UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN

# "SISTEMA DE CONTROL DE ACCESO A OFICINAS"

# T E S I S

**QUE PARA OBTENER EL TÍTULO DE**
**INGENIERO EN COMPUTACION**

**P R E S E N T A:**

**OSCAR ANDRES ZARAGOZA GARCÍA**

**ASESOR: ING ANTONIA NAVARRO GONZÁLEZ**

**BOSQUES DE ARAGÓN** **2007**

Agradezco:

A Dios por el camino que me presento y con el cual espero llegar a mi destino en la vida.

A mis padres, que no solo me dieron la vida, si no también todo su amor, además de que con su guía y paciencia me dieron los medios para llevar a cabo este trabajo y ser la persona que soy hoy y llegar a ser la persona que espero ser en el futuro.

A mi abuelita, a mi tía Irma y a mi tío Rosendo por haber contribuido de una manera muy especial a mi formación.

A mis hermanos que siempre han estado a mi lado para contribuir a mi desarrollo personal y profesional.

A mis amigos, sobre todo a tres de ellos, con los cuales comencé la universidad, y aunque nuestras vidas toman rumbos distintos me honran con el regalo de su amistad y apoyo, sobre todo cuando me cerré a las posibilidades de la vida.

A mi asesora la Ingeniera Antonia Navarro González quien fue la primera persona que me dio la oportunidad de desarrollarme profesionalmente y que me guió en el correcto desarrollo de este trabajo.

A la Universidad Nacional Autónoma de México, sobre todo a la Facultad de Estudios Superiores Aragón por las facilidades prestadas para mis estudios profesionales.

## Introducción

La seguridad de inmuebles ha sido un aspecto ampliamente difundido por los medios en la actualidad y esta siendo considerado como una necesidad y no un lujo.

La demanda que se tiene de estas soluciones y la oferta de las que actualmente se ofrecen obliga a analizar cuidadosamente cada una de ellas. Sin embargo, como en cualquier solución cada una tiene sus ventajas y desventajas. De estas últimas la principal es la dificultad de administrar eficiente y centralizadamente los puntos de acceso. Otro aspecto preocupante es la posibilidad de duplicar los elementos que conceden el acceso, ya que la mayoría de las soluciones actuales usan la misma clave de acceso para su validación y no cuentan con un proceso automatizado que pueda cambiar dicha clave. Además, las opciones que permiten centralizar procesos de alarma son costosas lo que dificulta el que sean consideradas para incluirlas en una solución completa. El objetivo de este trabajo es proponer un sistema para estos problemas, recomendando, de manera teórica un sistema de control de acceso a oficinas centralizado y con un proceso automatizado de cambio de claves de acceso, haciendo uso de técnicas criptográficas para este fin.

## Capítulo 1 Teoría de sistemas de control de acceso

El objetivo de un sistema de control de acceso es *Limitar a un grupo de entidades seleccionadas el que puedan interactuar de manera confiable y disponible con un elemento o bien material.*

Esta función no es exclusiva de nuestro entorno actual, es una conducta básica el proteger un bien de otras entidades, lo cual se puede lograr de distintas maneras.

La primera de ellas es mantener una vigilancia constante sobre el bien específico, pareciera ser que así fuera la manera más efectiva de mantenerlo "seguro" pero limita al propietario su libertad y el uso libre del bien antes mencionado. Así es que si el propietario desea continuar con dicha solución debe de buscar a otra entidad a la cual delegarle la tarea de vigilancia, así surgen grupos de vigías que resguardan un bien o propiedad. La desventaja de esta situación es buscar un servicio de vigilancia que pueda ser confiable, es por esto además de su alto costo que solo puede ser empleada en ocasiones muy limitadas.

La segunda es bloquear el acceso al bien en cuestión por medios físicos ya sea el terreno o una construcción hecha para tal fin. El problema con estas soluciones es que pueden ser muy costosas y no permiten una disponibilidad del bien protegido. A esto cabe añadir que cuando se mantiene su ubicación en secreto se corre el riesgo de no volver a encontrar la propiedad ó que al no estar la entidad que conocía la locación ésta se pierde.

