INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD CULHUACÁN

# *SISTEMA DE AUTENTICACIÓN PARA IMÁGENES DIGITALES IMPLEMENTADO EN HARDWARE DEDICADO MEDIANTE ALGORITMOS CRIPTOGRÁFICOS.*

**TESIS**

**QUE PARA OBTENER EL TITULO DE:**

INGENIERO EN COMPUTACIÓN

PRESENTA

**ALEXIS JIMÉNEZ CALZADILLA**

ASESOR DE TESIS

DR. ROGELIO REYES REYES

Ciudad de México, junio de 2019

**IPN**
**ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA**
**UNIDAD CULHUACAN**

**TESIS INDIVIDUAL**

Que como prueba escrita de su Examen Profesional para obtener el Título de **INGENIERO EN COMPUTACIÓN** deberá desarrollar el C.:

**ALEXIS JIMÉNEZ CALZADILLA**

**"SISTEMA DE AUTENTICACIÓN PARA IMÁGENES DIGITALES IMPLEMENTADO EN HARDWARE DEDICADO MEDIANTE ALGORITMOS CRIPTOGRÁFICOS."**

Dentro de la seguridad informática el sistema propuesto proveerá la autenticidad de la imagen digital mediante una firma digital generada mediante la función hash SHA-256 y a su vez, la integridad de la misma mediante el algoritmo ECDSA que cuenta con llaves de longitud de 128 bits. El uso de hardware dedicado para la autenticación de imágenes digitales reducirá el tiempo de procesamiento en comparación a la ejecución en hardware de uso compartido.

Capitulado:

| | |
|---|---|
| | INTRODUCCIÓN. |
| CAPÍTULO I.- | ANTECEDENTES. |
| CAPÍTULO II.- | MARCO TEÓRICO. |
| CAPÍTULO III.- | DESARROLLO DEL SISTEMA PROPUESTO. |
| CAPÍTULO IV.- | PRUEBAS Y RESULTADOS. |
| | CONCLUSIONES. |

Ciudad de México, a 27 de mayo de 2019.

**DR. ROGELIO REYES REYES**
**ASESOR**

**M. EN C. JOSE ANTONIO LOAIZA BRITO**
**JEFE DE LA CARRERA DE I.C.**

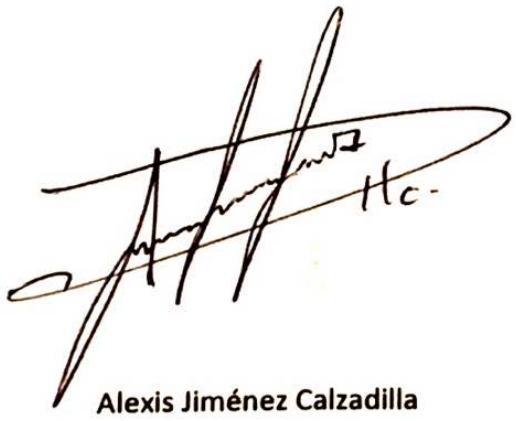
**DR. EUSEBIO RICARDEZ VÁZQUEZ**
**SUBDIRECTOR ACADÉMICO INTERINO**

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD CULHUACAN
SUBDIRECCIÓN ACADÉMICA
OFICINA DE TITULACIÓN PROFESIONAL

**CARTA DE AUTORIZACIÓN DE USO DE OBRA**

En la Ciudad de México, a **27** de **mayo** del año **2019**, el que suscribe **ALEXIS JIMENEZ CALZADILLA** alumno de la carrera de **Ingeniería en Computación**, con número de registro **R-027/19** , egresado de la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Culhuacán, manifiesto que soy el autor intelectual del presente trabajo de **Tesis,** bajo la asesoría del **DR. ROGELIO REYES REYES,** y autorizo el uso del trabajo titulado **"SISTEMA DE AUTENTICACIÓN PARA IMÁGENES DIGITALES IMPLEMENTADO EN HARDWARE DEDICADO MEDIANTE ALGORITMOS CRIPTOGRÁFICOS"**, al **Instituto Politécnico Nacional**, para su difusión con fines académicos y de investigación.

Los usuarios de la información no deberán reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o asesor del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección de correo: **alex-hc23@hotmail.com** Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Alexis Jiménez Calzadilla

Nombre y firma del alumno

# RESUMEN

Actualmente los avances en la tecnología nos permiten crear, almacenar y compartir información digital a través de dispositivos electrónicos con distintos tipos de formatos (de texto, audio, video e imagen) según sea la necesidad del usuario. Donde las imágenes se han popularizado debido a su facilidad de presentar información, su facilidad de ser creadas y la disponibilidad que tienen para ser consultadas [1].

Dentro de los avances tecnológicos también se ha creado software especialmente diseñado para manipular y modificar las características de las imágenes provocando la necesidad de proteger la originalidad del archivo para evitar falsificaciones o alteraciones de la información original

Para eso existen distintos tipos de sistemas o software que ayudan a proteger la originalidad de las imágenes con técnicas como marcas de agua, técnicas estenográficas o criptográficas; siendo estas últimas las más practicas al no necesitar modificar la información de la imagen original. Sin embargo, las técnicas robustas de criptografía requieren mayor procesamiento para su implementación, haciendo que los equipos que usan hardware de uso compartido requieran más tiempo para la implementación de la técnica [2].

Por lo tanto, este trabajo propone el uso de métodos criptográficos robustos para realizar la autenticación de la imagen utilizando una firma digital y para proteger la integridad de la función hash de la imagen, implementándolos en hardware dedicado para reducir el tiempo de procesamiento, permitiendo al usuario contar con la seguridad de que estará protegida la originalidad de su imagen, además de la identificación de la identidad del emisor de la imagen.

# Índice

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# Capítulo 1: Introducción

La seguridad informática se refiere a un conjunto de estrategias para gestionar herramientas y políticas necesarias para prevenir, detectar y contrarrestar amenazas a la información, es decir, sirve para mantener la integridad y privacidad de la información almacenada en medios informáticos ante la constante evolución de la tecnología y su globalización que parece un rubro incesante que genera la necesidad de incorporarse y hacer uso de los dispositivos que nos provee [1]. Toda esa gamma de dispositivos, con la que podemos crear y modificar toda clase de archivos digitales (imágenes, video o audio) presentan vulnerabilidades, por lo que es importante emplear alguna técnica que nos permita proteger archivos que contengan información de carácter personal.

Para satisfacer esta necesidad se ha creado una ciencia llamada criptografía, la cual se dedica a la investigación de técnicas que puedan proteger la integridad de la información donde solo el destino deseado pueda descifrarla. Con el tiempo se han diseñado diferentes sistemas de cifrado con variadas técnicas de índole matemática cumpliendo con la confidencialidad, integridad y autenticación.

La alternativa de autenticación, que se caracteriza por usar algoritmos que sirvan como una firma digital, resumen la información que contiene el archivo en una cadena de longitud establecida empleando el uso de un arreglo de compuertas lógicas y protegiendo su integridad implementando un algoritmo con el que se pueda validar su originalidad.

Esta es una buena estrategia debido a que las firmas digitales ocupan un espacio mínimo en memoria, pero cabe destacar que ese tipo de algoritmos están basados en problemas matemáticos de alta complejidad operacional para los sistemas y requieren de un gran procesamiento. A pesar de eso, es necesario emplearlos ya que, así como avanzan las técnicas criptográficas también avanzan los métodos para poder esquivarlos.

## 1.1 Planteamiento del problema

Al utilizar canales para compartir información, las alteraciones en archivos digitales (en este caso imágenes) pueden afectar al autor original si estas representan una fuente de información única y de uso en su trabajo; por ejemplo, si una aseguradora que maneja muchas imágenes (como pruebas de acontecimientos) necesitara de autenticar sus imágenes tras compartirlas con otras agencias vía internet y podría hacerlo con una firma digital para autenticar la originalidad de la imagen al llegar al receptor.

Un algoritmo óptimo para este propósito es el *algoritmo de firma digital con curva elíptica (ECDSA)* que utiliza valores hash de 256 bits solo que, debido a su gran complejidad matemática, al ser aplicado en imágenes requiere de mayor tiempo de procesamiento, por lo que la solución viable para agilizar los tiempos de procesamiento es utilizar un hardware dedicado para su implementación en comparación con hardware de uso compartido.

## 1.2 Objetivos

### 1.2.1 Objetivo general

Implementar un sistema de firma digital para autenticar imágenes digitales en una beaglebone black mediante funciones hash SHA-256 y el algoritmo ECDSA con la finalidad de verificar la integridad del archivo.

### 1.2.2 Objetivos específicos

- Programar la Beagle bone para el procesamiento de imágenes.
- Programar el sistema criptográfico con el hardware dedicado.
- Evaluar los resultados del sistema con un conjunto de imágenes de diferentes características.
- Optimizar el sistema conforme a los problemas que se vayan presentando.

## 1.3 Justificación

Dentro de la seguridad informática el sistema propuesto proveerá la autenticidad de la imagen digital mediante una firma digital generada mediante la función hash SHA-256 y a su vez, la integridad de esta mediante el algoritmo ECDSA que cuenta con llaves de longitud de 128 bits.

El uso de hardware dedicado para la autenticación de imágenes digitales reducirá el tiempo de procesamiento en comparación a la ejecución en hardware de uso compartido.

# Capítulo 2: Antecedentes

La autenticación de imágenes se ha vuelto importante para poder determinar si una imagen al viajar por un canal de comunicaciones no sufrió alguna alteración y también poder autenticar al emisor de la imagen. Eventualmente existen técnicas y sistemas que permiten realizar el proceso de autenticación, como son marcas de agua (visibles o no visibles), funciones hash o firmas digitales. De las anteriores técnicas existen diferentes estándares propuestos por organismos como el NIST, ANSI e IEEE que fueron avaladas después de exámenes rigurosos de fuerza bruta o técnicas que vencieron a anteriores versiones de algoritmos, permitiendo a investigadores crear nuevos sistemas utilizando estos algoritmos, combinándolos o incluso proponiendo nuevas mejoras. Tal el ese el caso del trabajo "*Geomtrically robust image hashing scheme for image authentication*" de Y. Wo y B. Zhang que describe un proceso de crear funciones hash para autenticar imágenes a través de la transformada exponencial compleja polar con la cual detectan los momentos de la imagen que no tienen varianza si sufre rotaciones o escalamiento, con los que obtienen una función hash robusta con la que se pueden autenticar imágenes, aunque estas hayan sufrido cambios geométricos [2].

Para la realización del proceso se calculan los PCETs de la imagen y luego se normalizan a través de la ecuación (1).

$$M_{nl}^{N} = 100x \frac{M_{nl}}{M_{00}} \tag{1}$$

Este proceso es necesario para lograr que los PCETs tengan invariancia en la escala y lograr que sean invariables en rotación y escalamiento con los cuales se creará la función hash. Una vez obtenidos los PCET serán usados como hash intermedio, posteriormente se cuantiza dicho hash por el método cuantización adaptativa determinística para obtener el valor de cuantización. Se codifica el valor de cuantización con el código Gray y se obtiene la función hash final H.

Finalmente, para el proceso de autenticación se aplica el mismo proceso a la imagen recibida y se obtendrá una función hash H0 que será comparada con la función H a través de la distancia de Hamming. Entre más grande sea la distancia entre las funciones menor será su similitud.

