Firma Digital Móvil Basada en Criptografía Hash

Leonardo Javier Montiel Arrieta, Flor Seleyda Hernández Mendoza, Luis Adrián Lizama Pérez, Luis Adrián Lizama Servín y Eric Simancas Acevedo

L. Montiel[1], F. Hernández[1], L. Lizama[1], L. Lizama[2], E. Simancas[1]
Universidad Politécnica de Pachuca[1]
Centro de Ingeniería y Desarrollo Industrial[2]
leoxavi8677@micorreo.upp.edu.mx

F. Trejo, (eds.). Ciencias Multidisciplinarias. Proceedings-©ECORFAN-México, Pachuca, 2017.

Abstract

Currently, most of the services offered on the internet are backed by the use of public key cryptography, which bases its security on modular arithmetic. The use of modular mathematics so far is safe. However, this research seeks to introduce a new approach to public key cryptography that improves execution times, reducing energy consumption. A different and promising option is the use of Hash functions, which can be exploited to make digital signatures. The protocol proposed in this work incorporates the role of a Certifying Authority (CA), which controls the keys used by the users and through the transaction log that stores allows to guarantee the non-repudiation of the information. This method is suitable for mobile network devices because of the speed and hardware requirements of the Hash functions.

Hash function, Hash chain, Merkle tree, digital signature, mobile devices

1. Introducción

Hoy en día se utilizan diversos métodos de firma digital para la integridad de las transacciones electrónicas en Internet. Algunos de ellos utilizan algoritmos basados en aritmética modular, la cual requiere de una gran cantidad de operaciones computacionales. Además, se necesitan números primos que deben estar en un rango de 200 a 700 dígitos (Hayashi, 2000). Actualmente existen diversos esquemas que utilizan algoritmos basados en aritmética modular, como lo es el RSA (Rivest, Shamir y Adleman) o las curvas elípticas (Hayashi, 2000).

En este trabajo se presentará un nuevo protocolo para firma digital basada en criptografía Hash. De manera preliminar se describirán otros métodos de firma relacionados, los cuales usan como base las funciones Hash. Sin embargo, este protocolo es vulnerable a un ataque de Hombre de En Medio (MITM por sus siglas en inglés), por lo cual se requiere implementar una AC (Autoridad Certificadora), que permita el intercambio seguro de llaves entre los usuarios. Cabe mencionar que en la actualidad no existen esquemas de llave pública basados en criptografía Hash, que proporcionen firma digital y servicios de confidencialidad (Buchmann, 2016).

Se debe señalar que los métodos de firma digital basados en aritmética modular requieren un tiempo considerable para la ejecución de la firma digital desde el proceso de generación de llaves hasta la verificación de la misma. Por lo anterior, en esta investigación se plantea utilizar la criptografía Hash para definir un nuevo esquema de firma digital, en el cual las llaves públicas y privadas, son elementos de una cadena Hash generada a partir de un valor inicial secreto definido por el usuario. En esta investigación se desarrolló un servicio web denominado Autoridad Certificadora (AC) como parte de un sistema cliente-servidor, la cual garantiza el servicio de no repudio entre los usuarios que intervienen en la firma digital de un documento o mensaje.

Los principales componentes del protocolo de firma digital basado en criptografía Hash son: Una función Hash, que recibe como entrada un conjunto de bits de longitud arbitraria o mensaje y genera como salida un conjunto de bits de longitud preestablecida (Smart, 2016). A esta representación se le llama código Hash o resumen del mensaje, el cual será representado en este documento como $f(x)$. Toda función o algoritmo Hash debe cumplir las siguientes características (Barreto, 2013):

a. Dada la firma digital debe ser computacionalmente imposible recuperar el mensaje original.
b. Compresión: La firma digital debe ser de longitud fija, independientemente de la longitud del mensaje original.
c. Facilidad de cálculo: Dado el mensaje original, debe ser fácil calcular la firma digital de este.

d. Difusión: La firma digital debe ser una función compleja de todos los bits del mensaje, de tal forma que, si se modifica un solo bit, el hash resultante deberá cambiar al menos la mitad de sus bits aproximadamente.

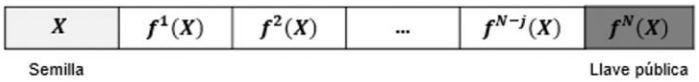
e. Resistente a colisiones. Existen dos tipos:

– Colisión Simple. Dado el mensaje x_0, debe ser computacionalmente imposible obtener otro mensaje x_1, tal que el hash de x_0 sea idéntico al de x_1.

– Colisión fuerte. Será computacionalmente complicado encontrar un par (x_0, x_1) de forma que el hash de x_0 sea idéntico al de x_1.

La cadena Hash, la cual es una secuencia de valores derivados consecutivamente de una función Hash y un valor inicial. Debido a las propiedades de la función hash, es relativamente fácil calcular sucesivamente valores encadenados (Horne, 2011).

Figura 3.1 Cadena Hash



El HMAC (Hash Message Authentication Code) es una función se usa el código de la función hash sin modificación alguna, esta construcción también involucra una sola clave de longitud arbitraria. Lo cual proporciona algunas ventajas en el nivel del manejo de las claves (Martínez, 2004). Un Árbol de Merkle es esencialmente una estructura que se construye con una función Hash resistente a colisiones para producir una pequeña descripción criptográfica de la raíz. Los nodos hoja contienen los valores de Hash de los datos de interés, es decir, el número de serie de los certificados revocados, y los nodos internos los valores de Hash que resultan de aplicar la función Hash de sus nodos hijos. De esta manera, un gran número de datos separados puede ser ligado a un único valor de Hash: el Hash del nodo raíz del árbol (Gañan, Caubet, Esparza, Mata-Díaz, & Montenegro).