Por último existen los sistemas de cerradura los cuales funcionan en conjunción con los bloqueos físicos pero brindan otra forma selecta de acceso aunque se ha comprobado que conforme estos métodos se desarrollen también lo harán las formas de sobrepasarlos.

En la actualidad la ciencia ficción nos muestra varias soluciones para proteger los bienes, cada una con diferentes ventajas y desventajas, pero principalmente la que más se impone es el costo de dichos sistemas y la dificultad que implica montarlos y supervisarlos. Dejando quizás para una elite el uso de sistemas electrónicos.

Algunos ejemplos de estos son:

a. Dispositivos Biométricos:

Permiten controlar el acceso por medio de la identificación de un rasgo corporal, los más comunes son los lectores de huellas digitales un ejemplo de estos se muestra en la Fig. 1, estos pueden trabajar solos o en conjunto, cada una de las unidades tiene un valor aproximado de US$ 1077.00 hasta los US$ 1790.00 y para conectar varias se necesita equipo independiente.

Fig. 1 Lector Biométrico

b. Lectores de Tarjeta Magnética

Utilizan tarjetas de tipo bancario, pueden almacenar información sobre varias tarjetas y ser programados para controlar varias áreas. Pueden ser utilizados con tarjetas adquiridas por el usuario o con tarjetas bancarias de cada usuario. Su rango de precio va desde los US$ 435.00 hasta los US$ 646.00.

Fig. 2 Lector de Tarjeta Magnética y Tarjetas Magnéticas

c. Lectoras de Proximidad

Funcionan con tarjetas o llaveros que contienen un diseño de circuito que se activa por inducción. También se puede arreglar que las tarjetas sirvan como gafete de identificación al imprimir en ellas los datos y fotos de los usuarios. Su precio aproximado es de US$ 232.00.

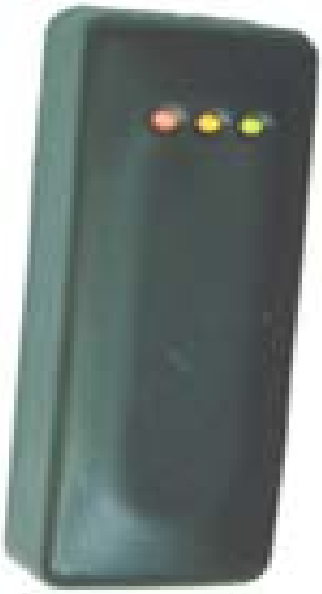

Fig. 3 Lector de Proximidad

d. Teclados Numéricos

Son una forma de control de acceso bastante común, su funcionamiento por lo general es en base a una clave de 4 dígitos decimales, a estos dispositivos se les puede asignar varias claves a una misma unidad para identificar los accesos. Su precio va desde los US$ 104.00.

Fig. 4 Teclado Numérico

Cada sistema requiere también de la instalación de los mecanismos para la apertura de las cerraduras por lo que su costo se eleva.

Así también, algunos de estos sistemas pueden utilizar el mismo elemento de identificación en varias terminales, pero hay que programar cada terminal.

Es por esto que el costo de estos sistema se eleva sobremanera además de que al no estar enlazados su administración se hace poco práctica. Es por esto que se propone en este

trabajo un sistema que permita una administración sencilla del control de acceso y que sea factible implementarlo, este sistema será modular por lo que podrá expandirse como sea necesario, el sistema en su nivel más básico queda conformado como lo indica la Figura 5.

