del mensaje.

Los tipos de ataque que pueden sufrir los mensajes cifrados son:

- **Ataque sólo con texto cifrado:** Cuando el criptoanalista conoce únicamente el criptograma.
- **Ataque con texto original conocido:** Cuando el criptoanalista conoce el tema al que se refiere el mensaje lo que permite establecer una relación entre las palabras más usuales.
- **Ataque con texto original escogido:** Cuando el criptoanalista conoce el criptograma y el mensaje cifrado que haya escogido, tomando en cuenta que dicha información ha sido conseguida ya de esa manera más no ha sido descifrada.
- **Ataque con texto cifrado escogido:** Cuando el criptoanalista obtiene el mensaje original adecuado a ciertos mensajes cifrados que haya elegido.

Para prevenir o recobrarse de los ataques activos o pasivos existen mecanismos de seguridad, donde cada mecanismo puede suministrar uno o varios servicios de seguridad solicitados por el usuario, pero se necesita de varios mecanismos para brindar todos los servicios al mismo tiempo, excepto si se aplican técnicas criptográficas.

Los tipos de mecanismos de seguridad son:

- Intercambio de autenticación
- Cifrado
- Firma digital
- Control de acceso
- Tráfico de relleno
- Control de encaminamiento
- Unicidad.

2.2. Criptografía Simétrica

Se denomina criptografía simétrica o de clave secreta debido a que utiliza la misma clave para encriptar y desencriptar el mensaje o la clave de desencriptación puede ser calculada a través de la clave de encriptación en un tiempo moderado (Amaya, 2014). Se entiende por encriptar al proceso de convertir un mensaje original a un mensaje clave, mientras que desencriptar toma el mensaje clave para convertirlo en el mensaje original.

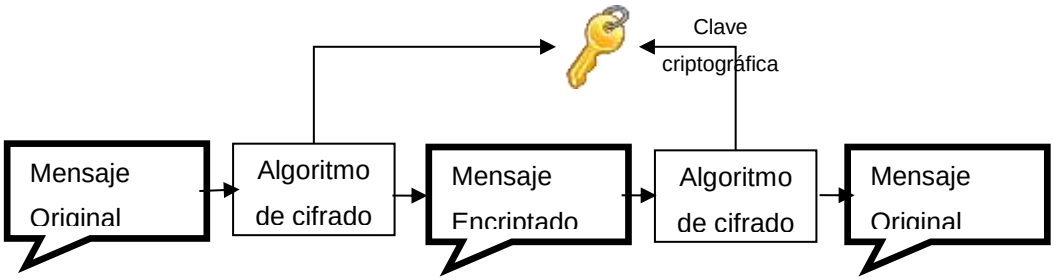

Figura 3 Esquema de cifrado por criptografía simétrica

Ambos procedimientos están relacionados con las claves, que son aquellas de las cuales depende el cambio del mensaje original al mensaje clave. En los algoritmos simétricos tanto el emisor como el receptor deben conservar en secreto la clave, y coordinar su uso, es decir, si U es el conjunto de mensajes enviados entre dos usuarios, (X y Y), entonces Z, es el conjunto de los mensajes encriptados, y w a las posibles claves criptográficas a ser utilizadas. Un criptosistema de clave simétrica se detalla a continuación:

E_w: U → Z y D_w: Z → U, de modo que para cada mensaje u del conjunto U se verifique que: D_w (E_w (u)) = u.

Los usuarios X y Y deben llegar previamente a un convenio y establecer secretamente un clave w para poder utilizar este tipo de criptosistema. Si X desea enviar un mensaje u a Y, encripta el mensaje por medio de la función E_w (E_w (u) = z), y envía el resultado z a Y. Para recuperar el mensaje original, el usuario Y desencripta el mensaje encriptado (z) que ha recibido utilizando la función D_w (D_w (E_w (u)) = u).

Los algoritmos criptográficos (E_w, D_w) deben ser “sencillos” de implementar para los usuarios (X,Y), pero deberían ser “complicados” de resolver para un intruso (H), de modo que no pudiera recuperar ni u ni w. (Amaya, 2014)

Se consideran como requisitos básicos de este tipo de algoritmo, que si se llegase a conocer el criptograma, a través de éste, no se pueda obtener ni el mensaje ni la clave. De igual manera si se conoce el mensaje original y el mensaje cifrado debe implicar un gasto mayor en tiempo y dinero descifrar dicha clave criptográfica.

La alta redundancia de la fuente se ha convertido en una amenaza criptoanalítica trascendente, y es por ello que Shannon propuso dos

métodos básicos que frustren un criptoanálisis estadístico que aproveche la propiedad en mención: la difusión y la confusión.

La difusión radica en invalidar la influencia de la redundancia sobre la fuente sobre el texto cifrado, disipándola. Para conseguirlo existen las siguientes maneras de ejecutarlo: “Transposición, el cual impide los criptoanálisis basados en las frecuencias de n-palabras. Cada letra del texto cifrado dependerá de un gran número de letras del texto original.” (Pino Caballero Gil)

Por otro lado la confusión, radica en lograr que la relación entre la clave y el mensaje cifrado sea lo más complicada como sea posible, logrando que las estadísticas del mensaje cifrado no se encuentren influenciadas por el texto original. Mediante la técnica de sustitución se consigue la relación mencionada.

A continuación, se lista los algoritmos de criptografía simétrica:

Tabla 5

Algoritmos de criptografía simétrica

Algoritmo Simétrico	**Técnica**	
	Cifrado Bloque	**Cifrado Flujo**
DES	X	
IDEA	X	
AES	X	
RC4		X
OFB		X

2.2.1. Cifrado en Bloque

"El cifrado en bloque opera sobre textos formados por palabras, convirtiendo cada una de ellas en una nueva palabra." (Pino Caballero Gil)

En este tipo de cifrado se define porque cada uno de sus bloques son independientes tanto en su posición como de los bloques contiguos, por ello se procesa un solo bloque de datos cada vez. La longitud del bloque varía dependiendo del tipo del algoritmo, por ejemplo si se trata de DES (Data Encryption Standard) la longitud de bloque es de 64 bits. Y en el caso de poseer mayor cantidad de bits es necesario fraccionarlo en bloques de 64bits, y de ser necesario el último bloque debe ser rellenado hasta completar los bits faltantes. El método más común es el ECB (electronic codebook), en el que cada bloque se cifra con la misma clave. Se denomina codebook debido a que para una clave provista existe un único texto cifrado por cada bloque de 64 bits de texto claro.

Con el ECB, si se repite el bloque de texto claro, éste producirá el mismo texto cifrado, lo cual sería poco seguro si se tratase de mensajes con gran longitud.

2.2.2. Cifrado en Flujo

Es un tipo de cifrado por sustitución, que se aplica a un carácter o bit a través de un flujo de clave que está compuesto por una clave y un vector de inicialización, ambos son combinados (XOR) con el flujo de entrada de datos. (Cristina Alcaraz).

Existen diversos tipos de cifrado síncronos y auto síncronos. Los síncronos son en los que cada estado depende del anterior y no de los caracteres de la secuencia del mensaje original. Si por algún evento se pierde la sincronización entre el emisor y receptor, se debe volver a sincronizar sus generadores antes de continuar. Los cifrados de tipo auto síncronos son los contrario de los síncronos, es decir que el texto cifrado

depende de un número finito de símbolos de texto cifrado anterior. (Pino Caballero Gil)

2.3. Criptografía Asimétrica

Llamada también de clave pública al contrario de la clave simétrica, la clave de descifrado no puede ser calculada a partir de la clave de cifrado. En este tipo de criptografía se utilizan dos claves, una pública y una privada. Este par de llaves criptográficas están asociadas matemáticamente, por lo tanto si una de las claves es utilizada para cifrar el mensaje, entonces, la otra clave es utilizada para descifrar el mensaje encriptado. Esto remedia la problemática de la clave simétrica sobre cómo compartir las claves de desencriptado.

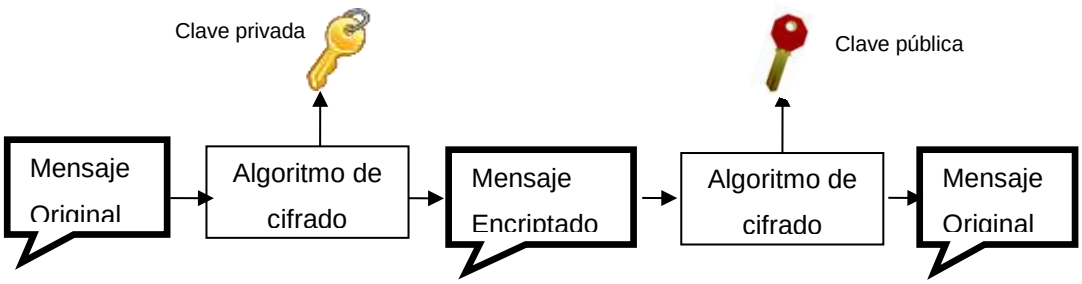

Figura 4 Esquema de cifrado por criptografía asimétrica

Es importante mencionar, que se requiere el uso de certificados digitales para autentificar la clave pública, puesto que la clave privada debe permanecer bajo resguardo.

En los algoritmos asimétricos tanto el emisor (X) como el receptor (Y) deben conservar en secreto la clave privada. Si U es el conjunto de mensajes enviados entre dos usuarios (X,Y), entonces Z, es el conjunto de mensajes encriptados. Por lo tanto, Wp y Ws son las posibles claves criptográficas a ser utilizadas. Un criptosistema de clave asimétrica se detalla a continuación:

E_{Wp}: U → Z y D_{Ws}: Z → U, de modo que para cada mensaje u del conjunto U se verifique que: D_{Ws} (E_{Wp} (u)) = u.

Tabla 6

Diferencias entre algoritmos Simétricos y Asimétricos

Algoritmos Simétricos	Algoritmos Asimétricos
La clave pública presenta información adicional que puede ser usada para un criptoanálisis.	Mayores restricciones en el diseño.
Algoritmos como el DES, poseen ecuaciones matemáticas no resolubles analíticamente.	Basa su seguridad en la dificultad de resolver algún problema matemático conocido.
La clave pública se genera de forma pseudoaleatoria	Se necesita un procedimiento para calcular la clave pública a partir de la privada que sea tan eficiente que su cálculo inverso sea “imposible” de lograr.
Son de cálculo rápido.	Son de cálculo lento.

2.3.1. Base de la Criptografía Asimétrica

Los sistemas de clave asimétrica se fundamentan en funciones-trampa de un solo sentido, es decir que su cálculo es sencillo, mientras que el procesamiento de la inversa es extremadamente complicado. El término “trampa” se refiere a que si se conociera parte de la información, sería fácil computar el inverso.

Según Diffie y Hellman, los algoritmos criptografía asimétrica deben cumplir las siguientes propiedades:

1. Cualquier usuario puede calcular sus propias claves pública y privada en tiempo polinomial.

2. El emisor puede cifrar su mensaje con la clave pública del receptor en tiempo polinomial
3. El receptor pude descifrar el criptograma con la clave privada en tiempo polinomial.
4. El criptoanalista que intente averiguar la clave privada mediante la pública, no podrá resolverlo.
5. El criptoanalista que intente descifrar un criptograma teniendo la clave pública no podrá resolverlo.

Para construir un criptosistema de clave asimétrica (Pino Caballero Gil) en su libro Introducción a la Criptografía propone el siguiente esquema:

1. Establecer un problema de difícil solución P, que sea posiblemente intratable.
2. Escoger un subproblema fácil (P_{facil}), que pueda ser resuelto en tiempo polinomial, de preferencia en tiempo lineal.
3. Transformar el problema P_{facil} de forma que el problema resultante, $P_{dificil}$ no se parezca al inicial, pero sí al problema original P.
4. Publicar el problema $P_{dificil}$ y la forma en que debe ser usado, estableciendo en este proceso la clave pública. La forma de transformar de $P_{dificil}$ a P_{facil} constituye la clave privada.

A continuación, se lista los algoritmos de criptografía asimétrica clasificados por su técnica de cifrado.

Tabla 7

Algoritmos de criptografía asimétrica

Algoritmo Asimétrico	**Técnica**	
	Logaritmo Discreto	**Factorización de enteros grandes**
RSA		X
Diffie - Hellman	X	

CONTINÙA →

El Gammal	**X**	
Shor		X
Rabin		X

2.4. Criptografía de Curvas Elípticas

La criptografía de Curvas Elípticas (ECC) es una tecnología eficiente que permite implementar infraestructuras de clave pública (PKI). Los algoritmos de curvas elípticas pueden basarse en el logaritmo discreto, teniendo en cuenta que permite usar claves más pequeñas que se relacionan directamente con el almacenamiento, debido a que permiten la utilización de menos memoria y hardware de menor capacidad de procesamiento.

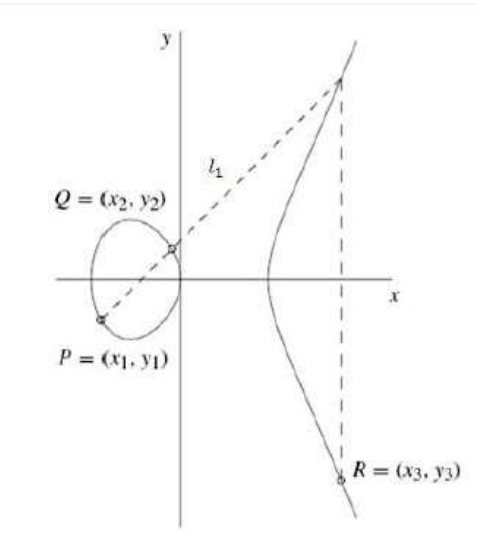

Figura 5 Representación geométrica de la suma de puntos de Curva Elíptica

Fuente: (Trujillo, 2011)

En la criptografía de curvas elípticas, tal como lo ilustra la Figura 5, el punto R se utiliza como clave pública la cual proviene de la suma de los puntos P y Q, los cuales corresponden a la clave de tipo privada. De este modo, al conocer un punto de una curva, como es el caso de R, la información contenida y cifrada con este valor no se compromete, ya que la curva elíptica tiene infinidad de puntos y para poder descifrar el mensaje se necesitan conocer los parámetros P y Q.

