# DESARROLLO DE UN ALGORÍTMO DE CIFRADO SIMÉTRICO DE RESUMEN

**JONATHAN DAVE ORJUELA NAVARRETE**

**UNIVERSIDAD DE SAN BUENAVENTURA**
**FACULTAD DE INGENIERÍA**
**PROGRAMA DE INGENIERÍA DE SISTEMAS**
**BOGOTÁ D.C.**
**2008**

# DESARROLLO DE UN ALGORÍTMO DE CIFRADO SIMÉTRICO DE RESUMEN

**JONATHAN DAVE ORJUELA NAVARRETE**

**Proyecto de Grado como requisito para optar por el título de Ingeniero de Sistemas**

**Asesor:**
**ING. IVAN MÉNDEZ ALVARADO**

**UNIVERSIDAD DE SAN BUENAVENTURA**
**FACULTAD DE INGENIEÍA**
**PROGRAMA DE INGENIERÍA DE SISTEMAS**
**BOGOTÁ D.C.**
**2008**

Nota de acpetación:

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

Bogotá, D.C. 7 de Julio de 2008

A mi familia, mi padre Luis Enrique Orjuela, mi madre Clara Ines Navarrete y mi hermano David Orjuela Navarrete, quienes son la fuerza y alegría de mi vida, quienes impulsan mi esfuerzo.  A todos aquellos que han hecho posible este sueño, a compañeros y amigos Fernando Garzón, Jairo Puentes, Camilo Ruiz, Carolina Vargas, entre muchos otros a los cuales ofrezco disculpas por no nombrarlos, pero saben que han aportado en el crecimiento cognitivo y humano en mi...

**CONTENIDO**

## LISTA DE TABLAS

pág.

## LISTA DE FIGURAS

## GLOSARIO

**ACCESO (S) – ACCESS:** con respecto a la privacidad, es la habilidad de un individuo para ver, modificar y refutar lo completa y precisa que pueda ser la información personal identificable (PII) reunida sobre él o ella. Acceso es un elemento de las Prácticas Honestas de Información.

**ANONIMATO (S) – ANONYMITY:** condición en la que la verdadera identidad de un individuo es desconocida.

**ATAQUE LOCAL (S) - LOCAL ATTACK**: ataque dirigido a la PC en el cual el atacante ha iniciado una sesión interactiva.

**ATAQUE POR SERVICIO DENEGADO - DOS (S) - DENIAL OF SERVICE ATTACK:** es un asalto computarizado llevado a cabo por un atacante para sobrecargar o congelar un servicio de red, como un servidor Web o de archivos. Por ejemplo, un ataque puede causar que el servidor esté tan ocupado tratando de responder, que ignorará cualquier petición legítima de conexión.

**ATAQUE REMOTO (S) - REMOTE ATTACK:** es un ataque que tiene como objetivo una PC diferente en la que el atacante ha iniciado una sesión interactiva. Por ejemplo, un atacante puede iniciar una sesión en una estación de trabajo y atacar a un servidor en la misma red o en una diferente.

**AUTENTICACIÓN (S) – AUTHENTICATION:** es el proceso de verificar que alguien o algo es quien o lo que dice ser. En redes de equipos públicos y privados (incluyendo Internet), la autenticación se lleva a cabo comúnmente a través de contraseñas de inicio de sesión.

**AUTORIZACIÓN (S) – AUTHORIZATION:** con referencia a la computación, especialmente en los equipos remotos en una red, es el derecho otorgado a un individuo o proceso para utilizar el sistema y la información almacenada en éste. Típicamente la autorización es definida por un administrador de sistemas y verificado por el equipo basado en alguna identificación del usuario, como son un código o una contraseña.

**BÚFER (S) – BUFFER:** es una región de la memoria reservada para servir como receptáculo intermediario en el cual la información es temporalmente retenida antes de su transferencia entre dos ubicaciones o dispositivos.

**CABALLO DE TROYA (TROYANO) (S) - TROJAN HORSE:** es un programa

computacional que aparentemente es útil pero que en realidad causa daño.

**CAPA DE SOCKETS SEGUROS - SSL (S):** es un protocolo para establecer un canal de comunicaciones encriptado que ayuda a prevenir la interceptación de información crítica, como números de tarjeta de crédito en World Wide Web y en otros servicios de Internet.

**CERTIFICADO (S) – CERTIFICATE:** es un archivo encriptado que contiene información de identificación del usuario o servidor, la cual es utilizada para verificar la identidad y ayudar a establecer un vínculo de seguridad mejorada.

**CERTIFICADO DIGITAL (S) - DIGITAL CERTIFICATE:** Véase certificado.

**CIFRADO (S) – CIPHER:** es un método de encriptación, que utiliza típicamente una clave predefinida y un algoritmo para transformar texto simple en texto cifrado.

**CLAVE (S) – KEY:** en encriptación y firmas digitales, es un valor utilizado en combinación con un algoritmo para encriptar o decriptar información.

**CLAVE PRIVADA (S) - PRIVATE KEY:** una de las dos claves en la encriptación de clave pública. El usuario mantiene la clave privada secreta y la utiliza para encriptar firmas digitales y para decriptar los mensajes recibidos.

**CLAVE PÚBLICA (S) - PUBLIC KEY:** una de las dos claves en la encriptación de clave pública. El usuario da a conocer esta clave al público y cualquiera puede utilizarla para encriptar mensajes que serán enviados al usuario y decriptar la firma digital del usuario. Compare con clave privada.

**CÓDIGO DE AUTENTIFICACIÓN DE MENSAJES - MAC (S):** es un algoritmo que permite a un receptor asegurarse que un bloque de información ha conservado su integridad desde el momento en que se envió hasta el momento en que se recibió.

**CÓDIGO MALINTENCIONADO (S) - MALICIOUS CODE:** software que cuando se ejecuta lo hace con un deliberado propósito de ser perjudicial. Virus, gusanos y caballos de Troya (troyanos) son algunos ejemplos de código malintencionado.

**COLISIÓN HASH:** cuando se obtiene como resultado de dos textos diferentes el mismo hash de salida.

**CONDICIÓN DE DESINCRONIZACIÓN (S) - RACE CONDITION:** es una condición causada por el tiempo de los eventos dentro o entre los componentes de un software. Típicamente, las condiciones de desincronización están asociadas con errores de sincronización que proporcionan una ventana de oportunidad durante la cual uno de los procesos puede interferir con otro, posiblemente introduciendo una vulnerabilidad de seguridad.

**CONTRASEÑA (S) – PASSWORD:** es una cadena de caracteres que el usuario escribe para verificar su identidad en una red o en una PC local.

**CRIPTOGRAFÍA (S) – CRYPTOGRAPHY:** es la utilización de códigos para convertir información por medio de una clave para que sólo un receptor específico pueda leerla. La criptografía es utilizada para permitir la autenticación y el no repudio, y para ayudar a preservar la confidencialidad y la integridad de datos.

**FIRMA DIGITAL (S) - DIGITAL SIGNATURE:** es la información que es incluida con un mensaje o es transmitida separadamente que se utiliza para identificar y autentificar al emisor y la información del mensaje. Una firma digital también puede confirmar que el mensaje no haya sido alterado.

**HASH:** se refiere a una función o método para generar claves o llaves de manera casi unívoca a un documento, registro, archivo, entre otros. El resultado de aplicar esta función se le conoce como Hash.

**LLAVE (S) - KEY:** es una palabra con o sin sentido que se usa como elemento de orden para cifrar, la llave puede variar en su longitud; a mayor longitud, mayor seguridad. Esta pieza de información controla la operación de un algorítmo criptográfico.

# INTRODUCCIÓN

Desde los años del Emperador Romano Julio César, la transmisión de mensajes debía ser protegida, tanto así que fue desarrollado un elemento tan rústico, pero al igual tan seguro para esa época: constaba de un tubo, al cual se le enrollaba una tira de papel, sobre la tira se escribía el mensaje a ser enviado, el receptor debería tener un tubo con el mismo diámetro y tamaño, para poder entender y leer el mensaje correctamente.

Este tipo de técnicas fueron mejorando a través de los años, es asi como nació la criptografía; una ciencia, un arte, un saber, del cual hoy en día se toman muchos elementos. Se ha evolucionado tanto en esta área que para la Seguna Guerra Mundial se desarrolló una máquina de cifrado, exclusiva para la protección de los mensajes enviados entre los generales de más alto rango de Alemánia, dicha máquina se conoció como “La máquina de cifrado de Lorenz”.

Después de pasar por elementos físicos para ocultar mensajes, se pasa a elementos lógicos, como los algorítmos de cifrado actuales, entre los que resaltan DES, IDEA, AES, y otros cuya función son el solo cifrar para obtener la huella digital, conocidos bajo el nombre de Algorítmos de Resúmen, o bien, el nombre técnico algorítmos de cifrado simétrico de resumen. Con el proyecto se presentará un nuevo camino y modelo de cifrado para esta última clase de algorítmos.

# 1. PLANTEAMIENTO DEL PROBLEMA

## 1.1 DESRIPCIÓN Y FORMULACIÓN DEL PROBLEMA

La sistematización de la información requiere algúno de los métodos de cifrado para proteger estos datos y evitar así, que se encuentre comprometia la privacidad, la estabilidad económica o hasta la misma identidad. El problema real de los algorítmos actuales de cifrado, sin importar cual sea su función, es la masificación de la capacidad de computación. Ahora se requiere de menos tiempo para realizar una labor de fuerza bruta o descifrado del mensaje por medio de la busqueda de la clave.

Los algorítmos de cifrado simétricos de resumen presentan grandes debilidades en su seguridad, ya que los más usados MD5 y SHA-1 no son lo suficientemente aptos para la actualidad computacional. De nada nos sirve tener un algorítmo que de como salida un Hash que puede ser duplicado.

¿Cómo se puede desarrollar un nuevo algorítmo de cifrado simétrico de resumen que supere la deficiencia en el hash de los actuales?

## 1.2 JUSTIFICACIÓN

Es importante abrir nuevos caminos, o remodelar los algorítmos, crear un nuevo modelo de cifrado permitirá ampliar las posibilidades de protección y disminuir los riesgos. Los algorítmos de cifrado son elementos fundamentales de la seguridad informática, para la transmisión de mensajes, o para la identificación digital.

## 1.4 OBJETIVOS

### 1.4.1 Objetivo General

Desarrollar e implementar un algoritmo de cifrado simétrico de resumen.

### 1.4.2 Objetivos Específicos

- Analizar el funcionamiento y vulnerabilidades de los diferentes tipos de cifrado actuales.
- Diseñar un nuevo algorítmo simétrico para el cifrado de datos.
- Implementar y realizar pruebas del nuevo algorítmo.

## 1.5 ALCANCES Y LIMITACIONES

- El proyecto culmina con la implementación del algorítmo en el lenguaje de programación web PHP para la demostración y la elaboración del algorítmo en Pseudocódigo para la migración a cualquier lenguaje de programación.
- Se tendrá en cuenta que en la Universidad de San Buenaventura Bogotá no hay un criptoanalista especializado.

# 2. MARCO DE REFERENCIA

## 2.1 MARCO TEÓRICO-CONCEPTUAL

La criptografía (del griego *kryptos*, "ocultar", y *grafos*, "escribir", literalmente "escritura oculta") es el arte o ciencia de cifrar y descifrar información utilizando técnicas matemáticas que hagan posible el intercambio de mensajes de manera que sólo puedan ser leídos por las personas a quienes van dirigidos[1].

Con más precisión, cuando se habla de esta área de conocimiento como ciencia se debería hablar de criptología, que engloba tanto las técnicas de cifrado, la criptografía propiamente dicha, como sus técnicas complementarias: el criptoanálisis, que estudia los métodos que se utilizan para romper[2] textos cifrados con objeto de recuperar la información original en ausencia de la clave[3].

La finalidad de la criptografía es, en primer lugar, garantizar el secreto en la comunicación entre dos entidades (personas, organizaciones, etc.) y, en segundo lugar, asegurar que la información que se envía es auténtica en un doble sentido: que el remitente sea realmente quien dice ser y que el contenido del mensaje enviado, habitualmente denominado criptográma, no haya sido modificado en su tránsito.

Otro método utilizado para ocultar el contenido de un mensaje es ocultar el propio mensaje en un canal de información, pero en realidad, esta técnica no se considera criptografía, sino esteganografía[4]. Por ejemplo, mediante la esteganografía se

1 Wikipedia “Criptografía” Disponible en: http://es.wikipedia.org/wiki/Criptograf %C3%ADa. Consultado: 18 de Febrero de 2008, 6:22 pm.
2 Haciendo referencia a la violación de la seguridad del criptosistema.
3 Al igual que un candado se requiere de una llave especifica para poder abrirlo, esta llave es conocida como clave, aunque tambien se usa la misma palabra.
4 Wikipedia “Esteganografía” Disponible en: http://es.wikipedia.org/wiki/Esteganograf %C3%ADa. Consultado: 18 de Febrero de 2008, 6:37 pm.

puede ocultar un mensaje en un canal de sonido, una imagen o incluso en reparto de los espacios en blanco usados para justificar un texto. La esteganografía no tiene por qué ser un método alternativo a la criptografía, siendo común que ambos métodos se utilicen de forma simultánea para dificultar aún más la labor del criptoanalista[5].

En la actualidad, la criptografía no sólo se utiliza para comunicar información de forma segura ocultando su contenido a posibles fisgones. Una de las ramas de la criptografía que más ha revolucionado el panoráma actual de las tecnologías informáticas es el de la firma digital[6]: tecnología que busca asociar al emisor de un mensaje con su contenido de forma que aquel no pueda posteriormente repudiarlo.