Este sistema funciona concretamente para imágenes y utiliza un método hash que no tiene un estándar avalado por ningún instituto ya que no tiene las características y delimitaciones con respecto a colisiones.

En el trabajo "PNG image Copyright protection and authentication using SVD hash and AES" de A. Jain y V. Jain [3] se muestra una forma de hacer autenticación sobre imágenes PNG, que tienen la característica de tener transparencia a través de un canal Alpha. En este articulo utilizan una combinación entre el AES (Advanced Encryption Standard) que es un estándar avalado por el *National institute of standards and technology (NIST)* y una transformación geométrica para imágenes llamada SVD (Singular Value Descomposition). Esta última es una matriz que guarda los valores de energía de una señal en los menores coeficientes que se pueda.

La idea entonces será guardar un mensaje de derechos de autor dentro de la imagen que no sea perceptible a simple vista, por lo cual utilizaran la técnica del bit menos significativo para insertar la información dentro de la imagen huésped sobre el canal Alpha de la imagen. La información guardada será cifrada con el algoritmo AES y el SVD, después se creará un token que determinará la ubicación del mensaje dentro de la imagen.

En el trabajo "A novel robust image-hashing method for content authentication" de C. Jiang, Y. Pang y A. Wu se desarrolla un método para crear valores hash de una imágen a través de la transformada de Wavelet y GA-BP network (modelo utilizado para generar funciones hash para imágenes) donde la transformada de Wavelet se utiliza para detectar las frecuencias contenidas en la imagen con las cuales se crearan matrices que funcionaran como entrada para el GA-BP network, el cual dará como salida un vector lineal, este será la función hash de la imagen. Posteriormente para el proceso de verificación, también se calculará la función a la imagen recibida y se calculará su distancia con Hamming [4].

Finalmente, en el trabajo “A secure, self-recovery, and high capacity blind digital image information hiding and authentication scheme using DCT moments” de O. Habbouli y D. B. Megherbi se conjuntan dos técnicas criptográficas: esteganografía y marcado de agua usando el dominio de la transformada discreta del coseno. Como entrada se tendrán tres imágenes: la portadora, una imagen que se usará como marca de agua y una imagen que será ocultada. Donde la imagen como marca de agua servirá para verificar la autenticidad de la imagen oculta y a su vez para detectar manipulaciones en caso de que existiera.

La imagen oculta y la marca de agua son divididas en bloques de 16x16 cada una y se les aplica la transformada discreta del coseno de los cuales solo se seleccionan veintiocho coeficientes DCT que son guardados en bloques de 8x8 arbitrarios. Después se reorganizan los bloques en cuatro diferentes cuadrantes de 256x256 que resultan como cuatro redundantes imágenes DCT de 128x128; obteniendo una imagen del mismo tamaño que la portadora.

Una vez creados los bloques DCT se introducen dentro de la imagen portadora a través de un proceso de escalamiento y factores de peso e iluminación. Esto permitirá que, si la imagen recibiera manipulaciones, existirán cuatro componentes DCT que permitirían verificar la imagen, ya que el proceso de autenticación requiere extraer las marcas de agua y reconstruir los bloques DCT para compararlos y así comprobar si la imagen es auténtica y si recibió manipulaciones [1].

# Capítulo 3: Marco teórico

## 3.1 Imágenes digitales

Una imagen es una captura de información que funciona como una representación de un evento físico en forma de una matriz de tamaño MxN, donde M representa el número de columnas y N el número de filas y cada coordenada formada es conocida como pixel, el cual almacena la información del tono o luminosidad captado, donde el tono blanco es el valor 255 y el negro el valor 0 que es representado por un máximo de tres bits, aunque también existen imágenes binarias donde solo se tienen dos niveles de luminosidad, blanco y negro reflejados como un 1 o un 0 [4].

Hay tres tipos de imágenes digitales principales, clasificadas dependiendo de la escala de colores que utilicen y son:

1. Imágenes blanco y negro (binarias)
2. Imágenes en escala de grises
3. Imágenes a color (RBG)

### 3.1.1 Imágenes binarias

Las imágenes binarias conocidas como en blanco y negro son representadas por una matriz de tamaño MxN que solo contiene dos niveles de iluminación representados por el cero para el negro y el uno para el color blanco.

### 3.1.2 Imágenes en escala de grises

Las imágenes en escala de grises son una matriz de M x N, en la que cada valor en un píxel está determinado por un nivel de gris. El rango de colores es representado por tres bits y va de 0 a 255, donde el 0 representa el negro y el 255 el blanco, el resto de los valores intermedios son tonos de grises.

### 3.1.3 Imágenes a color

Las imágenes a color están conformadas por tres matrices de tamaño M x N. Donde cada matriz corresponde a un color de los tres que componen el modelo RGB (rojo, verde y azul), el rango de iluminación para cada matriz de color también es de 0-255 y con combinaciones entre estas tres se producen imágenes a color.

Debido a esto, el modelo RGB es considerado como un formato de color aditivo, puesto que la creación de colores depende de las componentes individuales de cada matriz. Este proceso puede imaginarse como el traslape de tres hojas con translucidas con cada color RGB, La intensidad de las componentes (x, y) de cada hoja determinara tanto el color resultante como su luminosidad.

## 3.2 Formatos de imágenes digitales

Existen distintas aplicaciones para imágenes digitales, en unas se requiere que la imagen conserve la calidad y tamaño que tiene para percibir los detalles o apreciar la variedad de colores que tiene y otras donde las imágenes solo sirven como una ilustración por lo que se pueden comprimir a través de algún algoritmo para reducir su tamaño en memoria.

Para todas estas posibilidades existen distintos algoritmos que comprimen las imágenes (algunos con pérdidas y otros sin) ya que una imagen digital puede ocupar mucho espacio en memoria y su manipulación o su carga vía internet es más lenta. Para la compresión sin perdidas hay algoritmos como RLE, LZW y ZIP que son utilizados con imágenes digitales con contienen áreas de color de gran tamaño o con una extensa gama de colores.

En el caso de los formatos con pérdidas se desecha información redundante en la imagen perdiendo parte de los datos con el cual se reduce el espacio en memoria de la imagen. En el caso del formato jpeg compensan la perdida de información con técnicas que suavizan los bordes y áreas que tienen un color similar haciendo que la falta de información sea invisible a simple vista. Otros formatos sin perdida son PCX, BMP, PSD, EPS, TIF, PNG Y GIF.

Los formatos con perdida son más utilizados comúnmente principalmente por su uso en internet ya que se requieren archivos de tamaño medio para que puedan cargar rápidamente. También son el estándar usado en cámaras digitales en dispositivos inteligentes, por lo que su popularidad y utilidad es mayor. Dentro de los más populares se encuentran [2].

1. JPEG
2. GIF
3. BMP
4. TIF
5. PNG

### 3.2.1 Formato JPEG

JPEG es un formato estándar de compresión para imágenes digitales y es el más utilizado comúnmente en dispositivos como cámaras fotográficas digitales o celulares. Al ser un algoritmo de compresión, este tiene pérdidas para poder reducir el tamaño de las imágenes y así ocupen menos espacio en memoria.

El algoritmo JPEG se basa en dos fenómenos visuales del ojo humano: el primero es que detecta más fácilmente los cambios de brillo (luminancia) que de color (crominancia) y el segundo es que nota más fácilmente pequeños cambios de brillo en zonas homogéneas que en zonas donde la variación de crominancia es mayor [2].

### 3.2.2 Formato GIF

Por sus siglas en ingles *Graphics interchange format* fue creado por eCompuServe y es preferiblemente usado para imágenes que no tienen tonos continuos o cuando hay áreas grandes de un mismo color ya que utiliza una paleta de color indexado con un máximo de 256 colores.

Una ventaja de GIF es que se pueden elegir uno o varios colores para que sean transparentes y se puedan observar los elementos debajo de estos y su uso en animaciones es muy común porque permite reproducir varios cuadros secuencialmente además de esta diseñado para disminuir el tiempo de transferencia de datos por líneas telefónicas.

Cuenta con una profundidad de color de ocho bits, no tiene canales alfa, soporta la compresión LZW y es del tipo de mapa de bits [4][5].

### 3.2.3 Formato BMP

Es un formato de imagen creado por Microsoft y usado exclusivamente en sus sistemas operativos. El tamaño de imagen de este formato es muy grande ya que no alcanzan una gran compresión, aunque gracias a eso su uso es muy extendido en el mundo del diseño además de que su profundidad de color varia de blanco y negro (un bit), escala de grises (cuatro a ocho bits), color indexado (ocho bits) y RGB (24 bits); no contiene canales alfa y soporta compresión RLE en 4 y 8 bits.

Funciona con imágenes de mapa de bits que guardan la información en una matriz donde a cada coordenada se le llama píxel y este tiene un valor de iluminación y color independiente al del resto [5].

### 3.2.4 Formato TIFF

El formato TIFF por sus siglas en inglés (*Tag Image File Format*) es un formato de mapa de bits que tiene una profundidad de color de 32 bits, tiene canales alfa y es usado para gráficos debido a la calidad de imagen e impresión que presenta. Fue creado por Aldus corporation en 1987 y es reconocido por prácticamente todos los programas en Mac y Windows.

Su principal característica es el uso de etiquetas que contienen información sobre la imagen y así almacenar todas las características de la misma, pero esto hace que el archivo sea bastante pesado por lo que usa la compresión LZW (*Lemple-Zif-Welch*) [5].

### 3.2.5 Formato PNG

Lleva las siglas del grupo que lo desarrollo *Portable Networks Graphics* y fue pensando para una fácil distribución en internet tomando las mejores características de GIF y JPEG permitiendo altos niveles de compresión y el uso de la técnica de indexación para hacer colores transparentes, semitransparencias o transparencias degradadas sin estar limitadas a una paleta de 256 colores ya que usa 24 bits permitiendo usar hasta 8 millones de colores, también tiene canales alfa y su compresión no tiene perdidas. La única diferencia es que con PNG no se pueden hacer animaciones como con GIF [5].

## 3.3 Funciones hash

Las funciones hash criptográficas toman un mensaje como entrada de longitud arbitraria y lo convierten a una cadena de una longitud determinada; a esa salida se le llama *huella digital* y sirve como una representación del archivo digital. Estas juegan un papel determinante en la criptografía moderna debido a su uso para proteger la integridad de la información digital en conjunto con los esquemas de firmas digitales. [6]

Existen diferentes algoritmos para de hash, cuya huella digital varía de 160 a 512 bits dependiendo del estándar que se implemente y el algoritmo criptográfico con el que se combine, como son firmas digitales,

Dependiendo del algoritmo de hash que se utilice, el largo de la huella digital varía entre 160 y 512 bits. Los algoritmos de hash son normalmente usados con otros algoritmos criptográficos como algoritmos de firma digital, códigos de autenticación de mensajes cifrados con hash o en la generación de números aleatorios.

Para que un algoritmo hash sea considerado como seguro, debe ser computacionalmente imposible recuperar el mensaje original a través de su valor hash y que no existan dos o más archivos que tengan un mismo valor hash. Cualquier cambio en el mensaje original, por muy mínimo que sea, debería entregar un valor hash diferente al menos en la mitad de los bits de la cadena original. Este suceso provocaría un error de verificación cuando el valor hash es usado con algoritmos de firma digital.