Generalmente, en la criptografía de curvas elípticas cuando se desea obtener una clave privada se aplica una sumatoria exhaustiva en tiempo y espacio de todos los puntos de la curva elíptica hasta que el resultado de dicha búsqueda sea el punto R, el cual contiene la información almacenada. Por esa razón para impedir que un ataque de este tipo sea satisfactorio y pueda lograr que las claves sean lo más eficientes posible se escoge un campo de Galois lo adecuadamente amplio. También se cuida que la curva elíptica elegida no sea singular, es decir, que no pase por el punto cero, y por último se toma en cuenta que la curva no sea anómala, es decir, que el campo de Galois sobre el que se define la curva no tenga números racionales.

Tomando en cuenta que las aplicaciones criptográficas requieren rapidez y precisión algebraica, es necesario emplear el grupo de curvas elípticas sobre un campo finito de GF(p) (campo de Galois) pertenecientes a los campos primos y luego aplicando un procedimiento para obtener las claves pública y privada, que es muy similar en la multiplicación de un punto por un escalar. De tal forma, que se utiliza un punto de la curva elíptica, que sea un generador de todos los demás puntos de la misma para utilizarla como clave pública, en el campo de Galois sobre el que definimos dicha curva. Finalmente, se utiliza un número entero aleatorio definido sobre el mismo campo de Galois que será la clave privada del algoritmo criptográfico.

Se debe tener en cuenta que la clave privada en el campo de Galois elegido, debe ser un número primo lo suficientemente grande para evitar que el punto R pueda ser descifrado mediante el método de la búsqueda exhaustiva. Las aplicaciones de criptografía basada en curvas elípticas son:

a) **Test de primalidad:** Es un algoritmo que permite decidir si un número natural n es primo o compuesto y actúa como una herramienta que se utiliza en la configuración de los criptosistemas de clave pública ya sean los del tipo RSA como ElGamal.

El test de primalidad más sencillo y conocido a su vez es del test de divisiones sucesivas mismo que consiste en tomar un número entero impar m y ver si divide o no a n. Si m no es ni n ni 1, entonces n es compuesto; caso contrario, n pasa a dividirse por m. (Wilder, 2006)

b) **Algoritmos de factorización:** Son algoritmos de tiempo de ejecución sub exponencial para la factorización de enteros que emplea curvas elípticas. Al momento se lo conoce como el algoritmo más eficiente para divisores que no superen los 20 o 25 dígitos decimales.

2.4.1 Campo de Galois

El campo de Galois **GF (n)** es una agrupación de elementos finitos generados por n, donde n, es un número primo. Todos los elementos de dicha agrupación, con excepción del cero, tienen inversa, razón por la cual se puede sumar, restar, multiplicar o dividir de la misma forma que en R, por lo que se cumple la ecuación:

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

Quedando así definido el conjunto **E(GF(n))**.

2.5. Firma Digital

La firma digital es un documento que garantiza la legitimidad e integridad de la información contenida en un archivo (documento) mediante un cifrado que es el resultado del uso de una llave privada o “valor secreto” que solo la posee el firmante; además permite verificar la autenticidad al comparar el documento con la llave privada del firmante.

2.5.1. Funciones Hash

Es una función matemática que transforma el texto original que tiene una longitud variable (n-bits) a una secuencia de bits de tamaño fijo (s-bits) que lo identifica, a dicha secuencia se la denomina valor hash. El objetivo de las funciones hash es comprobar la integridad de la información, para lo que si

el texto original no sufrió cambio emite un valor, y otro distinto caso contrario. Las propiedades que debe cumplir la firma digital son:

- Difícil de hallar un mensaje m, tal que hash(m) =h
- Difícil de hallar dos mensajes m1 y m2, tal que hash(m1)= hash(m2).

Lo que significa que una función has es unidireccional, es decir que es prácticamente imposibles encontrar dos textos originales con el mismo valor hash. Existen varios tipos de funciones Hash, siendo SHA el estándar norteamericano utilizado.

a) **SHA-1 (Secure Hash Algorithm):** Según el Federal Information Processing Standard o por sus siglas en inglés FIPS, fue diseñado por el National Institute of Standards and Technology. El proceso de SHA se basa en el proceso de MD5. Su cálculo produce una función hash de 160 bits que se utiliza para comprobar la integridad. Mientras mayor sea la longitud de hash más seguridad representa.

b) **MD5:** En su proceso realiza cuatro pases sobre bloques de datos y utiliza una constante numérica distinta para cada palabra del mensaje en cada pase. El número de constantes de bits que se utilizaba durante el cálculo de MD5 es 64, lo que origina una función hash de 128 bits que se utiliza para comprobar la integridad del mensaje.

c) **RIPEMD-160:** Maneja claves más robustas, normalmente de 160 bits, aunque existen versiones de 128 y están en proceso nuevas de 256 y 320 bits. Su ventaja se establece en que es de código abierto, es rápido y de libre acceso.

2.5.2. Comprobación de la firma digital

Al introducir el concepto de firma digital, se adjunta la función denominada “Hash” o resumen del documento. “Esta función lo que hace es

que a partir de un documento de tamaño N bits entrega una cadena de M bits. No hay límite para el tamaño de N, pero M siempre es de tamaño constante de acuerdo con el algoritmo usado, normalmente es de 128 o 256 bits". (Paredes, Introducción a la Criptografía, 2006)

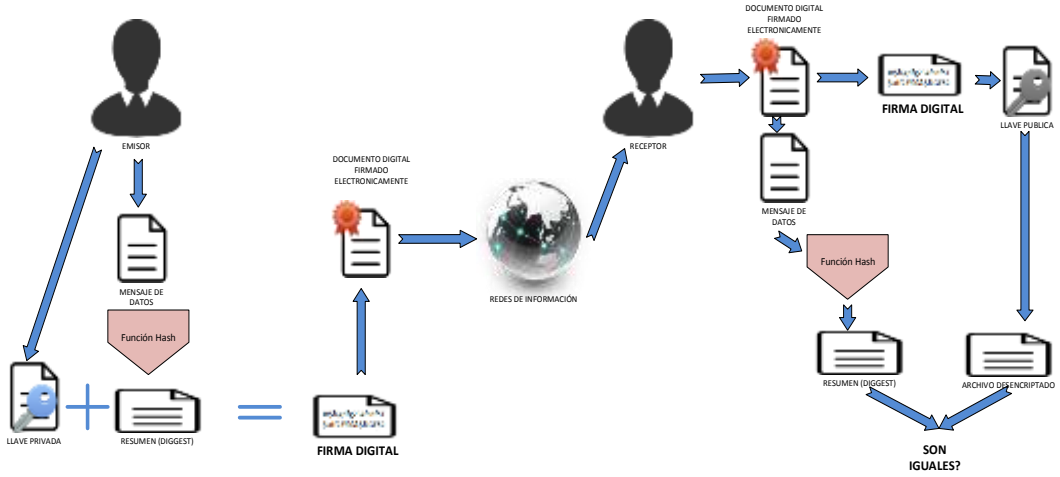

Figura 6 Comprobación de firma digital

CAPÍTULO III

FUNDAMENTACIÓN MATEMÁTICA DE ALGORITMOS DE FIRMA DIGITAL ASIMÉTRICA

3.1. Algoritmo ElGamal

Es un algoritmo basado en el cálculo logaritmos discretos y fuertemente relacionado con la factorización de números enteros; su patente correspondía a Diffie-Hellman, sin embargo ésta expiró el 29 de abril de 1997, lo que convierte a El Gamal al primer algoritmo "libre" de la reclamación de patentes en Estados Unidos.

Para la generación de un par de claves, se escoge un número primo **p** y dos números aleatorios **g** y **x** menores que **p**. Se calcula entonces:

$$y = g^x \pmod p$$

La llave pública es **y**, mientras la llave privada es **x**.

ElGamal se diferencia de otros algoritmos asimétricos porque el cifrado se efectúa utilizando, además de la clave pública del receptor, la clave secreta del emisor.

Al seleccionar **p** primo, se garantiza que sea cual sea el valor de **g**, el conjunto de $\{g, g^2, g^3, \ldots.\}$ es una permutación del conjunto $\{1,2,\ldots..,p-1\}$. El valor de **p** no debe ser necesariamente primo para que el algoritmo funcione, sin embargo bajo esta condición se debe asegurar que el conjunto generado por las potencias de **g** sea lo suficientemente grande.

Dado el caso que un grupo de personas llegue a utilizar el mismo número **p** y **g** la seguridad no se verá afectada.

a) Proceso de encriptación de un mensaje M

Siendo Lola el emisor y Juan el receptor. Se asume que Lola desea enviar un mensaje **M** a Juan para lo que:

- Juan toma un número aleatorio $\mathbf{k_B}$ (clave privada) cuyo máximo común divisor entre **p-1** sea 1 (mcd ($\mathbf{k_B}$, **p-1**)=1).
- Calcular:

$$K_B = g^{k_B} \pmod p$$

Dónde:

- $\mathbf{K_B}$**:** clave pública de Juan
- **p:** número primo
- **g:** número aleatorio de gran tamaño
- $\mathbf{k_B}$**:** número aleatorio, tal que (mcd ($\mathbf{k_B}$, **p-1**)=1). Clave privada.

- Lola selecciona un número entero $\mathbf{k_A}$, tal que mcd ($\mathbf{k_A}$, **p-1**)=1 como su clave privada y calcula la clave de cifrado a partir de su clave privada y la clave pública de Juan, $Q = K_B^{k_A} mod\ p$
- Se calcula el mensaje cifrado, tal que C c QM(mod p).

El proceso de descifrado que Juan realiza es el siguiente:

- Obtiene **Q** a través de la clave pública de Lola, $\mathbf{K_A}$ y su propia clave secreta $Q = K_A^{k_B} mod\ p$
- Se recupera M a partir de M ≡ $(Q^{-1})_p$ * C(mod p), donde $(Q^{-1})_p$ expresa el inverso de Q en módulo p.

b) Proceso de firma digital de un mensaje M

- Tomar un número aleatorio **k** cuyo máximo común divisor entre **p-1** sea 1 (mcd (**k**,**p-1**)=1).
- Calcular:

$$a = g^k \pmod p$$

Dónde:

- **p:** número primo
- **g:** número aleatorio menor que **p**
- **k:** número aleatorio, tal que (mcd (**k**, **p-1**)=1).

- Calcular **s** tal que, $M = k_A a + kb \pmod{(p-1)}$

Dónde:

- $\mathbf{k_A}$: clave privada de Lola

El par de valores enteros **(a, b)**, constituye la firma. Para el caso el valor **k**, debe permanecer en secreto y ser diferente cada vez que se realice el proceso.

- Si Juan desea confirmar la firma debe calcular:

$$a^b \pmod{p} \text{ y } (g^{k_A})^a \pmod{p}$$

- Calcular el producto del proceso anterior de modo que se tiene: y $(g^{k_A})^a (g^k)^b \pmod{p}$ y comprobar que coincide con $g^M \pmod{p}$.

Las necesidades de almacenamiento al utilizar el algoritmo El Gamal aumentan de manera considerable debido al tamaño del texto cifrado **C,** razón por la cual no es tan popular entre los usuarios aunque sea igual de seguro que RSA.

3.2. Algoritmo Rabin

En 1979, Rabin (Pino Caballero Gil) propuso un sistema criptográfico basado en la factorización de números enteros y que cumpla con los siguientes teoremas:

- **Teorema 1**

Se puede construir una solución de la congruencia $M^2+B*M \equiv C \pmod{p*q}$ mediante las soluciones **u** y **v** de las congruencias u^2 +

B*u ≡ C (mod p) y v^2+ Bv ≡ C(mod p), y de dos enteros **a** y **b** que verifican a ≡ 1 (mod p), a ≡ 0(mod q), b ≡ 0 (mod p), b1 ≡ 1(mod q), a partir de la expresión M = au + bv.

- **Teorema 2**

Si **p** y **q** son primos, siempre se pueden encontrar, mediante el algoritmo euclideo, los enteros **a** y **b** que satisfagan las condiciones del lema anterior.

Se llaman residuos cuadráticos módulo **p** a los enteros **d** tales que y^2 ≡ d (mod p). Para encontrar los residuos cuadráticos módulo **p** y módulo **q** son útiles los siguientes resultados. El primero se conoce como criterio de Euler.

- **Teorema 3**

Un entero a tal que 1 ≤ a ≤ p-1 es un residuo cuadrático módulo un número primo **p** sí y sólo si:

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

- **Teorema 4**

Si **p** es un número primo de la forma 4k-1 y d es un residuo cuadrático módulo **p**, entonces una solución de la congruencia y^2 ≡ d (mod p) viene dada por y ≡ d^k (mod p).

Mediante ambos resultados queda claro que si los dos primos p y q son congruentes con 3 (mod 4), entonces el procedimiento de descifrado se puede realizar en tiempo polinomial.

a) Proceso de encriptación de un mensaje m

- Escoger la clave privada correspondiente a dos números primos (**p** y **q**), ambos congruentes con 3 mod 4 (los dos últimos bits a 1).
- Calcular la clave pública **n = p*q**

- Codificar el mensaje **m**:

$$c = m^2 \pmod n$$

b) Proceso de desencriptación de un mensaje M

- Calcular lo siguiente:

$$m1 = c^{(p+1)/4} \pmod p$$
$$m2 = (p - c^{(p+1)/4}) \pmod p$$
$$m3 = c^{(q+1)/4} \pmod p$$
$$m4 = (q - c^{(q+1)/4} \pmod p$$

- Escoger **a** y **b** tal que **a** = **q (q^{-1} (mod p))** y **b** = **p (p^{-1} (mod q))**. Lo que permite deducir los potenciales mensajes originales expresados de la siguiente manera:

$$m_a = (am_1 + bm_3) \pmod n$$
$$m_b = (am_1 + bm_4) \pmod n$$
$$m_c = (am_2 + bm_3) \pmod n$$
$$m_d = (am_2 + bm_4) \pmod n$$

Al término del proceso no existe un medio que permita decidir cuál de las cuatro opciones es legítima, por lo que el mensaje **m** deberá incluir algún tipo de información que permita al destinatario distinguirlo de los otros.