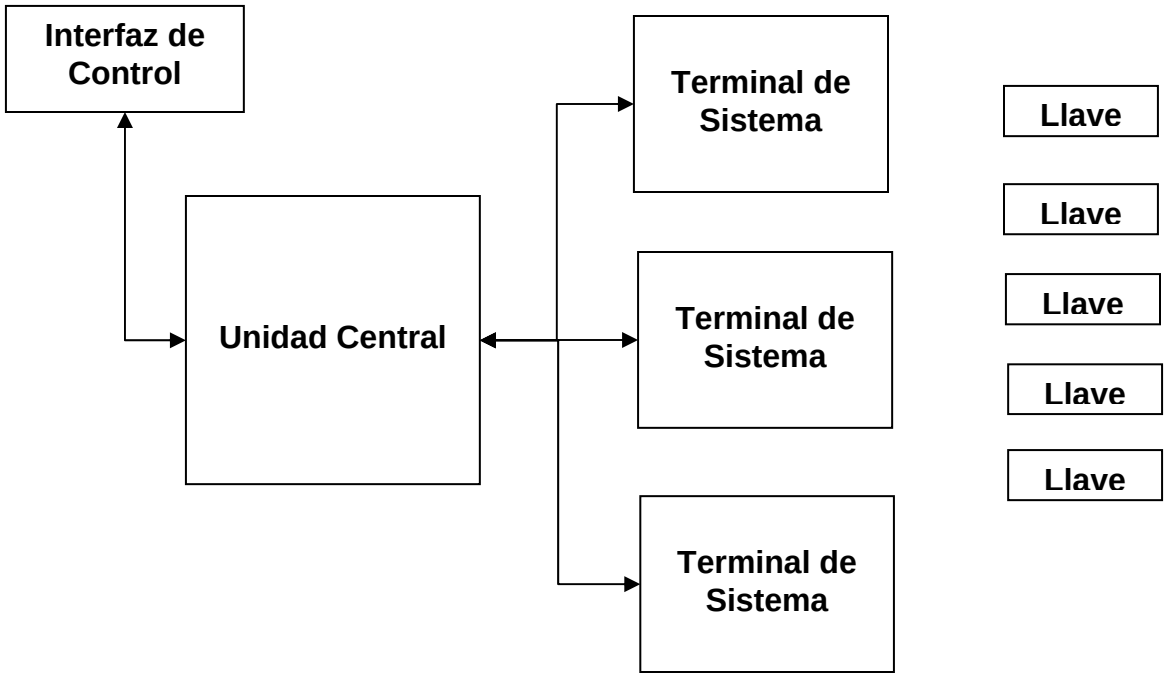


Figura 5. Estructura Básica del Sistema

De esta forma se controlan las operaciones generales del sistema desde la **Unidad Central**, lo cual se realiza por medio de una **Interfaz de Control**, también se propone que en cada **Terminal de Sistema** se pueden controlar funciones específicas de la forma en la que reaccionan estas a situaciones de alerta. Por lo demás es similar a los sistemas ya descritos, cada usuario del sistema debe contar con un elemento para identificarse que en este caso es la **Llave**.

A continuación se brinda un antecedente teórico de la forma en la que se crea la clave que se almacena en las llaves.

## Capítulo 2 Nociones básicas de criptología

Se le llama Criptología al conjunto de técnicas que nos ayudan a asegurar la información y a recuperar la información original, llamando a estas dos operaciones *CIFRAR* y *DESCIFRAR* respectivamente.

Es por ello que se divide en dos ramas la Criptografía y el Criptoanálisis.

La primera viene del griego *KRYPTOS* que significa oculto y *GRAFÍA* que significa escritura y su definición es *"Arte de escribir con clave secreta o de un modo enigmático"*, en estos días ha pasado de ser solamente un arte a un conjunto de técnicas para proteger el acceso de la información de entidades no autorizadas a obtenerla.

Por su lado el Criptoanálisis es el conjunto de técnicas enfocado a romper o penetrar los códigos generados por medio de la Criptografía.

Existen varios métodos para ocultar la información como lo es escribir de una manera especial (cifrar) u ocultar el mensaje en otro que tenga un significado propio (esteganografía). El objetivo final es que solo un conjunto selecto de entidades puedan interpretar correctamente la información.

Es por esto que la criptografía es una herramienta esencial en el mundo moderno donde la información es de un valor elevado para su dueño y para sus competidores. Así se usa para asegurar el viaje de esta información por las redes de datos y que sea recibido e interpretado en su destino exclusivamente.