### 3.3.1 Propiedades de las funciones hash [6]

*Unidireccional*: Existiendo una función hash H(m), debe ser imposible calculablemente encontrar mensaje de origen m a partir de dicho resumen.

*Compresión*: Con todo mensaje de entrada m de cualquier longitud se debe generar una función hash H(m) de longitud fija ya establecida por el algoritmo.

*Facilidad de cálculo*: Teniendo un mensaje m debe ser fácil calcular su función hash H(m)

*Difusión*: La función H(m) debe ser una fusión compleja de todos los bits del mensaje m; donde si el mensaje m es modificado en alguno de sus bits, la función H(m) deberá cambiar aproximadamente en la mitad de sus bits.

*Resistencia a colisiones*: Teniendo un mensaje m, será imposible encontrar algún mensaje m´, que cumpla con H(m)=H(m´). Y teniendo un mensaje m y su función H(m) no se cumpla que H(m)= H(H(m)).

### 3.3.2 Familias SHA

Por sus siglas en inglés, *secure hash algorithm* (Algoritmo de hash seguro) es una familia de funciones hash publicadas y avaladas por el Instituto Nacional de Estándares y Tecnología (NIST) que inició con SHA-0, que es conocida como la versión inicial. Después se creó SHA-1 y consecuentemente SHA-2 que abarca los estándares SHA-224, SHA-256, SHA384 y SHA-512.

Aunque SHA-2 ya existía y tenía mejoras con respecto a SHA-1, que había funcionado como estándar debido a su seguridad, su tamaño de bits que era de 160 y que no existían ataques exitosos contra él, hasta el 23 de febrero del 2017 cuando Google y CWI Amsterdam provocaron una colisión poniendo en duda la seguridad del algoritmo [6].

SHA-2 ya había sido diseñado para que no existirán colisiones entre sus funciones hash, siendo está su mejor aportación respecto a SHA-1 y en 2012 se convirtió en el estándar y publicado por el NIST.

Cada algoritmo especificado en las familias hash SHA, se puede describir en dos etapas: el preprocesamiento y el cálculo del valor hash. Donde el preprocesamiento abarca la división del mensaje de entrada en *N*-bloques y la inicialización de valores que serán usados en el cálculo del valor hash. El cálculo del valor hash se genera a partir del mensaje dividido y lo usa junto con otras funciones, constantes y operaciones sobre palabras para iterativamente una serie de valores hash. El valor hash que se genera al final por el cálculo del valor hash es usado como huella digital.

### 3.3.2.1 Algoritmo SHA-256

Como su nombre lo indica es un hash de 256 bits que está pensada para proveer seguridad en una llave 128 bits a prueba de colisiones. SHA-256 opera de la misma forma de otros algoritmos para valores hash como MD4, MD5 y la familia SHA-1 pero usa bloques de 512 bits y valores intermedios de 256 bits.

Se compone de dos partes principales: La función de compresión SHA-256 y la fijación de mensajes SHA-256 que utilizan un arreglo de compuertas lógicas para realizar el proceso. La notación para las ecuaciones se presenta en la siguiente tabla donde cada operador funciona con palabras de 32 bits.

**Tabla 3.1** Operaciones lógicas usadas para el cálculo del valor hash

| | |
|---|---|
| $\oplus$ | bitwise XOR |
| $\wedge$ | bitwise AND |
| $\vee$ | bitwise OR |
| $\neg$ | bitwise complement |
| $+$ | mod $2^{32}$ addition |
| $R^n$ | right shift by n bits |
| $S^n$ | right rotation by n bits |

Lo primero que debe hacerse es inicializar un arreglo $H^{(0)}$ de 32 bits que contiene la parte fraccional de la raíz cuadrada de los primeros 8 números primos como se muestra en (2).

$$
\begin{aligned}
H_1^{(0)} &= 6a09e667 \\
H_2^{(0)} &= bb67ae85 \\
H_3^{(0)} &= 3c6ef372 \\
H_4^{(0)} &= a54ff53a \\
H_5^{(0)} &= 510e527f \\
H_6^{(0)} &= 9b05688c \\
H_7^{(0)} &= 1f83d9ab \\
H_8^{(0)} &= 5be0cd19
\end{aligned} \tag{2}
$$

Una vez inicializados los valores se aplica una etapa de preprocesamiento al mensaje *M* de entrada donde se define *l* como la longitud del mensaje. Al final del mensaje se anexa un bit 1 y un valor *k* se inicializa con 0 bits donde k es solución no negativa más pequeña que cumple con (3).

$$l+1+k = 448 \bmod 512 \tag{3}$$

Una vez resuelta la ecuación se anexa un bloque de 64 bits que representa la longitud del mensaje en binario; al final se obtendrá un mensaje que debe ser un múltiplo de 512.

El mensaje se dividirá en N bloques de 512 de bits con la notación $M^{(N)}$ y a su vez cada bloque será dividido en 16 partes de 32 bits con la notación $M_j^{(N)}$ con la finalidad de ajustar los tamaños adecuados para el cálculo de las operaciones lógicas dentro del proceso.

Después del preprocesamiento se inicia un ciclo principal donde se recorren los N bloques del mensaje original; la primera parte es realizar la asignación mostrada en (4) donde los registros a, b, c, d, e, f, g, h, se inicializan con los valores hash intermedios.

$$
\begin{aligned}
a &= H_1^{(i-1)} \\
b &= H_2^{(i-1)} \\
c &= H_3^{(i-1)} \\
d &= H_4^{(i-1)} \\
he &= H_5^{(i-1)} \\
f &= H_6^{(i-1)} \\
g &= H_7^{(i-1)} \\
h &= H_8^{(i-1)}
\end{aligned} \tag{4}
$$

Después se utiliza la función de compresión SHA-256 donde se utilizan los registros a, b, c, d, e, f, g, y h según corresponda, para calcular las ecuaciones (5) (6) (7) y (8) dentro de un ciclo de 0 hasta 63.

$$\mathrm{Ch(e,f,g)} = (\mathrm{e} \wedge \mathrm{f}) \oplus (\neg \mathrm{e} \wedge \mathrm{g}) \tag{5}$$

$$\mathrm{Maj(a,b,c)} = (\mathrm{a} \wedge \mathrm{b}) \oplus (\neg \mathrm{a} \wedge \mathrm{c}) \oplus (\neg \mathrm{b} \wedge \mathrm{c}) \tag{6}$$

$$\Sigma 0(\mathrm{a}) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \tag{7}$$

$$\Sigma 1(\mathrm{a}) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x) \tag{8}$$

$$\sigma_0(\mathrm{x}) = S^7(x) \oplus S^{18}(x) \oplus R^3(x) \tag{9}$$

$$\sigma_1(\mathrm{x}) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x) \tag{10}$$

Luego se calculan los bloques de mensaje extendido $W_j$ que se conoce como fijación de mensajes SHA-256, donde se igualan los valores $W_j$ y $M_j^{(i)}$ y se realiza el cálculo mostrado en (11) en un ciclo de 16 hasta 63 para cada arreglo de bloques $M^{(N)}$.

$$W_k = \sigma_1(W_{k-2}) + W_{k-7} + \sigma_0(W_{k-15}) + W_{k-16} \tag{11}$$

Una vez obtenidos los bloques de mensaje extendido y las operaciones Ch (e, f, g), Maj(a, b, c), $\Sigma_0$(a), $\Sigma_1$(e) se realiza la asignación como se indica en (12). Los valores $K_j$ están dados por la tabla 3.2 y son constantes con los valores fraccionales de las raíces cubicas de los primeros 64 números primos

$$\begin{gathered} T1 \leftarrow h + \Sigma 1(e) + Ch(e;f;g) + Kj + Wj \\ T2 \leftarrow \Sigma 0(a) + Maj(a;b;c) \\ h \leftarrow g \\ g \leftarrow f \\ f \leftarrow e \\ e \leftarrow d + T1 \\ d \leftarrow c \\ c \leftarrow b \\ b \leftarrow a \\ a \leftarrow T1 + T2 \end{gathered} \tag{12}$$

**Tabla 3.2**. Primeros 32 bits en hexadecimal de las partes fraccionales de las raíces cubicas de los primeros 64 números primos [7]

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 428a2f98 | 71374491 | b5c0fbcf | e9b5dba5 | 3956c25b | 59f111f1 | 923f82a4 | ab1c5ed5 |
| d807aa98 | 12835b01 | 243185be | 550c7dc3 | 72be5d74 | 80deb1fe | 9bdc06a7 | c19bf174 |
| e49b69c1 | efbe4786 | 0fc19dc6 | 240ca1cc | 2de92c6f | 4a7484aa | 5cb0a9dc | 76f988da |
| 983e5152 | a831c66d | b00327c8 | bf597fc7 | c6e00bf3 | d5a79147 | 06ca6351 | 14292967 |
| 27b70a85 | 2e1b2138 | 4d2c6dfc | 53380d13 | 650a7354 | 766a0abb | 81c2c92e | 92722c85 |
| a2bfe8a1 | a81a664b | c24b8b70 | c76c51a3 | d192e819 | d6990624 | f40e3585 | 106aa070 |
| 19a4c116 | 1e376c08 | 2748774c | 34b0bcb5 | 391c0cb3 | 4ed8aa4a | 5b9cca4f | 682e6ff3 |
| 748f82ee | 78a5636f | 84c87814 | 8cc70208 | 90befffa | a4506ceb | bef9a3f7 | c67178f2 |

Aquí termina el ciclo de la función de compresión SHA-256 y se actualizan los registros $H_j^{(i)}$ como se indica en (13).

$$\begin{aligned} H_1^{(i)} &= a + H_1^{(i-1)} \\ H_2^{(i)} &= b + H_2^{(i-1)} \\ H_3^{(i)} &= c + H_3^{(i-1)} \\ H_4^{(i)} &= d + H_4^{(i-1)} \\ H_5^{(i)} &= e + H_5^{(i-1)} \\ H_6^{(i)} &= f + H_6^{(i-1)} \\ H_7^{(i)} &= g + H_7^{(i-1)} \\ H_8^{(i)} &= h + H_8^{(i-1)} \end{aligned} \tag{13}$$

El ciclo principal reinicia y se continua con el mismo proceso. Cuando el ciclo principal termina el valor hash de M será el mostrado en (14):

$$H^{(N)} = H_1^{(N)}, H_2^{(N)} \ldots, H_8^{(N)} \tag{14}$$

## 3.4 Criptografía

La criptografía se trata del diseño y análisis de técnicas matemáticas que permiten comunicaciones seguras en presencia de adversarios maliciosos [7]. Ya sea diseñando algoritmos, protocolos o sistemas para dotar de seguridad a las comunicaciones, a la información y a las entidades que se comunican.

Para el funcionamiento de los algoritmos criptográficos se requiere de un mensaje de entrada y en el esquema más básico, se requiere una llave de cifrado de determinada longitud que funcionen como entrada para el algoritmo, que transformará el mensaje para que sea incomprensible para aquel que no tenga la llave que permita descifrar el mensaje.