3.3. Algoritmo Fiat – Shamir

El algoritmo de Shamir basa su complejidad en la fragmentación del cifrado en varias entidades por lo que el cifrado puede reconstruirse cuando un subconjunto autorizado de participantes reúnes sus fragmentos. Para la idea detrás de los sistemas de secretos compartidos (Shared Secret Systems) es la protección de la información, vía su distribución. (TENA, 2003). Se asume que el distribuidor los de fragmentos es un ente confiable y que son entregados de manera segura a los participantes, tal como lo ilustra la siguiente figura:

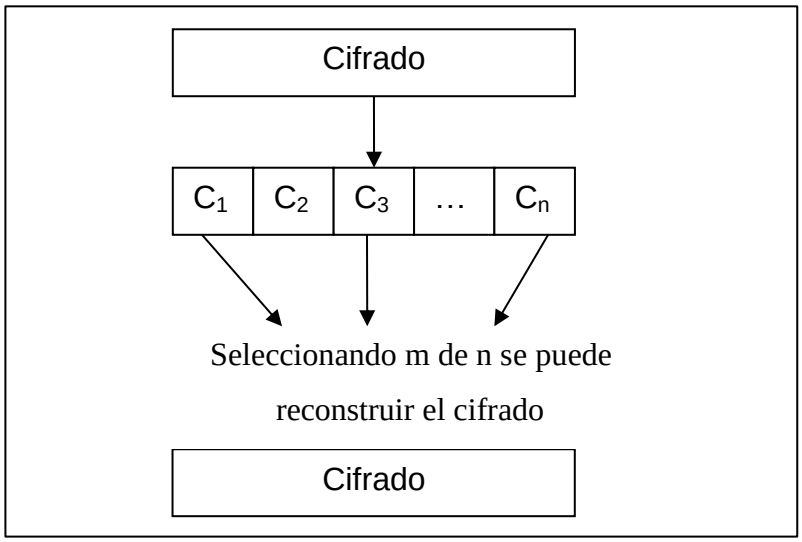

Figura 7 Sistema de Secreto Compartido

Fuente: (Vasquez, 2006)

El sistema de secreto compartido está definido a través de algoritmos probabilísticos, cuya entrada es el texto cifrado o secreto mismo que es un elemento de un conjunto finito, y su salida son sus fragmentos. El cifrado es recuperado con el umbral t, es decir, en el que al menos t fragmentos pueden recuperar el secreto; y en el caso que menos de t fragmentos son conocidos quiere decir que no se conoce nada sobre el mismo (de manera teórica) (Miguel Alfredo Acedo Arias, 2008).

Shamir en su publicación “Identity Based Cryptosystems and Signature Schemes” precisa las terminologías de privacidad y exactitud en los esquemas de umbral, además propuso la elección de un polinomio (aleatorio) de grado **t-1**, y el recobro del cifrado se basa en la interpolación polinomial de Lagrange.

Shamir propone la solución al esquema del umbral de la siguiente manera (Miguel Alfredo Acedo Arias, 2008):

Entrada: Números enteros positivos **l** y **t <= l,** y un secreto **K** Є {**0,…s-1**}

Salida: Números enteros positivos fragmentos del secreto distribuidos en **S_i**

- Elegir un número primo **p** >= **max{s,l+1}** (preferiblemente mayor que el número de partes en el que se seccionará)
- Elegir aleatoria e independientemente **$a_1,\ldots,a_{t-1}$ Є Ƶ**
- Construir el polinomio de grado **t-1,q(x) = K +** $\sum_{i=1}^{t-1} a_i x^i$
- Distribuir el secreto en las particiones **s_i = q(i) Є Ƶ, i=1**

Dónde:

- **p:** número primo
- **l:** número entero positivo
- **t:** número entero positivo
- **q(x):** polinomio

Los elementos del conjunto **t** conseguirán recobrar el polinomio **q(x)** debido a que conocen las imágenes de **t** puntos y el término independiente de éste es el cifrado.

El protocolo de Fiat-Shamir por su parte es distinguido como el primer protocolo de conocimiento cero; permite la autenticación segura del usuario proveyendo la información estrictamente necesaria sobre la identidad de la entidad a ser autenticada. Su complejidad se basa en el cálculo de la raíz cuadrada de un número entero de módulo n, con factorización desconocida.

3.3.1. Proceso de la raíz cuadrada de modulo n

Sea **n = pq** donde **p** y **q** son números primos impares. Para el problema de residuos cuadráticos de módulo **n**, la cuestión es, si dado **a Є Ƶ,** la ecuación tiene solución:

$$x^2 = a \pmod n$$

Se ha demostrado que el resolver el problema de la raíz cuadrada de módulo **n** es equivalente a encontrar la factorización de **n**.

El protocolo involucra de tres partes:

- Lola que quiere demostrar su identidad
- Juan (generalmente el servidor) que quiere verificar la identidad de Lola
- Un testigo que desempeña el papel de garante (Trusted Third Party)

El proceso del algoritmo de Fiat – Shamir viene dado de la siguiente manera:

- El testigo genera de forma aleatoria dos números primos **p** y **q** que se mantienen en secreto y publica el producto **n = pq**
- Lola escoge su clave secreta que corresponde a un número entero:

  $$\mathbf{1 \leq a \leq n-1},$$

- Lola calcula $\mathbf{v = s^2 \pmod n}$, donde s es el secreto.
- El testigo registra **(n, v)** como clave pública de Lola.

El protocolo descrito por Elisa Bragaglia en su tesis “Tesis di Laurea in Algoritmi della Teoria dei Numeri e Criptografia” (Bragaglia, 2012) está dado según el siguiente proceso**:**

- Lola selecciona un número aleatorio $\mathbf{r \in \mathbb{Z}_n}$ y envía a Juan

  $$\mathbf{x = r^2 \pmod n}$$

- Juan envía a Lola un número de entrada $\mathbf{e \in \{0,1\}}$ (desafío)
- Lola responde el desafío calculando y enviando a Juan

  $$\mathbf{y = r\, s^e \pmod n}$$

  dónde, si:

  a. **e = 0** Lola envía **y = r (mod n)**

  b. **e = 1** Lola envía **y = r s (mod n)**

Nótese que, sólo Lola conoce la clave secreta **s** que pude calcular correctamente **y**.

- Juan comprueba la exactitud del resultado utilizando la clave pública de Lola descrita anteriormente como **(n, v),** a través de la igualdad

$$\mathbf{y^2 \equiv x\, v^e \pmod n},$$

dónde, si:

c. **e = 0** Bob envía $\mathbf{y^2 \equiv x \equiv r^2 \pmod n}$

d. **e = 1** Lola envía $\mathbf{y^2 \equiv xv \equiv r^2\, s^2 \pmod n}$

Para que Juan esté seguro de la identidad de Lola puede iterar cuantas veces crea necesario el proceso hasta obtener un alto resultado probabilístico siempre y cuando Lola siga respondiendo correctamente al desafío enviado. Como es de suponerse, si el proceso es enviado una única vez la probabilidad que la identidad sea veraz o fraudulenta es de 50%.

3.3.2. Proceso de Firma Fiat-Shamir

El proceso de generación de firma digital ha considerado la argumentación dada por Pino Caballero Gil en su libro “Introducción a la Criptografía” (Pino Caballero Gil). El cual se describe a continuación:

- Elegir aleatoriamente un conjunto de enteros $\mathbf{r_1, r_2, \ldots r_i}$ que vayan de **0** a **n-1**
- Calcular $\mathbf{a_i = r_i^{\,2} \pmod n}$ para cada **i= 1,2,…,t**
- Calcular $\mathbf{f\ (x, a_1, a_2, \ldots, a_t)}$ donde **f** es una función unidireccional pública que utiliza los primeros **kt** bits como entradas $\mathbf{e_{ij}}$ de una matriz binaria **txk**, E.

- Calcular $\mathbf{y_i} = \mathbf{r_j} \prod_{e_y=1} \mathbf{S_j} \ (\mathbf{mod\ n})$ para **i = 1,2,….t.**

  Dónde:

  - **n:** producto de dos números primos **p** y **q**
  - **r:** parte de un conjunto de números enteros de **0** a **n-1**
  - **t:** número entero positivo

La firma para **x** de Lola es **(y, E)** donde **y** es el vector **(y_1, y_2,…y_t)** y **E** es la matriz **(e_{ij}).** Para que Juan verifique la veracidad de la firma se procede de la siguiente manera:

- Calcular **Zj =** $\mathbf{y^2_j} \prod_{e_y=1} \mathbf{V_j} \ (\mathbf{mod\ n})$ para **i = 1,2,….t,** siendo $\mathbf{V_j}$ la información pública de Lola.

- Comprobar que los **kt** bits primeros de **f (x, z_1, z_2,…, z_t)** son las entradas de $\mathbf{e_{ij}}$ en la matriz **E**

3.4. Algoritmo Shor

Su complejidad se basa en el cálculo de la factorización cuyas amplitudes sobrepasan los límites establecidos para el procesamiento de una computadora “normal”, es decir que considera valores que no pueden ser leídos, para lo que Peter Shor en 1994 demostró que un computador cuántico puede factorizar un número eficientemente dando solución a la problemática y proveyendo así protocolos criptográficos de clave privada más seguros.

El algoritmo de Shor sobrepasa la efectividad del sistema criptográfico RSA debido a que sus técnicas se cimientan en una propiedad característica de la mecánica cuántica: los estados cuánticos no se pueden copiar (clonar). (W. K . Wootters, 1982). Los prototipos de ordenadores cuánticos donde se ha probado esta técnica poseen un máximo de 10 qubits.

Al igual que la mayoría de algoritmos cuánticos, el algoritmo de Shor se fundamenta en la probabilidad estadística lo que significa que mientras mayores repeticiones se realicen, mayor es la probabilidad de éxito.

3.4.1. Problema de factorización clásico

El proceso que involucra el problema de factorización clásico se según Jesús García Lacalle, (Lacalle, 2004), para encontrar el factor propio de un número **N** impar es:

- Escoger un número aleatorio **$1 \leq a \leq N-1$**
- Si **mcd (a, N) ≠ 1** devolver **mcd (a, N)**
- Calcular el orden **t** de **a** en **U_N**, donde **t** es un número natural
- Si **t** es impar devolver fallo
- Si **mcd ($a^{t/2}$+1, N) ≠ N** devolver **mcd ($a^{t/2}$+1, N)**
- Devolver fallo.

Si el algoritmo se ejecuta exitosamente entonces el resultado es un factor propio de **N**.

El algoritmo de Shor reside en ejecutar cuánticamente la parte más elevada del problema de factorización clásico definido anteriormente, lo que corresponde a efectuar el cálculo de orden **t** de **a** en **U_N**, equivalente al cálculo del periodo **T** de la función **$f(k) = a^k \pmod N$** definida sobre **$\mathbb{Z}_{\phi(N)}$**. Al desconocer el valor de ϕ se debe establecer un nuevo dominio para la función **f** no periódica que es el conjunto **$\mathbb{Z}_Q$** donde **$Q = 2^n$** que además verifica **$N^2 < Q < 2N^2$**.

La función **f** necesita exactamente **m** qubits, donde **m** es un número entero que cumple con la condición **$N < 2^m < 2N$**, por lo que el algoritmo trabajará con un **(n+m)-qubit**.

3.4.2. Proceso del algoritmo de Shor

El proceso del algoritmo de Shor según Jesús García Lacalle, (Lacalle, 2004), viene dado según el siguiente proceso.

- Elegir un número aleatorio **a** tal que **$1 \leq a \leq N-1$**
- Si **mcd (a, N) ≠ 1** devolver **mcd (a, N)**
- Calcular el periodo **T** de la función **$f(k) = a^k \pmod N$** definida sobre $\mathbb{Z}_{\phi(N)}$:
    - ○ Inicializar el **(n, m)-qubit $\Psi = |0\} \otimes |0\}$**
    - ○ Aplicar **F_n** al primer registro, es decir aplicar **$F_n \otimes I$**
    - ○ Aplicar el operador **U_f** asociado a la función **f**
    - ○ Aplicar **Fn** al primer registro, es decir aplicar **$F_n \otimes I$**
    - ○ Obtener la medida **$k \in \{0, 1,\ldots,Q-1\}$** del primer registro
    - ○ Si es posible, calcular **T** a partir de **k** y si no devolver fallo
- Si **t** es impar devolver fallo
- Si **mcd ($a^{T/2}+1$, N) ≠ N** devolver **mcd ($a^{T/2}+1$, N)**
- Devolver falló.

Dónde:

- ○ **a:** número aleatorio
- ○ **N:** número impar
- ○ **m:** número e qubits (número entero tal que, **$N < 2^m < 2N$**)
- ○ **f:** function no periódica
- ○ **Q:** equivale a **2^n** , y que además verifica **$N^2 < Q < 2N^2$**
- ○ **n:** producto de dos número aleatorios **p** y **q**

3.5. Algoritmo RSA

Desarrollado en 1977 por Ronald Rivest, Adi Shamir, Leonard Adelman y registrado el 20 de septiembre de 1983 el algoritmo RSA es el sistema de

encriptación más conocido y considerado como el más seguro en el medio; su complejidad se basa en la factorización de números de gran tamaño y como varios de los algoritmos analizados anteriormente su clave pública proviene de la multiplicación de dichos números.