### 2.1 Criptosistemas

Se le llama así al conjunto de cinco elementos (quíntupla) que representa a los elementos involucrados en las operaciones de criptología, así se cuenta con estos nombres para los elementos (***M, C, K, E, D***) donde:

- ***M*** representa el conjunto de todos los mensajes sin cifrar (texto plano) que pueden ser enviados.
- ***C*** representa el conjunto de todos los mensajes cifrados, o **criptogramas**.
- ***K*** representa el conjunto de claves que se pueden utilizar en el criptosistema.
- ***E*** es el conjunto de *transformaciones de cifrado* o familia de funciones que se aplica a cada elemento de ***M*** para obtener un elemento de ***C***. Existe una transformación diferente $E_k$ para cada valor posible de la clave ***k***.
- ***D*** es el conjunto de *transformaciones de descifrado*, análogo a ***E***.

Todo criptosistema debe cumplir con lo siguiente:

$$D_k(E_k(m)) = m$$

Es decir, por un mensaje ***m***, al cifrarlo por la clave ***k*** y al descifrarlo por la misma clave ***k***, obtenemos el mismo mensaje ***m***.

Para la criptología existen dos tipos de Criptosistemas:

a. *Simétricos o de Clave Privada*: Emplean la misma clave ***k*** para cifrar y descifrar. Por eso las dos partes (Emisor y Receptor) deben tener la clave, de ahí surge el problema de cómo hacer llegar de manera segura la clave ***k***.
b. *Asimétricos o de Clave Pública*: Se usa una doble clave ($k_p$, $k_P$). $k_p$ se conoce como *clave privada* y $k_P$ se conoce como *clave pública*. La primera es para la transformación ***E*** de cifrado y la otra para la transformación ***D*** de descifrado. Pueden ser intercambiables (si se usa una para cifrar la otra se puede usar para descifrar y viceversa). Lo importante es que a partir de una de las claves no se puede obtener su pareja.

2.2 Métodos de la Criptografía moderna

Con este objetivo se han desarrollado varios algoritmos que transforman la información para que esta pueda realizar este viaje, que, por ser en redes públicas no debe ser interpretado más que por su receptor autorizado.

Así, de acuerdo a su uso podemos catalogar a estos algoritmos:

Digestión de Mensajes (Hash)

Toma una entrada de cualquier tamaño y produce una salida de tamaño fijo, sin alterar la entrada original. Su principal característica es que tienen una probabilidad muy baja de que dos entradas diferentes generen una salida idéntica, además de que es muy improbable que se pueda obtener el mensaje de entrada en base a la cadena de salida.

Dos algoritmos de este tipo son el MD5 (Message Digest 5) y el SHA (Secure Hash Algorithm).

MACs (Message Authentication Codes) Códigos de Autenticación de Mensajes.

Realizan la misma función que una Digestión de Mensajes usando una llave, su función es comprobar que la información (entrada de la función) no haya sido modificada antes de llegar al receptor autorizado de la misma. Para realizar esta acción existen dos variables de los algoritmos de Digestión Hmac MD5 (pide que la longitud de la llave sea al menos de 128 bits). Hmac SHA-1 (su llave debe de ser de al menos 160 bits).

Firmas Digitales (Digital Signatures)

El resultado de esta función se envía junto con la entrada para que él o los receptores puedan corroborar que la información ha sido enviada por el emisor declarado, para realizar esta función se utiliza lo que se llama “Arquitectura de llave Pública y Privada” en la cual el firmante genera las dos llaves haciendo de conocimiento a los receptores su llave pública y, cuando el emisor desea enviar un mensaje debe procesarlo con el algoritmo y su llave privada y enviar el resultado y la entrada, él o los receptores al recibir este conjunto de información toman el agregado y al procesarlo con el algoritmo y la llave pública deben obtener un resultado satisfactorio.

Un ejemplo de estos algoritmos es el DSA (Digital Signature Algorithm). Sus llaves son de entre 512 y 1024 bits.

## 2.3 El algoritmo SHA

Para el desarrollo de este proyecto se seleccionó un algoritmo de resumen (HASH) para que con los datos de los usuarios, se genere una clave única la que debe ser almacenada en un circuito programable y al tener una longitud fija se les pueda entregar el mismo tipo de dispositivo a todos los usuarios.

Las características de este algoritmo de digestión están reguladas por el *Institute of Standard and Technology (NIST),* el Departamento de comercio de E. E. U. U. y el *Information Technology Laboratory (ITL)*.