El esquema anterior es conocido como de clave secreta o bien, criptografía simétrica donde precisamente solo se tiene una llave que sirve para cifrar y descifrar el mensaje como se muestra en la Fig. 3.1.

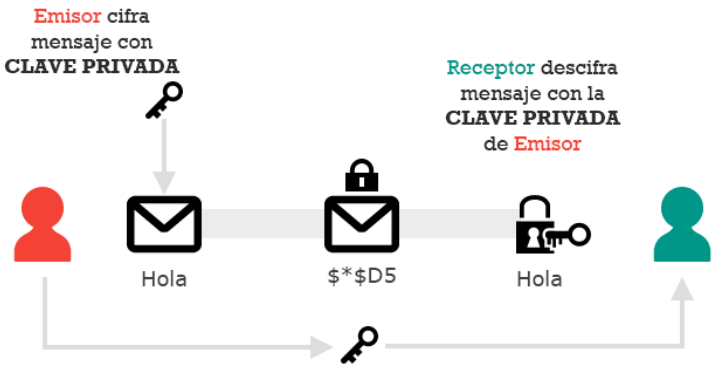



**Figura 3.1**. Primeros 32 bits en hexadecimal de las partes fraccionales de las raíces cubicas de los primeros 64 números primos [7]

Para este esquema, la llave representa toda la seguridad del cifrado, ya que a un atacante le sería inútil conocer el algoritmo con el que se cifró el mensaje si desconoce la llave.

El principal problema de la criptografía simétrica no radica en la seguridad de su algoritmo, sino en la distribución de las llaves, puesto que se debe buscar una forma segura para compartirlas y que no sean interceptadas junto con el mensaje.

### 3.4.1 Criptografía de llave publica

También conocida como criptografía asimétrica, un par de llaves son seleccionadas donde el problema de derivar la llave privada desde la correspondiente llave publica es equivalente a resolver un problema computacional que es pensado para ser intratable. De esta forma, los problemas de teoría numérica cuya intratabilidad forman la base para la seguridad de esquemas de llave publica comúnmente usados [8].

Para transmitir un mensaje por un canal, tanto el emisor como el receptor poseen dos llaves (una publica y una privada) donde el proceso se conforma por usar la llave pública del receptor para cifrar el mensaje y una vez transmitido, este debe ser descifrado con la llave privada del receptor tal como se muestra en Fig.3.2.


**Fig. 3.2.** Esquema de criptografía asimétrica para transmitir un mensaje cifrado [12]

De esta forma se resuelve el problema de la criptografía simétrica porque, aunque se interceptara el mensaje junto con la llave publica, se necesitaría de la llave privada (esta nunca es transmitida y debe permanecer oculta por el dueño) para poder descifrar el mensaje y así se asegura que solo el receptor podrá ver el mensaje ya que solo el posee la llave privada.

## 3.4.2 Criptografía de curva elíptica

Las curvas elípticas han sido utilizadas para resolver un diverso rango de problemas matemáticos y a partir de mil novecientos ochenta y cinco, Neal Koblitz y Victor Miller propusieron el uso de las curvas elípticas para diseñar sistemas criptográficos de llave pública. Desde entonces existe gran variedad de investigaciones publicadas acerca de la seguridad con curvas elípticas.

En 1990 los sistemas de curva elíptica empezaron a recibir aceptación comercial cuando organizaciones de estándares acreditadas especificaron protocolos de curva elíptica y compañías privadas incluyeron estos protocolos en sus productos de seguridad. [9]

Una curva elíptica está definida por una ecuación de la forma mostrada en (15) donde a, b pertenecen a un campo finito $F_p$ y que el discriminante $4a^3 + 27b^2 \neq 0\ mod\ (p)$. La función E ($F_p$) consiste en todos los puntos (x, y) pertenecientes al campo finito $Fp$ y que satisfacen la ecuación 14 y con el punto especial 0, llamado punto en el infinito.

$$y^2 = x^2 + ax + b \tag{15}$$

Dentro de la teoría de las curvas elípticas, existe una regla de adición para puntos llamada *regla de la tangente* que junto al conjunto de puntos E ($F_p$) se forma un grupo que tiene como identidad al punto en el infinito 0. Esta regla es la base para construir los cryptosistemas de curva elíptica y se define teniendo dos puntos iniciales P ($x_1$, $y_1$) y Q ($x_2$, $y_2$) cuya suma dará un tercer punto R($x_3$,$y_3$). Geométricamente se traza una línea que atraviese *P* y *Q* que intersecará un punto de la curva elíptica como se muestra en Fig. 3.3. Entonces *R* es la reflexión de este punto sobre el eje *x* [9].

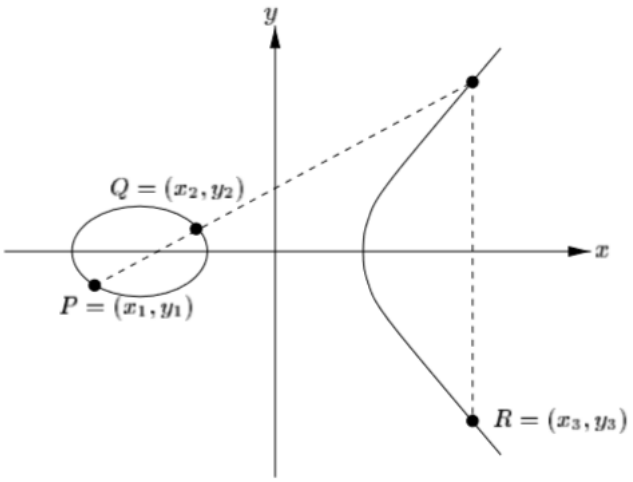



**Fig. 3.3.** Representación geométrica para la obtención para la obtención del punto R sobre la curva elíptica

Para obtener las coordenadas del punto R ($x_3$, $y_3$) se utilizan las ecuaciones (16) y (17)

$$x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2 \tag{16}$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 (x_1 - x_3) - y_1 \tag{17}$$

### 3.4.3. Autenticación

El desarrollo de software que permite manipular archivos digitales facilita que existan falsificaciones o alteraciones maliciosas para usar ilegalmente el contenido del archivo, dañando los derechos de autor provocando pérdidas económicas y daños morales. Por lo cual se han creado métodos que permiten verificar si un archivo conserva su originalidad después de viajar por un canal de comunicación.

Un esquema de criptografía asimétrica enfocado a la autenticación requiere cifrar el mensaje con la llave privada del emisor (en lugar de la pública como en el esquema de cifrado) permitiendo que su identidad dependa de su propia llave pública justo como se muestra en la Fig. 3.4. De esta forma, la autenticación solo será exitosa si la llave publica corresponde al emisor y dueño de la llave privada con la que se cifró el mensaje [7].

Al proceso de cifrar un mensaje con una llave privada se le conoce como *firmado digital* y como se describió, son algoritmos que sirven para autenticar al emisor del mensaje como la integridad de este.


**Fig. 3.4.** Esquema de criptografía asimétrica para la autenticación de mensajes [8}

## 3.4.4 Algoritmos de firma digital

Estos permiten que el receptor de algún mensaje identifique la entidad original del mensaje y que este no ha sido alterado desde que se hizo la firma digital por parte del emisor. Su aplicación es importante para poder verificar la autenticidad e integridad de la información enviada, ya que son una herramienta útil para detectar falsificaciones y manipulación de información [10].

Existen algunas propiedades necesarias para que las firmas digitales puedan garantizar la seguridad, las cuales son:

- Deben ser únicas al momento de ser generadas por el firmante y por lo tanto infalsificables.
- Ser infalsificables, donde el atacante tendría que resolver problemas matemáticos de complejidad elevada para poder descifrar el mensaje.
- Deben ser fácilmente verificables por los receptores de las mismas
- Deben ser viables, es decir, tienen que ser fácilmente generables por el firmante

Los algoritmos de firma digital también utilizan criptografía asimétrica, es decir, ambos participantes poseen una llave publica y otra privada.

Pero el esquema de transmisión es diferente, ya que el mensaje debe ser cifrado con la misma llave privada del emisor (a este proceso se le conoce como firmar digitalmente), después el mensaje es transmitido por algún canal y al llegar al receptor este debe utilizar la llave pública del receptor, confirmando que fue el quien firmó el mensaje y que el mensaje no sufrió alteraciones durante su transmisión.

### 3.4.4.1 Algoritmo de firma digital con curva elíptica (ECDSA)

Fue aceptado por ANSI como un estándar en 1999 y después en 2000, fue aceptado por el IEEE y el NIST. El ECDSA es análogo a su predecesor, el DSA con la diferencia de que basa su seguridad en el problema del logaritmo discreto en curvas elípticas que ha demostrado su eficiencia para crear pares de llaves más cortas, pero con la misma seguridad que otros algoritmos que usan llaves de mayor longitud [9].

El ECDSA tiene tres funciones principales, que son: generar pares de llaves (pública y privada), crear firmas digitales y verificar firmas digitales, donde el generar un par de llaves indica que el algoritmo funciona con criptografía asimétrica.

Para poder generar el par de llaves se deben de cumplir los parámetros de dominio $D = (q, FR, a, b, G, n, h)$. La llave publica es un múltiplo aleatorio de un punto base $P$ que existe sobre la curva elíptica y la llave privada es el entero utilizado para generar dicho múltiplo. El proceso de generar y verificar una firma digital ECDSA también debe cumplir con los parámetros de dominio $D = (q, FR, a, b, G, n, h)$ y con los asociados al par de llaves (d,Q). Si todas estas condiciones son cumplidas entonces el proceso de generación de firma está compuesto por [10]:

1. Seleccionar aleatoriamente un entero $k$, donde $1 < k < n-1$
2. Calcular $kG=(x_1, y_1)$ y $r=x_1 \bmod n$
3. Calcular $k^{-1} \bmod n$
4. Hacer $e=SHA(m)$
5. Calcular $s= k^{-1}(e + dr) \bmod n$
6. La firma para el mensaje m es (r, s)

Para el proceso de verificación de la firma se deben seguir los siguientes pasos:

1. Verificar que *r* y *s* sean enteros pertenecientes al intervalo [1, n-1]
2. Hacer e=SHA(m)
3. Calcular $w=s^{-1} \bmod n$
4. Calcular $u_1 = ew \bmod n$ y $u_2 = rw \bmod n$
5. Calcular $X = u_1G + u_2Q$, donde $X= (x_{1, y1})$
6. Calcular $v= x_1 \bmod n$
7. La firma verifica si v=r

## 3.5 Hardware dedicado para implementar el sistema

Los algoritmos SHA-256 y ECDSA pueden ser implementados manualmente con cualquier otro lenguaje de programación, pero con la idea de reducir el tiempo de procesamiento para calcular ambos algoritmos matemáticos se utilizan dos componentes de hardware: La BeagleBone Black Rev-C y una extensión para esta misma llamada CryptoCape.

La BeagleBone es una plataforma de costo reducido que funciona con sistemas operativos de Linux, para el caso de la versión Rev-C viene precargada con Debian y sus características principales son las siguientes [12].