3.5.1. Fundamento Criptográfico

Escoger dos números primos grandes aleatorios **p** y **q**, estos pueden ser mayores que $10^{100.}$

- Calcular la clave pública **n = pq**
- Calcular **z = (p - 1) (q - 1)**
- Seleccionar un número aleatorio **d** primo de **z**, donde **1 < d < z** y **mcd (z, d) = 1**
- Calcular **e** de forma que **e*d ≡ 1 (mod z)**

La clave pública $\mathbf{K_E}$ viene dada por el par **(e, n)** y la clave privada $\mathbf{K_D}$ viene dada por el par **(d, n).**

Una vez aclarados estos términos se puede proceder al cifrado de un mensaje **M**, por ejemplo si el emisor Lola desea mandar un mensaje a Juan. Juan deberá tener la clave pública de Lola y el módulo **n** para cifrar el mensaje **M** de forma que se calcula:

$$\mathbf{C = M^e \ (mod\ n)}$$

Dónde:

- **e:** valor calculado, de forma que **e*d ≡ 1 (mod z)**
- **M:** mensaje
- **p y q:** par de números primos aleatorios
- **n:** producto de **p** y **q**

Para el proceso de descifrado el receptor Juan debe utilizar la clave privada **d** y el modulo **n** de forma que se calcule:

$$\mathbf{M = C^4 \pmod{n}}$$

3.5.2. Aplicación del algoritmo RSA

- Escoger dos números primos aleatorios **p = 13 y q = 5**
- Calcular la clave pública **n = pq → n = 65**
- Calcular **z = (p - 1) (q - 1) → z = 12*4 → z= 48**
- Seleccionar un número aleatorio **d** primo de **z**, donde **1 < d < 48**.
- Entonces si **mcd (48, 5) = 1 ,** por lo que **d = 5**
- Calcular **e** de forma que:

  e*d ≡ 1 (mod z) → e*5 ≡ 1 (mod 48).

  Por lo tanto **e = 29**

La clave pública está dada por $\mathbf{K_E = (e, n) \rightarrow (29, 65)}$ y la clave privada por

$\mathbf{K_D = (d, n) \rightarrow (5, 65)}$

Si **M** =2, entonces el cifrado es:

$\mathbf{C = M^e \pmod{n} \rightarrow C = 2^{29} \pmod{65} \rightarrow C = 536870912 \pmod{65} \rightarrow C = 32}$

Su respectivo descifrado es:

$\mathbf{M = C^d \pmod{n} \rightarrow M = 32^5 \pmod{65} \rightarrow M = 33554432 \pmod{65} \rightarrow M = 2}$

Por ejemplo si la información de entrada es un texto plano: “mensaje”, e=7, n = 33, d=3; el proceso de cifrado y descifrado se vería reflejado en las tablas 8 y 9.

Tabla 8

Cifrado de un mensaje M

Mensaje (M)			Cifrado (C)	
símbolo	**número**	**M^7**	**M^7 (mod 33)**	**símbolo**
m	13	62748517	7	**g**
e	05	78125	14	**n**
n	14	105413504	20	**t**
s	19	893871739	13	**m**
a	01	1	1	**a**
j	10	10000000	10	**j**
e	05	78125	24	**x**

Tabla 9

Descifrado de un mensaje M

Cifrado (C)			Mensaje (M)	
símbolo	**número**	**M^3**	**M^3 (mod 33)**	**símbolo**
g	07	343	13	**m**
n	14	2744	05	**e**
t	20	8000	14	**n**
m	13	2197	19	**s**
a	01	1	01	**a**
j	10	1000	10	**j**
x	14	2744	05	**e**

Como se puede notar los procedimientos de cifrado y descifrado son inversos por lo que si un intruso quisiera romper la seguridad del mensaje podría factorizar **n** que es la clave pública en sus números primos **p** y **q**, a partir de los cuales puede calcular **z**, obtener **e** y hallar **d** utilizando el algoritmo euclidiano. Más al tratarse de números grandes (idea principal del algoritmo) a nivel computacional es improbable.

3.5.3. Firma digital RSA

Para generar firma digital se puede utilizar el algoritmo asimétrico RSA cuyo procedimiento inicial es el mencionado en el fundamento criptográfico anteriormente visto. Una vez obtenidas las claves, pública $\mathbf{K_E}$ que viene dada por el par **(e, n)** y la clave privada $\mathbf{K_D}$ que viene dada por el par **(d, n)** la firma digital sigue el siguiente procedimiento:

- Calcular la firma de Lola tal que **y ≡** M^{K_E}**(mod n)**
- Entonces el par **(M, y)** se vuelve público.
- Para verificar que Lola firmó el mensaje Juan debe seguir los siguientes pasos:
    - Obtener (e_A, n) de Lola
    - Calcular **z ≡ ye_A (mod n).** Y si **z =m**, entonces Juan reconoce la firma como válida caso contrario no lo es.

Dónde:

- **M:**mensaje
- $\mathbf{K_E}$**:**clave pública
- **n:**resultado de la multiplicación de dos número primos **p** y **q**
- **z :** resultado de **(p - 1) (q - 1)**
- $\mathbf{e_A}$**:** número, tal que, e*d ≡ 1 (mod z)

En el caso de firmas digitales el esquema criptográfico RSA muestra algunos inconvenientes tal como lo hace al ser utilizado en el cifrado, es decir que cabe la posibilidad de que el intruso factorice los números enteros o en su defecto, al tener un mensaje M y la llave de cifrado de otro usuario (e, n), calcular la firma digital **s** tal que **m -s^e (mod n)** puede ser mucho más fácil si se tiene, además, **(s^1; m^1)**, donde $\mathbf{s^1}$ es la firma digital del usuario legítimo para un mensaje m^1 muy parecido al mensaje **M**. En otras palabras, podría resultar fácil falsificar firmas digitales para algún mensaje dado después de haber visto las firmas digitales auténticas de varios mensajes parecidos. (Montiel, 2003)

Para cifrar un mensaje M en el proceso de firma digital RSA, se ingresa con una función hash que da como resultado una asimilación que se codifica según el estándar en una cadena de octetos, siguiente al paso mencionado se divide en bloques y cada sucesión de octetos se convierte a enteros con lo que puede así aplicar el fundamento criptográfico RSA descrito anteriormente y el resultado a su vez pasa de enteros a octetos obteniendo así la firma digital. El procedimiento mencionado se lo puede apreciar de mejor manera en el siguiente gráfico:

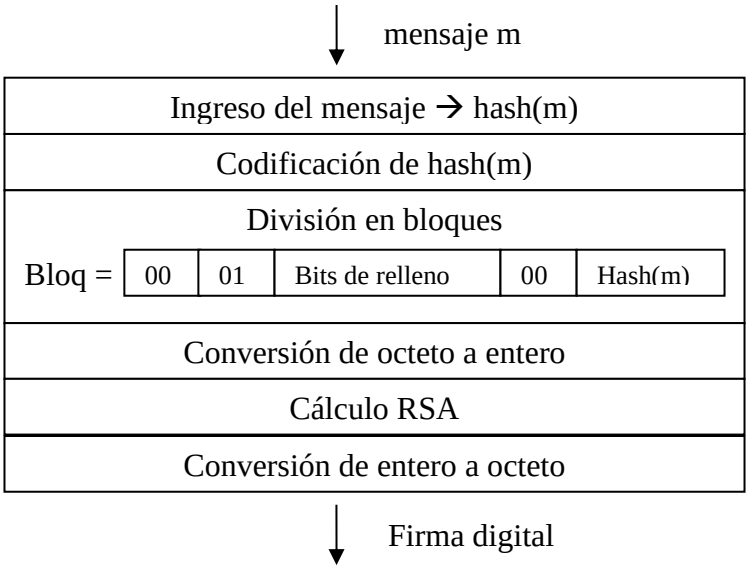

Figura 8 Proceso de firma RSA
Fuente: (Montiel, 2003)

Como la firma digital RSA se trata de un proceso inverso para su comprobación, a partir de la firma y el mensaje m, lo primero que se debe realizar es la conversión de la cadena de octetos a enteros, a lo que se le aplica el fundamento criptográfico RSA, la cadena de enteros resultado del proceso en mención nuevamente se transforma a secuencia de octetos al que se le administra un análisis para recuperar el bloque cifrado ingresado hash (m)', se ingresa el mensaje m y como resultante, hash(m) debe ser igual a hash(m)'. Proceso representado en LA Figura 9.

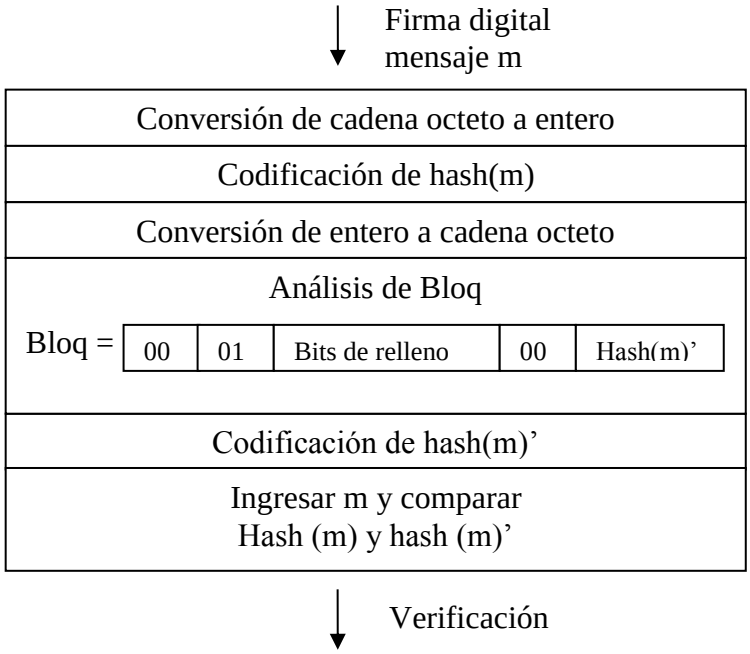

Figura 9 Verificación de firma RSA

Fuente: (Montiel, 2003)

CAPÍTULO IV

FUNDAMENTACIÓN MATEMÁTICA DE ALGORITMOS DE FIRMA DIGITAL ECC

El estudio comparativo del presente documento considera curvas elípticas con parámetros sobre **Zp** por lo que es importante conocer su fundamentación expresada a continuación.

Existen dos tipos de parámetros de dominio, entre los cuales se encuentran: parámetros de dominio de curva elíptica sobre $\mathbf{Z_p}$ y parámetros de dominio de curva elíptica sobre $\mathbf{Z_2^m}$. Sin embargo para el estudio se enfatizará los parámetros sobre $\mathbf{Z_p}$.

Al tratarse de firmas digitales basados en ECC es importante mencionar los parámetros de dominio de Curva Elíptica.

Los parámetros de dominio de curva elíptica sobre **Zp** son un séxtuple:

T= (p,a,b,G,n,h)

Dónde:

- **p:** número entero sobre Zp
- **Zp:** espacio finito
- **a**: elemento de Zp
- **b:** elemento de Zp
- **n:**número primo de orden **G**
- **h:** cofactor tal que, $\mathbf{h = E(Z_p)/n}$
- **G:** punto base

Que consiste en un entero **p** sobre el espacio finito **Zp**, dos elementos **a** y $\mathbf{b \in Z_p}$ sobre una curva elíptica $\mathbf{E(Z_p)}$ definida por la ecuación:

$$\mathbf{E: y^2 \equiv x^3 + ax + b \pmod{p},}$$

Un punto base **G = (X_G, Y_G)** sobre **E**($\mathbb{Z}_p$), un primo **n** que es el orden de **G**, y un entero **h** que es el cofactor tal que, **h = E($\mathbb{Z}_p$)/n**

El proceso de generación de primitivas de parámetros de dominio de curva elíptica sobre $\mathbb{Z}_p$ es descrito a continuación:

- Seleccionar un número primo **p** tal que **[log2p]** = **2t** siendo **t ≠ 256** y tal que [**log2p**] = **521** si **t = 256** para determinar el campo finito $\mathbb{Z}_p$
- Seleccionar elementos **a** y **b** Є $\mathbb{Z}_p$ para determinar la curva elíptica E($\mathbb{Z}_p$) definida por la ecuación:

  $$E: y^2 \equiv x^3 + ax + b \pmod{p},$$

- Un punto base G = (X_G, Y_G) sobre **E**($\mathbb{Z}_p$), un primo **n** que es el orden de **G**, y un entero **h** que es el cofactor tal que, **h = E($\mathbb{Z}_p$)/n ,** sujeto a las siguientes condiciones:
  - $4a^3 + 27b^2 \neq 0 \pmod{p}$
  - #E($\mathbb{Z}_p$) ≠ p
  - $p^B \neq 1 \pmod{\mathbf{n}}$ para cualquier $1 \leq B < 20$
  - $h \leq 4$
- Salida T = (p,a,b,G,n,h).

La primitiva permite que cualquiera de los métodos conocidos de curva elíptica puedan ser usados.

4.1. Algoritmo Diffie – Hellman

El protocolo de intercambio de claves Diffie-Hellman es uno de los primeros métodos prácticos sobre un canal inseguro que basa su complejidad en el problema del logaritmo discreto en un cuerpo finito.

4.1.1. Proceso clásico del algoritmo Diffie-Hellman (logaritmo discreto)

- Escoger un número primo **p** y un generador **g** $\in \mathbb{Z}_p$. Ambos públicos, es decir conocidos por Lola, Juan e inclusive un intruso.
- Lola escoge un número aleatorio **a** $\in \mathbb{Z}_{p-1}$, calcula $\mathbf{A = g^a \bmod p}$ y lo envía a Juan
- Juan escoge un número aleatorio **b** $\in \mathbb{Z}_{p-1}$, calcula $\mathbf{B = g^b \bmod p}$ y lo envía a Lola

Al intercambiar información Lola y Juan se puede notar que ambos tienen la habilidad de calcular $\mathbf{K = g^{ab} \bmod p}$. La demostración para el emisor Lola y el receptor Juan se muestra a continuación.