En la jerga de la criptografía, la información original que debe protegerse se denomina texto en claro. El cifrado es el proceso de convertir el *texto plano* en un galimatías ilegible, denominado texto cifrado o criptograma. Por lo general, la aplicación concreta del algoritmo de cifrado (también llamado cifra) se basa en la existencia de una clave: información secreta que adapta el *algoritmo de cifrado* para cada uso distinto. Cifra es una antigua palabra arábiga para el cero, en la antigüedad cuando Europa empezaba a cambiar del sistema de numeración romano al arábigo, se desconocía el cero por lo que este resultaba misterioso, de ahí probablemente cifrado significa misterioso. Las dos técnicas más básicas de *cifrado* en la criptografía clásica son la sustitución (que supone el cambio de significado de los elementos básicos del mensaje -las letras, los dígitos o los símbolos-) y la transposición (que supone una reordenación de las mismas); la gran mayoría de las *cifras* clásicas son combinaciones de estas dos operaciones básicas. El descifrado es el proceso inverso que recupera el *texto plano* a partir del

5 Persona de profesion capaz de analizar el funcionamiento del criptosistema y encontrar las posibles vulnerabilidades que este pueda presentar, para que de esta manera se pueda tener un acceso al mensaje cifrado.

6 O firma electrónica es, en la transmisión de mensajes telemáticos y en la gestión de documentos electrónicos, un método criptográfico que asocia la identidad de una persona o de un equipo informático al mensaje o documento, en función al tipo de firma puede, además, asegurar la integridad del documento o mensaje.

*criptograma* y la *clave*. El protocolo criptográfico especifica los detalles de cómo se utilizan los *algoritmos* y las *claves* (y otras operaciones primitivas) para conseguir el efecto deseado. El conjunto de *protocolos*, *algoritmos de cifrado*, procesos de gestión de claves y actuaciones de los usuarios, en su globalidad es lo que constituyen un criptosistema, que es con lo que el usuario final trabaja e interactúa.

Existen dos grandes grupos de *cifras*: los algoritmos que utilizan una única *clave* tanto en el proceso de *cifrado* como en el de *descifrado* y los que utilizan una *clave* para *cifrar* mensajes y una *clave* distinta para *descifrarlos*. Los primeros se denominan cifras simétricas o de clave simétrica y son la base de los algoritmos de cifrado clásico. Los segundos se denominan cifras asimétricas, de clave asimétrica o de clave pública y clave privada y forman el núcleo de las técnicas de cifrado modernas[7].

En el lenguaje cotidiano, la palabra *código* se usa de forma indistinta con *cifra*. En la jerga de la criptografía, sin embargo, el término tiene un uso técnico especializado: los códigos son un método de criptografía clásica que consiste en sustituir unidades textuales más o menos largas o complejas, habitualmente palabras o frases, para ocultar el mensaje; por ejemplo, "cielo azul" podría significar "atacar al amanecer". Por el contrario, las *cifras* clásicas normalmente sustituyen o reordenan los elementos básicos del mensaje[8]; en el ejemplo anterior, "rcnm arcteeaal aaa" sería un criptograma obtenido por *transposición*. Cuando se usa una técnica de códigos, la información secreta suele recopilarse en un *libro de códigos*.

Con frecuencia los procesos de cifrado y descifrado se encuentran en la literatura como encriptado y desencriptado, aunque ambos son neologismos[9] todavía sin reconocimiento académico. Hay quien hace distinción entre "cifrado/descifrado" y

7 Peiper “Encriptación (parte 1)” Disponible en: http://www.peiper.com.ar/edicion02/encriptacion.pdf. Consultado: 18 de Ferbero de 2008, 7:09 pm
8 Letras, dígitos y símbolos.
9 Anglicísmos de los términos ingleses Encryp y Decryp

"encriptado/desencriptado" según esté hablando de criptografía simétrica o asimétrica, pero la mayoría de los expertos en el mundo académico prefiere evitar ambos neologismos.

**2.1.1 Advanced Encryption Standard, AES[10]** En criptografía, Advanced Encryption Standard (AES), también conocido como Rijndael, es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. Se espera que sea usado en el mundo entero y analizado exhaustivamente, como fue el caso de su predecesor, el Data Encryption Standard (DES). El AES fue anunciado por el Instituto Nacional de Estandares y Tecnologia[11] (NIST) como FIPS[12] PUB 197 de los Estados Unidos (FIPS 197) el 26 de Noviembre de 2001 después de un proceso de estandarización que duró 5 años. Se transformó en un estándar efectivo el 26 de Mayo de 2002. Desde 2006, el AES es uno de los algoritmos más populares usados en criptografia simétrica.

El cifrador fue desarrollado por dos criptólogos belgas, Joan Daemen y Vincent Rijmen, ambos estudiantes de la Katholieke Universiteit Leuven, y enviado al proceso de selección AES bajo el nombre "Rijndael", una mezcla empaquetada de los nombres de los inventores.

Rijndael fue un refinamiento de un diseño anterior de Daemen y Rijmen, Square; Square fue a su vez un desarrollo de Shark.

Al contrario que su predecesor DES, Rijndael es una red se sustitución-

10 NIST “Advanced Encryption Standard (AES)” Disponible en: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Consultado: 18 de Febrero de 2008, 7:48 pm.
11 Es una agencia de la Administración de la Tecnología.
12 Federal Information Processing Standards, son estandáres anunciados públicamente desarrollados por el gobierno de los Estados Unidos para la utilización por parte de todas las agencias del gobierno no militares y por los contratistas del gobierno.

permutación, no una red de Feistel[13]. AES es rápido tanto en software como en hardware, es relativamente fácil de implementar, y requiere poca memoria. Como nuevo estándar de cifrado, se está utilizando actualmente a gran escala.

Estrictamente hablando, AES no es precisamente Rijndael (aunque en la práctica se los llama de manera indistinta) ya que Rijndael permite un mayor rango de tamaño de bloques y longitud de claves; AES tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 ó 256 bits, mientras que Rijndael puede ser especificado por una clave que sea múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits.

La clave se expande usando el esquema de claves de Rijndael.
La mayoría de los cálculos del algoritmo AES se hacen en un campo finito determinado.

AES opera en una matriz de 4×4 de bytes, llamada state[14]. Para el cifrado, cada ronda de la aplicación del algoritmo AES (excepto la última) consiste en cuatro pasos:

SubBytes — en este paso se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla de búsqueda.
ShiftRows — en este paso se realiza un transposición donde cada fila del state es rotado de manera cíclica un número determinado de veces.

MixColumns — operación de mezclado que opera en las columnas del «state», combinando los cuatro bytes en cada columna usando una transformación lineal.
AddRoundKey — cada byte del «state» es combinado con la clave «round»; cada

13 Es el nombre común que se le dió al Cifrado de Feistel, este es usado por un gran número de algorítmos, del cual se destaca el Data Encryption Algorithm (DES).
14 Algunas versiones de Rijndael con un tamaño de bloque mayor tienen columnas adicionales en el state.

clave «round» se deriva de la clave de cifrado usando unaiteración de la clave[15]. La ronda final reemplaza la fase MixColumns por otra instancia de AddRoundKey.

En la etapa SubBytes, cada byte en el arreglo es actualizado usando la caja-S de Rijndael de 8 bits. Esta operación provee la no linealidad en el cifrado. La caja-S utilizada proviene de la función inversa alrededor del GF(28), conocido por tener grandes propiedades de no linealidad. Para evitar ataques basados en simples propiedades algebráicas, la caja-S se construye por la combinación de la función inversa con una transformación afin inversible. La caja-S también se elige para evitar puntos estables (y es por lo tanto un derangement), y también cualesquiera puntos estables opuestos.

El paso ShiftRows opera en las filas del state; rota de manera cíclica los bytes en cada fila por un determinado offset[16]. En AES, la primera fila queda en la misma posición. Cada byte de la segunda fila es rotado una posición a la izquierda. De manera similar, la tercera y cuarta filas son rotadas por los offsets de dos y tres respectivamente. De esta manera, cada columna del state resultante del paso ShiftRows está compuesta por bytes de cada columna del state inicial. (variantes de Rijndael con mayor tamaño de bloque tienen offsets distintos).

En el paso MixColumns, los cuatro bytes de cada columna del state se combinan usando una transformación lineal inversible. La función MixColumns toma cuatro bytes como entrada y devuelve cuatro bytes, donde cada byte de entrada influye todas las salidas de cuatro bytes. Junto con ShiftRows, MixColumns implica difusión en el cifrado. Cada columna se trata como un polinomio GF(28) y luego se multiplica el módulo x4 + 1 con un polinomio fijo c(x). El paso MixColumns puede

15 NIST “Advanced Encryption Standard (AES)” Disponible en: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Consultado: 18 de Febrero de 2008, 7:48 pm.

16 En informática, un offset es un entero que indica la distancia desde un punto a otro punto en diferentes estrucutras de datos.

verse como una multiplicación matricial en el campo finito de Rijndael[17].

En el paso AddRoundKey, la subclave se combina con el state. En cada ronda se obtiene una subclave de la clave principal, usando la iteración de la clave; cada subclave es del mismo tamaño del state. La subclave se agrega combinando cada byte del state con el correspondiente byte de la subclave usando XOR.

En sistemas de 32 bits o de mayor tamaño de palabra, es posible acelerar la ejecución de este algoritmo mediante la conversión de las transformaciones SubBytes, ShiftRows y MixColumn en tablas. Se tienen cuatro tablas de 256 entradas de 32 bits que utilizan un total de 4 kilobytes (4096 bytes) de memoria, un Kb cada tabla. De esta manera, una ronda del algoritmo consiste en 16 búsquedas en una tabla seguida de 16 operaciones XOR de 32 bits en el paso AddRoundKey. Si el tamaño de 4 kilobytes de la tabla es demasiado grande para una plataforma determinada, la operación de búsqueda en la tabla se pude realizar mediante una sola tabla de 256 entradas de 32 bits mediante el uso de rotaciones circulares.

Hasta 2005, no se ha encontrado ningún ataque exitoso contra el AES. La Agencia de Seguridad Nacional de los Estados Unidos (NSA) revisó todos los finalistas candidatos al AES, incluyendo el Rijndael, y declaró que todos ellos eran suficientemente seguros para su empleo en información no clasificada del gobierno de los Estados Unidos. En junio del 2003, el gobierno de los Estados Unidos anunció que el AES podía ser usado para información clasificada:

"*The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems*

17 NIST “Advanced Encryption Standard (AES)” Disponible en: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Consultado: 18 de Febrero de 2008, 7:48 pm.

*and/or information must be reviewed and certified by NSA prior to their acquisition and use.*"[18]

Este hecho marca la primera vez que el público ha tenido acceso a un cifrador aprobado por la NSA para información super secreta (TOP SECRET). Es interesante notar que muchos productos públicos usan llaves de 128 bits por defecto; es posible que la NSA sospeche de una debilidad fundamental en llaves de este tamaño, o simplemente prefieren tener un margen de seguridad para documentos super secretos, (which may require security decades into the future)[19].

El método más común de ataque hacia un cifrador por bloques consiste en intentar varios ataques sobre versiones del cifrador con un número menor de rondas. El AES tiene 10 rondas para llaves de 128 bits, 12 rondas para llaves de 192 bits, y 14 rondas para llaves de 256 bits. Hasta 2005, los mejores ataques conocidos son sobre versiones reducidas a 7 rondas para llaves de 128 bits, 8 rondas para llaves de 192 bits, y 9 rondas para llaves de 256 bits (Ferguson et al, 2000).

Algunos criptógrafos muestran preocupación sobre la seguridad del AES. Ellos sienten que el margen entre el número de rondas especificado en el cifrador y los mejores ataques conocidos es muy pequeño. El riesgo es que se puede encontrar alguna manera de mejorar los ataques y de ser así, el cifrador podría ser roto[20]. De modo que un ataque contra el AES de llave de 128 bits que requiera 'sólo' 2120 operaciones sería considerado como un ataque que "rompe" el AES aún tomando en cuenta que por ahora sería un ataque irrealizable. Hasta el momento, tales preocupaciones pueden ser ignoradas. El ataque de fuerza bruta más largamente

18 NIST “Advanced Encryption Standard (AES)” Disponible en: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Consultado: 18 de Febrero de 2008, 7:48 pm.

19 NIST “Advanced Encryption Standard (AES)” Disponible en: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Consultado: 18 de Febrero de 2008, 7:48 pm.

20 En el contexto criptográfico se considera “roto” un algoritmo si existe algún ataque más rápido que una búsqueda exhaustiva (ataque por fuerza bruta).

publicitado y conocido ha sido contra una clave de 64 bits RC5 por distributed.net.

Otra preocupación es la estructura matemática de AES. A diferencia de la mayoría de cifradores de bloques, AES tiene una descripción matemática muy ordenada. Esto no ha llevado todavía a ningún ataque, pero algunos investigadores están preocupados que futuros ataques quizá encuentren una manera de explotar esta estructura.

En 2002, un ataque teórico, denominado "ataque XSL", fue anunciado por Nicolas Courtois y Josef Pieprzyk, mostrando una potencial debilidad en el algoritmo AES. Varios expertos criptográficos han encontrado problemas en las matemáticas que hay por debajo del ataque propuesto, sugiriendo que los autores quizá hayan cometido un error en sus estimaciones. Si esta línea de ataque puede ser tomada contra AES, es una cuestión todavía abierta. Hasta el momento, el ataque XSL contra AES parece especulativo; es improbable que nadie pudiera llevar a cabo en la práctica este ataque.