- 512MB de memoria RAM DDR3 a 606 MHz
- Procesador ARM Cortex AM335X a 1 GHz
- Almacenamiento eMMC incorporado de 4GB
- Acelerador de gráficos 3D SGX530

La BeagleBone Black Rev-C esta ilustrada en Fig. 3.5 y, además, cuenta con los siguientes puertos para conectividad:

- Puerto USB para fuente o comunicaciones
- USB como host
- Puerto Ethernet
- Puerto HDMI
- 2 grupos para pines

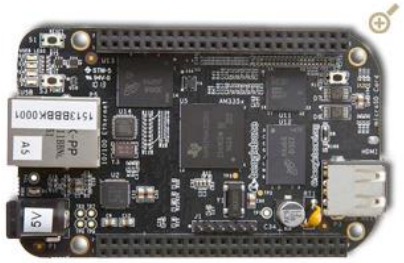


**Fig. 3.5.** Vista superior de la BeagleBone Black Rev-C

La CryptoCape es un hardware específico para la BeagleBone que permite extender sus capacidades al contener chips que implementan diferentes operaciones criptográficas utilizables para realizar seguridad informática; la CryptoCape esta ilustrada en Fig. 3.6 y los chips que contiene son los siguientes [13]:

- Reloj en tiempo real
- EEPROM
- AES-128 EEPROM
- Módulo de plataforma de confianza (TPM)
- ATSHA 204
- Generador de curva elíptica para ECDSA

El chip de interés para el implementar el sistema es el generador de curva elíptica para ECDSA (chip ATECC108) que tiene la versatilidad de implementar curvas estandarizadas por el NIST llamadas P256, B282 y K283.

Sparkfun proporciona una página dedicada para poder descargar e instalar las librerías necesarias para poder utilizar los distintos chips que contiene la CryptoCape, además de contener las guías de las distintas funciones que se pueden realizar con cada uno de ellos [14].

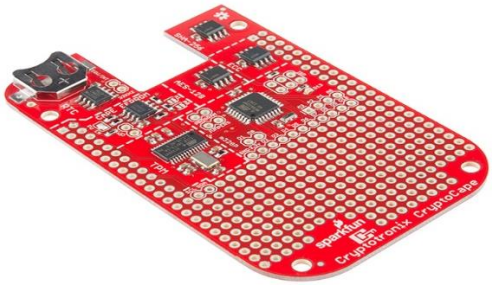


**Fig. 3.6.** Vista superior de la CryptoCape donde son visibles todos los chips con los que cuenta.

# Capítulo 4: Desarrollo del sistema propuesto

La Fig., 4.1 muestra un diagrama a bloques del sistema de autenticación de imágenes mediante digitales mediante hardware dedicado. En donde se puede observar que las imágenes deben estar previamente almacenadas en una memory stick de cualquier marca o capacidad de almacenamiento que se conecta al puerto microSD de la Beagle bone black

El sistema propuesto usó el *algoritmo de firma digital de curva elíptica* con funciones hash del estándar SHA-256 descrito en el apartado dos y fueron implementados en una Beaglebone black en conjunto con la tarjeta cryptocape permitiendo mejores tiempos de procesamiento comparado con equipos de hardware compartido. El sistema cuenta con una interfaz con dos procesos principales: la generación de firmas digitales y la verificación de estas que son mostrados en los diagramas de las siguientes figuras.

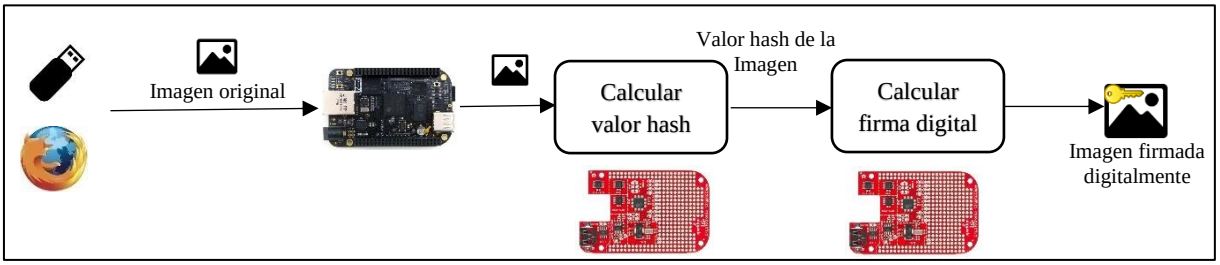



**Fig. 4.1.** Diagrama general del sistema para firmar imágenes digitalmente con el algoritmo ECDSA

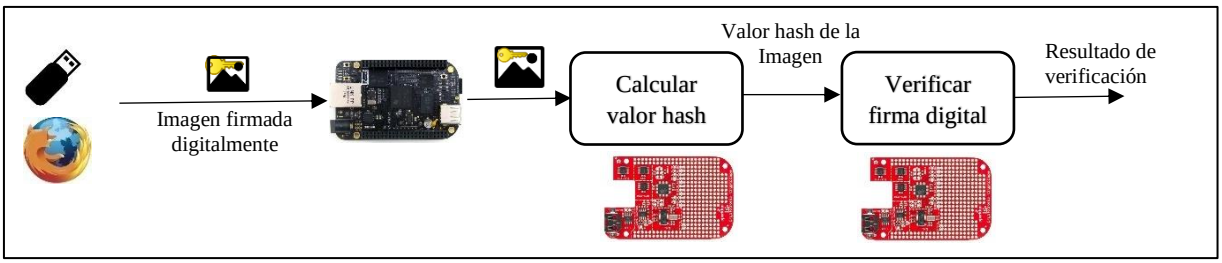



**Fig. 4.2.** Diagrama general del sistema para verificar imágenes firmadas digitalmente con el sistema propuesto

## 4.1 Generación de firmas digitales con el sistema propuesto

En este proceso, la interfaz gráfica abre el explotador de archivos para que el usuario pueda seleccionar una imagen ya sea que este almacenada en el sistema o se obtenga de un dispositivo externo como memorias USB, SD o microSD, tal como se muestra en la Fig. 4.3. posteriormente el sistema guarda la imagen y calcula el valor hash de la imagen como se muestra en el diagrama de flujo de la imagen 4.4 para poder ejecutar el *algoritmo de firma digital de curva elíptica* y generar la firma digital como se muestra en el diagrama a bloques de la Fig. 4.5.

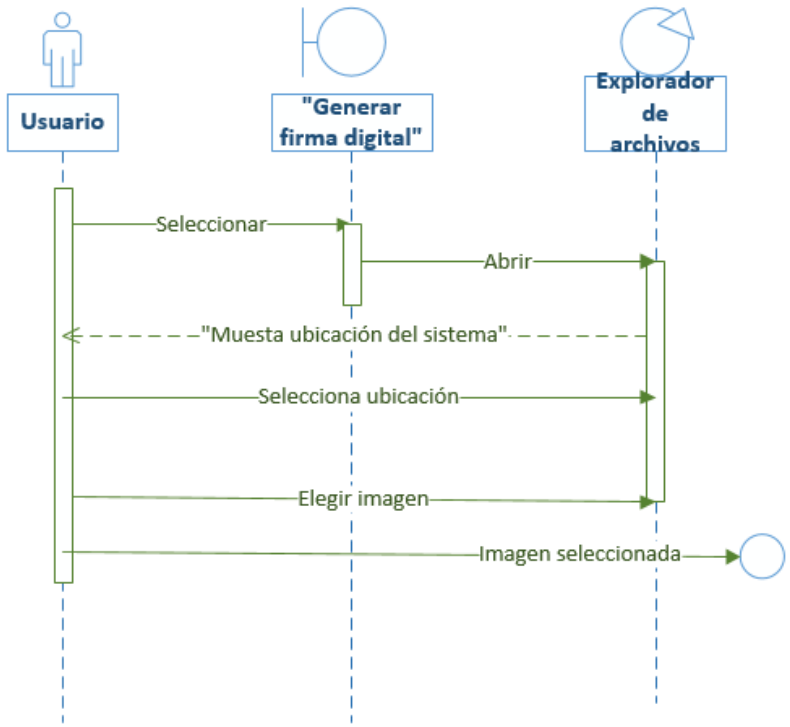



**Fig. 4.3**. Diagrama de secuencia para la generación de una firma digital con el sistema propuesto

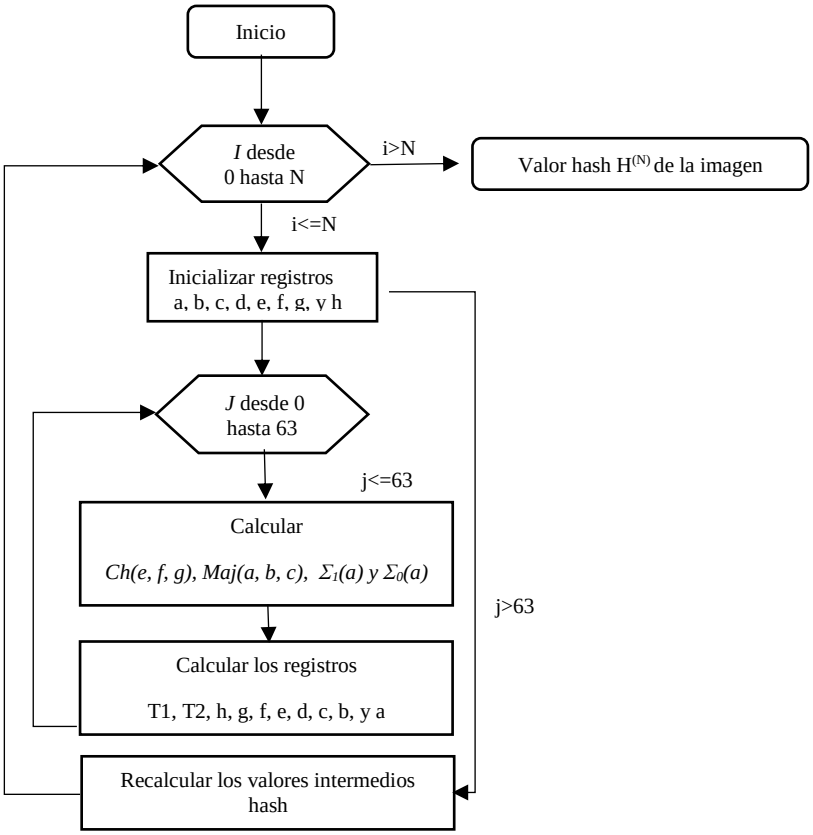



**Fig. 4.4**. Diagrama de flujo para calcular el valor hash con SHA-256.

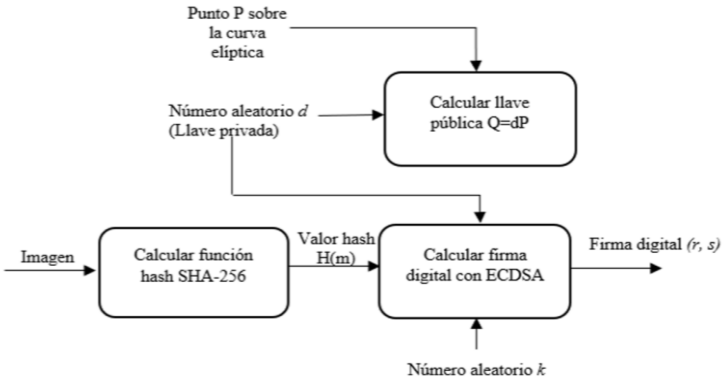



**Fig. 4.5**. Diagrama a bloques del proceso de generación de firma digital del sistema

**Bloque "Calcular valor hash SHA-256":** Requiere una sola entrada externa, que es la imagen a la que se requiere calcular su valor hash. El proceso a bloques para calcular el valor hash se muestra en la Fig. 4.6, donde se observan tres procesos indispensables y la forma en que interactúan, así como sus entradas y salidas de las que hace uso la función principal SHA-256 para lograr el cálculo del valor hash de la imagen.