El NIST es el encargado de realizar la publicación de estos estándares a través de los *Federal Information Processing Standards Publications (FIPS PUBS)* en el FIPS PUB 180-2 es la última revisión publicada del estándar y es en la que se basa la descripción que se hace en este trabajo.

En el FIPS PUB 180-2 se tiene la referencia de la evolución que ha seguido el algoritmo SHA, teniendo en total 5 versiones la original se le llama SHA-0 por que ha quedado en desuso ya que presentaba vulnerabilidades en su diseño, de esta manera se tiene otro conjunto de algoritmos los cuales se implementan dependiendo de la aplicación en la que se les requiera. El criterio que define su uso es la longitud de la llave requerida pudiendo ser desde 160 bits hasta 512 bits. Para el desarrollo de este sistema se propone el uso del algoritmo SHA-1 al ser el que tiene menor longitud en su cadena de resultado es adecuado para grabarlo en memorias no volátiles como lo son las **Llaves** propuestas.

### 2.3.1 Términos

A lo largo de la descripción de este algoritmo se utilizaran las siguientes definiciones:

**Bit** Dígito binario que puede tener el valor de 0 o 1.
**Byte** Grupo de ocho bits.
**FIPS** *Federal Information Processing Standard* (Estándar Federal de Procesamiento de la Información)
**Palabra** Grupo de 32 bits (4 bytes)

### 2.3.2 Parámetros símbolos y términos utilizados en el algoritmo

#### 2.3.2.1 Parámetros

**a, b, c,...,h** Son las variables de trabajo que son las palabras usadas en el cálculo de los valores Hash, $H^{(i)}$.
**$H^{(i)}$** El $i^{seavo}$ valor Hash. Siendo $H^{(0)}$ el valor Hash inicial; $H^{(N)}$ es el valor Hash final y es usado para determinar la digestión del mensaje.
**$H_j^{(i)}$** La $j^{seava}$ palabra del $i^{seavo}$ valor Hash, donde $H_0^{(i)}$ es la palabra extrema

izquierda del valor Hash *i*.

**$K_t$** Valor constante a ser usado para la iteración *t* del calculo Hash.

**k** Número de ceros agregados al mensaje durante la etapa de relleno (*padding*).

L Longitud del mensaje, *M*, en bits.

**M** Mensaje a ser procesado.

**$M_j^{(i)}$** La $j^{seava}$ palabra del $i^{seavo}$ bloque de mensaje, donde $M_0^{(i)}$ es la palabra extrema izquierda del bloque de mensaje *i*.

**n** Número de bits a ser rotados o cambiados cuando una palabra esta siendo procesada.

**N** Número de bloques en el mensaje rellenado.

**T** Palabra temporal de *w-bit* usada en el cálculo del valor Hash.

**w** Número de bits en una palabra.

**$W_t$** La $t^{seava}$ palabra de w-bit del itinerario del mensaje.

2.3.2.2 Símbolos

$\wedge$ Operación binaria AND.

$\vee$ Operación binaria OR (OR inclusiva).

$\oplus$ Operación binaria XOR (OR exclusiva).

$\neg$ Operación binaria complemento.

+ Adición modulo $2^w$.

<< Operación de sustitución a la izquierda, donde *x* << *n* es obtenida descartando los *n* bits de la extrema izquierda de la palabra *x* y completando el resultado con *n* ceros a la derecha.

>> Operación de sustitución a la derecha, donde *x* >> *n* es obtenida descartando los *n* bits de la palabra *x* y completando el resultado con *n* ceros a la izquierda.

2.3.3 Notaciones y convenciones del algoritmo

Cadenas de Bits y Enteros.

Se utilizara la siguiente terminología relacionada a las cadenas de bits y enteros.