Caso Lola:

- $B^a \bmod p = (g^b \bmod p)^a \bmod p = ((g^b \bmod p)(g^b \bmod p)\ldots(g^b \bmod p)) \bmod p = g^{ba} \bmod p = g^{ab} \bmod p =$ **K**

Caso Juan:

- $A^b \bmod p = (g^a \bmod p)^b \bmod p = ((g^a \bmod p)(g^a \bmod p)\ldots(g^a \bmod p)) \bmod p = g^{ab} \bmod p =$ **K**

Como se ha comprobado que ambos pueden generar K ésta funciona como su clave compartida.

4.1.2. Primitivas Diffie-Helman de Curvas Elípticas

Antes de definir el esquema de Diffie-Hellman es necesario analizar sus primitivas que son el cimiento para operación del ECAES (Elliptic Curve Augmented Encryption Scheme), y el esquema de curvas elípticas.

Se consideran dos primitivas; la base de ambas es generar un valor secreto compartido a partir de una clave privada recogida por A y una clave pública recogida por una entidad distinta B, y según el proceso clásico del algoritmo Diffie-Helman los dos podrán recuperar la clave compartida.

La diferencia entre las primitivas a analizarse se encuentra en que la primitiva Diffie-Hellman de curva elíptica es semejante del algoritmo clásico, mientras que la primitiva Diffie-Hellman de cofactor de curva elíptica, como su nombre lo indica incluye el cofactor **h** en el proceso de cálculo del valor compartido para proporcionar una resistencia eficiente contra ataques de subgrupo pequeño. (Belingueres). La descripción de las primitivas se muestra a continuación:

- Primitiva Diffie-Hellman de curvas elíptica. Para que Lola pueda calcular la clave compartida con Juan la primitiva Diffie-Helman el proceso toma como primera instancia parámetros validados de curva elíptica **T = (p,a,b,G,n,h)** o a su vez **T = (m, f(x), a, b, G, n, h),** la clave privada $\mathbf{d_L}$ relacionada a **T** perteneciente a Lola, y una clave pública $\mathbf{Q_J}$ relacionada con **T** perteneciente a Juan. Su proceso es:
    - Calcular el punto **P** de la curva elíptica tal que $\mathbf{P = (xp, yp) = d_L Q_J}$
    - Verificar que $\mathbf{P \neq 0}$, caso contrario retornar “no válido” y finalizar.
    - Calcular $\mathbf{z = x_P}$ (elemento del campo finito que es el valor secreto compartido) y retornar z.

- Primitiva Diffie-Hellman de cofactor de curva elíptica. Su procedimiento para el cálculo de la clave secreta compartida es similar al establecido para la primitiva Diffie-Hellman de curva elíptica y su diferencia se encuentra en el primer paso correspondiente a la multiplicación escalar donde el lugar de multiplicar únicamente por la clave privada $\mathbf{d_L}$, Lola debe calcular **P** tal que:

$$\mathbf{P = (xp, yp) = hd_L Q_J.}$$

  Dónde:
    - **xp:** punto x de la curva elíptica

- **yp:** punto y en la curva elíptica
- **d_L :** clave privada
- **$Q_{J.}$:** clave pública

4.2. Algoritmo Elgamal

Para el correcto entendimiento del algoritmo de cifrado ElGamal es necesario conocer su esquema de Cifrado, mencionado consecutivamente.

Un texto claro **m** está representado por primera vez como un punto **M**, y se encripta añadiéndolo a **kQ** donde **k** es un número entero seleccionado al azar, y **Q** es la clave pública del destinatario. El emisor transmite los puntos **C1 = k P** y **C2 = M + KQ** al destinatario que utiliza su clave privada **d** para calcular **dC1 = d (k P) = k (d P) = kQ**, y se recupera a partir de entonces **M = C2 - kQ**.

Un intruso que quiera recuperar **M** tiene que calcular **kQ**. Esta tarea de calcular **kQ** partir de los parámetros de dominio, **Q**, y **C1** = **k P**, es el análogo de curva elíptica del problema de Diffie-Hellman.

a) Algoritmo básico de Cifrado de curva elíptica ElGamal

ENTRADA: parámetros de dominio de curvas elípticas **(p, E, P, n), Q** y **m.**

Dónde:

- **p:** número entero sobre Zp
- **E:** Una curva elíptica sobre **Zp** de la forma **y2 = x3 + ax + b (a, b en Zp)**
- **n:**número primo de orden P
- **P:** punto base
- **Q:** clave pública
- **m:** texto plano

SALIDA: texto cifrado (C1, C2).

- Representar al mensaje m como punto M en E (Zp).
- Seleccione k **Є** R [1, n -1].
- Calcular C1 = k P.
- Calcular C2 = M + kQ.
- Retorno (C1, C2).

b) Algoritmo de descifrado de curva elíptica ElGamal

ENTRADA: Parámetros de dominio **(p, E, P, n), D** clave privada, texto cifrado **(C1, C2).**

SALIDA: **m** Texto plano.

- Calcular **M = C2 – D C1** y extracto **m** de **M**.
- Retorno **(m)**.

c) Esquema de firma digital ElGamal:

Para este esquema precisamos, igual que antes, un punto **P** generador de un subgrupo grande **$E(F_q)$** de orden primo **p**, ambos datos públicos.

Lola desea enviar la firma de un mensaje **w**, codificado como un número entero y que cumple **$1 \leq w \leq p-1$** (Si fuera mayor se podría usar una función hash para reducirlo).

Se requiere una función

$f: E(F_q) \rightarrow Zp$

Que asigne a cada punto de la curva elíptica un entero. Esta función no necesita tener características especiales, es suficiente con que el entero resultante de aplicar la función sea más grande y que muy pocos puntos correspondan con el mismo entero. Por ejemplo, en el caso de que la curva

este definida sobre $\mathbf{Z_p}$, es suficiente tomar la coordenada "*x*" de un punto como su imagen, **f(x,y)=x**. De esta forma tan solo dos puntos como máximo (un punto y su opuesto) se corresponderán con la misma imagen. Por simplicidad supondremos que podemos tomar directamente la coordenada "*x*".

d) Procedimiento de generación de firma digital:

- Lola escoge aleatoriamente una clave privada $\mathbf{k_A}$, con un **r** tal que, $\mathbf{1 \leq r \leq p-1}$, con su clave pública correspondiente $\mathbf{K_A = k_A p.}$ Donde **p** es un número primo sobre Zp

- Lola, además, escoge un entero **r**, con $\mathbf{1 \leq r \leq p-1}$, donde p es un número primo sobre Zp, que cumpla la propiedad mcd(r,p)=1, con lo que garantizamos la existencia de $\mathbf{r^{-1}}$ **(mod p).** Como **p** es primo, cualquier **r** es válido. Tras ello, calcula.

$$\mathbf{R=rp}.$$

- Después obtiene un entero **s** como:

$$\mathbf{s= r^{-1}(w - k_A f(R)) \bmod p}$$

Dónde:

- **p:** número primo sobre Zp
- **r:** número, tal que $\mathbf{1 \leq r \leq p-1}$
- **w:** mensaje codificado como número entero tal que $\mathbf{1 \leq w \leq p-1}$
- $\mathbf{K_A}$**:** clave pública
- **R:** valor calculado anteriormente

La firma de Lola para el mensaje **w** quedara determinada por el par (**R, s).**

Para que Juan pueda verificar la firma, debe recibir de Lola tanto su firma **(R,s)** como el mensaje "**w**". El proceso de verificación es el siguiente:

- Juan calcula $\mathbf{V_1=f(R)K_A+sr}$.
- Además, calcula $\mathbf{V_2=wp.}$
- Si $\mathbf{V_1=V2}$ la firma queda verificada.

El esquema de firma funciona ya que:

- $\mathbf{V_1}$ = f(R) K_A p + sR
  = f(R) k_A P + srP
  = f(R) k_A P + (w - k_A f(R))P
  = wP
  = $\mathbf{V_2}$

4.3. Algoritmo Menezes Vanstone

La solución al problema de codificar un mensaje en un punto es el Criptosistema Menezes-Vanstone de curva elíptica.

Sus características son:

- Utiliza un punto sobre una curva elíptica para "enmascarar" un punto en el plano.
- Funciona sobre **GF (p),** con **p** primo y **p > 3**.
- Es rápido y sencillo.

Una desventaja importante es que debido a apuntar los gastos generales, los mensajes cifrados se duplicaron en longitud.

Si Lola quiere enviar un mensaje a Juan usando su clave pública:

- **Dado**: Lola y Juan jugaran los siguientes convenios, todos los cuales son públicos, se debe:

- Seleccionar un número primo grande **p** (por lo menos debe ser mayor que 3)

- **Fp**: Un campo de Galois de tamaño **p** (**p** es primo, por lo que funciona como aritmética modular)

- **E**: Una curva elíptica sobre **Fp** de la forma **y2 = x3 + ax + b (a, b en Fp)**
- **P**: Un punto seleccionado al azar en **E** (llamado el punto base) que generará subgrupo **H**
- **H**:Un subgrupo de **E** que es preferiblemente del mismo tamaño que **E**

- **Clave privada:** la clave privada de Juan. Sólo él lo sabe.

- **A:** la clave privada de Juan es un número natural seleccionada al azar.

- **Clave Pública:** clave púbico de Juan. Lo ideal sería que se distribuye al mundo.

- **ß:** la clave pública de Juan se calcula como **ß = aP.** Es un punto en **H**.
- **Secreto**: En este esquema, Lola también tiene un secreto.
- **k**: Aleatoriamente seleccionada por Lola.

El proceso de Cifrado se basa en que Lola tiene **m** secreto, que ella divide en **m1** y **m2**, entonces:

- Calcula Lola **(y1, y2) = k ß.**
- Lola calcula $\mathbf{c_0 = kP.}$ ← Dónde c_0 es un punto.
- Lola calcula $\mathbf{c_1 = y_1 m_1 \pmod p}$.
- Lola calcula $\mathbf{c_2 = (y_2 m_2 \bmod p).}$
- Lola envía un mensaje encriptado $\mathbf{C = (c_0, c_1, c_2)}$ para Juan.
- Se debe tener en que **C** es dos veces tan grande como el mensaje original **m**.

Si Juan quiere volver el mensaje **m** de **C**.

- Juan calcula $\mathbf{AC_0 = (y1, y2)}$

- Juan recupera el mensaje **m** calculando:

$$m = (c_1 y_1 \text{-} 1 \pmod p), c_2 y_2 \text{-} 1 \pmod p)$$

4.4. Algoritmo de Firma Digital basado en Curvas Elípticas ECDSA

ECDSA fue propuesto por Scott Vanstone en 1992; fue aceptado en 1998 por como estándar ISO (ISO 14888-3), en 1999 como estándar ANSI (ANSI X9.62), como estándar IEEE (1363-2000) y en 2000 como estándar NIST (FIPS 186-2).

Considerado como una variante del algoritmo ElGamal, ECDSA (Eliptic Curve DSA) es el equivalente al DSA, la diferencia se encuentra en que el lugar de trabajar sobre el subgrupo de orden **n** en **Zp**, se trabaja sobre un grupo de curva elíptica **E (Z_p)**. Si un intruso quisiera descubrir la clave secreta **d**, dada una curva **E** prima de un número primo **n**, entonces el cálculo de **dP** tomaría aproximadamente $\mathbf{2^{n/2}}$ operaciones, es decir que si la clave tiene un tamaño **n** de **192** bits (tamaño más pequeño recomendado por NIST para curvas definidas sobre **GF(p)**), entonces el intruso requerirá de $\mathbf{2^{96}}$ operaciones. Incluso si el intruso tuviera una súper computadora con un performance de un billón de operaciones por segundo, le tomaría alrededor de dos y medio trillones de años para encontrar la clave secreta, y este el problema del algoritmo de curvas elípticas discretas tras ECDSA. (Stadick, 2005) ECDSA es un esquema de firma digital con agregado basado en ECC, cuyo diseño se estima es existencialmente infalsificable, es decir que es improbable para un intruso falsificar una firma sobre algún mensaje que aún no ha sido firmado por la entidad verídica del esquema.

4.4.1. Proceso de firma digital ECDSA

Considerar una curva elíptica **E** definida sobre **Z_p** con número de puntos **E (Z_p)** divisibles para un número primo de gran tamaño **n**.

Lola genera claves ECDSA:

- Seleccionar un punto **P** de orden **n** tal que **P Є (Z_p)**

- Seleccionar un entero aleatorio **d** talque **$1 \leq d \leq n-1$**
- Calcular **$Q = dP$**
- Su clave privada es **d**, mientras que su clave pública es **Q**.

Si Lola desea firmar un mensaje **m**:

- Seleccionar un entero aleatorio **k** talque **$1 \leq k \leq n-1$**
- Calcular **$kP = (x_1, y_1)$** y **$r = x_1 \pmod n$** donde **x_1** es un número entero y si:
    - **$r = 0$**, y se considera la firma **s** igual a **$k^{-1} \{h(m)+ dr\} \pmod n$;** al multiplicar **d** por **$r = 0$** se pierde la seguridad porque no se usa la clave privada **d**. Razón por la que se debe volver al paso anterior.

- Calcular **$k^{-1} \bmod n$**
- Calcular **$s = k^{-1} \{ h(m)+ dr \} \pmod n$**, donde **h(m)** es una función hash que por estándar se especifica que la función debe ser SHA-1.
- Comprobar si s =0, entonces **$s^{-1} \bmod n$** no existe, al ser así volver al paso 1 del como firmar un mensaje.
- La firma para el mensaje m es el par de enteros (r, s).