1.1. Antecedentes

En esta sección abordaremos los sistemas de firma digital, clasificándolos en aquellos que basan su seguridad en aritmética modular y los que se basan en criptografía Hash. Los primeros son susceptibles de ser analizados por una computadora cuántica, ya que puede procesar más información en menos tiempo que un ordenador clásico. Los ordenadores clásicos pueden resolver problemas de complejidad NP y las computadoras cuánticas pueden resolver problemas de la clase BQP (bounded error quantum polynomial time), la cual incluye la clase P y algunos problemas de la clase NP como factorización entera y logaritmo discreto (Lizama-Pérez, 2015) y (Prieto, 2015) .

1.1.1. Algoritmos pre-cuánticos

Los algoritmos considerados como pre-cuánticos se sustentan en problemas derivados de la aritmética modular. A continuación, se describen brevemente algunos de los algoritmos más representativos.

– El método RSA (Rivest-Shamir-Adleman) desarrollado en 1977, es el algoritmo de cifrado y autenticación más utilizado en Internet (Network Dictionary, 2007). Este algoritmo se basa en el problema de la factorización entera, cuya dificultad radica en encontrar dos números primos grandes, dado el producto de ambos.

- La Criptografía de Curva Elíptica (ECC) fue propuesto por Koblitz. La principal ventaja de este método es que los tamaños de llave son más pequeños en comparación a otros criptosistemas como RSA. Sin embargo, ECC ofrece un nivel de seguridad similar a RSA (Zilong, 2014). La seguridad de ECC se basa en la dificultad de problemas de curva elíptica de logaritmos discretos.
- El-Gamal, es uno de los criptosistemas más establecidos de llave pública, este sistema adoptó el algoritmo de intercambio de claves Diffie-Hellman. La seguridad de este criptosistema se basa en problemas de Logaritmo Discreto sobre campos finitos (A. Mandangan, 2014).
- El Algoritmo de Firma Digital (DSA) está basado en la dificultad computacional para resolver logaritmos discretos (Stallings, 2014), fue desarrollado por la Agencia de Seguridad Nacional de EEUU para generar una firma digital para la autenticación de documentos electrónicos. DSS fue propuesta por el Instituto Nacional de Estándares y Tecnología (NIST) en 1994, y se ha convertido el estándar gobierno de los Estados Unidos para la autenticación de la electrónica documentos (Network Dictionary, 2007). Este estándar puede ser implementado utilizando DSA, RSA o Curva elíptica (ECDSA) (Gallagher, 2013).

1.1.2. Algoritmos post-cuánticos

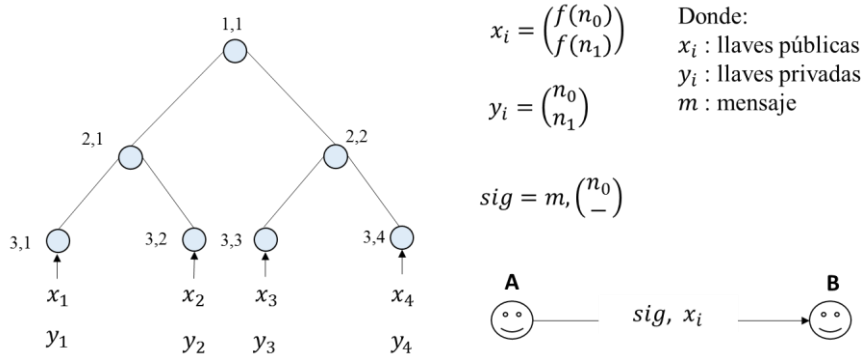
El advenimiento de las computadoras cuánticas podría conducir al rompimiento de los sistemas que actualmente están en uso en internet. Esto debido a que el algoritmo de Shor fue diseñado para llevar a cabo el análisis de los métodos basados en matemáticas modulares (NIST, 2016) y (Lizama-Pérez, 2015). Actualmente el National Institute of Standards and Tecnology (NIST) ha publicado una convocatoria para evaluar y estandarizar uno o más algoritmos criptográficos de llave pública de resistencia cuántica (NIST, 2016). Las funciones Hash están consideradas como una forma de criptografía post-cuántica (Buchmann, 2016). El algoritmo de firma digital Hash que se describe en este trabajo es por tanto, un método post-cuántico. Sin embargo, a continuación se describirán otros métodos que usan funciones Hash.

Uno de los primeros algoritmos que usa funciones Hash para lograr servicios de autenticación es la Contraseña de Una Sola Vez (OTP) (Lamport, 1981). Este algoritmo genera una cadena de valores Hash para ser usadas durante *n* sesiones de autenticación. El último elemento de la cadena se conoce como llave pública $z_0 = f(z_1)$, la cual se publica y se comparte con el servidor. Posteriormente el cliente usa la llave anterior z_1 en la primera sesión de autenticación. El servidor no conoce z_1 pero como conoce el valor de z_0 es posible verificar su identidad con la relación $z_0 = f(z_1)$. La siguiente llave a utilizar es z_2 para que pueda ser verificada con la relación $z_1 = f(z_2)$ y así hasta agotar el número de sesiones.