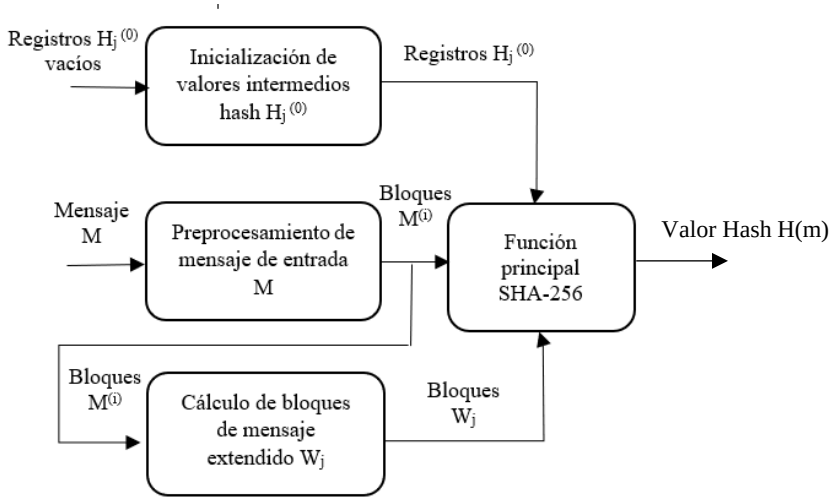



**Fig. 4.6**. Diagrama a bloques del proceso para calcular el valor hash de la imagen

El algoritmo para la función principal SHA-256 se define como en la Fig. 4.4 donde se especifica a través de un diagrama de flujo. Una vez obtenido el valor hash H(m) este sirve como entrada para el ECDSA.

**Bloque "Calcular firma digital con ECDSA":** Una vez obtenida la función hash, el sistema procede a calcular la firma digital usando como entrada el valor hash recién calculado, además de la llave privada *d* y un número aleatorio *k*, después esta se muestra en pantalla para que pueda ser usada para verificar la imagen desde otro dispositivo y comprobar la integridad del archivo y la identidad del emisor tal como es muestra en el diagrama de la Fig. 4.7.

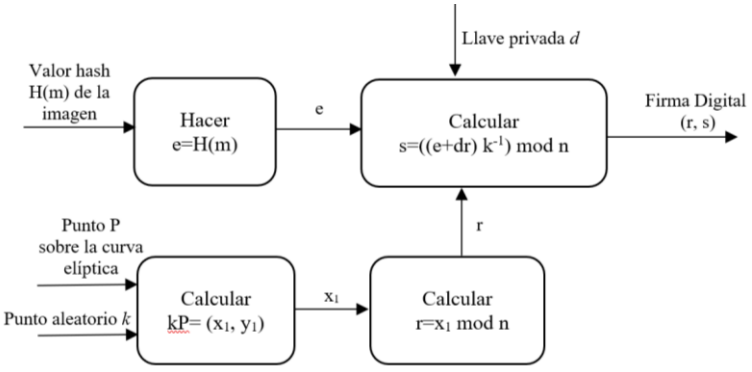



**Fig. 4.7**. Diagrama a bloques de generación de la firma digital del sistema propuesto con el algoritmo ECDSA

## 4.2 Verificación de firmas digitales con el sistema propuesto

Para el caso de la verificación, la interfaz de usuario permite seleccionar nuevamente una imagen que sea de formato jpeg, tiff, gif, png o bmp a la que se le aplicará el algoritmo SHA-256 para conseguir las variables necesarias para que se pueda verificar la imagen con el algoritmo ECDSA cuya interacción está especificada en la imagen de la Fig. 4.8.

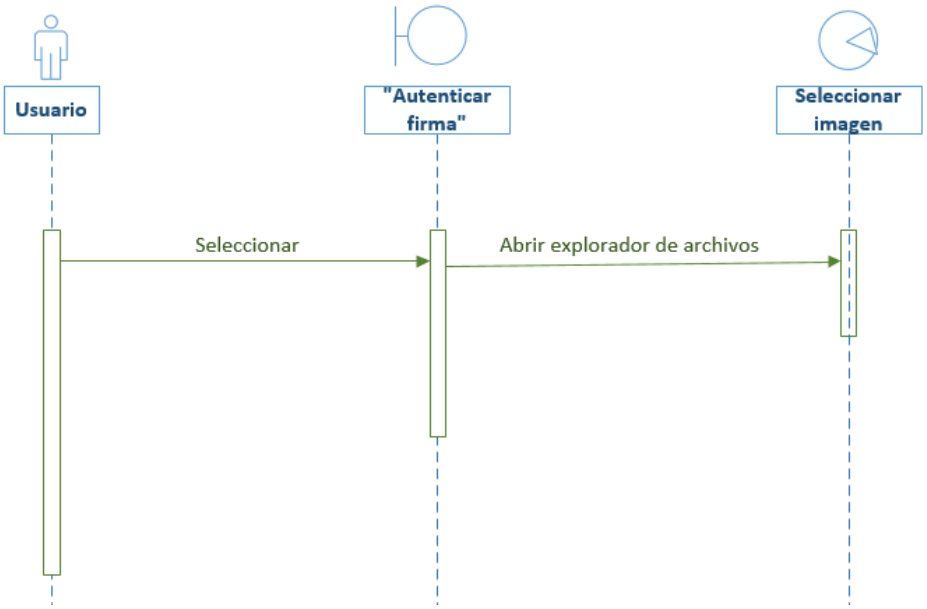



**Fig. 4.8.** Diagrama de secuencia del proceso de autenticación del sistema

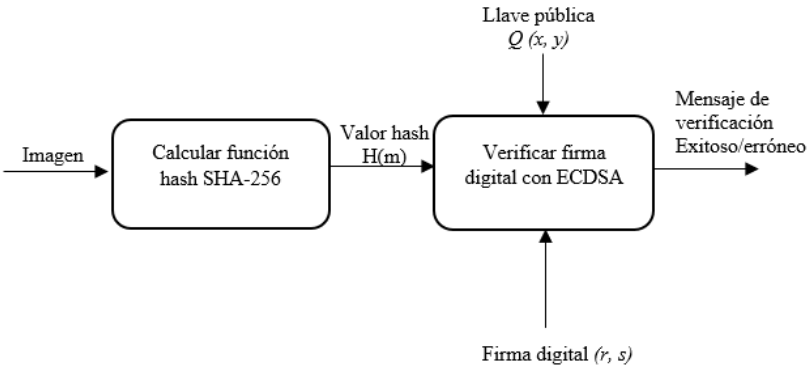



**Fig. 4.9** Diagrama a bloques del proceso de verificación del sistema propuesto

La figura 4.9 muestra el proceso para verificar una firma digital a través de un diagrama a bloques, donde es necesario calcular el valor hash de la imagen que puede ser la primera causa para que la firma no verifique correctamente; esto se debe a que si la imagen sufrió alguna modificación su valor hash cambiará como se muestra en la imagen de la Fig. 4.10, donde se calculó el valor hash de dos textos con el estándar SHA-224, estos textos solo se diferencian por el punto final pero su valor hash cambia completamente. Esto mismo pasaría con la imagen si sufriera modificaciones, por lo que verificación entregará un mensaje de verificación errónea.

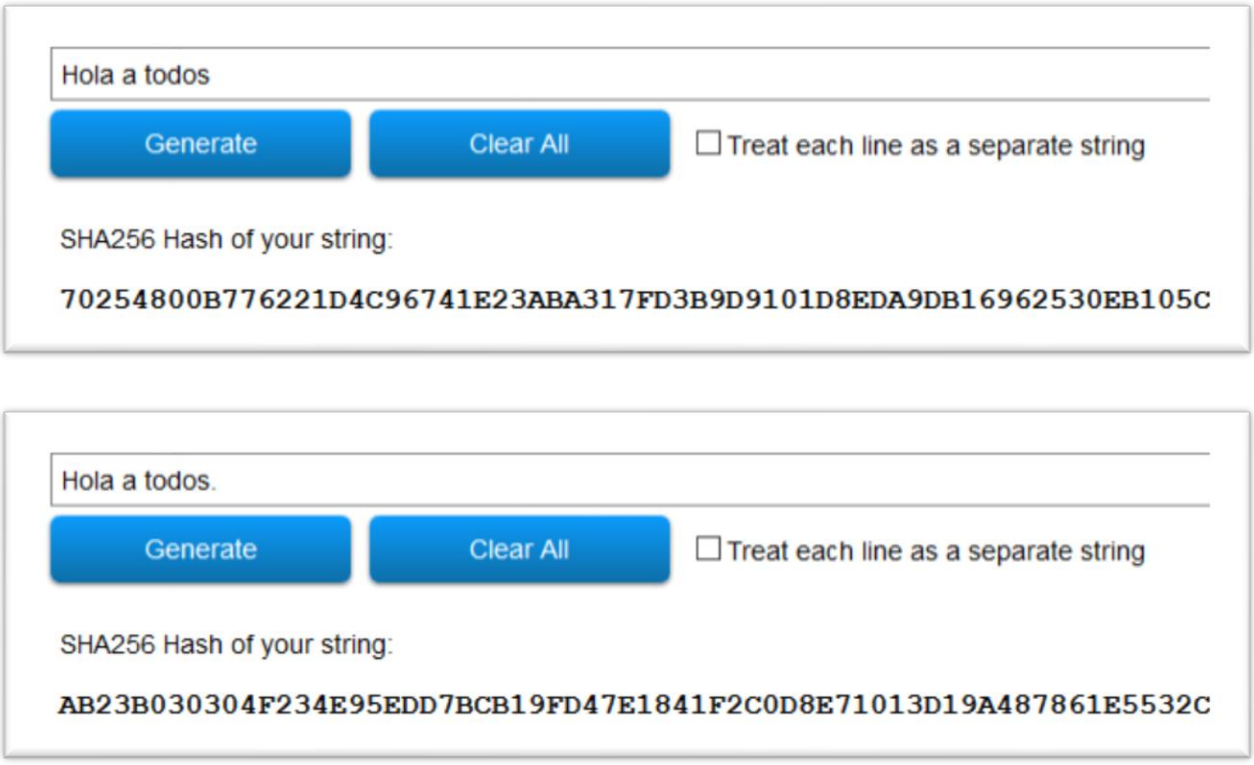



**Fig. 4.10**. Función hash de 2 textos similares con el algoritmo SHA-256.

Si el valor hash de la imagen fuera correcto, la siguiente opción que haría fallar al proceso de verificación es el usar una llave pública incorrecta, que matemáticamente haría fallar la verificación como se muestra en el algoritmo 3 del apartado 3.

El proceso que se muestra en la Fig. 4.11 corresponde a la verificación que realiza el sistema para finalmente mostrar en la interfaz el resultado, si todos los parámetros son correctos entonces la verificación será exitosa.