Si Juan desea verificar la firma sobre el mensaje **m** de Lola:

- Juan debe tener una copia de la clave pública de Lola **(Q)**
- Verificar que **$1 \leq r \leq n-1$** y **$1 \leq s \leq n-1$.** Si no se cumplen estas condiciones finalizar.
- Calcular **$w = s^{-1} \pmod n$** y **h(m)**
- Calcular **$u_1 = h(m)w \pmod n$** y **$u_2 = rw \pmod n$**
- Calcular **$R = u_1P + u_2Q = (x_o, y_o)$** y **$v= x_o \bmod n$.** Si R = o, finalizar.
- Si y solo si **v = r**, aceptar la firma.

Para que cada entidad no genere su propia curva elíptica se puede elegir la misma curva **E (Zp)** y **P** de orden **n** y establecerlas como parámetros del sistema.

CAPÍTULO V

IMPLEMENTACIÓN DE ALGORITMOS DE FIRMA DIGITAL

Los algoritmos de generación de firma digital se implementaron en lenguaje de programación Java Standard Edition 8 actualización 60.

Java es un lenguaje de programación originado en 1990 por Sun Microsystems para pequeños dispositivos electrónicos, sin embargo su lanzamiento oficial se realizó en 1996. Su sintaxis se basa en otros lenguajes de programación como C y C++.

En la actualidad se estima que Java posee más de 9 millones de desarrolladores y habilitado para su uso en diferentes dispositivos. En enero del 2010, Oracle Corporation adquirió Sun Microsytems por lo la tecnología de Java ahora es gestionada por Oracle.

Las versiones aún utilizadas para el desarrollo de aplicaciones Java son:

- Java SE 6 (2006)
- Java SE 7 (2011)
- Java SE 8 (2014)

La seguridad en java se encuentra compuesta por dos elementos:

5.1. JCA (Java Cryptography Architecture)

JCA o Arquitectura Criptográfica de Java presenta interfaces, clases abstractas, modelos para la implementación de los algoritmos criptográficos pero no para otras implementaciones de funcionalidades criptográficas concretas. Por lo que si se desea cifrar de manera simétrica un documento, se necesita hacer uso de JCE, la cual será descrita posteriormente.

Encargado de la generación de firmas digitales y resúmenes de mensajes. Sus principios básicos son la Independencia e interoperabilidad de las implementaciones, es decir que las llaves generadas por una implementación, pueden ser utilizadas por otra y la independencia y extensibilidad de los algoritmos, es decir que, los nuevos algoritmos puedan ser agregados de manera sencilla.

Para mejor entendimiento se requiere el conocimiento de ciertas terminologías como las que se listan a continuación:

- Motor: representación abstracta de un servicio criptográfico que no tiene una implementación concreta.

- Algoritmo: implementación de un motor. Por ejemplo, el algoritmo SHA1 es una implementación del motor de resumen de mensajes. Cada implementación puede diferir dependiendo del código interno que proporcione la clase

- Proveedor: paquete o conjunto de paquetes que proporcionan una implementación concreta de las funcionalidades criptográficas de la API de seguridad de Java

Para conseguir la independencia de las implementaciones se debe establecer una arquitectura fundamentada en proveedores.

Desde la versión de Java SE 7, Oracle incluyó un nuevo motor criptográfico para trabajar con algoritmos de curvas elípticas llamado “SunEC”, el cual permite el uso de llaves con una longitud va desde 112 a 571 bits. El aplicativo diseñado para la comparación del estudio realizado en utiliza este motor con el algoritmo de curva elíptica ECDSA. En el caso de que el proveedor no sea especificado como en el caso de la generación de la firma basada en RSA la consulta se realiza en el listado de proveedores

propio de Java (java.security) hasta encontrar el que le corresponda, lo que significa que utilizará el motor “SunRsaSign” instalado en el equipo de pruebas. En caso de no encontrar ningún motor adecuado se retorna una excepción. Para verificar los proveedores instalados en el ordenador se implementó el siguiente código:

```java
package algortimoRSA;

import java.security.Provider;
import java.security.Security;
import java.util.Enumeration;

public class TestPropiedades {
    @SuppressWarnings("rawtypes")
    public static void main(String[] args) {
        /*
         * listado de proveedores
         */
        Provider[] listaProv = Security.getProviders();
        for (Provider provider : listaProv) {

            System.out.println("Nombre          : " + provider.getName());
            System.out.println("Versión         : " + provider.getVersion());
            System.out.println("Información     :\n   " + provider.getInfo());
            System.out.println("Propiedades     :");

            Enumeration propiedades = provider.propertyNames();
            while (propiedades.hasMoreElements()) {
                String clave = (String) propiedades.nextElement();
                String valor = provider.getProperty(clave);
                System.out.println("   " + clave + " = " + valor);
            }
            System.out.println("----------------");
        }
    }
}
```

Figura 10 Código en lenguaje Java para listar los proveedores existentes

El resultado de la implementación del algoritmo se lo puede apreciar en el Anexo A.

5.2. JCE(Java Cryptography Extension)

JCE o extensión criptográfica de java brinda un framework e implementaciones de los acuerdos de claves, encriptación, y algoritmos de generación de claves y de autenticación de mensajes a partir de las bases

definidas en JCA, es decir que se implementa un proveedor de funciones para generar operaciones criptográficas con nuevos algoritmos de forma sencilla.

El esquema general de la API de seguridad de Java es el siguiente:

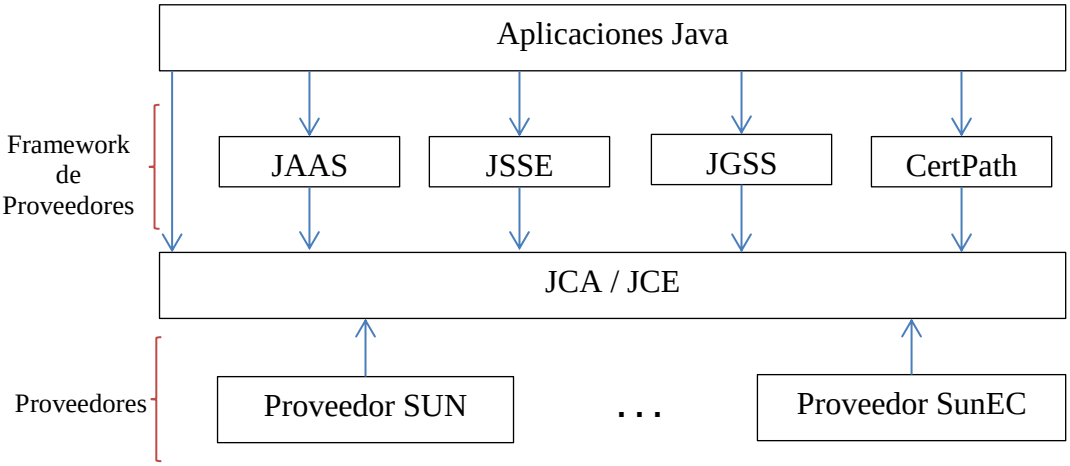

Figura 11 Esquema de seguridad de Java

Fuente: (Zaikin, 2007)

Donde JAAS(Java Authentication and Authorization Service), JSSE(Java Secure Socket Extension), JGSS(Java Generic Secure Services) y CertPath(Java Certification Path API) son frameworks de proveedores.

5.3. Implementación de algoritmos

Los algoritmos a implementarse son el algoritmo de firma digital RSA y el algoritmo basado en curvas elípticas ECDSA.

5.3.1. Simulación del algoritmo de firma digital RSA

Conforme lo analizado en el numeral 3.5 se desarrolló en lenguaje Java la implementación del algoritmo dando como resultado lo que se muestra a continuación.

a) Proceso de Encriptación de la firma digital basada en el algoritmo RSA

Generación de números aleatorios implementando el algoritmo SHA1PRNG del proveedor SUN.

```
SecureRandom aleatorio = SecureRandom.getInstance("SHA1PRNG", "SUN");
```

Basándose en el concepto de JCA, para la generación de un par de llaves (pública y privada) a ser utilizadas para el algoritmo RSA se implementó:

```
KeyPairGenerator genClave = KeyPairGenerator.getInstance("RSA");
//se inicializa la generación de claves como parámetro el tamaño de la //clave en bits y el número aleatorio generado anteriormente
    genClave.initialize(1024, aleatorio);

//se extrae el par de claves generados con la línea anterior
KeyPair pardeClaves = genClave.generateKeyPair();

//se asigna la clave pública y privada según corresponda
PrivateKey clavePrivada = pardeClaves.getPrivate();
PublicKey clavePublica = pardeClaves.getPublic();
```

Implementación del motor Signature para el firmado de datos. getInstance("SHA512withRSA") solicita a la clase java.security.Security que le proporcione el objeto solicitado con el proveedor resultante de la búsqueda en el listado de java.security analizado en punto 5.1.

```
Signature firmaRSA = Signature.getInstance("SHA512withRSA");
```

```
firmaRSA.initSign(clavePrivada);
```

Para el estudio se recibe un archivo llamado “mensaje”, el cual se recorre y es firmado, posteriormente se genera un archivo de salida denominado “firmaRSA”.

```
FileInputStream archivo = new FileInputStream(mensaje);
BufferedInputStream buferEntrada = new BufferedInputStream(
 archivo);
byte[] buffer = new byte[1024];
int longitud;
while (buferEntrada.available() != 0) {
longitud = buferEntrada.read(buffer);
 firmaRSA.update(buffer, 0, longitud);
}

buferEntrada.close();
byte[] firmaReal = firmaRSA.sign();
// Guardando los datos firmados en un archivo
FileOutputStream archivoFirma = new FileOutputStream(
 "firmaRSA.txt");
archivoFirma.write(firmaReal);
archivoFirma.close();
```

Finalmente se escribe la clave pública en un archivo diferente para poder realizar las comparaciones posteriores.

```
byte[] clave = clavePublica.getEncoded();
FileOutputStream archivoClavePublica = new FileOutputStream
("clavePublicaRSA.txt");
archivoClavePublica.write(clave);
archivoClavePublica.close();
```

b) Proceso de Verificación de la firma digital basada en el algoritmo RSA.

Para el proceso de verificación se requiere de tres archivos, el primero que corresponda a la clave pública, el segundo al archivo firmado y por último el mensaje original.

En primera instancia se lee el archivo de clave pública y se emplea X509EncodedKeySpec para codificar en formato X509. Para mayor información respecto al formato del certificado X509 ver Anexo B.

```
FileInputStream clavepublica = new FileInputStream(files[0]);
byte[] clave = new byte[clavepublica.available()];
clavepublica.read(clave);
clavepublica.close();
X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(clave);
```

Empleo del motor KeyFactory que es una clase que convierte llaves criptográficas de tipo key en la llave del tipo especificado.

```
KeyFactory keyFactory = KeyFactory.getInstance("RSA");
PublicKey clavePublica = keyFactory.generatePublic(pubKeySpec);
```

Lectura del archivo firmado utilizando e motor Signature con el mismo proveedor especificado para el proceso de firma realizado anteriormente.

```
FileInputStream archivoFirmado = new FileInputStream(files[1]);
byte[] firmaVerificada = new byte[archivoFirmado.available()];
```

```
archivoFirmado.read(firmaVerificada);
archivoFirmado.close();
Signature firma = Signature.getInstance("SHA512withRSA");
```

Inicialización del proceso de verificación de la firma obtenida.

```
firma.initVerify(clavePublica);
```

Lectura del archivo con el mensaje original para comparación con la firma generada a partir de la clave pública.

```
FileInputStream mensaje = new FileInputStream(files[2]);
BufferedInputStream buferEntrada = new
BufferedInputStream(mensaje);

byte[] buffer = new byte[1024];
int longitud;

while (buferEntrada.available() != 0) {
 longitud = buferEntrada.read(buffer);
 firma.update(buffer, 0, longitud);
}

buferEntrada.close();
//variable que almacena el resultado de la operación
boolean verifica = firma.verify(firmaVerificada);
```

c) Proceso de Encriptación de la firma digital basada en el algoritmo ECDSA

Uso del motor KeyPairGenerator junto con el algoritmo específico para la generación de curvas elípticas “EC” y proveedor “SunEC”.

KeyPairGenerator kpg = KeyPairGenerator.*getInstance*("EC", "SunEC");

La clase ECGenParameterSpec se utliza para especificar los parámetros a utilizarse al generar los parámetros de dominio de curva elíptica. El motor recibe uno de los nombres dados a los estándares de parámetros de dominio especificados en la siguiente tabla.

Tabla 10

Parámetros de dominio de curva elíptica

Tamaño de llave	NIST	SEC		RFC
160	**P-160**	**secp160k1**	**secp160r1**	**Brainpool P160r1**
192	P-192	secp192k1	secp192r1	brainpoolP192r1
224	P-224	secp224k1	secp224r1	brainpoolP224r1
256	P-256	secp256k1	secp256r1	brainpoolP256r1
320	-	-	-	brainpoolP256r1
384	P-384	secp384r1	-	brainpoolP384r1
512	-	-	-	brainpoolP512r1
521	P-521	secp521r1	-	-
169	K-163	sect163k1	sect163r1	-
233	K-233	sect233k1	sect233r1	-
239	-	sect239k1	-	-
283	K-283	sect283k1	sect283r1	-
409	K-409	sect409k1	sect409r1	-
571	K-571	sect571k1	sect571r1	-

Fuente: (Corporation, 2015)

Las instituciones encargadas de la estandarización especificadas en la Tabla 8 se describen a continuación.