Los ataques de canal auxiliar[21] no atacan al cifrador por debajo, sino las implementaciones del cifrador en sistemas que pierden datos involuntariamente. En abril de 2005, D.J. Bernstein anunció a Ataque temporizado de cache que solía romper un servidor a medida que usaba el cifrado AES para OpenSSL[22]. Este servidor fue diseñado para dar la mayor cantidad de información acerca del tiempo como fuera posible, y el ataque requería cerca de 200 millones de archivos de texto plano. Se dice que el ataque no es práctico en implementaciones del mundo real; Bruce Schneier llamó a esta investigación un "bonito ataque temporizado".

21 Este ataque se le realiza al software o hardware que ha implementado el algorítmo de cifrado, y en su implementación lleva errores de programación, lo cual trae la perdida de datos involuntaria.

22 Es un proyecto de software desarrollado por los miembros de la comunidad Open Source. Consiste en un robusto paquete de herramientas de administración y libreías relacionadas con la criptografía.

En Octubre de 2005, Adi Shamir y otros dos investigadores presentaron un artículo demostrando varios ataques temporizados de cache contra AES. Uno de los ataques obtuvo una clave de AES entera luego de tan sólo 800 escrituras, en 65 milisegundos. Este ataque requiere que el atacante pueda ejecutar programas en el mismo sistema[23] que realiza el cifrado de AES[24].

**2.1.2 Message-Digest Algorithm 5, MD5[25]** En criptografía, MD5 (acrónimo de Message-Digest Algorithm 5, Algorítmo de Resumen del Mensaje 5) es un algorítmo de reducción criptográfico de 128 bits ampliamente usado. El código MD5 fue diseñado por Ronald Rivest en 1991. Durante el año 1996 fúe anunciada la posibilidad de realizar colisiones de Hash, obtenidas por ataques criptoanalíticos, lo que hará que en un futuro cercano se cambie de este sistema a otro más seguro.

La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal. El siguiente código de 28 bytes ASCII será tratado con MD5 y se verá su correspondiente hash[26] de salida:

MD5("Esto si es una prueba de MD5")
dará como resultado de Hash
e07186fbff6107d0274af02b8b930b65

Un simple cambio en el mensaje nos da un cambio total en la codificación hash, en este caso cambiamos dos letras, el "si" por un "no".

23 Acceso directo a la computadora o al sistema al cual esta implentando el algorítmo.
24 NIST “Advanced Encryption Standard (AES)” Disponible en: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Consultado: 18 de Febrero de 2008, 7:48 pm.
25 MIT Laboratory for Computer Science and RSA Data Security “The MD5 Message-Digest Algorithm”. Disponible en: http://www.ietf.org/rfc/rfc1321.txt. Consultado: 19 de Febrero de 2008, 12:27 am.
26 Cadena de carácteres final como resultado de la aplicación del algorítmo.

MD5("Esto no es una prueba de MD5")
Y es notable el cambio del Hash
dd21d99a468f3bb52a136ef5beef5034
Otro ejemplo sería la codificación de un campo vacío:
MD5("") = d41d8cd98f00b204e9800998ecf8427e
En este documento "palabra" es una entidad de 32 bits y byte es una entidad de 8 bits. Una secuencia de bits puede ser interpretada de manera natural como una secuencia de bytes, donde cada grupo consecutivo de ocho bits se interpreta como un byte con el bit más significativo al principio. Similarmente, una secuencia de bytes puede ser interpretada como una secuencia de 32 bits (palabra), donde cada grupo consecutivo de cuatro bytes se intepreta como una palabra en la que el byte menos significativo está al principio.

El símbolo "+" significa suma de palabras.

X <<< s se interpreta por una rotación a la izquierda 's' posiciones
not(x) se entiende como el complemento de x

Se empieza suponiendo que se tiene un mensaje de 'b' bits de entrada, y se quiere encontrar su resumen. Aquí 'b' es un valor arbitrario entero no negativo, pero puede ser cero, no tiene por qué ser múltiplo de ocho, y puede ser muy largo. Se puede imaginar los bits del mensaje escritos así:

m0 m1 ... m{b-1}

Los siguientes cinco pasos son efectuados para calcular el resumen del mensaje.
El mensaje será extendido hasta que su longitud en bits sea congruente[27] con 448, módulo 512. Esto es, el mensaje se extenderá hasta que se forme el menor número múltiplo de 512 bits. Esta extensión se realiza siempre, incluso si la longitud del

27 Es un término usado en la teoría de números, para designar que dos números enteros *a* y *b* tienen el mismo resto al dividirlos por un número naturla *m* llamado el módulo.

mensaje es ya congruente con 448, módulo 512.

La extensión se realiza como sigue: un sólo bit "1" se añade al mensaje, y después bits "0" se añaden hasta que la longitud en bits del mensaje extendido se haga congruente con 448, módulo 512. En todos los mensajes se añade al menos un bit y como máximo 512[28].

Una representación de 64 bits de 'b' (la longitud del mensaje antes de añadir los bits) se concatena[29] al resultado del paso anterior. En el supuesto no deseado de que 'b' sea mayor que 2^64, entonces sólo los 64 bits de menor peso de 'b' se usarán. En este punto el mensaje resultante (después de rellenar con los bits y con 'b') se tiene una longitud que es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es múltiplo de 16 palabras (32 bits por palabra). Con M[0 ... N-1] denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Aquí cada una de las letras A, B, C, D representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bits de menor peso primero:

palabra A: 01 23 45 67
palabra B: 89 ab cd ef
palabra C: fe dc ba 98
palabra D: 76 54 32 10

Primero se definen cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits.

28 MIT Laboratory for Computer Science and RSA Data Security “The MD5 Message-Digest Algorithm”. Disponible en: http://www.ietf.org/rfc/rfc1321.txt. Consultado: 19 de Febrero de 2008, 12:27 am.

29 La acción en la cual dos cadenas de texto se juntan para conformar así una única cadena de texto.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Los operadores $\oplus, \wedge, \vee, \neg$ son las funciones XOR, AND, OR y NOT respectivamente[30].

En cada posición de cada bit F actúa como un condicional: si X, entonces Y sino Z. La función F podría haber sido definida usando + en lugar de v ya que XY y not(x) Z nunca tendrán unos ('1') en la misma posición de bit. Es interesante resaltar que si los bits de X, Y y Z son independientes y no sesgados, cada uno de los bits de F(X,Y,Z) será independiente y no sesgado.

Las funciones G, H e I son similares a la función F, ya que actúan "bit a bit en paralelo" para producir sus salidas de los bits de X, Y y Z, en la medida que si cada bit correspondiente de X, Y y Z son independientes y no sesgados, entonces cada bit de G(X,Y,Z), H(X,Y,Z) e I(X,Y,Z) serán independientes y no sesgados. Nótese que la función H es la comparación bit a bit "xor" o función "paridad" de sus entradas.

Este paso usa una tabla de 64 elementos T[1 ... 64] construida con la función Seno[31]. Denotaremos por T[i] el elemento i-ésimo de esta tabla, que será igual a la parte entera del valor absoluto del seno de 'i' 4294967296 veces, donde 'i' está en radianes. Código del MD5:

/* Procesar cada bloque de 16 palabras. */

para i = 0 hasta N/16-1 hacer

30 MIT Laboratory for Computer Science and RSA Data Security “The MD5 Message-Digest Algorithm”. Disponible en: http://www.ietf.org/rfc/rfc1321.txt. Consultado: 19 de Febrero de 2008, 12:27 am.

31 Función trigonométrica.

```
  /* Copiar el bloque 'i' en X. */

  para j = 0 hasta 15 hacer
    hacer X[j] de M[i*16+j].
  fin para /* del bucle 'j' */[32]
/* Guardar A como AA, B como BB, C como CC, y D como DD. */
AA = A
BB = B
CC = C
DD = D

/* Ronda 1. */
/* [abcd k s i] denotarán la operación
     a = b + ((a + F(b, c, d) + X[k] + T[i]) <<< s). */
/* Hacer las siguientes 16 operaciones. */
[ABCD  0  7  1] [DABC  1 12  2] [CDAB  2 17  3] [BCDA  3 22  4]
[ABCD  4  7  5] [DABC  5 12  6] [CDAB  6 17  7] [BCDA  7 22  8]
[ABCD  8  7  9] [DABC  9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
[ABCD 12  7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

/* Ronda 2. */
/* [abcd k s i] denotarán la operación
    a = b + ((a + G(b, c, d) + X[k] + T[i]) <<< s). */
/* Hacer las siguientes 16 operaciones. */
[ABCD  1  5 17] [DABC  6  9 18] [CDAB 11 14 19] [BCDA  0 20 20]
[ABCD  5  5 21] [DABC 10  9 22] [CDAB 15 14 23] [BCDA  4 20 24]
[ABCD  9  5 25] [DABC 14  9 26] [CDAB  3 14 27] [BCDA  8 20 28]
[ABCD 13  5 29] [DABC  2  9 30] [CDAB  7 14 31] [BCDA 12 20 32]
```

32 MIT Laboratory for Computer Science and RSA Data Security “The MD5 Message-Digest Algorithm”. Disponible en: http://www.ietf.org/rfc/rfc1321.txt. Consultado: 19 de Febrero de 2008, 12:27 am.

/* Ronda 3. */

/* [abcd k s t] denotarán la operación

a = b + ((a + H(b, c, d) + X[k] + T[i]) <<< s). */[33]

/* Hacer las siguientes 16 operaciones. */

[ABCD  5  4 33] [DABC  8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]

[ABCD  1  4 37] [DABC  4 11 38] [CDAB  7 16 39] [BCDA 10 23 40]

[ABCD 13  4 41] [DABC  0 11 42] [CDAB  3 16 43] [BCDA  6 23 44]

[ABCD  9  4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA  2 23 48]

/* Ronda 4. */

/* [abcd k s t] denotarán la operación

a = b + ((a + I(b, c, d) + X[k] + T[i]) <<< s). */

/* Hacer las siguientes 16 operaciones. */

[ABCD  0  6 49] [DABC  7 10 50] [CDAB 14 15 51] [BCDA  5 21 52]

[ABCD 12  6 53] [DABC  3 10 54] [CDAB 10 15 55] [BCDA  1 21 56]

[ABCD  8  6 57] [DABC 15 10 58] [CDAB  6 15 59] [BCDA 13 21 60]

[ABCD  4  6 61] [DABC 11 10 62] [CDAB  2 15 63] [BCDA  9 21 64]

/* Ahora realizar las siguientes sumas. (Este es el incremento de cada uno de los cuatro registros por el valor que tenían antes de que este bloque fuera inicializado.) */

A = A + AA

B = B + BB

C = C + CC

D = D + DD

33 MIT Laboratory for Computer Science and RSA Data Security “The MD5 Message-Digest Algorithm”.  Disponible en: http://www.ietf.org/rfc/rfc1321.txt. Consultado: 19 de Febrero de 2008, 12:27 am.

fin para /* del bucle en 'i' */[34]

*** Seguridad** MD5 ha sido ampliamente usado, y originalmente se pensaba que era criptográficamente seguro. No obstante, ciertas investigaciones han destapado vulnerabilidades que hacen cuestionable un futuro uso del MD5. El 17 de Agosto del año 2004 Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu anunciaron que habían descubierto colisiones de hash para MD5. Su ataque sólo llevó una hora de cálculo con un clúster IBM P690.

Aunque el ataque de Wang era analítico, el tamaño del hash (128 bits) es suficientemente pequeño para poder contemplar ataques de 'fuerza bruta' tipo 'cumpleaños'[35]. MD5CRK era un proyecto distribuido que comenzó en Marzo del 2004 con el propósito de demostrar que MD5 es inseguro encontrando una colisión usando un ataque de 'fuerza bruta', aunque acabó poco después del aviso de Wang.

Debido al descubrimiento de un método fácil para generar colisiones de hash, muchos investigadores recomiendan otros algoritmos tales como SHA-1 o RIPEMD-160, para ser usados en lugar de MD5[36].

*** Aplicaciones** Los resúmenes MD5 se utilizan extensamente en el mundo del software para proporcionar la certeza de que un archivo descargado de internet no se ha alterado. Comparando una suma MD5 publicada con la suma de

34 MIT Laboratory for Computer Science and RSA Data Security "The MD5 Message-Digest Algorithm". Disponible en: http://www.ietf.org/rfc/rfc1321.txt. Consultado: 19 de Febrero de 2008, 12:27 am.

35 Es un tipo de ataque criptográfico que se basa en la matemática detrás de la paradoja del cumpleaños, haciendo uso de una situación de compromiso espacio-tiempo informática. Concretamente, sí una función matemática produce $N$ resultados diferentes igualmente probables y $N$ es lo suficientemente grande, entonces, después de evaluar la función sobre $1.2\sqrt{N}$ argumentos distintos, se espera encontrar un par de argumentos $X_1$ y $X_2$ diferentes de manera que $f(X_1)=f(X_2)$, hecho conocido como una colisión de Hash.

36 Wikipedia "MD5". Disponible en: http://es.wikipedia.org/wiki/MD5. Consultado: 19 de Febrero de 2008, 1:49 am.

comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores. Esto protege al usuario contra los 'Caballos de Troya' o 'Troyanos' y virus que algún otro usuario malicioso pudiera incluir en el software. La comprobación de un archivo descargado contra su suma MD5 no detecta solamente los archivos alterados de una manera maliciosa, también reconoce una descarga corrupta o incompleta.