El inconveniente de este protocolo de autenticación es que las llaves se agotan y no existe una manera segura de renovación, a menos que se inicie el proceso de autenticación nuevamente entre el cliente y el servidor. Además, este proceso lo debe realizar el usuario de manera personal y en sitio como el servidor. De otro modo, si el protocolo se ejecuta de manera remota la contraseña inicial podría ser del conocimiento del adversario, lo cual comprometería la cadena Hash que se usará para las sesiones de autenticación.La Firma de Una Sola Vez, también conocida como One Time Signature (OTS) fue introducido por Leslie Lamport (Bicakci, 2003). OTS ofrece una alternativa viable a las firmas digitales basadas en llaves públicas. Para la descripción de este método utilizaremos una función Hash hipotética de 1 bit. Para firmar un mensaje se necesitan dos números aleatorios n_0, n_1 que sirven como llave privada. El firmante comparte con el destinatario la llave pública que se obtiene después de realizar el cómputo de la función Hash sobre los valores de la llave privada, es decir $f(n_0), f(n_1)$, donde $f()$ denota la función Hash. Para la firma del mensaje es necesario obtener el Hash del mensaje el cual puede ser 0 ó 1. El par $(n_0, f(n_0))$ representa la firma del mensaje 0 y $(n_1, f(n_1))$ la firma del mensaje 1 como se ve en la Figura 3.2

Por lo anterior, la firma del mensaje requiere que uno de los números aleatorios siempre sea revelado. Para verificar la autenticidad de la firma se calcula $f()$ del número aleatorio recibido, el cual debe coincidir con la llave pública del emisor. La firma que se obtiene de este método como su nombre lo indica es útil solo una vez, por lo que no puede reusarse ya que siempre es revelada la mitad de la llave privada lo que comprometería la seguridad del protocolo.

Figura 3.2 Firma de Una Sola Vez con árboles de Merkle



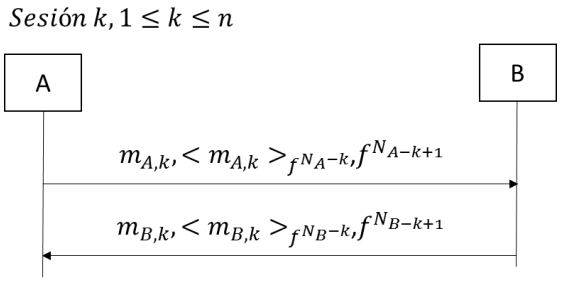
Ralph Merkle introdujo una mejoría a este esquema, de modo que las firmas quedan incrustadas en una estructura de árbol, lo que permite ampliar el número de firmas del protocolo 3.2. El emisor genera un árbol de Merkle como se describió en el apartado anterior. El tamaño del árbol se determina considerando que el número de hojas constituye el número de veces que se puede realizar la firma. La llave pública del emisor es la raíz del árbol. Sin embargo, cada hoja del árbol contiene una llave pública que se usará para firmar un mensaje, véase la Figura 3.2.

Como ejemplo consideremos la firma del primer mensaje: el emisor debe enviar el mensaje m, la llave pública x_1 y la porción correspondiente de la llave privada y_1. Además, la verificación de la raíz del árbol requiere los nodos 2,2 y 3,2, los cuales deberán ser enviados por el emisor. En este protocolo las firmas requieren mayor capacidad de procesamiento, ya que el cómputo del árbol recae en el usuario firmante. Por lo anterior, este método no es óptimo cuando se trata de usuarios móviles debido a sus limitaciones de cómputo y almacenamiento.

1.1.3. Trabajos relacionados

El protocolo de Autenticación de Mensajes Retardados o Delayed Message Authentication (DeMA) se describe en (Groza, 2006): Es un protocolo peer-to-peer, es decir, no es un protocolo que funciona como cliente-servidor. Se usa una cadena Hash para cada entidad A y B. El protocolo está compuesto por un número variable de sesiones k; una sesión equivale a 2 rondas. Cada sesión proporciona información necesaria que brinda la autenticación del mensaje anterior. El protocolo puede ser suspendido en cualquier sesión únicamente por A, la autenticación del mensaje de la sesión donde se detuvo se confirmará cuando el protocolo se reinicie. De lo anterior se deduce que este protocolo requiere una etapa de sincronía al inicio de la comunicación.En la Figura 3.3 se observa el protocolo de comunicación donde m es el mensaje de la sesión k, el cual es firmado por medio de una función HMAC con la llave f^{N_A-k} o f^{N_B-k} respectivamente.Una vez que el firmante da a conocer la llave el receptor puede verificar la autenticidad de la firma anterior. Por ejemplo, con la llave f^{N_A-k} la cual se da a conocer en la sesión $k+1$, se puede verificar la firma del mensaje $m_{A,k}$ de la sesión k.

Figura 3.3 Protocolo de Autenticación de Mensajes Retardados (DeMA)



El protocolo de Autenticación de Mensajes Directos (DiMA) corrige el problema de que el mensaje sea verificado en la sesión posterior, el cual tiene como principal característica el uso de dos cadenas Hash por cada entidad. En cada ronda dos elementos de cada cadena Hash son enviados y la autenticación del mensaje es recuperada de estos elementos. Al igual que el protocolo DeMA éste protocolo también tiene como restricción que solo una entidad es capaz de iniciar o parar la comunicación. Además de que la comunicación se da solamente entre entidades punto a punto. Otra desventaja de este protocolo es que las cadenas hash se agotan rápidamente y se necesitan de muchas sesiones para enviar pequeñas cantidades de información.

2. Desarrollo

En esta sección se describirá el protocolo de firma digital con la AC basado en criptografía Hash, y el registro de usuarios ante la AC. Posteriormente, en el apéndice A se presenta el análisis de seguridad del protocolo cuando el atacante compromete uno o más nodos del árbol de Merkle. El protocolo básico de firma digital basado en criptografía Hash es equivalente al protocolo DeMA descrito en el apartado anterior. Este protocolo es vulnerable a un Ataque de Hombre de En Medio porque en un ambiente multiusuario se producen saltos en el número de la llave por lo cual es imposible verificar la validez de la última llave recibida, haciendo posible que un atacante use una llave obsoleta como si fuera una llave fresca. En contraste, nuestro protocolo está enfocado para que funcione bajo el paradigma cliente-servidor a través de un servicio de certificación en la nube denominado AC.