1. Un *dígito hexadecimal* es un elemento del conjunto {0, 1,…, 9, a,…, f}. Es la representación de una cadena de 4-bit. Por ejemplo, el dígito hexadecimal “7” representa la cadena de 4-bit “0111”, y el dígito hexadecimal “a” representa la cadena de 4-bit “1010”.
2. Una palabra es una cadena de w-bit que puede ser representada como una secuencia de dígitos hexadecimales. Para convertir una palabra a dígitos hexadecimales, cada cadena de 4-bit es convertida a su equivalente en hexadecimal.
3. Un entero puede ser representado como una palabra o par de palabras. Una representación en palabra de la longitud del mensaje L**,** en bits, es requerida para las técnicas de relleno.
   Un entero entre 0 y $2^{32}$-1 inclusivo puede ser representado como una palabra de 32-bit. El menos significativo de los 4 bits que componen una palabra es representado por el dígito hexadecimal de la extrema derecha de la representación de la palabra.
   Lo mismo se mantiene correcto para un entero entre 0 y $2^{64}$-1 inclusivo, que puede ser representado como una palabra de 64-bit.

Si $Z$ es un entero, $0 \leq Z < 2^{64}$, entonces $Z - 2^{32}X + Y$, donde $0 \leq X < 2^{32}$ y $0 \leq Y < 2^{32}$. Ya que $X$ e $Y$ pueden ser representados como palabras de 32-bit $x$ e $y$, respectivamente, el entero $Z$ puede ser representado por el par de palabras ($x$, $y$).

4. Para los algoritmos seguros Hash, la longitud del *bloque de mensaje – m bits –* depende del algoritmo, para el caso del SHA-1 cada bloque de mensaje tiene 512 bits, los cuales pueden ser representados como una secuencia de **16 palabras de 32-bit**.

2.3.4 Operaciones en palabras

a. Las operaciones en binarias en palabras: $\wedge$, $\vee$, $\oplus$, $\neg$.

b. Adición Módulo $2^w$*:*

Tomemos $x + y$ que se definirá como sigue:

Las palabras $x$ y $y$ representan enteros $X$ y $Y$, donde $0 <\_ X < 2^w$ y $0 <\_ Y < 2^w$. Para enteros positivos $U$ y $V$, ***U mod V*** será el residuo de dividir $U$ entre $V$. Calcúlese:

$$Z=(X + Y) \bmod 2^w.$$

Entonces $0 <\_ Z < 2^w$. Conviértase el entero $Z$ a una palabra, $z$, y se define $z = x + y$. En el caso del algoritmo SHA-1 el orden de la operación es de $\mathbf{2^{32}}$.

**Nota**: Esta operación presenta una situación muy especial en computación ya que $w$ es muy grande y si fuera calculada produciría un error por la falta de capacidad de las computadoras de manejar variables tan grandes. Es por esto que se utilizan operaciones de la *Aritmética Modular* en especifico el **Algoritmo de Exponenciación Rápida**.

c. Rotación a la izquierda (desplazamiento circular a la izquierda). $\mathbf{ROTL^n(x)}$ donde $x$ es una palabra de w-bit y $n$ es un entero con $0 \leq n < w$, es definida por:

$$ROTL^n(x) = (x << n) \vee (x >> w - n).$$

Así, $ROTL^n(x)$ es equivalente a una sustitución circular (rotación) de $x$ a $n$ posiciones a la izquierda.

Nótese la equivalencia entre estas dos relaciones, donde $w$ esta ajustada en cada relación:

$$ROTL^n(x) \approx ROTR^{w-n}(x)$$
$$ROTR^n(x) \approx ROTL^{w-n}(x)$$

2.3.5 Operaciones con números grandes

Para realizar su función los algoritmos de criptografía emplean claves con un elevado número de bits, y usualmente se puede medir su calidad por la cantidad de esfuerzo que se necesita para romperlos. Así algoritmos como el DES tienen $2^{56}$ (56 bits) posibles claves, las cuales si una computadora capaz de realizar un millón de operaciones por segundo tardaría 2200 años en probar todas, si la clave tuviera 128 bits el tiempo necesario sería de $10^{24}$

años. Un ejemplo de estas operaciones es la **adición módulo $2^w$**.