Para comprobar la integridad de un archivo descargado de Internet se puede utilizar una herramienta MD5 para comparar la suma MD5 de dicho archivo con un archivo MD5SUM con el resumen MD5 del primer archivo. En los sistemas UNIX[37], el comando de md5sum es un ejemplo de tal herramienta. Además, también está implementado en el lenguaje de scripting PHP[38] como MD5("") entre otros.
En sistemas UNIX y GNU[39]/Linux[40] se utiliza el algoritmo MD5 para cifrar las claves de los usuarios. En el disco se guarda el resultado del MD5 de la clave que se introduce al dar de alta un usuario, y cuando éste quiere entrar en el sistema se compara la entrada con la que hay guardada en el disco duro, si coinciden, es la misma clave y el usuario será autenticado. He ahí el problema de encontrar y generar colisiones de hash a voluntad.

El MD5 también se puede usar para comprobar que los correos electrónicos no han sido alterados usando llaves públicas y privadas[41].

37 Sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.
38 Es un lenguaje de programación interpretado, diseñado originalmente para el desarrollo de páginas Web dinámicas.
39 El proyecto GNU (Gnu is Not Unix) fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre.
40 Es el núcleo de GNU, y fue desarrollado en 1991 por el hacker finlandés Linus Torvalds.
41 Wikipedia “MD5”. Disponible en: http://es.wikipedia.org/wiki/MD5. Consultado: 19 de Febrero de 2008, 1:49 am.

**2.1.3 Secure Hash Algorithm, SHA** La familia SHA (Secure Hash Algorithm, Algoritmo de Hash Seguro[42]) es un sistema de funciones hash criptográficas relacionadas de la Agencia de Seguridad Nacional de los Estados Unidos y publicadas por el National Institute of Standards and Technology (NIST). El primer miembro de la familia fue publicado en 1993 es oficialmente llamado SHA. Sin embargo, hoy día, no oficialmente se le llama SHA-0 para evitar confusiones con sus sucesores. Dos años más tarde el primer sucesor de SHA fue publicado con el nombre de SHA-1. Existen cuatro variantes más que se han publicado desde entonces cuyas diferencias se basan en un diseño algo modificado y rangos de salida incrementados: SHA-224, SHA-256, SHA-384, y SHA-512 (todos ellos son referidos como SHA-2).

En 1998, un ataque a SHA-0 fue encontrado pero no fue reconocido para SHA-1, se desconoce si fue la NSA quien lo descubrió pero aumentó la seguridad del SHA-1.

*** SHA-1** Ha sido examinado muy de cerca por la comunidad criptográfica pública, y no se ha encontrado ningún ataque efectivo. No obstante, en el año 2004, un número de ataques significativos fueron divulgados sobre funciones criptográficas de hash con una estructura similar a SHA-1; esto ha planteado dudas sobre la seguridad a largo plazo de SHA-1.

SHA-0 y SHA-1 producen una salida resumen de 160 bits de un mensaje que puede tener un tamaño máximo de 264 bits, y se basa en principios similares a los usados por el profesor Ronald L. Rivest del MIT[43] en el diseño de los algoritmos de

42 Cisco Systems “US Secure Hash Algorithm 1” Disponible en: http://www.ietf.org/rfc/rfc3174.txt. Consultado: 19 de Febrero de 2008, 5:20 pm.

43 Instituto Tecnológico de Massachusetts (Massachusetts Institute of Technology), es una de las principales instituciones dedicadas a la docencia y a la investigación en los Estados Unidos.

resumen del mensaje MD4 y MD5[44].

La codificación hash vacía para SHA-1 corresponde a:

SHA1("") = da39a3ee5e6b4b0d3255bfef95601890afd80709

* **Ataques contra SHA-1** La resistencia del algoritmo SHA-1 se ha visto comprometido a lo largo del año 2005. Después de que MD5, entre otros, quedara seriamente comprometido en el 2004 por parte del equipo de investigadores chinos, el tiempo de vida de SHA-1 quedó visto para sentencia[45].

El mismo equipo de investigadores chinos, compuesto por Xiaoyun Wang, Yiqun Lisa Yin y Hongbo Yu[46] (principalmente de la Shandong University en China), ha demostrado que son capaces de romper el SHA-1 en al menos 269 operaciones, unas 2000 veces más rápido que un ataque de fuerza bruta (que requeriría 280 operaciones). Los últimos ataques contra SHA-1 han logrado debilitarlo hasta 263. Según el NIST:

"*Este ataque es de particular importancia para las aplicaciones que usan firmas digitales tales como marcas de tiempo y notarías. Sin embargo, muchas aplicaciones que usan firmas digitales incluyen información sobre el contexto que hacen este ataque difícil de llevar a cabo en la práctica*"[47].

A pesar de que 263 suponen aún un número alto de operaciones, se encuentra

44 Cisco Systems "US Secure Hash Algorithm 1" Disponible en: http://www.ietf.org/rfc/rfc3174.txt. Consultado: 19 de Febrero de 2008, 5:20 pm.
45 Wikipedia "SHA" Disponible en: http://es.wikipedia.org/wiki/SHA. Consultado: 19 de Febrero de 2008, 6:12 pm.
46 Epoch Times "Chinese Profesor Cracks Fifth Data Security Algotithm". Disponible en: http://en.epochtimes.com/news/7-1-11/50336.html. Consultado: 19 de Febrero de 2008, 6:55 pm.
47 NIST "Secure Hash Algothim". Disponible en: http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html. Consultado: 19 de Febrero de 2008, 6:44 pm.

dentro de los límites de las capacidades actuales de cálculos, y es previsible que con el paso del tiempo romper esta función sea trivial, al aumentar las capacidades de cálculo y al ser más serios los ataques contra SHA-1[48].

La importancia de la rotura de una función hash se debe comprender de la siguiente manera: Un hash permite crear una huella digital, en teoría única, de un archivo. Si un hash fuese roto podría haber otro documento con la misma huella. La similitud podría ser que hubiese personas que compartiesen las mismas huellas digitales, o peor aún, el mismo ADN: No habría manera de poder diferenciarlos usando estos medios de discriminación.

A pesar de que el NIST contempla funciones de SHA de mayor tamaño (por ejemplo, el SHA-512, de 512 bits de longitud), expertos de la talla de Bruce Schneier abogan por, sin llamar a alarmismos, buscar una nueva función hash estandarizada que permita sustituir a SHA-1. Los nombres que se mencionan al respecto son Tiger, de los creadores de Serpent, y WHIRLPOOL, de los creadores de AES[49].

48 Cisco Systems “US Secure Hash Algorithm 1” Disponible en: http://www.ietf.org/rfc/rfc3174.txt. Consultado: 19 de Febrero de 2008, 5:20 pm.
49 Bruce Schneier “SHA-1 Broken”. Disponible en: http://www.schneier.com/blog/archives/2005/02/sha1_broken.html. Consultado: 19 de Febrero de 2008, 7:17 pm.

# 3. METODOLOGÍA

## 3.1 ENFOQUE DE LA INVESTIGACIÓN

### 3.1.1 Línea de Investigación

Tecnologías actuales y sociedad.

*** Sublínea de Facultad**

Sistemas de Información y comunicación.

*** Campo temático**

Algorítmia.

## 3.2 TÉCNICAS DE RECOLECCIÓN

El análisis de la bibliografía es supremamente importante, aunque existen libros físicos que claramente han aportado al desarrollo y mejoramiento de los algorítmos de cifrado para la aplicación y adaptación en la seguridad informática, la bibliografía electrónica simpre esta actualizada, y ello permite un mejor estudio y actualización permanente.

El estudio de los actuales algorítmos de cifrado es sumamente importante para el entendimiento de su funcionamiento y busqueda de posibles vulnerabilidades aun no publicadas, ya que es tambien es imprecindible el estudio de las vulnerabilidades públicadas.

## 3.5 HIPÓTESIS

Desarrollar un nuevo algorítmo de cifrado simétrico de resumen, y expandir el tamaño de las llaves y del Hash, generará una mayor seguridad en el cifrado de datos y de información.

## 3.6 VARIABLES

### 3.6.1 Independientes

- Tamaño del Hash.
- Conversión del texto plano a Binario.
- Completar la cadena binaria a una longitud de 128 Bits.

### 3.6.2 Dependientes

- La cantidad de Bits de complemento.
- El Hash de resultado.

# 4. DESARROLLO INGENIERÍL

## 4.1 ANÁLISIS FUNCIONAL DE ALGORÍTMOS Y VULNERABILIDADES

Los algorítmos de cifrado simétricos de resumen o algorítmos de Hash más utilizados y aprobados por la NIST y la NSA son el MD5 y SHA-1, por esa razón son los que se estudiarán y analizarán.

### 4.1.1 Message-Digest Algorith, MD5

Desarrollado por Ron Rivest, ha sido hasta los últimos años el algorítmo Hash más usado, procesa mensajes de una longitud arbitraria en bloques de 512 bits generando un resumen de 128 bits. Gracias a la gran capacidad de procesamiento actual esos 128 bits son insuficientes para asegurar la integridad del algorítmo, además de que una serie de ataques criptoanalíticos han puesto en manifiesto algunas vulnerabilidades del algorítmo, de las cuales se hablará más adelante.

### 4.1.2 Secure Hash Algorithm 1, SHA-1

Toma como entrada un mensaje de longitud máxima de $2^{64}$ bits y produce como salida un resumen de 160 bits.

En realidad, lo seguros o inseguros que estos algorítmos sean no depende de los conocimientos telemáticos o informáticos que se tengan, sino de sus conocimientos matemáticos. Se demostrará cual es el fallo de seguridad que existe en los algorítmos de cifrado simétrico de resumen y la dificultad computacional que presentan.

Aproximadamente desde el año 2004, cuando saltaron las primeras noticias sobre la ruptura del MD5, la seguridad que ofrecen los algorítmos de cifrado simétricos de resumen se ha puesto en duda.

Matemáticamente el funcionamiento de un algorítmo de cifrado simétrico de resumen esta dada por una función. Suponiendo que se tiene un mensaje *m*, al cual se le aplica el algorítmo que será llamado *a*, al resultado de esta operación se le conocerá como *h*, es decir, *h* es el Hash de *m* y se representa matemáticamente así:

$a(m)=h$

Esta función debe ser sencilla de realizar para un computador, pero computacionalmente imposible realizar la operación inversa, por lo menos para usuarios normales. Una característica de la función: el tamaño del mensaje de entrada no es de longitud fija, esto tiene como consecuencia que dos mensajes de entrada, *m* produzcan el mismo mensaje de salida *h*. Es decir, se puede encontrar un mensaje *n* tal que:

$a(n)=h$

Pero encontrar ese mensaje debe ser, al igual que la particularidad antes mencionada, muy complejo desde el punto de vista computacional. Para los algorítmos de cifrado simétrico de resumen esto es lo que se conoce como una colisión de hash: Que dos mensajes diferentes de entrada produzcan el mismo mensaje de salida.

Hasta este punto se han establecido dos posibles vulnerabilidades de los algorítmos de cifrado simétrico de resumen, la primera posibilidad es realizar la operación:

$a^{-1}(h)=m$

A esta operación, de invertir el algorítmo de cifrado simétrico de resumen

comprobando todas las posibilidades para los bits de salida, se le llama ataque de fuerza bruta. Este ataque es el que debe ser computacionalmente impracticable, supondria aplicar la funcion $h$ $2^n$ veces hasta encontrar la coincidencia, donde $n$ es el número de bits de la función.

Y la segunda vulnerabilidad es la posibilidad de hallar colisiones:

$a(m)=h$ y $a(n)=h$, siendo m y n diferentes.

Estas dos posibilidades, dan lugar a 4 tipos de ataques:

- Ataque 1: El atacante es capaz de encontrar dos mensajes al azar que colisionan pero es incapaz de hacerlo de forma sistemática. Si es capaz de dar sólo con dos mensajes que provocan colisión, esta no es razón suficiente para tildar al algorítmo de ineficiente.
- Ataque 2: El atacante es capaz de generar dos mensajes distintos de forma que sus Hash colisionen, pero sin saber con anterioridad cual será el Hash que resultará. Es decir, el atacante no podría generar el Hash que necesite para sus fines.
- Ataque 3: El atacante es capaz de construir un mensaje sin sentido de forma que su Hash colisione con el de un mensaje con sentido. Si éste es el caso, el atacante puede comprometer algorítmos de cifrado asimétrico con firma digital, haciendo que se firmen mensajes sin sentido y que el destinatario los acepte como fiables.
- Ataque 4: El atacante es capaz de crear un mensaje con sentido y falso cuyo Hash colisiona con el de un mensaje verdadero, en éste caso el atacante puede actuar con total impunidad.

Los problemas de los algorítmos de cifrado simétrico de resumen, se establecen bajo las siguientes preguntas; ¿Qué tan difícil es encontrar una solución?, ¿Qué se gana incrementando el número de bits de salida del algorítmo?, ¿Qué ataques

reales son practicables?.

Si se aumenta el número de bits de salida del algorítmo, el ataque de fuerza bruta será más impracticable y también lo será encontrar los mensajes que colisionen, pues teóricamente se cumple que para tener la certeza de encontrar dos mensajes que colisionen no es necesario realizar $2^n$ operaciones, si no sólo $2^{n/2}$, realizando los cálculos matemáticos para una clave de 12 dígitos, escrita con un teclado de 97 caracteres (base 97), se necesitaria realizar:

$97^{12} = 693.842.360.995.438.000.295.041$ de comprobaciones.

Para MD5 la salida es de 128 bits, sería necesario realizar:

$2^{128} = 3.401.823.669 \times 10^{38}$ operaciones.