El principal objetivo de la AC es garantizar la frescura de las llaves realizando el proceso básico de firma con cada usuario. Para ello, la AC debe ejecutar el protocolo punto a punto con cada usuario, lo que implica que debe prevenir el crecimiento indefinido de su llave pública a través de un método para el manejo de las llaves. El método usado en nuestro protocolo es el árbol de Merkle. Cada hoja del árbol de Merkle contiene una llave pública, es decir, el último elemento de la cadena Hash. Esto significa que la AC mantiene una semilla o valor secreto por cada usuario que debe permanecer resguardada por la AC.

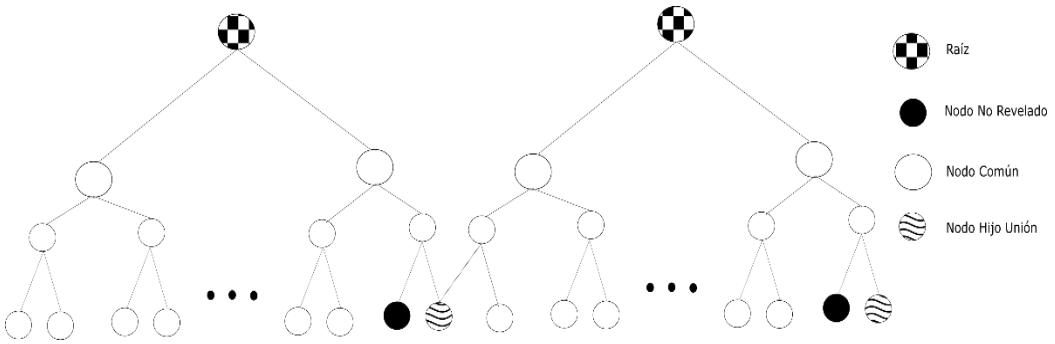
2.1. Autenticación de la AC

Para la autenticación de la llave pública de la AC, se usa un árbol de Merkle donde la raíz del árbol es la llave pública de la AC. Cuando un usuario se registra ante la AC, ésta le envía una lista de nodos para que pueda llegar a la raíz de la AC. El funcionamiento de un árbol de Hashes de Merkle se realiza de la siguiente manera: cada nodo padre del árbol concatena los hashes de sus nodos hijos y procede al cómputo del Hash de dicha concatenación. El proceso se repite en cada nodo hasta alcanzar la raíz del árbol. La estrategia anterior, implica que la AC debe crear de antemano un árbol del tamaño suficiente para los usuarios que van a ser registrados en el sistema. Dado que este enfoque no es eficiente el problema de escalabilidad se resuelve creando un encadenamiento de árboles de Merkle. No obstante, cada nuevo árbol se crea conforme se requiera registrar más usuarios.

Para la creación del nuevo árbol las dos últimas hojas del primer árbol se mantienen en secreto por la AC, es decir no se publican hasta que la AC crea el nuevo árbol. El nuevo árbol se crea tomando como primera hoja la última hoja del árbol anterior, de esta manera el nuevo árbol queda enlazado con el primer árbol. La AC publica la raíz del nuevo árbol y da a conocer el valor Hash de los dos nodos hojas ocultas, así como la lista de nodos que permiten verificar la raíz del nuevo árbol.

En la Figura 3.4 se observa que el último nodo del primer árbol sirve para enlazarse con el siguiente árbol, es decir el "Nodo Hijo Unión" del árbol izquierdo se une con el árbol derecho, logrando así el encadenamiento de árboles.

Figura 3.4 Encadenamiento de árboles de Merkle



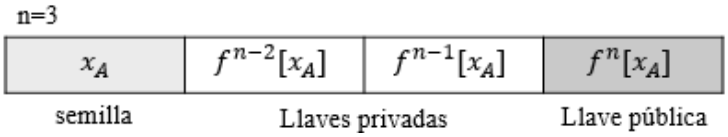
2.2. Protocolo de Firma Digital con la Autoridad Certificadora

Para la descripción del Protocolo de Firma Digital con la Autoridad Certificadora se usará un ejemplo básico donde los usuarios y la AC crean cadenas de longitud tres. Para ello se requieren las variables que aparecen listadas y descritas en la Tabla 3.1.

Tabla 3.1 Nomenclatura de firma digital

Símbolo	**Significado**
A	Alice
B	Bob
AC	Autoridad Certificadora
f	Función Hash
h	Hash del mensaje
$<>_f$	Función HMAC, donde f es la llave requerida
$f^3[x_i]$	Llave pública de i, donde i puede ser A o B
$f^2[x_i]$	Primera llave privada que usa i como medio de verificación ante la AC, donde i puede ser A o B
$f^1[x_i]$	Segunda llave privada de i
x_i	Semilla de i
$f^3[x_{AC,i}]$	Llave pública que comparte la AC con i, donde i puede ser A o B
$f^2[x_{AC,i}]$	Primera llave privada que comparten la AC con i, donde i puede ser A o B
$f^1[x_{AC,i}]$	Segunda llave privada que comparte la AC con i, donde i puede ser A o B
$x_{AC,i}$	Semilla que comparte la AC con i
$s_0 =< h >_{f^1[x_A]}$	HMAC del Hash del mensaje de A firmado con su segunda llave privada
$s_1 =< s_0 >_{f^2[x_{AC,B}]}$	HMAC de s_0 firmado con la segunda llave privada que la AC comparte con Bob $f^1[x_{AC,B}]$
Para la descripción de este ejemplo usaremos una longitud de la cadena 3, simbólicamente $n = 3$. En este ejemplo, la llave publica de A es $f^3[x_A]$ como se muestra en la **¡Error! La autoreferencia al marcador no es válida.**	

Figura 3.5 Cadena Hash de llaves privadas y pública de Alice (A)