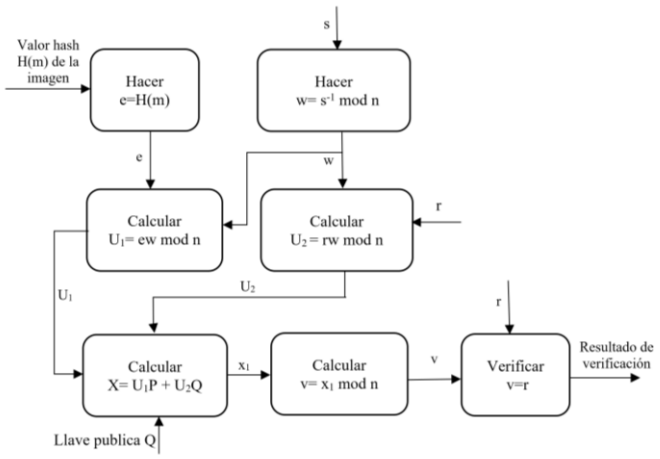



**Fig. 4.11**. Diagrama a bloques de verificación de la firma digital del sistema propuesto con el algoritmo ECDSA.

# Capítulo 5: PRUEBAS Y RESULTADOS

Para las pruebas existen dos procesos principales del sistema: la generación de firmas digitales y la verificación de estas. Se utilizaron diversos escenarios con diferentes imágenes de distintas características para probar la efectividad de los algoritmos SHA-256 y ECDSA, además del tiempo requerido por el sistema para realizar los procesos. Debido a que se verificará la integridad de la imagen, los escenarios considerados incluyen alteraciones sobre las imágenes que son perceptibles para la vista humana y otros donde las alteraciones son más discretas, tales que la vista humana no puede encontrarlas con facilidad.

## 5.1 Pruebas sobre la generación de firmas digitales

Como se mostró en el diagrama a bloques de la figura 4.5, el sistema requiere del valor hash de la imagen, la llave privada del emisor y un número aleatorio *k* para generar la firma digital. Para comprobar la generación de firmas digitales se utilizaron las imágenes mostradas en la figura 5.1 que fueron generadas con dispositivos móviles, que cuentan con diferentes dimensiones y están en formato JPEG.


(a) ESIME.jpg (b) EXPO.jpg (c) LOGO.jpg (d) ROCC.jpg

**Fig. 5.1**. Imágenes utilizadas para generar una firma digital

En la tabla 5.1 se muestran los resultados obtenidos de las firmas digitales generadas por el sistema acompañadas del valor hash que se obtuvo inicialmente. Los valores hash calculados recientemente serán los determinantes para la verificación de la firma digital, es decir si estos cambian (ya sea por alguna alteración) la firma digital no verificará. El otro caso es el uso de una llave pública que no corresponda a la llave privada del sistema, lo que también provocaría un error de verificación.

**Tabla 5.1** Resultados para la generación de firmas digitales de las imágenes de prueba

| Imagen | Firma digital de la imagen (128 bits) | Valor hash de la imagen (256 bits) |
|---|---|---|
| ESIME.jpg | 042FF0E1DD1699003E4169297ECB07DDD40F35792B4687B03D9D53A25D8473EFB9A04A86AB797E4B43FBBD702CFE0D816953676B7F99F1649063482CC52462BB51 | 0xCA 0x82 0x1D 0x63 0xB2 0x40 0x78 0x0B 0xD2 0x95 0xD1 0xE2 0x49 0x5A 0x94 0x69 0x62 0x6F 0x6C 0x1C 0x66 0xD0 0xA9 0x51 0x12 0x83 0xC1 0x49 0xB0 0xF4 0xE7 0x71 |
| EXPO.jpg | 54178D08DD43D6879A18CAA6B1F33AAA5E849D0D7E33450DFEAA8F3231A2BD012F07984DBBB3D97E753919A41A00CD7EF5CB3AA0B607FE06A37182E79C6967A1 | 0xDE 0x6D 0xB8 0xFD 0xD1 0xB9 0x0B 0x0A 0x93 0x4C 0x4C 0x87 0x50 0x54 0xA5 0xD8 0xF5 0x7B 0xE7 0x42 0x1E 0xFB 0xA7 0x43 0xDE 0x1A 0x3B 0xB5 0x4E 0x9A 0x92 0xC4 |
| LOGO.jpg | 2D5C86E8CE09DC4AB62582328DA228B72090C336AB5B7D1A342487C1E7E934A316610FCDF55ABB55D3D0D3BBB0342F8578DA95955D6B7FBA69EA60C50B6C4036 | 0x60 0xDF 0x00 0x14 0xE8 0xA4 0x49 0xA7 0xBF 0xD4 0x32 0xD0 0x3A 0x2F 0x21 0x6A 0xAA 0x87 0xEC 0x7E 0x45 0x05 0xAB 0x89 0x08 0xBD 0x61 0xF0 0xA5 0x39 0x0B 0x2D |
| ROCC.jpg | 2F2B84D4D94EEEF8BFCE8574DA8762B52F0C13E63799EE4594FBC062409CADBBA5630C806559D8F45608F116F11A7E9128FEDE1D1B9F22D93B57252A230DA89E | 0x79 0xB6 0xC7 0x63 0x89 0x77 0xB8 0x77 0xC1 0x7E 0x93 0xBD 0x1A 0x8E 0x27 0x58 0x14 0x74 0x95 0x8F 0x7D 0xE8 0x4B 0x7C 0xCE 0xA4 0x79 0xFC 0x66 0x63 0xCD 0x54 |

## 5.2 Pruebas sobre la verificación de firmas digitales

Para el proceso de verificación de las firmas se requiere calcular nuevamente el valor hash de la imagen a autenticar, utilizar la llave pública del emisor y la firma digital generada previamente, lo que nos asegura un proceso de verificación exitoso. El emisor fue el sistema propuesto así que la llave pública que necesita usarse es la misma para todos los casos, en la tabla 5.2 se muestran los resultados de las verificaciones cuando todos los parámetros son legítimos.

**Tabla 5.2** Resultados de la verificación de las imágenes de la figura 5.1

| Imagen | Valor hash (256 bits) | Llave pública (128 bits) | Resultado |
|---|---|---|---|
| ESIME.jpg | 0xCA 0x82 0x1D 0x63 0xB2 0x40 0x78 0x0B 0xD2 0x95 0xD1 0xE2 0x49 0x5A 0x94 0x69 0x62 0x6F 0x6C 0x1C 0x66 0xD0 0xA9 0x51 0x12 0x83 0xC1 0x49 0xB0 0xF4 0xE7 0x71 | 042FF0E1DD1699003E4169297EC B07DDD40F35792B4687B03D9D 53A2528473EFB9A04A86AB797E 4B43FBBD702CFE0D816953676B 7F99F1649063482CC52462BB51 | Exitoso |
| EXPO.jpg | 0xDE 0x6D 0xB8 0xFD 0xD1 0xB9 0x0B 0x0A 0x93 0x4C 0x4C 0x87 0x50 0x54 0xA5 0xD8 0xF5 0x7B 0xE7 0x42 0x1E 0xFB 0xA7 0x43 0xDE 0x1A 0x3B 0xB5 0x4E 0x9A 0x92 0xC4 | 042FF0E1DD1699003E4169297EC B07DDD40F35792B4687B03D9D 53A2528473EFB9A04A86AB797E 4B43FBBD702CFE0D816953676B 7F99F1649063482CC52462BB51 | Exitoso |
| LOGO.jpg | 0x60 0xDF 0x00 0x14 0xE8 0xA4 0x49 0xA7 0xBF 0xD4 0x32 0xD0 0x3A 0x2F 0x21 0x6A 0xAA 0x87 0xEC 0x7E 0x45 0x05 0xAB 0x89 0x08 0xBD 0x61 0xF0 0xA5 0x39 0x0B 0x2D | 042FF0E1DD1699003E4169297EC B07DDD40F35792B4687B03D9D 53A2528473EFB9A04A86AB797E 4B43FBBD702CFE0D816953676B 7F99F1649063482CC52462BB51 | Exitoso |
| ROCC.jpg | 0x79 0xB6 0xC7 0x63 0x89 0x77 0xB8 0x77 0xC1 0x7E 0x93 0xBD 0x1A 0x8E 0x27 0x58 0x14 0x74 0x95 0x8F 0x7D 0xE8 0x4B 0x7C 0xCE 0xA4 0x79 0xFC 0x66 0x63 0xCD 0x54 | 042FF0E1DD1699003E4169297EC B07DDD40F35792B4687B03D9D 53A2528473EFB9A04A86AB797E 4B43FBBD702CFE0D816953676B 7F99F1649063482CC52462BB51 | Exitoso |

Para otro caso de prueba, se alteraron las imágenes previo a su verificación lo que provocaría un valor hash diferente y la firma no verificaría permitiendo entender que hubo alguna alteración durante su transmisión. En la figura 5.2 se muestran las imágenes alteradas donde la alteración realizada se resalta con un círculo y en la tabla 5.3 se muestran comparados los valores hash de las imágenes originales y alteradas.

**Tabla 5.3** Comparación de valores hash entre imágenes originales y alteradas

| Nombre de la imagen | Original (256 bits) | Alterada (256 bits) |
|---|---|---|
| ESIME.jpg | 0xCA 0x82 0x1D 0x63 0xB2 0x40 0x78 0x0B 0xD2 0x95 0xD1 0xE2 0x49 0x5A 0x94 0x69 0x62 0x6F 0x6C 0x1C 0x66 0xD0 0xA9 0x51 0x12 0x83 0xC1 0x49 0xB0 0xF4 0xE7 0x71 | 0x26 0xB3 0x9A 0xC2 0x35 0x2D 0xDD 0xB5 0xD7 0xC9 0x98 0x22 0xC4 0xC0 0x63 0xC1 0xBB 0xFC 0xC2 0xCA 0x75 0xC4 0xA6 0x39 0xA3 0xDF 0xE9 0x39 0x62 0x96 0xF3 0xE1 |
| EXPO.jpg | 0xDE 0x6D 0xB8 0xFD 0xD1 0xB9 0x0B 0x0A 0x93 0x4C 0x4C 0x87 0x50 0x54 0xA5 0xD8 0xF5 0x7B 0xE7 0x42 0x1E 0xFB 0xA7 0x43 0xDE 0x1A 0x3B 0xB5 0x4E 0x9A 0x92 0xC4 | 0xD7 0x26 0x19 0x42 0xFA 0x18 0x40 0x1E 0x62 0x60 0x51 0x5F 0xAB 0xA4 0xE2 0x78 0xBF 0x55 0x79 0x41 0x46 0x71 0x19 0x40 0x25 0x72 0x15 0xDF 0xE5 0xFE 0x5A 0x11 |
| LOGO.jpg | 0x60 0xDF 0x00 0x14 0xE8 0xA4 0x49 0xA7 0xBF 0xD4 0x32 0xD0 0x3A 0x2F 0x21 0x6A 0xAA 0x87 0xEC 0x7E 0x45 0x05 0xAB 0x89 0x08 0xBD 0x61 0xF0 0xA5 0x39 0x0B 0x2D | 0x3A 0x03 0xD0 0x83 0x35 0x20 0x10 0x06 0xD4 0x95 0x4E 0x6D 0x35 0xD7 0x87 0x41 0x08 0xB3 0x21 0xA7 0x76 0xA9 0x68 0xB0 0x6C 0x98 0xD2 0x58 0x95 0x69 0x0A 0x5F |
| ROCC.jpg | 0x79 0xB6 0xC7 0x63 0x89 0x77 0xB8 0x77 0xC1 0x7E 0x93 0xBD 0x1A 0x8E 0x27 0x58 0x14 0x74 0x95 0x8F 0x7D 0xE8 0x4B 0x7C 0xCE 0xA4 0x79 0xFC 0x66 0x63 0xCD 0x54 | 0x06 0x5A 0x2D 0x51 0x8D 0xDA 0x07 0x03 0xA9 0x82 0x73 0x17 0xD2 0x5F 0xDC 0x6E 0xB1 0x47 0x57 0x54 0x5E 0xF0 0x07 0xD9 0xFB 0x9F 0xC5 0xB8 0x0C 0xC2 0xE5 0xF6 |