Tomando la vulnerabilidad de colisiones, y teniendo en cuenta que para hallar una colisión teóricamente se deben realizar la mitad de operaciones, para MD5 se tiene que:

$2^{64} = 18446.744.073.709.551.616$ operaciones.

Y para SHA, cuya salida es de 160 bits la cantidad es de:

$2^{80} = 1.208.925.819.614.629.174.706.176$ operaciones.

Lo realmente excepcional de estos algorítmos, es que para encontrar una colisión de Hash, con 1.000.000 de computadores, capaces de procesar en 1 μs (un microsegundo) cada operación tardarían más de 38.000 años en las $2^{80}$ operaciones.

Para romper SHA-0 completo se requirió un supercomputador BULL de 256 procesadores durante 9 años de proceso. Otro grupo de Investigadores Feig, Wang, Yu y Lei reportaron haberlo conseguido con una complejidad aproximadamente 2.000 veces menor. Esta reducción equivale a una necesidad de calculo menor a 24 horas, pero los mismos investigadores han reportado necesitar sólo un día con un IBM P690 en clúster, para romper el MD5.

## 4.2 DISEÑO DEL ALGORÍTMO

### 4.2.1 Análisis de requerimientos

Para que el algorítmo funcione, requiere de dos datos, el texto a ser cifrado, y la llave con la cual se cifrará el texto. El texto se conoce como “texto plano”, y puede ser una entrada de cualquier tamaño de bits, la clave debe tener un máximo de 1024 bits, entre más extensa la clave mayor será la complejidad del cifrado.

### 4.2.2 Complejidad Algorítmica

La complejidad Algorítmica representa la cantidad de recursos temporales que necesita un algorítmo para resolver un problema y por tanto permite determinar la eficiencia de dicho algorítmo. Los criterios que se emplean para evaluar la complejidad algorítmica no proporcionan medidas absolutas sino medidas relativas al tamaño del problema.

El algorítmo es de Orden de Complejidad Constante y Orden de Complejidad lineal, ya que sus funciones principales son los búcles y ejecuciones únicas. Y depende del tamaño de entrada el tiempo de duración, como se muestra en la tabla 1. tiempos de ejecución del algorítmo.

Tabla 1. Tiempos de ejecución del algorítmo

| Prueba | Tamaño de Entrada | Tiempo de Ejecución (seg) |
|---|---|---|
| 1 | 1 | 0.00624895095825 |
| 2 | 2 | 0.0095641555792 |
| 3 | 3 | 0.047181200981 |
| 4 | 4 | 0.047184920311 |
| 5 | 5 | 0.0508989095688 |
| 6 | 10 | 0.663584947586 |
| 7 | 100 | 0.70773005486 |
| 8 | 1000 | 0.925750017166 |
| 9 | 5000 | 0.965337963104 |
| 10 | 10000 | 0.971342096329 |

### 4.2.3 Diagráma de flujo de datos

El diagráma de flujo de datos se presenta a continuación. La figura 1. Diagráma general muestra todo el proceso principal, haciendo llamadas a subprocesos.

Figura 1. Diagráma general

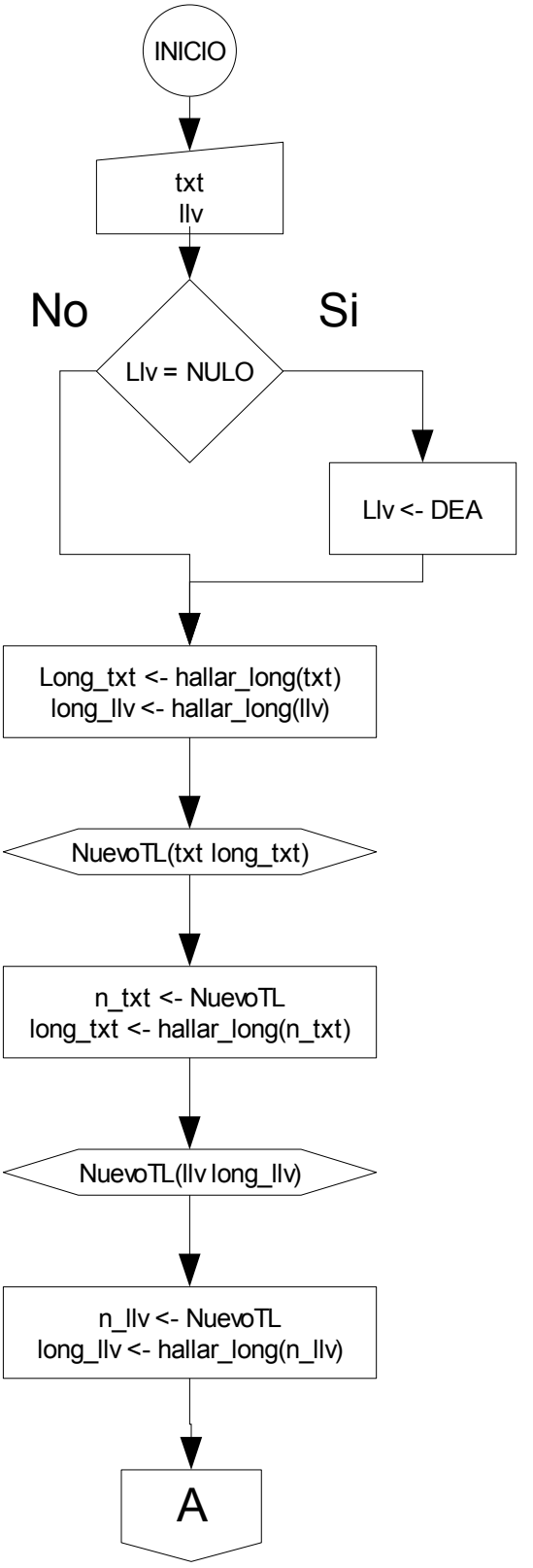


Figura 1. Diagráma general (continuación)