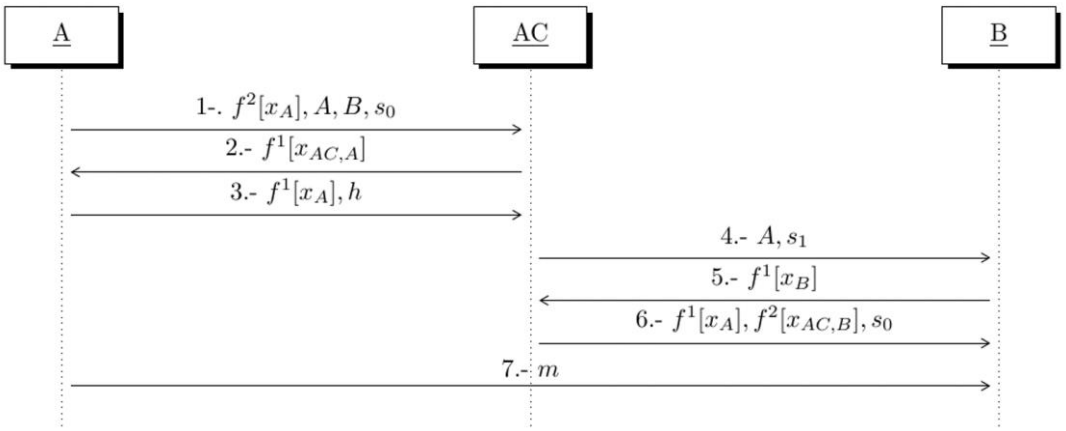


El proceso de firma inicia cuando Alice (A) envía su petición a la AC, la cual contiene en primer término la llave privada $f^2[x_A]$, que se usa como medio de autenticación ante la AC. Esta medida de autenticación del usuario se usa para evitar que la AC procese solicitudes de firma falsas. El proceso de autenticación se realiza de la siguiente manera: la AC realiza el cómputo del Hash de $f^2[x_A]$, simbólicamente $f\left[f^2[x_A]\right] = f^3[x_A]$ que debe corresponder con la llave pública de A.

Además de la llave de autenticación, A envía s_0 que es la firma del Hash del mensaje, así como el nombre de Alice (A) y el nombre de Bob (B). La AC responde enviando la primera llave privada que comparte con A, para que ésta verifique que se está comunicando con la AC. Este proceso corresponde a una doble autenticación entre el usuario y la AC. Entonces A revela la llave $f^1[x_A]$ así como el Hash del mensaje h. La AC realiza el proceso de verificación de firma mediante el cómputo del HMAC sobre h usando la llave $f^1[x_A]$ el cual debe coincidir con el HMAC que la AC recibió previamente en el paso 1, como se observa en la Figura 3.6.

Como siguiente paso, la AC reenvía a B la petición de A, que contiene el nombre de A así como s_1, el cual es el HMAC de s_0 firmado con $f^2[x_{AC,B}]$, que es la primera llave privada que la AC comparte con B. Ahora, B envía su llave de autenticación a AC, es decir $f^1[x_B]$. Cuando la AC ha autenticado a B, como se describió anteriormente entonces la AC revela s_0 , así como las llaves $f^1[x_A]$ y $f^2[x_{AC,B}]$, ésta última es la que usó para firmar s_0. Posteriormente, B realiza la verificación de la firma de s_1 de la misma forma que la AC verificó s_0 enviado por A.A continuación A comparte el mensaje m a B a través de un medio de comunicación como correo electrónico, redes sociales o algún servicio de mensajería en la nube. Finalmente, B verifica el mensaje de la siguiente manera: B realiza el cómputo del HMAC tomando como entradas el Hash del mensaje m, es decir $f(m)$ y la llave $f^1[x_A]$. Dicho cómputo debe coincidir con s_0 , que ha recibido previamente de la AC como se describe en la Figura 3.6

Figura 3.6 Protocolo de Firma Digital con la Autoridad Certificadora



2.3 Registro de los usuarios ante la AC

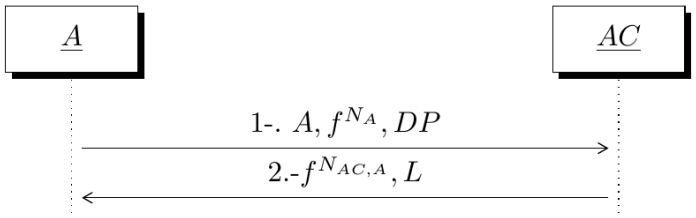
Para que un usuario pueda firmar un documento o mensaje, es necesario que se registre ante la AC la cual ha creado un árbol de Merkle para el registro de usuarios. El intercambio de mensajes que se requiere para el registro de usuarios se aprecia en la Figura 3.7, donde las variables no están definidas a ciertos valores específicos como en ejemplo anterior. A continuación se describe cada paso del protocolo:

1. Alice envía su nombre de usuario (A), su llave pública (f^{N_A}) y sus Datos Personales (DP) a la AC. Dónde:
   a. f: función Hash
   b. N_A: longitud de la cadena Hash de A.

2. La AC responde enviando la llave pública ($f^{N_{AC,A}}$) que compartirá con A y una lista de Hashes (L) para que ésta obtenga la raíz del Árbol de Merkle. Dónde:
   a. $N_{AC,A}$: contador que comparten AC y A.

El usuario A realiza la verificación del árbol de Merkle como se describió en la página 25. Esta verificación corresponde al proceso de autenticación de la AC. Con esto se garantiza que el usuario se está registrando ante la instancia legítima. Este proceso de registro es esencial en el protocolo de firma porque un oponente podría tomar control de los registros de usuario y aprovecharlos para registrar a nuevos usuarios.