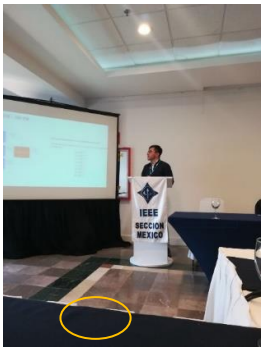
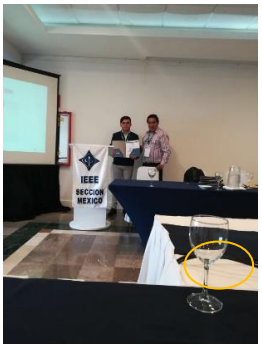
| Nombre | Originales | Alteradas |
|---|---|---|
| ESIME.jpg |  |  |
| EXPO.jpg |  |  |
| LOGO.jpg |  |  |
| ROCC.jpg |  |  |

**Fig. 5.2**. Imágenes alteradas (donde señalan las formas de colores) para mostrar error en la verificación de la firma digital.

Finalmente, en la tabla 5.4 se muestran los tiempos para la generación de la firma digital y la verificación de esta en las imágenes JPEG de prueba.

**Tabla 5.4** Tiempos obtenidos en el sistema propuesto para la generación de imágenes

| Imagen | Dimensiones de imagen | Tiempo de generación de firma | Tiempo de verificación de firma |
|---|---|---|---|
| ESIME.jpg | 2049x1537 | 71 microsegundos | 95 microsegundos |
| LOGO.jpg | 1536x2048 | 71 microsegundos | 95 microsegundos |
| ROCC.jpg | 960x1280 | 71 microsegundos | 95 microsegundos |
| EXPO.jpg | 960x1280 | 71 microsegundos | 95 microsegundos |

## 5.3 Pruebas para imágenes de otros formatos

JPEG se ha convertido en el formato más utilizado para imágenes digitales del tipo mapa de bits convirtiéndose en un estándar, pero existen algunos otros formatos de gran aceptación como TIFF, PNG, GIF y BMP. Por lo que el sistema está optimizado para generar firmas digitales y verificaciones para estos formatos. En la figura 5.3 se muestran las imágenes de prueba para generar firmas digitales con el sistema propuesto y comprobar su funcionamiento.

(a) Playa.tiff

(b) Tiger.bmp


(c) logoesime.png


(d) Beagle_logo_hdr.gif

**Fig. 5.3**. Imágenes con formatos diferentes a JPEG para realizar pruebas

Finalmente, en la tabla 5.5 se muestran los valores hash y firmas digitales obtenidas con el sistema propuesto. El tamaño de las cadenas es el mismo que utilizando JPEG y la verificación realizada con el sistema tampoco tiene problemas para determinar si hubo o no alteraciones.

**Tabla 5.5** Firmas digitales y valores hash obtenidos para las imágenes de la figura 5.3

| Imagen | Firma digital de la imagen (128 bits) | Valor hash de la imagen (256 bits) | Verificación |
|---|---|---|---|
| Playa.tiff | 0xCA 0x82 0x1D 0x63 0xB2 0x40 0x78 0x0B 0xD2 0x95 0xD1 0xE2 0x49 0x5A 0x94 0x69 0x62 0x6F 0x6C 0x1C 0x66 0xD0 0xA9 0x51 0x12 0x83 0xC1 0x49 0xB0 0xF4 0xE7 0x71 | 0x26 0xB3 0x9A 0xC2 0x35 0x2D 0xDD 0xB5 0xD7 0xC9 0x98 0x22 0xC4 0xC0 0x63 0xC1 0xBB 0xFC 0xC2 0xCA 0x75 0xC4 0xA6 0x39 0xA3 0xDF 0xE9 0x39 0x62 0x96 0xF3 0xE1 | Exitosa |
| Tiger.bmp | 0xDE 0x6D 0xB8 0xFD 0xD1 0xB9 0x0B 0x0A 0x93 0x4C 0x4C 0x87 0x50 0x54 0xA5 0xD8 0xF5 0x7B 0xE7 0x42 0x1E 0xFB 0xA7 0x43 0xDE 0x1A 0x3B 0xB5 0x4E 0x9A 0x92 0xC4 | 0xD7 0x26 0x19 0x42 0xFA 0x18 0x40 0x1E 0x62 0x60 0x51 0x5F 0xAB 0xA4 0xE2 0x78 0xBF 0x55 0x79 0x41 0x46 0x71 0x19 0x40 0x25 0x72 0x15 0xDF 0xE5 0xFE 0x5A 0x11 | Exitosa |
| Logoesime.png | 0x60 0xDF 0x00 0x14 0xE8 0xA4 0x49 0xA7 0xBF 0xD4 0x32 0xD0 0x3A 0x2F 0x21 0x6A 0xAA 0x87 0xEC 0x7E 0x45 0x05 0xAB 0x89 0x08 0xBD 0x61 0xF0 0xA5 0x39 0x0B 0x2D | 0x3A 0x03 0xD0 0x83 0x35 0x20 0x10 0x06 0xD4 0x95 0x4E 0x6D 0x35 0xD7 0x87 0x41 0x08 0xB3 0x21 0xA7 0x76 0xA9 0x68 0xB0 0x6C 0x98 0xD2 0x58 0x95 0x69 0x0A 0x5F | Exitosa |
| Beagle_logo_hdr .gif | 0x79 0xB6 0xC7 0x63 0x89 0x77 0xB8 0x77 0xC1 0x7E 0x93 0xBD 0x1A 0x8E 0x27 0x58 0x14 0x74 0x95 0x8F 0x7D 0xE8 0x4B 0x7C 0xCE 0xA4 0x79 0xFC 0x66 0x63 0xCD 0x54 | 0x06 0x5A 0x2D 0x51 0x8D 0xDA 0x07 0x03 0xA9 0x82 0x73 0x17 0xD2 0x5F 0xDC 0x6E 0xB1 0x47 0x57 0x54 0x5E 0xF0 0x07 0xD9 0xFB 0x9F 0xC5 0xB8 0x0C 0xC2 0xE5 0xF6 | Exitosa |

# CONCLUSIONES

El uso del algoritmo SHA-256 para crear los valores hash de 256 bits de las imágenes digitales asegura una correcta verificación de autenticidad de este tipo de material digital, ya que el algoritmo utiliza la información en bits de la imagen, permitiendo que el más mínimo cambio sea detectado y cualquier falsificación o modificación sea detectada además de no permitir colisiones, otra ventaja es que tiene un amplio futuro a comparación de SHA-1 que ya se ha vuelto obsoleta por su tamaño de hash de 128 bits que alcanzó su límite hace algunos años.

Así también el empleo del algoritmo ECDSA y el problema del logaritmo discreto en curvas elípticas (considerado intratable) brinda la seguridad necesaria para asegurar la identidad del remitente y la integridad del archivo cuando se verifica la imagen. Cabe mencionarse, que el proceso es ágil en cuestión de tiempo con un tiempo promedio aproximado de respuesta de 55 µs, lo que resulta como la mayor aportación del sistema propuesto gracias al uso de hardware dedicado con la BeagleBone y la CryptoCape, permitiendo la aplicación en sistemas que manejen un gran volumen de imágenes de diferentes formatos como se comprobó en el aparado de PRUEBAS Y RESULTADOS, donde resaltan los principales tipos: JPEG, PNG, TIFF, BMP y GIF.

# TRABAJO A FUTURO

Desarrollar e implementar un algoritmo de marca de agua invisible, el cual inserte como marca de agua la llave publica Q y la firma digital (r, s) para optimizar la transmisión de los distintos parámetros que se requieren para verificar la integridad de la imagen.

# Referencias


[1] Othmane H. and Dalila B., "Robustness-to-Noise Analysis of a Secure High Capacity Full-gray-scale-Image Information Hiding Via A New DCT-Moments-Based Scheme", IEEE International Symposium on Technologies for Homeland Security, pp. 1-5, Estados Unidos, 2018.

[2] Yan W. and Bo Z., "Geometrically robust image hashing scheme for image authentication", International Conference on Machine Learning and Cybernetics, pp. 1152-1157, China, 2012.

[3] Amol V., Yogesh J. and Swapan B., "Video tamper detection techniques based on DCT-SVD and multi-level SVD", IEEE TENCON Region 10 Conference, China, pp. 1-6, 2015.

[4] Pornchai A., Suphakant P. and Sasipa R., "Enhancing trustworthy of document using a combination of image hash and cryptographic hash", International Joint Conference on Computer Science and Software Engineering, Estados Unidos, pp. 1-6, 2017.

[5] Acharya T. and Thasi P., "JPEG2000 Standard for Image Compression", John Wiley & Sons, Vol. 30, pp. 128-153, Estados Unidos, 2005.

[6] Katz J., "Introduction to Modern Cryptography", Chapman & Hall CRC, Vol. 1143, pp. 324-378, Estados Unidos, 2007.

[7] Alfred M. and Paul van O., "Handbook of applied cryptography", T&F India, Vol. 1, pp. 276-302, India, 1997.

[8] Wikipedia.org," Criptografía asimétrica ", noviembre 2017. [En línea]. Disponible: https://es.wikipedia.org/wiki/Criptograf%C3%ADa_asim%C3%A9trica

[9] Stanoyevitch A., "Introduction to cryptography with mathematical foundations and computer implementations", Chapman & Hall/CRC, Vol. 37, pp.504-538, Estados Unidos, 2011.

[10] Schneier B., Kohno T. and Ferguson N., "Cryptography engineering", Wiley, Vol. 3, pp.84-173, Estados Unidos, 2013.

[11] Paar C., Preneel B. and Pelzl J., "Understanding Cryptography", Springer, Vol. 2, pp.127-132, Estados Unidos, 2014.

[12] Beaglebord.org, "BeagleBone Black", abril 2014, [En línea]. Disponible: https://beagleboard.org/black

[13] Sparkfun.com, "CryptoCape", marzo 2015, [En línea]. Disponible: https://www.sparkfun.com/products/retired/12773

[14] Sparkfun.com, "CryptoCape: Hookup Guide", marzo 2015, [En línea]. Disponible: https://learn.sparkfun.com/tutorials/cryptocape-hookup-guide?_ga=2.194604781.1323002197.1560142088-34581923.1560142088