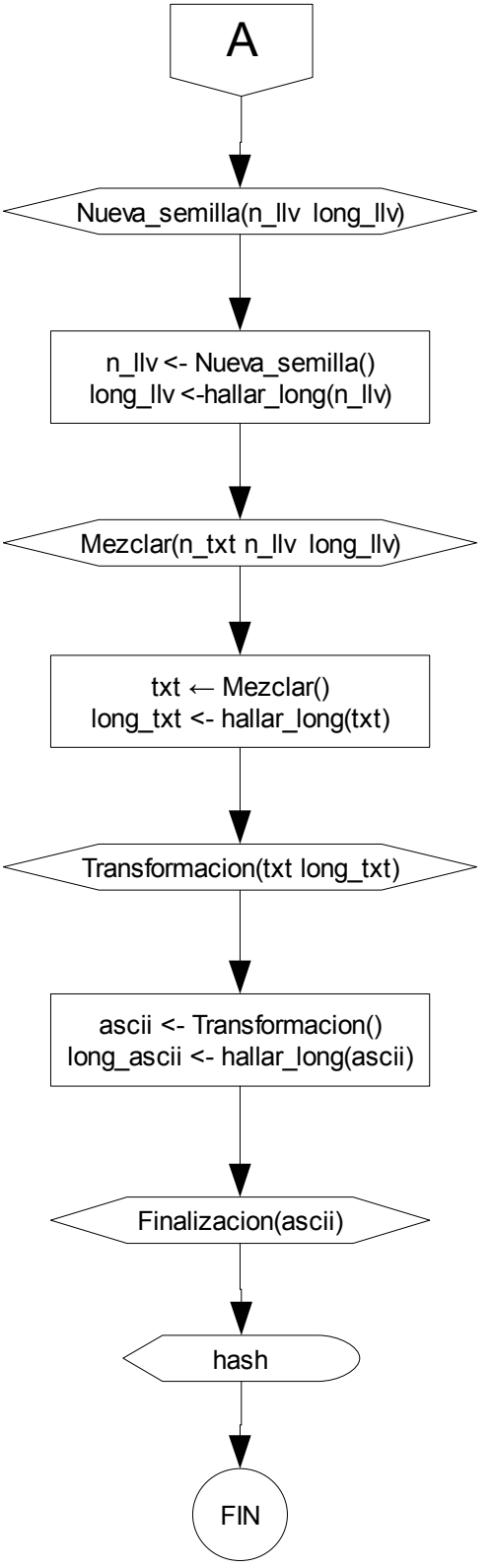


Figura 2. Diagráma del proceso NuevoTL

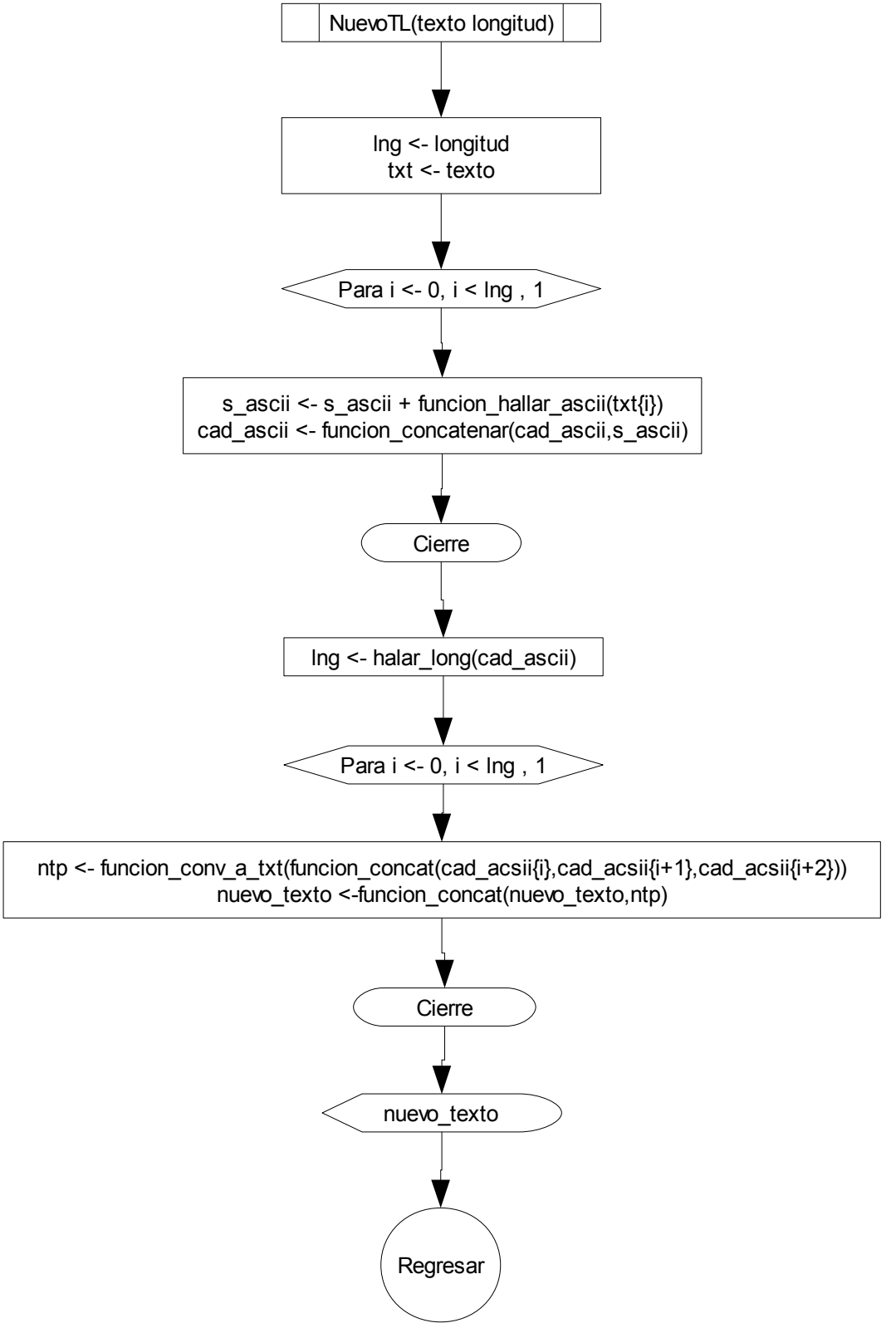


Figura 3. Diagráma del proceso Nueva_semilla

Figura 4. Diagráma del proceso Mezclar

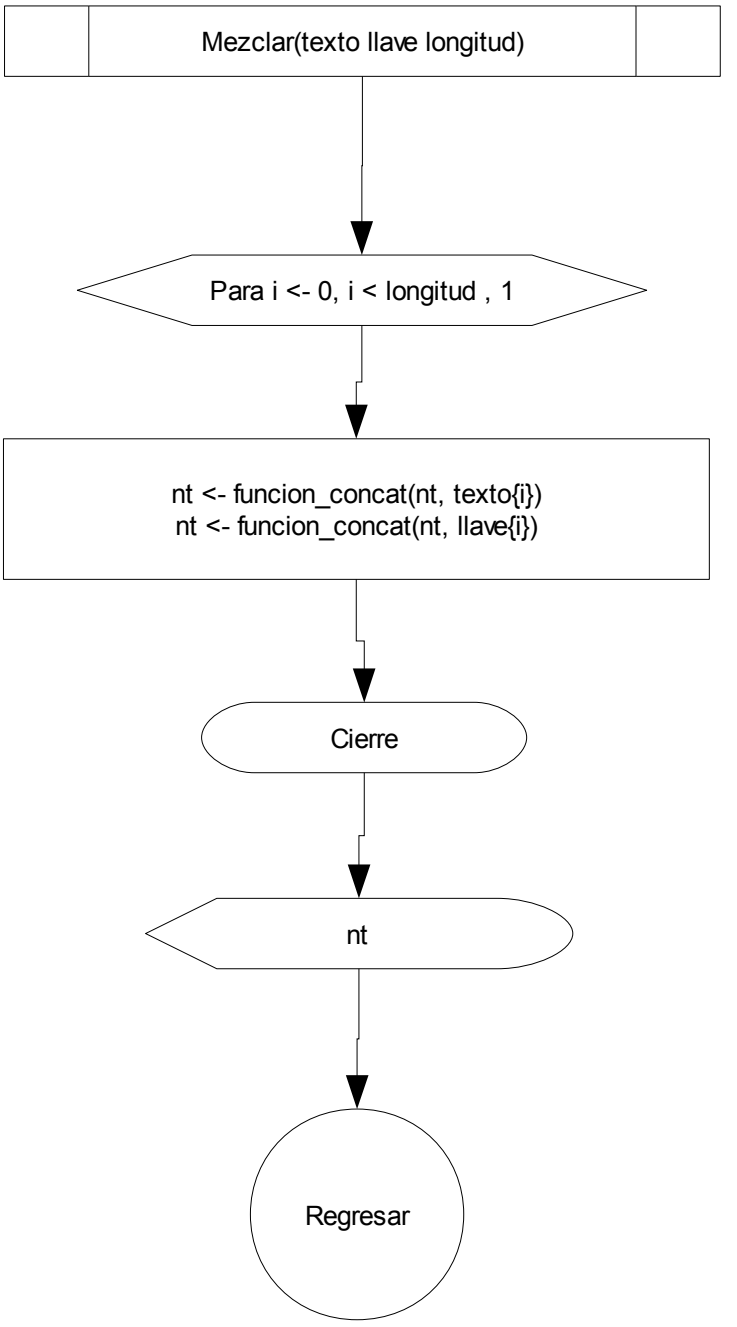


Figura 5. Diagráma del proceso Transformación

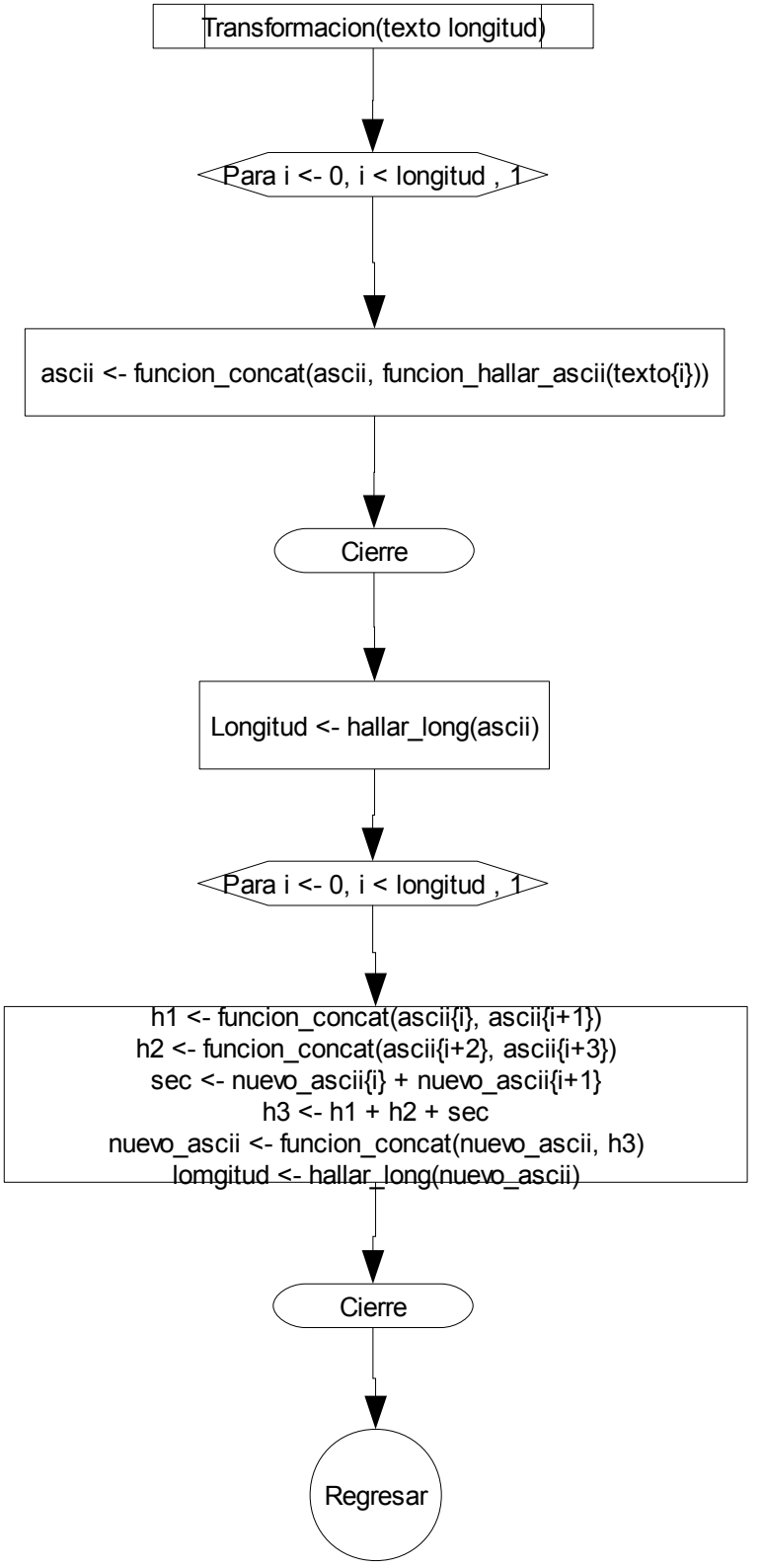


Figura 6. Diagráma del proceso Finalización

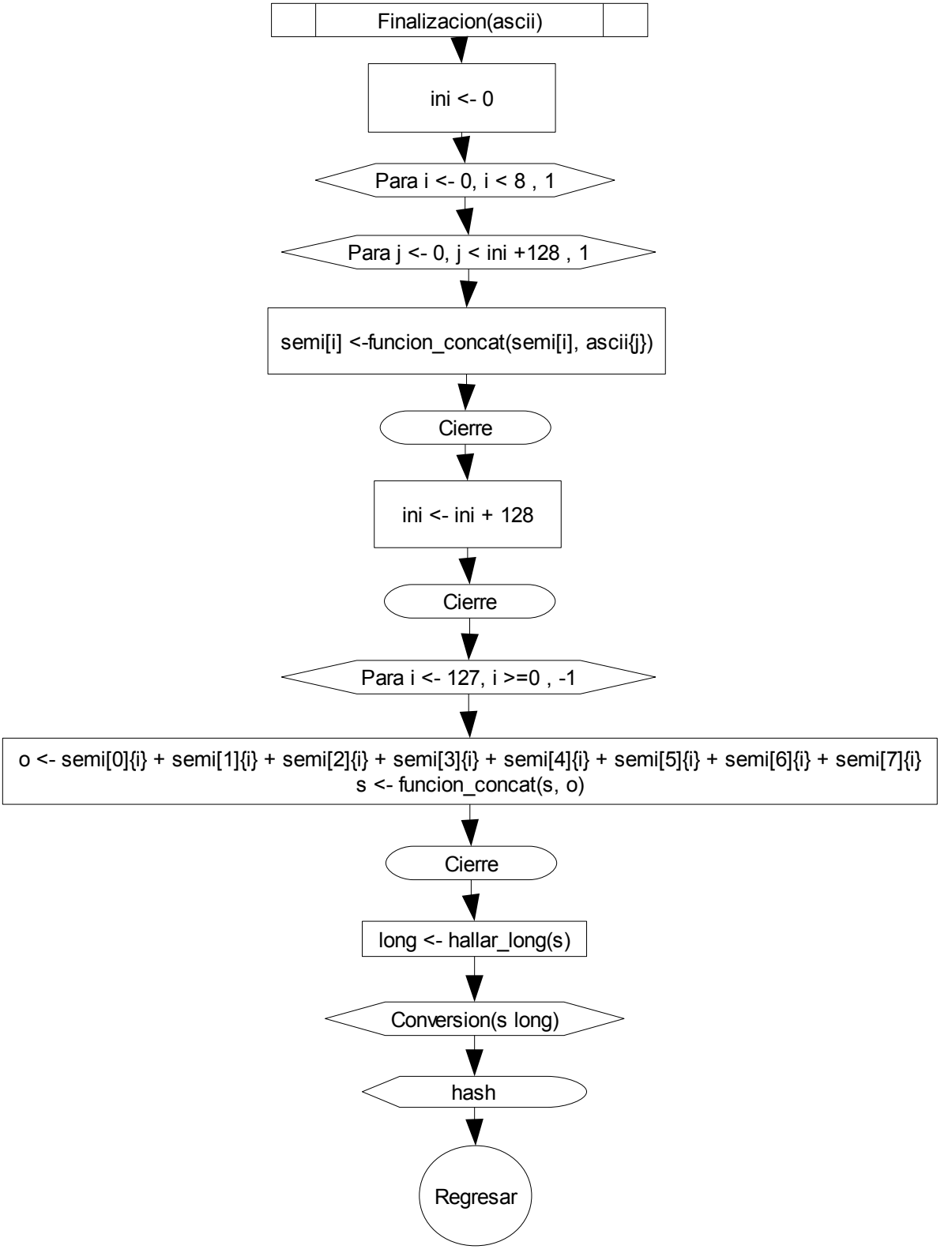
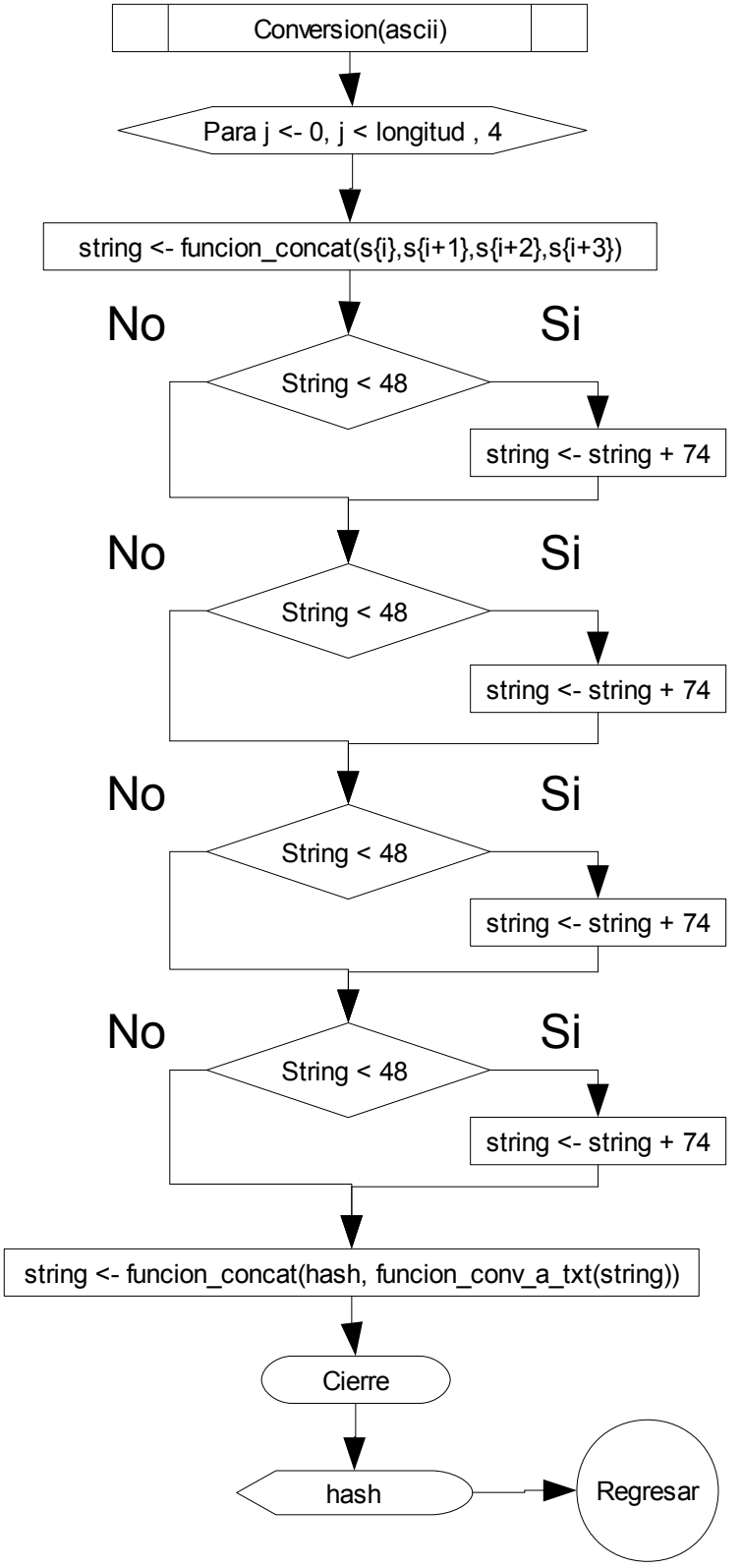