2.3.6 Funciones y constantes del algoritmo SHA-1

SHA-1 usa una secuencia lógica de funciones, $f_0, f_1, f_2,..., f_{78}, f_{79}$. Cada función $f_i$ donde $0 \leq t < 79$, opera en tres palabras de 32-bit, *x, y,* y *z*, que produce una palabra de 32-bit como resultado. La función $f_t(x, y, z)$ se define como sigue:

| | |
|---|---|
| $Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$ | $0 \leq t \leq 19$ |
| $Parity(x, y, z) = x \oplus y \oplus z$ | $20 \leq t \leq 39$ |
| $Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$ | $40 \leq t \leq 59$ |
| $Parity(x, y, z) = x \oplus y \oplus z$ | $60 \leq t \leq 79$ |

SHA-1 usa una secuencia de ochenta palabras constantes de 32-bit, $K_0, K_1,\ldots, K_{79}$, las cuales están dadas por:

| | |
|---|---|
| 5a827999 | $0 \leq t \leq 19$ |
| 6ed9eba1 | $20 \leq t \leq 39$ |
| 8f1bbcdc | $40 \leq t \leq 59$ |
| ca62c1d6 | $60 \leq t \leq 79$ |

2.3.7 Etapas del algoritmo SHA-1

2.3.7.1 Pre-proceso

El preproceso toma lugar antes de que empiece el cálculo del valor Hash. Este preproceso consiste de tres pasos:

Complementar el mensaje, *M*, analizar el mensaje complementado en bloques de mensaje y definir el valor Hash inicial $H^{(0)}$.

a. Complementar el mensaje

El mensaje, *M*, debe ser complementado antes de que inicie el cálculo Hash. El propósito de esto es asegurar que el mensaje complementado es un múltiplo de 512.

Suponiendo que la longitud del mensaje, *M*, es $L$ bits. Se concatena el bit “1” al final del mensaje, seguido por *k* bits en cero, donde *k* es la menor, solución no negativa a la ecuación $L + 1 + k \equiv 448 \bmod 512$. Entonces complemente con el bloque de 64-bit que es igual al número $L$ expresado utilizando representación binaria. Por ejemplo tome la cadena “abc” cada carácter tiene longitud de 8-bit en código ASCII por lo que tiene longitud 8 * 3 = 24 por lo que se le complementa un bit en *1* al mensaje, entonces 448 – (24 + 1) = 423 bits en cero, entonces la longitud del mensaje original (“abc” cuya longitud es 24) para convertirse en un mensaje complementado de 512-bit.

b. División del mensaje

Una vez que el mensaje ha sido complementado debe de ser analizado para dividirlo en *N*

bloques de m-bit antes del cálculo del valor Hash. Así el mensaje queda dividido en *N* bloques de 512-bit, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$. Esto es para el algoritmo SHA-1.

c. Definir el valor Hash inicial ($H^{(0)}$)

Antes de iniciar el cálculo del valor Hash, el valor inicial Hash $H^{(0)}$ debe ser dispuesto. La longitud y número de palabras en $H^{(0)}$ depende de la longitud del resultado. Para el algoritmo SHA-1 el valor inicial Hash, $H^{(0)}$, debe de consistir de las siguientes cinco palabras de 32-bit, en hexadecimal:

$H_0^{(9)}$ = 67452301
$H_1^{(0)}$ = efcdab89
$H_2^{(0)}$ = 98badcfe
$H_3^{(0)}$ = 10325476
$H_4^{(0)}$ = c3d2e1f0

2.3.7.2 Uso del algoritmo SHA-1

El algoritmo puede ser usado para calcular el valor Hash de un mensaje *M*, que tenga una longitud de bits L, donde $0 \leq L < 2^{64}$. El algoritmo usa 1) un esquema de mensaje de ochenta palabras de 32-bit, 2) cinco variables de 32-bit cada una, y 3) un valor Hash de cinco palabras de 32-bit. El resultado final de SHA-1 es un mensaje de 160-bit llamado *digestión*.

Las palabras del esquema del mensaje son llamadas $W_0, W_1, \ldots, W_{79}$. Las cinco variables de trabajo son llamadas *a, b, c, d,* y *e*. Las palabras del valor Hash son llamadas $H_0^{(i)}, H_1^{(i)}, \ldots, H_4^{(i)}$ las cuales contienen al valor Hash inicial, $H^{(0)}$, remplazada cada vez por un *valor Hash intermedio* (después de que cada bloque de mensaje es procesado), $H^{(i)}$, y concluyendo con el *valor Hash final*, $H^{(N)}$, el algoritmo SHA-1 también usa una palabra temporal, *T*.