NIST (Instituto Nacional de Mediciones de Estados Unidos). Especializados en mediciones, normas y tecnología para incrementar productividad, facilitar el comercio y mejorar la calidad de vida. (W.W Grainger, 2015)

SEC (Standards for Efficient Cryptography). Recomendados por SECG(Standards for Efficient Cryptography Group), una industria para desarrollar normas comerciales que faciliten la adopción de la criptografía y la interoperabilidad eficiente a través de una amplia gama de plataformas de computación. (SECG)

RFC. Documento formal perteneciente de Internet Engineering Task Force (IETF).

Considerando lo descrito en la parte superior para la realización del estudio se tomaron en cuenta diversos estándares según el tamaño de la llave como "sect163k1".

```
ECGenParameterSpec ecsp = new ECGenParameterSpec ("sect163k1");
kpg.initialize(ecsp);
```

Se extrae el par de llaves generadas previamente.

```
KeyPair kp = kpg.genKeyPair();
PrivateKey clavePrivada = kp.getPrivate();
PublicKey clavePublica = kp.getPublic();
```

Implementación del motor Signature para el firmado de datos. getInstance("SHA1withECDSA","SunEC") solicita a la clase

java.security.Security que le proporcione el objeto solicitado del proveedor “SunEC”.

```
Signature firmaECDSA = Signature.getInstance("SHA1withECDSA", "SunEC");
firmaECDSA.initSign(clavePrivada);
```

Para el estudio se recibe un archivo llamado “mensaje”, el cual se recorre y es firmado; posteriormente se genera un archivo de salida denominado “firmaECDSA”.

```
FileInputStream archivo = new FileInputStream(mensaje);
BufferedInputStream buferEntrada = new BufferedInputStream( archivo);
byte[] buffer = new byte[1024];
int longitud;

while (buferEntrada.available() != 0) {
 longitud = buferEntrada.read(buffer);
 firmaECDSA.update(buffer, 0, longitud);
}

buferEntrada.close();
byte[] firmaReal = firmaECDSA.sign();
```

Finalmente se escribe la clave pública en un archivo diferente para poder realizar las comparaciones posteriores.

```
FileOutputStream archivoFirma = new FileOutputStream( "firmaECDSA.txt");

archivoFirma.write(firmaReal);
```

```
archivoFirma.close();

//Guardando la clave pública en un archivo

byte[] clave = clavePublica.getEncoded();
FileOutputStream archivoClavePublica = new FileOutputStream(
                    "clavePublicaECDSA.txt");

archivoClavePublica.write(clave);
archivoClavePublica.close();
```

d) Proceso de Verificación de la firma digital basada en el algoritmo ECDSA

Para el proceso de verificación se requiere de tres archivos, el primero que corresponda a la clave pública, el segundo al archivo firmado y por último el mensaje original.

En primera instancia se lee el archivo de clave pública y se emplea X509EncodedKeySpec para codificar en formato X509.

```
//para manejo de la clave publica
FileInputStream clavepublica = new FileInputStream(files[0]);
byte[] clave = new byte[clavepublica.available()];

clavepublica.read(clave);
clavepublica.close();

X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec
(clave);
```

Empleo del motor KeyFactory que es una clase que convierte llaves criptográficas de tipo key en la llave del tipo especificado.

```
KeyFactory keyFactory = KeyFactory.getInstance("EC", "SunEC");

PublicKey clavePublica= keyFactory.generatePublic(pubKeySpec);
```

Lectura del archivo firmado utilizando e motor Signature con el mismo proveedor especificado para el proceso de firma realizado anteriormente.

```
FileInputStream archivoFirmado = new FileInputStream(files[1]);

byte[] firmaVerificada = new byte[archivoFirmado.available()];

archivoFirmado.read(firmaVerificada);

archivoFirmado.close();

Signature firma = Signature.getInstance("SHA1withECDSA","SunEC");
```

Inicialización del proceso de verificación de la firma obtenida.

```
firma.initVerify(clavePublica);
```

Lectura del archivo con el mensaje original para comparación con la firma generada a partir de la clave pública.

```
FileInputStream mensaje = new FileInputStream(files[2]);

BufferedInputStream buferEntrada = new BufferedInputStream(mensaje);

byte[] buffer = new byte[1024];
```

```
int longitud;

while (buferEntrada.available() != 0) {

        longitud = buferEntrada.read(buffer);

    firma.update(buffer, 0, longitud);

}

buferEntrada.close();

//variable que almacena el resultado de la operación
boolean verifica = firma.verify(firmaVerificada);
```

El aplicativo desarrollado para la implementación de los procesos: a), b), c) y d) puede ser apreciado en el Anexo C

5.4. Pruebas y Evaluación complementaria

Por motivos de estudio para ECC se ha considerado los dominios de parámetros de curva elíptica sobre **Zp** recomendados por SEC, listados en la Tabla 11.

Tabla 11

Propiedades de parámetros de dominio de curva elíptica recomendados sobre Zp

Parámetros de dominio de curva elíptica	Fuerza / Nivel de Seguridad (56-256)	ECC (bits)	RSA (bits)
secp192k1/ secp192r1	96	192	1536

CONTINÚA →

secp224k1/ secp224r1	**112**	**224**	**2048**
secp256k1/ secp256r1	128	256	3072
secp384r1	192	384	8192
secp521r1	256	521	15360

Fuente: (Corp, 2010)

Como se puede observar en la **Tabla 11** cada propiedad de parámetro de dominio de curva elíptica posee un pseudónimo compuesto por “sec” que denota “Standards Of Efficient Cryptography”, seguido por P que denota parámetros sobre curva **Zp,** seguido por un número que denota la longitud en bits del campo **p**, seguido por **k** que denota asociación con una curva Koblitz o **r** para denotar parámetros aleatorios verificables seguidos por un número de secuencia.

El detalle de los parámetros listados se puede apreciarse en el Anexo D.

Para el caso de la firma digital RSA se especifica el tamaño de la clave en la siguiente línea.

```
genClave.initialize(1024, aleatorio);
```

Además de los tamaños de clave especificados se utilizó un archivo de 1.53 Gb (1.647.085.568 bytes) dando como resultado la siguiente tabla de resumen:

Tabla 12

Resumen de generación y verificación de firma digital

Tamaño de Clave (bits)	Estándar	ECDSA					Equivalente RSA (bits)	RSA			
		Generación de la Firma			Verificación de la Firma			Generación de la Firma			Verificación de la Firma
		Tiempo ECDSA (ms)	Tam. Clave Pública ECC(bits)	Tam. Firma ECC (bits)	Tiempo ECDSA (ms)			Tiempo RSA (ms)	Tam. Clave Pública RSA (bits)	Tam. Firma RSA (bits)	Tiempo RSA (ms)
192	secp192r1	11072	600	440	10992		1536	11404	1808	1536	10974
224	secp224r1	11380	640	512	11329		2048	12006	2352	2048	10992
256	secp256k1	11307	704	568	11135		3072	12439	3376	3072	10945
384	secp384r1	11248	960	832	11098		7680	54656	7984	7680	11020
521	secp521r1	11313	1264	1112	11176		15360	1225960	19588	15360	11029

Para cada tamaño de firma se realizó 5 iteraciones con el mismo tamaño de archivo tanto para ECDSA como para la firma basada en RSA, de cada iteración se obtuvo el promedio a ser comparado dando como resultante las siguiente gráficas.

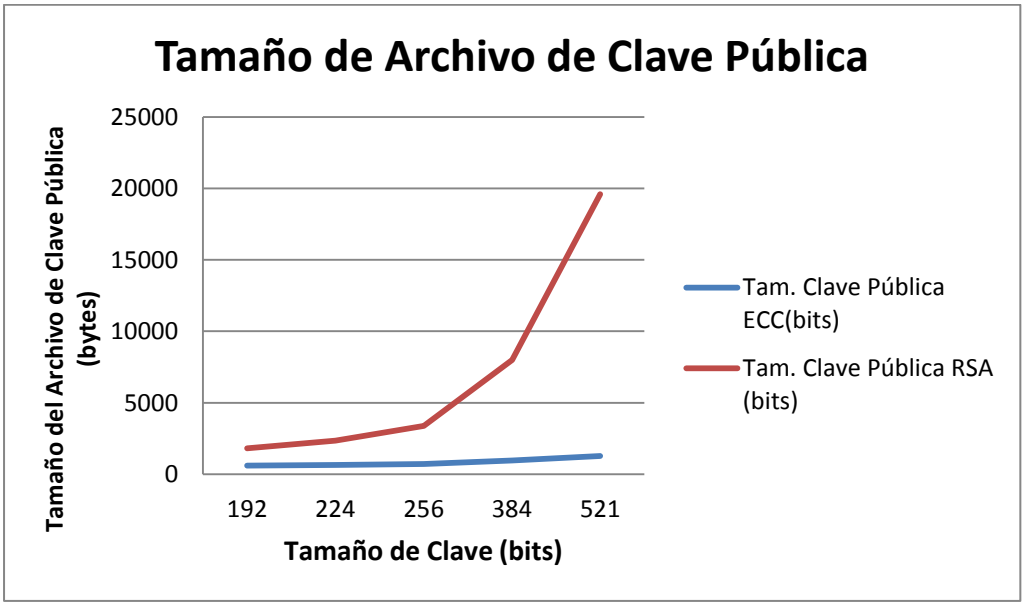

Figura 12 Comparación de archivos de clave pública generados

La Figura 12 nos indica que el tamaño de clave pública generado por el algoritmo RSA es mayor a su equivalente ECDSA y su diferencia se incrementa de manera notoria conforme incrementa el tamaño de clave utilizada.

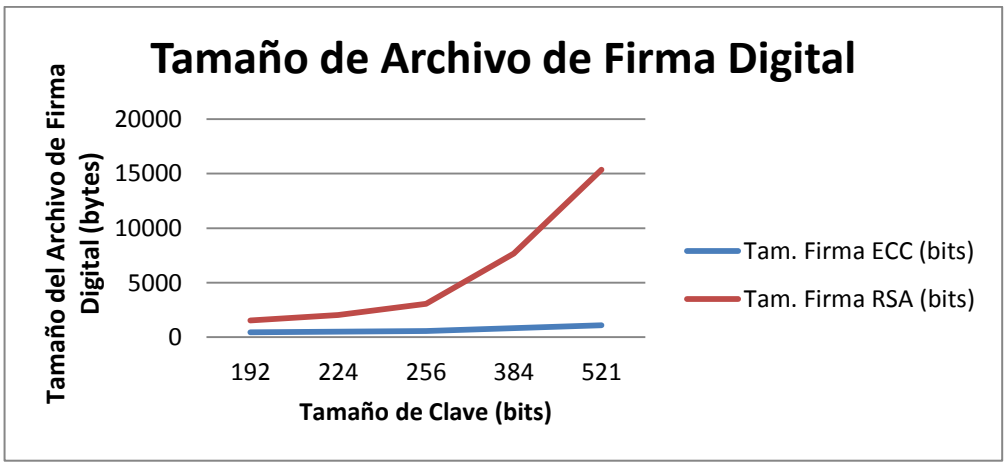

Figura 13 Comparación de archivos de firma digital generada

La Figura 13 nos indica que el tamaño de firma digital generada por el algoritmo RSA es mayor a su equivalente ECDSA y al igual que en la Figura 10 su diferencia se incrementa de manera notoria conforme incrementa el tamaño de clave utilizada.

En la actualidad la mayoría de aplicaciones están orientadas a la web o dispositivos móviles por lo que es importante que el tamaño de los archivos generados sea el menor posible para mejorar los tiempos de transaccionalidad, y es en estos ámbitos donde representa una ventaja poseer archivos de menor tamaño que no saturen la red.

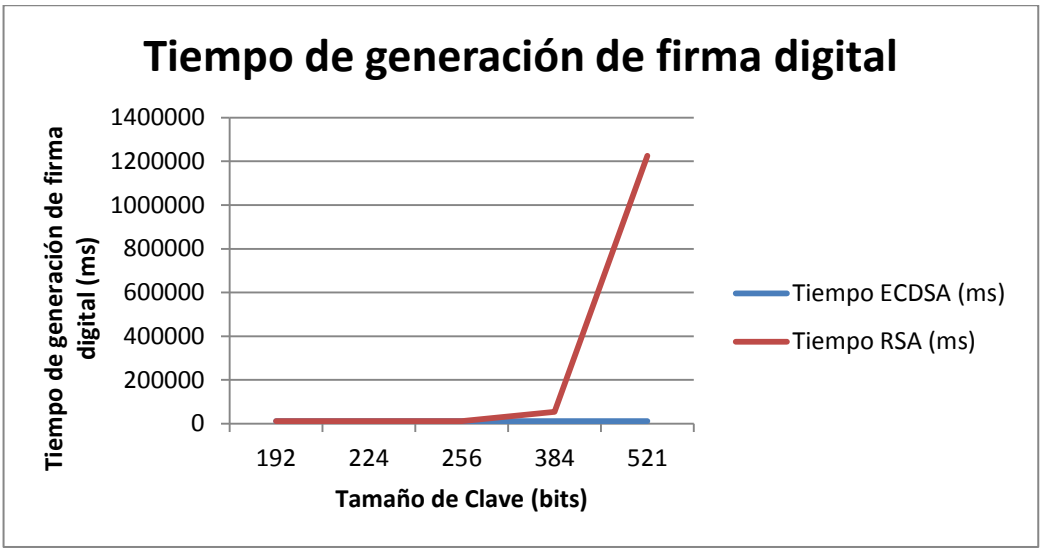

Figura 14 Comparación de Tiempo en generación de firma digital

Continuando con la argumentación de la necesidad de un mejor tiempo de respuesta, una vez más la Figura 14 muestra que la generación de firmas digitales es ejecutada de manera más rápida y su tiempo no varía significativamente conforme incrementa el tamaño de la clave, contrario a lo que sucede con la firma digital generada por RSA en los que se puede apreciar que con niveles más altos de seguridad más se demora en generar la firma.