Figura 7. Diagráma del proceso Conversión



### 4.2.4 Pseudocódigo

```
ALGORITMO: SBDA

Leer entrada
Leer clave

longitud_seed <- 64
iniciador <- 0
ini <- 0

texto <- entrada
llave <- clave

si llave = NULO entonces
        llave <- "DEA"
fin si

longitud_texto <- funcion_hallar_longitud(texto)
longitud_llave <- funcion_hallar_longitud(llave)

para i<-iniciador mientras i < longitud_texto hacer
        suma_ascii_texto <- suma_ascii_texto + funcion_hallar_ascii(texto{i})
        cadena_acsii_texto <-
funcion_concatenar(cadena_ascii_texto,suma_ascii_texto)
        i <- i + 1
fin para

longitud_cadena_acsii_texto <- funcion_hallar_longitud(cadena_acsii_texto)
para i<-iniciador mientras i < longitud_cadena_ascii_texto hacer
```

```
        nuevo_texto_temp <-
funcion_convertir_a_texto(funcion_concatenar(cadena_acsii_texto{i},cadena_acsii_t
exto{i+1},cadena_acsii_texto{i+2}))
        nuevo_texto <- funcion_concatenar(nuevo_texto,nuevo_texto_temp)
        i <- i + 2
fin para

para i<-iniciador mientras i < longitud_llave hacer
        suma_ascii_llave <- suma_ascii_llave + funcion_hallar_ascii(llave{i})
        cadena_acsii_llave <-
funcion_concatenar(cadena_ascii_llave,suma_ascii_llave)
        i <- i + 1
fin para

longitud_cadena_acsii_llave <- funcion_hallar_longitud(cadena_acsii_llave)

para i<-iniciador mientras i < longitud_cadena_ascii_llave hacer
        nueva_llave_temp <-
funcion_convertir_a_texto(funcion_concatenar(cadena_acsii_llave{i},cadena_acsii_ll
ave{i+1},cadena_acsii_llave{i+2}))
        nueva_llave <- funcion_concatenar(nueva_llave,nueva_llave_temp)
        i <- i + 2
fin para

para i<-iniciador mientras i < longitud_seed hacer
        para j<-iniciador mientras j < longitud_llave hacer
                si i = longitud_seed entonces
                        terminar_ciclos
                sino
                        nueva_seed <- funcion_concatenar(nueva_seed,
```

```
nueva_llave{j})
                fin si
        i <- i + 1
        fin para
        i <- i - 1
fin para

para i<-iniciador mientras i < longitud_seed hacer
        nt <- funcion_concatenar(nt, nuevo_texto{i})
        nt <- funcion_concatenar(nt, nueva_seed{i})
        i <- i + 1
fin para

longitud_nt <- funcion_hallar_longitu(nt)
para i<-iniciador mientras i < longitud_nt hacer
        ascii <- funcion_concatenar(ascii, funcion_hallar_ascii(nt{i}))
        i <- i + 1
fin para

longitud_ascii <- funcion_hallar_longitud(ascii)

para i<-iniciador mientras i < longitud_ascii hacer
        h1 <- funcion_concatenar(ascii{i}, ascii{i+1})
        h2 <- funcion_concatenar(ascii{i+2}, ascii{i+3})
        sec <- nuevo_ascii{i} + nuevo_ascii{i+1}
        h3 <- h1 + h2 + h3
        nuevo_ascii <- funcion_concatenar(nuevo_ascii, h3)
        i <- i + 1
fin para
```

```
longitud_nuevo_ascii <- funcion_hallar_longitud(nuevo_ascii)

si longitud_nuevo_ascii < 1024 entonces
        para i<-longitud_nuevo_ascii+1 mientras i < 1024 hacer
                nuevo_ascii <- funcion_concatenar(nuevo_ascii, "3")
                i <- i + 1
        fin para
fin si

para i<-iniciador mientras i < 8 haga
        para j<-ini mientras j < ini+128 haga
                semi[i] <- funcion_concatenar(semi[i], nuevo_ascii{j})
                j <- j + 1
        fin para
        ini <- ini + 128
fin para

para i<-127 mientas i >= 0 hacer
        o <- semi[0]{i} + semi[1]{i} + semi[2]{i} + semi[3]{i} + semi[4]{i} + semi[5]{i} +
semi[6]{i} + semi[7]{i}
        s <- funcion_concatenar(s, o)
        i <- i - 1
fin para

longitud_s <- funcion_hallar_longitud(s)

para i<-iniciador mientras i < longitud_s hacer
        string <- funcion_concatenar(s{i},s{i+1},s{i+2},s{i+3})
        si string < 48 entonces
                string <- string + 74
```

```
        fin si
        si string > 122 entonces
                string <- string - 74
        fin si
        si string > 57 y string < 65 entonces
                string <- string + 8
        fin si
        si string > 90 y string < 97 entonces
                string <- string - 8
        fin si
        hash <- funcion_concatenar(hash, funcion_convertir_a_texto(string))
        i <- i + 4
fin para

escribir hash

FINALGORITMO
```

### 4.2.5 Implementación

Tabla 2. Comparación entre C, PHP, ASP

| Características | C | Pts. | PHP | Pts. | ASP | Pts. |
|---|---|---|---|---|---|---|
| Licencia | Libre | **5** | Libre | **5** | No Libre | 1 |
| Multi-plataforma | Sí | **5** | Sí | **5** | No | 1 |
| Portabilidad | No | 1 | Sí | **5** | Sí | **5** |
| Conocimientos necesarios | Avanzado | 1 | Intermedio | 3 | Intermedio | 3 |
| Intuitividad | Baja | 1 | Alta | **5** | Alta | **5** |
| Rapidéz | Alta | **5** | Alta | **5** | Alta | **5** |
| Puntuación | TOTAL | 18 | **TOTAL** | **27** | TOTAL | 20 |

Debido a las características necesarias para implementar el algorítmo, y la calificación recibida según los items estipulados, el mejor lenguaje de programación para la implementación y demostración del algorítmo es PHP.

Para la demostración del funcionamiento del algorítmo este se escribió en el lenguaje de programación web PHP, y su código y explicación se encuentran a continuación. Para entender el funcionamiento del algorítmo hace falta tener conocimientos medianos en el lenguaje de programación estipulado.

Se declara la función DEA junto con los argumentos de entrada, los cuales son dos variables, $text que representa el texto plano a ser cifrado y la variable $key la cual representa la llave a ser usada, ya que el algorítmo acepta una llave estipulada por el usuario. Si no existe una llave estipulada por el usuario, el algorítmo usará una llave por defecto, la cual es 'DEA', el texto y la llave se copian en las variables $t y $l respectivamente a su vez, la variable $long almacena la longitud de la llave, esta longitud entre más alta sea, mayor será su seguridad, en este caso se usará una llave de 64 bits.

```
function DEA($text,$key){
	$long = 64;
	$count = 0;
	$t=$text;
	$l=$key;
	$texto = $t;
	if($l == NULL){
		$l = 'DEA';
	}
	$lt = strlen($t);
	$ll = strlen($l);
```

Las variables $lt y $ll almacenan la longitud del texto ingresado y la longitud de la llave respectivamente, esta longitud se obtiene por medio de la función strlen(). Luego de obtener los datos necesarios, iniciamos la conversión de cada carácter de la cadena de texto en su respectivo valor hexadecimal, estos valores se obtienen

usando la función ord() y se almacenan en la variable $at concatenando sus valores y dejando un solo texto hexadecimal al cual se le halla la longitud y se almacena en la variabale $lat.

```
for($a=0; $a<$lt ; $a++){
        $ab = $ab + ord($t{$a});
        $at = $at . $ab;
}
$lat = strlen($at);
```

Una vez obtenida la longitud de la cadena hexadecimal, se hacen grupos de 3 bits, para así obtener su valor ASCII por medio de la función chr() y estos se almacenan en la variable $ab concatenandolos.  El nuevo texto plano será almacenando en su totalidad en la variable $t, reemplazando el antiguo texto plano.

```
for($a=0; $a<$lat ; $a++){
        $ab = $ab . chr($at{$a}.$at{$a+1}.$at{+2});
        $a = $a + 2;
}
$t = $ab;
```

Se realizan los mismos procedimientos para cambiar la llave.

```
for($a=0; $a<$ll ; $a++){
        $ab = $ab + ord($l{$a});
        $at = $at . $ab;
}
$lat = strlen($at);
for($a=0; $a<$lat ; $a++){
        $ab = $ab . chr($at{$a}.$at{$a+1}.$at{+3});
```

```
        $a = $a + 2;
    }
    $l = $ab;
```

Una vez convertida la cadena de texto y la llave en un diferentes y nuevos textos planos, almacenados en $t y $l respectivamente, se procede a realizar grupos del tamaño de la longitud de la llave, entre más alto sea el valor de la longitud de la llave más probabilidades diferentes de hash existirán, el algorítmo fue programado para trabajar con llaves de 64 bits, por esta razón la longitud del hash es de 64 bits y la agrupación de los mismos también.

```
    for($a=0 ; $a<$long ; $a++){
        for($b=0 ; $b<$ll ; $b++){
            if($a == $long){
                break;
            }
            else{
                $l64 = $l64.$l{$b};
            }
        $a++;
        }
        $a = $a-1;
    }
```

Ahora se concatenan la llave y el nuevo texto hexadecimal, intercalando un bit del texto con un bit de la llave, hasta alcanzar una longitud de 128 bits y se almacena este nuevo texto en la variable $nt a la cual se le halla la longitud y se almacena en la variable $lnt, así se generarán grupos de 128 bits.

```
    for($j=0 ; $j<$long ; $j++){
```

```
        $nt = $nt.$t{$j};
        $nt = $nt.$l64{$j};
    }
    $lnt = strlen($nt);
```

Se convierte el texto en una nueva cadena ASCII de carácteres, con las siguientes funciones, y estas se almacenan en la variable $ascii al igual que las otras funciones se halla la longitud para mantener siempre un control y evitar un desbordamiento de buffer.

```
    for($j = 0; $j < $lnt ; $j++){
        $ascii = $ascii.$preascii = ord($nt{$j});
    }
    $lascii = strlen($ascii);
```

Una vez obtenidos los datos, realizamos una concatenación y sumatoria de bits en grupos de 4 bits, divididos en 2 bits y concatenandolos en la nueva variable $nascii, hallando tambien su longitud.

```
    for($j=0 ; $j < $lascii ; $j++){
        $h1 = $ascii{$j}.$ascii{$j+1};
        $h2 = $ascii{$j+2}.$ascii{$j+3};
        $sec = $nascii{$j} + $nascii{$j+1};
        $h3 = $h1 + $h2 + $sec;
        $nascii = $nascii . $h3;
    }
    $nlna = strlen($nascii);
```

Se realiza la concatenación en grupos de 4 bits.

```
if($nlna < 1024){
        for($j=$nlna+1 ; $j<=1024 ; $j++){
                $nascii = $nascii.'3';
        }
}
$nlna = strlen($nascii);
```

Se inicializa la variable $ini en cero y se realizan grupos de 128 bits. En un array llamado $semi.

```
$ini = 0;
for($i=0 ; $i<8 ; $i++){
        for($j=$ini ; $j<$ini+128 ; $j++){
                $semi[$i] = $semi[$i].$nascii{$j};
        }
        $ini = $ini + 128;
}
```

Ahora, luego de obtener ocho grupos de 128 bits, dentro de cada grupo se toman 8 bits y se suman entre ellos.

```
for($i=127 ; $i>=0 ; $i--){
        $o = $semi[0]{$i}+$semi[1]{$i}+$semi[2]{$i}+$semi[3]{$i}+$semi[4]{$i}+
$semi[5]{$i}+$semi[6]{$i}+$semi[7]{$i};
        $s = $s.$o;
}
$ls = strlen($s);
```

Luego se de la sumatoria, se realizan parejas de bits y se suman consecutivamente, y se le realiza un control de salida, para que solo muestre

carácteres alfanuméricos.

```
    for($k=0 ; $k<$ls ; $k++){
        $string = ($s{$k}.$s{$k+1})+($s{$k+2}.$s{$k+3});
        if($string<48){
            $string = $string + 74;
            }
        if($string>122){
            $string = $string - 74;
            }
        if(($string>57)&&($string<65)){
            $string = $string + 8;
            }
        if(($string>90)&&($string<97)){
            $string = $string - 8;
            }
```

Una vez confirmado el código hexadecimal final, se convierte en convierte en texto plano, y se concatena para obtener el hash final.

```
        $f = $f.chr($string);
        $k = $k + 3;
    }
    $final = $f;
    $msjdigest = return($final);
}
```

# 5. PRESENTACIÓN Y ANÁLISIS DE RESULTADOS

El Algorítmo fue sometido a diferentes pruebas -Por ejemplo, Criptoanálisis diferencial, entrada de carácteres especiales, criptoanálisis basados en fallos de Hardware-, a las cuales respondió satisfactoriamente, ya que la posibilidad matemática de que el algoritmo genere un mismo Hash con diferentes entradas, es casi imposible, pues la posibilidad de salida es $97^{1024}$ si se tomara un teclado base 97.

El algorítmo responde satisfactoriamente a la entrada de código ASCII haciendo que su seguridad aumente, ya que la entrada de carácteres fuera del alfabeto convencional, conocidos también como caracteres especiales, no generan ningún error en la salida, ni escape de información.

El algoritmo de cifrado simétrico, realiza un resumen de la cadena cifrada, por lo tanto, reduce la longitud de bits a 64 bits, se le puede llamar “Algoritmo de Cifrado Simétrico de Resúmen”, Al algorítmo se realizaron las siguientes pruebas:

## 5.1 ANÁLISIS DE PRUEBAS

### 5.1.1 Por criptoanálisis diferencial

El criptoanálisis diferencial es un ataque estadístico aplicable a cifrados producto, en los que se itera 'r' veces una transformación criptográficamente débil. Fue propuesto por Biham y Shamir.