Figura 3.7 Esquema de Registro de usuarios ante la AC



2.3. Análisis de Seguridad

A continuación se discutirá el ataque de Hombre de En Medio (MITM) cuando una entidad maliciosa se registra ante la AC para montar una AC falsa. Por lo tanto, para montar un ataque MITM en un nodo, es necesario que Eva se registre ante la AC usando un nombre de usuario en este caso E. A continuación, Eva ejecuta el protocolo de firma con algún usuario para obtener llaves privadas de la AC, las cuales utilizará para suplantar la identidad de la AC ante A y suplantar la identidad de A ante B. Con el propósito de validar las transacciones de firma digital los usuarios deben usar un servicio de contactos, de tal manera que una vez que se ha descargado la aplicación ésta les permite dar de alta continuamente a nuevos contactos. En el proceso de alta de contactos, los usuarios intercambian sus llaves públicas. Posteriormente, el usuario solicita a la AC que verifique los contactos añadidos para validar sus llaves públicas.Por lo anterior, es necesario que A y B se añadan previamente como contactos en su lista de direcciones. De esta manera, A y B validan sus llaves públicas ante la AC por un canal y en un momento distinto a los usados durante la etapa de registro. De esta forma, si la identidad de AC es suplantada por Eva ante A, la llave pública de A no podrá ser verificada por B, ni A podrá realizar la verificación de la llave de B. Por lo tanto, este ataque no se puede llevar a cabo eficientemente por un adversario con capacidades de cómputo convencionales. Este análisis se puede generalizar para escenarios donde el atacante pueda intervenir más de un nodo.

3. Resultados

El desarrollo de la AC se implementó en el entorno de desarrollo de software NetBeans 8.2 y el servidor de Base de Datos MySQL Community Server 5.7. El cliente móvil está desarrollado en Android Studio 2.3. Para las pruebas se usaron dos dispositivos móviles, las características específicas de estos dispositivos se presentan en la Tabla 3.2. Para medir el tiempo de los procesos de generación de llave, ejecución y verificación de la Firma Digital se realizaron los siguientes pasos:

a. Iniciar las variables de tiempo en 0 de $tiempoInicial$ y $tiempoFinal$ en cada proceso
b. Obtener el tiempo de inicio en milisegundos con el objeto System y su método currentTimeMillis que se encuentra definido en Java.
c. Al término de cada proceso se obtiene el tiempo final con el mismo objeto System y se hace el cómputo de la diferencia para obtener el $tiempoObtenido$ del proceso.

Tabla 3.2 Características de dispositivos móviles utilizados para pruebas

Dispositivo	**Modelo**	**CPU**	**Versión Android**	**Memoria Interna**
Samsung	GT I8190	1 GHz dual core ARM Cortex-A9	4.1.2 Jelly Bean	8 GB
Samsung	SM G920I	2.1 GHz, 1.5 GHz Octa-Core	6.0.1 Marshmallow	32 GB

Los resultados obtenidos a partir de las pruebas realizadas del sistema de firma digital hash en el equipo de cómputo donde se aloja la AC se presentan en las Tabla 3.3 y Tabla 3.3 Dichos resultados se muestran en relación a los tiempos que requiere el sistema para la generación de llave, firma y verificación en milisegundos y se comparan con los resultados publicados en (Alese, 2012) y (Mahto, 2016).

Tabla 3. 3 Resultados comparados con (Alese, 2012) en milisegundos usando un mensaje de 100 bytes

Tamaño de la Llave	**Generación de Llave**	**Firma**	**Verificación**
RSA 1024	1312.7±190.8	166.9±46.3	15.7±0.4
RSA 2048	6804.6±2540.6	290.2±29.8	122.4±9.1
RSA 3072	32180.0±18947.7	310.5±75.5	293.2±71.8
RSA 7680	322843.0±233809.0	352.1±154.1	2932.8±44.7
RSA 15360	N/A	N/A	N/A
ECC P-160	198.6±12.5	17.9±4.9	15.7±0.1
ECC P-224	208.3±13.4	95.9±6.8	18.7±5.5
ECC P-256	243.5±22.2	35.1±6.1	21.1±6.8
ECC P-384	294.0±26.5	74.9±7.1	47.7±3.2
ECC P-521	447.8±90.9	138.2±4.9	109.9±0.3
AC con SHA-256	240.25	39.21	12.5

Como se puede observar en la Tabla 3.3 , el servidor de la AC genera tiempos que son menores a los reportados de la ejecución del algoritmo RSA durante las fases de Generación de Llave, Firma y Verificación comparado con (Alese, 2012). Por el lado de ECC, la AC es más rápida a partir de la versión ECC P-384 en las tres fases de la firma digital.

Tabla 3.4 Resultados comparados con (Mahto, 2016) en milisegundos

	1 byte		**32 bytes**	
Tamaño de la Llave	Generación de Firma	Verificación de Firma	Generación de Firma	Verificación de Firma
RSA 1024	30.7	754.3	550	19310
RSA 2048	29.9	2707.5	580	102030
RSA 3072	30.5	6940.9	560	209600
ECC 160	488.5	1326.7	7920	22880
ECC 224	2203	1586.3	39700	26330
ECC 256	3876.3	1769	58430	27400
AC con SHA-256	32.7	18.5	39.87	17.8

Derivado de los resultados de la Tabla 3.4 se puede apreciar que la AC no es más rápida que RSA en la etapa de Generación de Firma (G.F.) tomando como entrada un mensaje de 1 byte, pero en el caso de que el mensaje sea de un tamaño de 32 bytes, la AC es más rápida. Sin embargo, en la etapa de Verificación de Firma (V.F.) el tiempo de ejecución es menor para cualquier tamaño de llave RSA. En cuanto a ECC, se puede observar que la AC es más rápida para cualquier tamaño de llave ECC con un mensaje de 1 byte como de 32 bytes en las fases G.F. y V.F. De lo anterior se desprende que el sistema de firma basado en Criptografía Hash se encuentra dentro de parámetros óptimos en el estado actual de la tecnología.