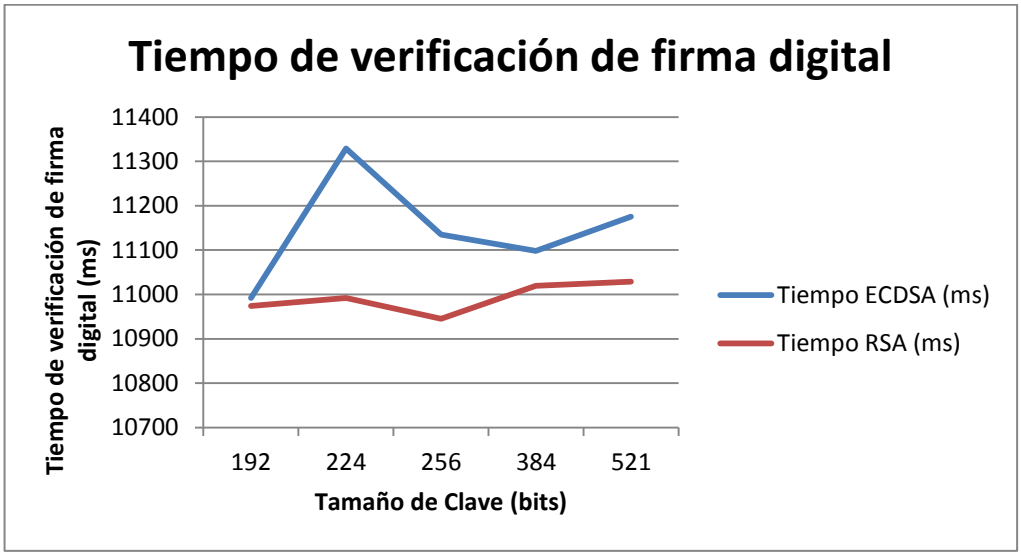

Figura 15 Comparación de Tiempo de verificación de firma digital

Respecto a la verificación de las firmas digitales generadas según lo muestra la Figura 15, es más compleja al tratarse de ECDSA incluso conociendo la clave pública que es el caso de estudio que se ha realizado. Lo que significa que es más seguro por el grado de dificultad que representa su descifrado.

El detalle de la tabla de resultados puede ser apreciado de mejor manera en el Anexo E.

5.5. Usos de firmas digitales ECC

A lo largo del tiempo con el afán de mejorar la seguridad de las aplicaciones se han desarrollado nuevas técnicas evitar la interrupción de intrusos como lo son los algoritmos de curva elíptica.

Un claro ejemplo es la “Implementación Del Esquema De Firma Digital Ecdsa Sobre El Procesador Embebido Microblaze” elaborada por los ingenieros Roberto González González, MSc. Humberto Díaz Pando e Ing. Ander Torres López, el propósito de este aplicativo es la implementación del Estándar de Firma Digital con Curvas Elípticas (ECD-SA) sobre un

dispositivo de hardware reconfigurable tipo FPGA, y se propone una arquitectura que sirve de base a la implementación del estándar de firma con curvas sobre el procesador de propósito general Microblaze. (Ing. Roberto González González, 2014)

Otro ámbito para su utilización es la protección a los protocolos de autenticación para redes inalámbricas demostrando que ECC es una opción criptográfica más efectiva para la implementación de WTLS con tiempos de ejecución más rápidos que RSA.

5.5.1. SSL (Secure Socket Layer)

Es un protocolo que tiene por objetivo asegurar la transmisión y recepción de información a través del cifrado y descifrado respectivamente de los datos enviados.

Existen varias aplicaciones configuradas para recibir de manera automática conexiones SSL tales como los navegadores web, entre ellos Google Chrome o Mozilla Firefox, aplicaciones de correo electrónico como Outlook, entre otros. Para que estas aplicaciones establezcan una conexión de tipo SSL deben tener una clave de cifrado en forma de Certificado asignada por una autoridad certificadora. Un certificado electrónico es la base para la firma digital.

A nivel comercial, este protocolo se utiliza mayoritariamente para asegurar transacciones monetarias como las compras, de esta manera se asegura al cliente que la información ingresada mantenga su privacidad. (DigiCert).

Un ejemplo claro de SSL puede ser apreciado en los correos electrónicos institucionales como lo es el de la Universidad de las Fuerzas Armadas ESPE, de donde se ha extraído las siguientes imágenes:

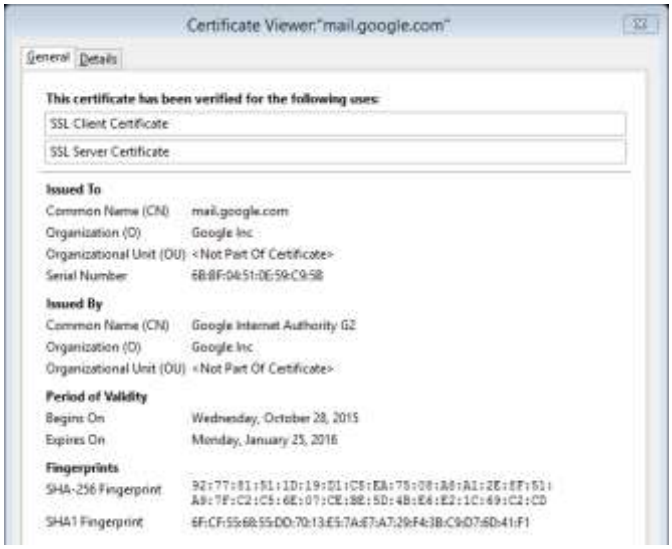

Figura 16 Vista del Certificado emitido por Google Inc.

La Figura 16. Indica la entidad que emitió el certificado, a quién se ha emitido, el período de validez y las “huellas dactilares” (impresión digital del certificado o hash)

Figura 17 Detalle del Certificado Digital

La Figura 17. Indica el tipo de algoritmo de encriptación utilizado para firmar el certificado, el cual es SHA-256 con RSA. Adicionalmente en la parte superior se puede observar la estructura jerárquica de la autoridad de certificación.

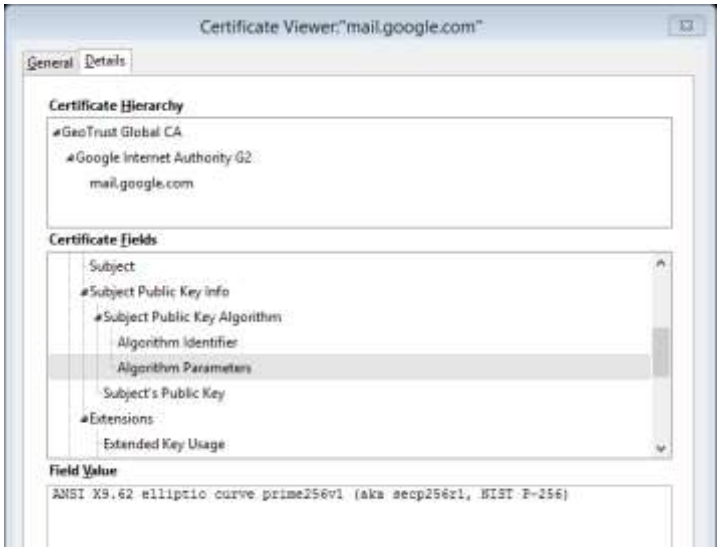

Figura 18 Parámetros del Algoritmo utilizado por la entidad certificadora

La Figura 18 indica que el algoritmo del cual proviene la clave pública, que para este caso se trata de un algoritmo de curva elíptica de 256bits, con su parámetro de dominio secp2561r1.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

Los algoritmos de firma digital basados en curvas elípticas son matemáticamente más complejos por lo que el proceso de descifrado en caso de un ataque es más demoroso, brindando así mayor seguridad de la información.

En la actualidad la mayoría de aplicaciones están orientadas a la web o dispositivos móviles por lo que es importante que el tamaño de los archivos generados sea el menor dado para mejorar los tiempos de transaccionalidad, y se ha comprobado con el presente estudio que el algoritmo de curva elíptica ECDSA es quien realiza este requerimiento posible en comparación con el algoritmo RSA.

ECDSA necesita un menor tamaño de clave para ofrecer el mismo nivel de seguridad brindado por RSA.

Los tiempos de generación de la firma digital obtenidos por el algoritmo RSA son altos en comparación son los obtenidos por ECDSA y su diferencia se incrementa notoriamente proporcionalmente al tamaño de clave.

6.2. Recomendaciones

Si el objetivo de la aplicación es respuesta inmediata y segura es recomendable el uso de algoritmos de curvas elípticas como ECDSA sobre todo en ambientes Java donde su utilización se la puede hacer de manera sencilla y su implementación no varía drásticamente si se desea seleccionar un algoritmo diferente durante el proceso de desarrollo.

Por otro lado si la el aplicativo se encuentra en un servidor robusto donde el tiempo de respuesta no es primordial se puede utilizar firma digital basada en RSA ya que su clave al ser más grande se torna más difícil de resolver o a su vez más demorosa.

Trabajos citados

Amaya, L. A. (2014). *Memorias de Seminario El algoritmo de Elgamal.* Colombia.

Belingueres, G. (s.f.). *Introducciòn a los criptosistemas de curva elíptica.*

Bragaglia, E. (2012). *Tesis di Laurea in Algoritmi della Teoria dei Numeri e Criptografia.* Bologna: Università di Bologna.

Corp, C. (2010). SEC 2: Recommended Elliptic Curve Domain Parameters. *Standards For Efficient Cryptography*, 4-12.

Corporation, O. (2015). *Class ECStandardizedParameterFactory*. Recuperado el 01 de 08 de 2015, de http://javadoc.iaik.tugraz.at/ECCelerate/current/iaik/security/ec/common/ECStandardizedParameterFactory.html

Cristina Alcaraz, R. R. (s.f.). *Análisis de Primitivas Criptográficas para Redes de Sensores.* Málaga.

Darrel Hankerson, A. M. (s.f.). *Guide to Curve Cryptography.* Library of Congress Cataloging in Publication Data.

DigiCert. (s.f.). *Capa de conexión segura SSL*. Recuperado el 01 de 09 de 2015, de https://www.digicert.com/es/ssl.htm

Grajales, T. (s.f.). *Tipos de Investigación*. Recuperado el 17 de 04 de 2014, de http://www.iupuebla.com/Maestrias/M_E_GENERO/MA_Maestria_Genero/Jose_Miguel_Velez/Tipos%20de%20investigacion.pdf

Ing. Roberto González González, M. H. (2014). IMPLEMENTACIÓN DEL ESQUEMA DE FIRMA DIGITAL ECDSA SOBRE EL PROCESADOR EMBEBIDO MICROBLAZE. *Revista Telem@tica*, 31-32.

Jesús J. Ortega Triguero, M. Á. (2006). *Introducción a la Criptografía Historia y Actualidad.* La Mancha: Ediciones de la Universidad de Castilla.

Koblintz, N. (1994). *A course in Number Theory and Cryptography, segunda edición.*

Lacalle, J. G. (2004). El diablo de los números. *La Gaceta de la RSME*, 515-537.

Marcelo Andrés Saravia Gallardo, M. A. (s.f.). Metodología de Investigación Científica. Valle de méxico.

Miguel Alfredo Acedo Arias, M. A. (2008). *Análisis de los secretos compartidos para la autenticación de nodos en las wireless sensor networks mediante el algoritmo de shamir.* Bogotá.

Montiel, L. I. (2003). *Estudio, diseño y evaluación de protocolos de autenticación para redes Inalámbricas.* México, D.F: Centro de Investigación y estudios avanzados del Instituto Politécnico Nacional.

Paredes, G. G. (2006). Introducción a la Criptografía. *Revista digital universitaria*, 17.

Paredes, G. G. (10 de Julio de 2006). Introducción a la Criptografía. *Revista Digital Universitaria*, 17.

Pino Caballero Gil. (s.f.). *Introducción a la Criptografía.* Alfaomega.

postgrado, i. p. (s.f.). *Metodología de la investigación para elaborar una tesis*. Recuperado el 20 de 04 de 2014, de http://www.slideshare.net/juanchojuancho/metodologa-de-investigacin-para-elaborar-una-tesis-25826565?qid=908c70e7-b3c8-4337-be2c-0fab403786ea&v=qf1&b=&from_search=10

Santiago Fernandez. (2004). *Criptografía Clásica.*

SECG. (s.f.). *Standards for Efficient Cryptography Group*. Recuperado el 01 de 08 de 2015, de http://www.secg.org/

Stadick, R. (2005). *Java Implementation of th Elliptic Curve Digital Signature Algorithm Using NIST Curves Over GF(p).* Oregon .

Talens-Oliag, S. (s.f.). *Introducción a los certificados digitales*. Recuperado el 01 de 11 de 2015, de http://www.uv.es/sto/articulos/BEI-2003-11/certificados_digitales.html

TENA, J. (2003). *Protocolos Criptográficos y Seguridad en Redes.* España: Servicio de Publicaciones de la Universidad de Cantabria.

Tilborg, H. C. (2005). Encyclopedia of Cryptography and Security. Springer.

Trujillo, R. T. (2011). *PROTOCOLO CRIPTOGRÁFICO*.

Vasquez, L. (2006). *Optimización de la eficiencia de los esquemas de compartición secretos.* España: Universidad Politécnica de Cataluña.

W. K . Wootters, W. H. (1982). *A single quantum cannot be cloned.*

W.W Grainger, I. (2015). *Grainger*. Recuperado el 01 de 08 de 2015, de http://espanol.grainger.com/content/qt-nist-and-iso-194

Wilder, M. (2006). *Test de primalidad, aplicación a la criptografía.* Palermo.

William Raúl García Muñoz, A. V. (2005). Estudio técnico para la implementación de una entidad certificadora para el CTT – ESPE CECAI. Sangolquí.

Zaikin, M. (2007). *SCEA 5 Study Guide*. Recuperado el 01 de 08 de 2015, de http://java.boot.by/scea5-guide/index.html