El criptoanálisis diferencial funciona estudiando grandes cantidades de parejas de

texto claro y sus correspondientes cifrados. Las parejas de textos pueden ser escogidas por el criptoanalista o tomados al azar. Naturalmente, en el primer caso se obtienen resultados satisfactorios con menos operaciones. Los textos claros han de ser tales que la diferencia entre dos de ellos satisfaga una diferencia conocida en donde la diferencia es la resta módulo 2 bit a bit (idéntica a la suma), que en lógica binaria equivale a la operación OR exclusiva. La intención es disminuir el número de pruebas requerido para rotura del cifrado respecto de un ataque por fuerza bruta para recuperar la clave.

Para cada criptosistema es preciso preparar un esquema de ataque personalizado. En el libro de Biham y Shamir se presentan varios posibles ataques a los algoritmos más populares. En la mayoría de los casos se obtiene una ventaja sobre un ataque por fuerza bruta. El DES ha constituido el principal objetivo de este tipo de criptoanálisis. Contra la versión completa con 16 vueltas, se puede obtener la clave después de analizar 2^47 parejas de textos claros escogidos y sus correspondientes cifrados, o bien 2^55 parejas de textos claros conocidos y sus correspondientes parejas de textos cifrados.

Al realizar el ataque del Criptoanálisis diferencial a SBDA se consiguieron resultados satisfactorios, ya que el cifrado de diferentes cadenas de texto, y la variación de unos cuantos Bits daban como resultado hashes totalmente diferentes (Ver tabla 3).

### 5.1.2 Por criptoanálisis basados en fallos de Hardware

En 1996 apareció un nuevo tipo de ataque, consistente en la recuperación de la clave contenida en sistemas con fallos de hardware. Esta vía de ataque ha sido propuesta por Dan Boneh, Rich DeMillo y Ricard Lipton, investigadores de la empresa Bellcore, que han postulado su viabilidad en criptosistemas que usan operaciones modulares, como la mayor parte de los sistemas de clave pública. Eli

Tabla 3. Resultados de Hash

| TEXTO | HASH |
|---|---|
| hola | yv6028v12x448ty7v3px26v6w4y13put0280EqxHvw0xty68tr360twxu3qw7v8r |
| Hola | sur3ywmswx21xtvuwsyt5xyy0wvuyvwy64rxvx32u72ww7yt1Duqxsowry5puuxr |
| HOla | 9xrt0unyw6nr34t7tuxtpv2n1wuprt9w3pwpuuy1r10Dyuw6u3182uw03rtH4q6y |
| HOLa | 00Dx4t0yy35u808v7C91y63t052Bw85y10x1vw6vBx6DBC488y2EFr4000vExux4 |
| HOLA | rp2nvs6l0mrykox7nsv9nwp8qpu0qsn22m584py85Dso57uy7Btyu2qsx50rk64j |
| HoLa | x4vssuwwqtxspqxqso6v0lutxnH4ro03rpv2vul54xq4BpsEBs5s1x7o5yj5y5ok |
| HoLA | v1t6305xw010oy3s0B76tx22ywxw203uyxt2m1w10o757yx01997Hu3213v8u19u |
| HolA | m15lo24qo272ur5xsm60vrtsoq44y1r61su83sx3Fty233r060400vwwp4r74txt |
| hOLa | v4sxuwxrm17kn7wlpu2xos50uq84lt58o2vAq2uCt2vx5u530uy10xBouyq7vrts |
| hOLA | 79wy00v33062039731301215yD87ypEvv05918x581ty43985307Bq04u1vFCw32 |
| hOla | tswu501xsw1u2xy4v3sv1x013svlr34ty8346vyy042y122w80111ywyt4114vvm |
| hOlA | xu4owtxxpr4Drqn1lCsBr7t2xvm0Dop32wyv050u43wurDyy5B03x18rn8rB243g |
| hoLA | no3Bkv7to0sFuumysmn83kxD8o417syo522vvFyqxx0837x6sE4vsv8ovwy02u1r |
| hoLa | nrqsy7vtlwv1031v2ltt3s720vromv1yy24o5v5wr6BsvvvwF65y3xmkyqx1x2oj |
| holA | soo0xvmqyt5n3tsvrrv12x1xwopo450r8upvuwvnuD1x421547yu3ys0ntxm1vqn |
| a | Hs82rGt2CrD3rB22Gn13nI1uQkAyqHv1Dm7urHyqFnD0qB8wBnGytHy4Bo69oBCp |
| aa | rx7tvxyI2840yq1CtC6py4BB215uu11vsw2v6v22Bu546v99wB910sy12518spk1 |
| aaa | Bs3rw9rxEusywE107wqut1vr1wstrx3qGysssyFvDytutx72Ittqu43pEuqmx04m |
| A | D2vr3s6q80vpttDuExykmx8nFtpkqtGkFusku5GmB3wk5u9m6qsmp43o24uxw0Ck |
| Aa | 8w03222v336uBx3E0x367w36p87u4280w004x12wnE7647BvCE66x68uF6w9nsur |
| AA | wvvswvwssv003u1r41wv327qvxs7tq2uw0w2mtsusp44H1220x21prqwotxsqrtt |
| Aaa | 00w0wurysprt2t0w3wxu6vsov07rwwxsyywwo42vnA3w54u3Buxx7v0swu33svwn |
| AAa | wqxtGr1x6vnpMn4wskw0IoBw7k1pGnGruk6tEx986qupHl76tlwuAmyuyoyvBo4y |
| AaA | 8qxrsBts6v0xoCvs1lurqDtw5mtupL9u3mtsnBywFq49tHy8Cm0pmC2o1nqus85q |
| aAA | 11q01Gr1v11m2Fn3uym1rGmFvvm0sEnBwlmysC2651vw8Bo8myyroCqxvsk0vBs4 |
| aaA | wrw3xrtx0um735p2xsrv4xv0stut420xv5syy1t084u0xCD6p201vyursut5sqto |
| aAa | tqy3ruu3tus4sxou2sr0Br3trux17s3wy3psy2pt6CyDutBwmxu1yv4rqxy9vkyp |
| AAA | x4yKx2Fsy1nGvC3ys34o2vE1sx4yt1y3yrv3772BxuwDp4826wF7620p8vqIr00o |
| qwertyuiop | xuy704x41Fyynnvr7Cx2wwu63o54s3tq2vtx7D2y34ut4rtoulvuw3Bttvom23yG |
| qwertyuiop x10 | SBxDC4D1Kx8BB93s259I9A8H9HHL5HFSI1JDCND0LB5FHisF3EyE0E30EL38D4JK |
| qwertyuiop x100 | 35671vBu75DGrFBFGFsDB9uECLADCGKBC8A9LEATHFwD6XHv93FC37CCFE1NEBBJ |
| qwertyuiop x1000 | 25x0D30014DC848GE29A9FxB67J9CFHFRMC9UEx8HM1D9EL4J3249746ELGHEGxL |
| 1 | qDx2v4w8qAtus0tBkDyxsp13sAxtq52AlDyqsytDtH5rrunFtHnws2wyrBrru585 |
| ~`!@#$%^&*() | E6437xx5214w7uAB9IG2w3860uFx3swC8tCp45m0C0oG29ruAys6mF1oC27vq63v |
| NULL | pwuqB33q7Gy5qvu2567Bu29ss1H00s0q6t5sr2H2yCqowyx7o5m2Jrwlr111v2FA |

Biham y Adi Shamir han extendido la validez de este ataque al DES y otros algorítmos de cifrado en bloque. Posteriormente, Anderson y Kuhn lo han optimizado en tiempo y en operaciones.

El ataque consiste en someter al equipo (hardware) de cifrado en el que reside la clave, a la que se desea acceder, a una perturbación física, como un bombardeo ligero con partículas ionizantes, falta de tensión, sobretensión, etc., de tal modo que se perturbe parcialmente su funcionamiento de forma temporal o permanente. Una vez logrado esto, se comparan textos cifrados, de manera parecida a como se hace en el criptoanálisis diferencial, obtenidos a partir de textos claros conocidos, antes y después de la perturbación. Las discrepancias se podrían explotar para recuperar la clave.

Este ataque resulta especialmente peligroso contra criptosistemas contenidos en tarjetas inteligentes, ya que, aunque están a prueba de lectura indebida, pueden ser sometidas fácilmente a agentes perturbadores externos. La versión más moderna, en lugar de provocar fallos definitivos en el sistema, provoca simplemente fallos temporales mediante 'glitches' en la alimentación y en el reloj. Esta técnica ya había sido utilizada por hackers para prolongar las suscripciones a la televisión por pago controlada por tarjetas chip (Sky, DirectTV, ...), pero de forma sencilla, sin combinarlo con criptoanálisis.

Al realizarle este ataque a SBDA, se encontró una falla en el tiempo de respuesta del servidor, la solución a este inconveniente fue aumentar el tiempo de respuesta al cliente, para asi conseguir que el servidor que hace uso del algorítmo pueda realizar el proceso de cifrado satisfactoriamente.

# 6. CONCLUSIONES

Los algorítmos de cifrado simétricos de resúmen aunque ofrecen una seguridad no es la apropiada, ya que las tecnologías evolucionan rápidamente, y son vulnerables o fáciles de romper debido a que existe mayor capacidad de procesamiento y esto conlleva al descifrado de la clave en menor tiempo.

El algorítmo SBDA (San Buenaventura Digest Algorithm) desarrollado como producto de esta investigación se diseñó cumpliendo con los parámetros de simplicidad, fiabilidad, calidad y a bajo costo requeridos por la organismos multinacionales generadores de estándares a nivel de seguridad como la NIST, NSA, IEEE e ISO.

SBDA es un algorítmo de cifrado simétrico de resúmen que empieza a sumplir en parte la vulnerabilidad de los mismos, ya que garantiza frente a sus similares como MD5 y SHA una mayor posibilidad de combinaciones en la salida del Hash; es decir, si se compara los Hashes de MD5 con SBDA se obtiene como resultado de MD5 un cifrado con una salida Hexadecimal (valores numéricos entre 0 y 9, y alfabéticos entre la A y la F mayúsculas), mientras que SBDA ofrece una salida de hash más amplia, usando como estándar el alfabeto Inglés diferenciando mayúsculas de minúsculas y usando los números de 0 a 9.

# 7. RECOMENDACIONES

Se recomienda la reestructuración de los algorítmos de cifrado simétricos de resúmen, ampliando su salida de hash, y habilitando la salida para un código ASCII más extenso, usando, como se hizo con SBDA, el alfabeto Inglés, y diferenciando mayúsculas de minúsculas.

Para seguir mejorando el algorítmo SBDA, y conseguir una mayor complejidad en el cifrado y en la salida, se puede extender el hash de resultado hasta 1024 bits, y aumentar el numero de iteraciones de conversión y agrupación de bits.

## BIBILIOGRAFÍA


COWEN, David L. Super utilidades hacker: hackers y seguridad. Madrid. Anaya Interactiva, 2006. 960 p.

GUTIERREZ, Juan Diego y LOPEZ GUISADO, Angel. Seguridad en redes locales: guía práctica. Madrid. Anaya Interactiva, 2008. 400p.

HUSEBY, Sverre H. Innocent code: A security wake-up call for web programmers. Lóndres. LEA, 2003. 256p.

JIMENO GARCIA, Maria Teresa. Hacker: guía práctica. Madrid. Anaya Interactiva, 2008. 368 p.

LOCKHART, Andrew. Seguridad de redes: los mejores trucos. Madrid. Anaya Interactiva, 2007. 528 p.

MONTERO AYALA, Ramon. Protección ante Internet: como proteger el ordenador. Madrid. Creaciones Copyright, 2007. 135 p.

ORTEGA TRIGUERO, Jesus J. Y LOPEZ GUERRERO, Miguel Angel. Introducción a la criptografía: historia y actualidad. Cuenca. Universidad de Castilla – La Mancha, 2006. 251 p.

PIATTINI VELTHUIS, Mario. Auditoria de tecnologías y sistemas de información. Madrid. Ra-Ma, 2008. 731 p.

PICOUTO RAMOS, Fernando. Hacking y seguridad en Internet. Madrid. Ra-Ma, 2007. 552 p.

VELTE, Anthony T. Manual de Cisco. Madrid. McGraw-Hill, 2008. 432 p.

Wikipedia “Criptografía” Disponible en: http://es.wikipedia.org/wiki/Criptograf %C3%ADa. Consultado: 18 de Febrero de 2008, 6:22 pm.

Wikipedia “Esteganografía” Disponible en: http://es.wikipedia.org/wiki/Esteganograf %C3%ADa. Consultado: 18 de Febrero de 2008, 6:37 pm.

Peiper “Encriptación (parte 1)” Disponible en: http://www.peiper.com.ar/edicion02/encriptacion.pdf. Consultado: 18 de Ferbero de 2008, 7:09 pm

NIST “Advanced Encryption Standard (AES)” Disponible en: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Consultado: 18 de Febrero de 2008, 7:48 pm.

MIT Laboratory for Computer Science and RSA Data Security “The MD5 Message-Digest Algorithm”. Disponible en: http://www.ietf.org/rfc/rfc1321.txt. Consultado: 19 de Febrero de 2008, 12:27 am.

Wikipedia “MD5”. Disponible en: http://es.wikipedia.org/wiki/MD5. Consultado: 19 de Febrero de 2008, 1:49 am.

Cisco Systems “US Secure Hash Algorithm 1” Disponible en: http://www.ietf.org/rfc/rfc3174.txt. Consultado: 19 de Febrero de 2008, 5:20 pm.

Bruce Schneier “SHA-1 Broken”. Disponible en: http://www.schneier.com/blog/archives/2005/02/sha1_broken.html. Consultado: 19 de Febrero de 2008, 7:17 pm.