En la Tabla 3.5 se presentan los resultados obtenidos en los dispositivos móviles descritos anteriormente, estos resultados son comparados con los resultados publicados en (Tayoub).

Tabla 3.5 Comparación en Dispositivos Móviles del lado del usuario

	ECC (ECDSA) (Tayoub)			
Dispositivo móvil	**Tamaño de llave de Operación (bits)**	**Generación de llave**	**Firma**	**Verificación**
GT-S6102	160	767	561	1285
	224	615	699	1217
	521	595	562	141
GT-I9100	160	303	412	681
	224	443	360	624
	521	595	562	141
	RSA (Tayoub)			
Dispositivo móvil	**Tamaño de llave de operación (bits)**	**Generación de llave**	**Firma**	**Verificación**
GT-S6102	1024	1088.66	14.33	1
	2048	3896	77.66	1
	15360	>1h	\	\
GT-I9100	1024	1150	28	0.33
	2048	2341.66	95	1
	15360	>1h	\	\
	Firma Digital Hash usando SHA-256			
Dispositivo móvil	**Tamaño de llave de operación (bits)**	**Generación de llave**	**Firma**	**Verificación**
GT-I8190	256	1678.47	566.75	468.91
SM-G920I	256	158.55	128.65	94.22

De los resultados anteriores se observa que el uso de funciones Hash mejora tiempo de generación de llaves públicas y privadas, así como el proceso de firma y verificación cuando es comparado con ECC. En cuanto a RSA, el protocolo de firma con criptografía Hash es más rápido en el proceso de generación de llaves mayores a 15360 bits.

4. Conclusiones

El sistema de firma digital con funciones Hash se encuentra dentro de los parámetros óptimos de la tecnología actual debido a que los tiempos obtenidos en las fases de generación de llaves, firma y verificación resultaron menores a los tiempos que arrojan los algoritmos basados en aritmética modular como RSA y ECC. Por lo anterior, la criptografía Hash es una tecnología que puede ser explotada para mejorar los protocolos de autenticación, especialmente para firma digital en dispositivos móviles. Esto debido a la fortaleza criptográfica de estas funciones y su facilidad de cómputo.

De igual manera al llevar a cabo el análisis de seguridad del Ataque de Hombre de En Medio, se tiene que si los usuarios intercambian sus llaves públicas antes de realizar cualquier firma de mensajes éstos verificarán que los involucrados en dicho proceso están registrados ante la AC. Por lo tanto se deduce que el Ataque de Hombre de En Medio no se puede implementar con ningún nodo intervenido.

5. Referencias

A. Mandangan, Y. L. (2014). ElGamal Cryptosystem with Embedded Compression-Crypto Technique. *AIP Conference Proceedings, 1635*(1), 455-460.

Alese, B. a. (2012). Comparative analysis of public-key encryption schemes. (Citeseer, Ed.) *International Journal of Engineering and Technology, 2*(9), 1552--1568.

Barreto, P. a. (2013). Introdução à criptografia pós-quântica. *Minicursos do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computaciona*.

Bicakci, K. a. (2003). How to construct optimal one-time signatures. (Elsevier, Ed.) *Computer Networks, 43*(3), 339--349.

Buchmann, J. A. (2016). Post-Quantum Cryptograph: State of the Art. En *The New Codebreakers* (págs. 88--108). Springer.

Gallagher, P. (2013). Digital signature standard (dss). *Federal Information Processing Standards Publications, volume FIPS*.

Gañan, C., Caubet, J., Esparza, O., Mata-Díaz, J., & Montenegro, J. A. (s.f.). Estructura de Datos Autenticadas para Gestionar Datos de Revocación en VANETs. 6.

Groza, B. (2006). Using one-way chains to provide message authentication without shared secrets. En *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006. SecPerU 2006. Second International Workshop on* (págs. 82--87). IEEE.

Hayashi, A. (2000). A new fast modular multiplication method and its application to modular exponentiation-based cryptography. *Electronics and Communications in Japan(Part III Fundamental Electronic Science), 83*(12), 88--93.

Horne, D. (2011). Hash chain. En Springer (Ed.), *Encyclopedia of Cryptography and Security* (págs. 542--543).

Lamport, L. (1981). Password authentication with insecure communication. (ACM, Ed.) *Communications of the ACM, 24*(11), 770--772.

Lizama-Pérez, L. A. (2015). Descifrando un Mundo Complejo.

Mahto, D. a. (2016). Security Analysis of Elliptic Curve Cryptography and RSA. En *Proceedings of the World Congress on Engineering.*

Martínez, I. a. (2004). Realizacion del esquema de autenticacion HMAC empleando la funcion HASH MD5. (E. U. Cuba, Ed.) *Ingeniería Electrónica, Automática y Comunicaciones, 25*(1), 76--82.

Network Dictionary. (2007). *Digital Signature Standard*, 150.

NIST. (2016). *POST-QUANTUM CRYPTO STANDARDIZATION*. Recuperado el 06 de 07 de 2017, de http://csrc.nist.gov/groups/ST/post-quantum-crypto/index.html",

Prieto, d. l. (2015). COMPUTACIÓN CUÁNTICA Y SU REALIZACIÓN FÍSICA.

Smart, N. P. (2016). Hash Functions, Message Authentication Codes and Key Derivation Functions. En S. I. Publishing (Ed.), *Cryptography Made Simple* (págs. 271--294). doi:10.1007/978-3-319-21936-3_14

Stallings, W. (2014). *Cryptography and network security: principles and practice* (Vol. 6). Pearson.

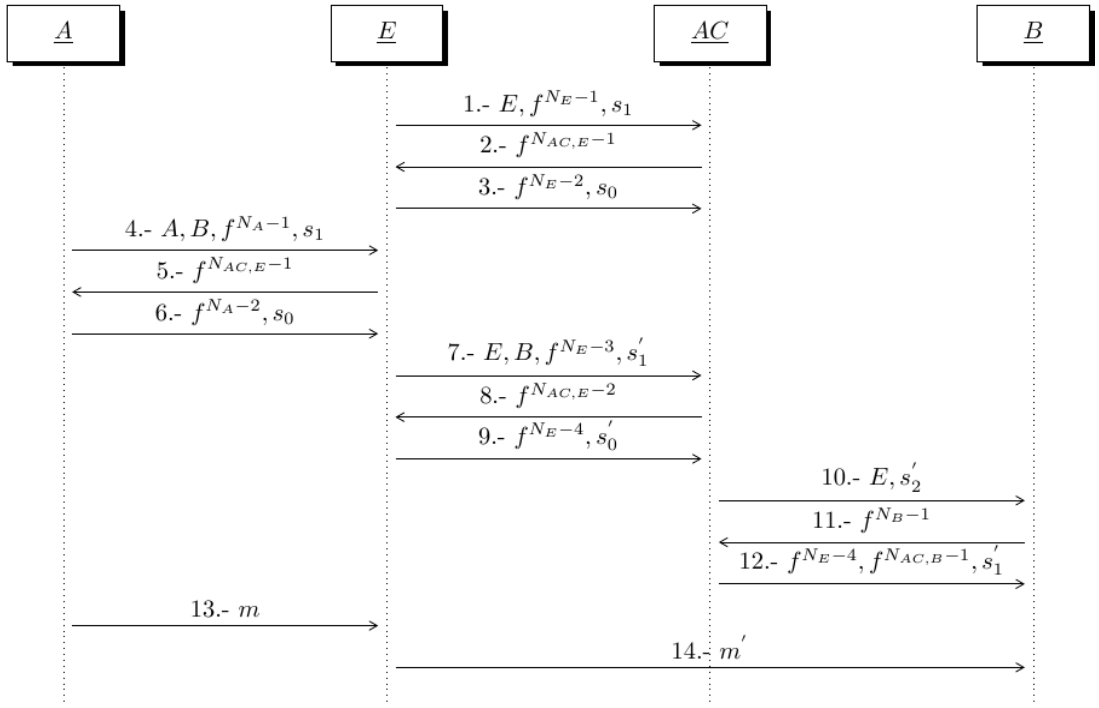
Tayoub, W. a. (s.f.). Implementation Of Public-Key Cryptographic Systems On Embedded Devices (Case: Computation Speed).

Zilong, L. D. (2014). Design of an Elliptic Curve Cryptography Processor for RFID Tag Chips. *Sensors (14248220), 21*(10), 17883-17904.

6. Anexos

Apéndice A

FigurA 3.8 Ataque de Hombre de En Medio con un nodo intervenido



1. Eva (E) envía E, f^{N_E-1} y s_1 hacía AC.
    a. E: nombre de usuario de Eva.
    b. f^{N_E-1}: llave privada de Eva.
    c. $s_1 =< s_0 >_{f^{N_{E-2}}}$: HMAC de s_0 firmada con $f^{N_{E-2}}$.
        i. $s_0 = f(x)$: Hash del mensaje x.
2. AC envía $f^{N_{AC,E}-1}$ hacía E.
    a. $f^{N_{AC,E}-1}$: llave que comparten AC y E.
3. E manda s_0 y $f^{N_{E-2}}$ hacia la AC.
4. Alice (A) envía A,B,f^{N_A-1} y s_1.
    a. A: nombre de usuario de Alice.
    b. B: nombre de usuario de Bob.
    c. f^{N_A-1}: llave de Autenticación de A.
    d. $s_1 =< s_0 >_{f^{N_A-2}}$: HMAC de s_0 firmado con f^{N_A-2}.
        i. $s_0 = f(m)$: Hash del mensaje (m).
5. Eva (E) envía $f^{N_{AC,E}-1}$para que A piense que se está comunicando con la AC.
    a. $f^{N_{AC,E}-1}$: llave que E comparte con AC.
6. A manda f^{N_A-2} y s_0 hacia la AC, para lo cual E lo intercepta.
7. E envía E, B, f^{N_E-3} y s_1' hacía AC.
    a. $s_1' =< s_0' >_{f^{N_E-4}}$: HMAC de s_0' firmado con f^{N_E-2}.
        i. $s_0' = f(m')$: Hash de m'.
8. AC manda $f^{N_{AC,E}-2}$ hacía E.
9. E envía f^{N_E-4} y s_0'.
10. AC manda E y s_2'.
    a. $s_2' =< s_1' >_{f^{N_{AC,B}-1}}$: HMAC de s_1' firmado con $f^{N_{AC,B}-1}$.

i. $f^{N_{AC,B}-1}$: llave que comparten AC y B.

11. B envía f^{N_B-1} hacia la AC.
12. AC manda f^{N_E-4}, $f^{N_{AC,B}-1}$ y s_1'.
13. A envía m hacia B pero E lo intercepta.
14. E modifica m y en lugar de que B lo reciba, éste recibe m'.

Cuando B realiza el cómputo de los valores anteriores con los datos obtenidos de la AC, B autentica el mensaje m' que fue modificado por E a partir de m. Sin embargo, B espera un mensaje de A y la AC notifica en el paso 10 que el mensaje proviene de E por lo que este ataque no se realiza con éxito. Bob conoce la llave pública de Alice f^{N_A} por el registro de lo anterior se puede observar que este ataque MITM con un nodo intervenido no es factible, debido a que Bob puede verificar que el mensaje no proviene del emisor auténtico.