Después de que el preproceso se ha concluido, cada bloque de mensaje, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$, es procesado en orden, usando los siguientes pasos:

Para $i = 1$ a $N$:

1. Preparar el esquema del mensaje, $\{W_t\}$:

Siendo $W_t$:

$M_t^{(i)}$ $\qquad 0 \leq t \leq 15$

$ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$ $\qquad 16 \leq t \leq 79$

2. Inicializar las cinco variables de trabajo, *a, b, c, d,* y *e* con el $(i-1)^{\text{seavo}}$ valor Hash:

$a = H_0^{(i-1)}$
$b = H_1^{(i-1)}$
$c = H_2^{(i-1)}$
$d = H_3^{(i-1)}$

$e = H_4^{(i-1)}$

3. Para $t = 0$ a $79$:

$T = ROTL^5(a) + f_t(b, c, d) + e + K_t + W_t$
$e = d$
$d = c$
$c = ROTL^{30}(b)$
$b = a$
$a = T$

4. Cálculo del $i^{seavo}$ *valor Hash intermedio* $H^{(i)}$:

$H_0^{(i)} = a + H_0^{(i-1)}$
$H_1^{(i)} = b + H_1^{(i-1)}$
$H_2^{(i)} = c + H_2^{(i-1)}$
$H_3^{(i)} = d + H_3^{(i-1)}$
$H_4^{(i)} = e + H_4^{(i-1)}$

Después de repetir los pasos uno a cuatro un total de $N$ veces (después de procesar $M^{(N)}$), el mensaje digerido resultante de 160-bit del mensaje, $M$, es:

$$H_0^{(N)} \ // \ H_1^{(N)} \ // \ H_2^{(N)} \ // \ _{H3}^{(N)} \ // \ H_4^{(N)}$$

2.3.8 Ejemplo del uso del algoritmo SHA-1

Consideremos al mensaje, $M$, sea la cadena de 448-bit ($\mathbb{L}$ = 448):

"**abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq**".

El mensaje se complementa concatenando un bit “1”, seguido de 511 bits “0”, y terminando con el valor hexadecimal 00000000 000001c0 (la representación en dos palabras de 32-bit de la longitud, 448). Por lo tanto el mensaje final consiste de dos bloques (N = 2).

Para el algoritmo SHA-1 el valor Hash inicial, $H^{(0)}$, es:

$H_0^{(0)}$ = 67452301
$H_1^{(0)}$ = efcdab89
$H_2^{(0)}$ = 98badcfe
$H_3^{(0)}$ = 10325476
$H_4^{(0)}$ = c3d2e1f0.

Las palabras del primer bloque de mensaje, $M^{(1)}$, son asignadas a las palabras $W_0$, $W_1,\ldots,W_{14}$, $W_{15}$.

$W_0$ = 61626364
$W_1$ = 62636465

$W_2$ = 63646566
$W_3$ = 64656667
$W_4$ = 65666768
$W_5$ = 66676869
$W_6$ = 6768696a
$W_7$ = 68696a6b
$W_8$ = 696a6b6c
$W_9$ = 6a6b6c6d
$W_{10}$ = 6b6c6d6e
$W_{11}$ = 6c6d6e6f
$W_{12}$ = 6d6e6f70
$W_{13}$ = 6e6f7071
$W_{14}$ = 80000000
$W_{15}$ = 00000000.

La siguiente tabla muestra los valores hexadecimales de las variables *a, b, c, d* y *e,* después de la vuelta *t* del ciclo "*t* = 0 hasta *t* = 79"

| | ***a*** | ***b*** | ***c*** | ***d*** | ***e*** |
|---|---|---|---|---|---|
| *t* = 0: | 0116fc17 | 67452301 | 7bf36ae2 | 98badcfe | 10325476 |
| *t* = 1: | ebf3b452 | 0116fc17 | 59d148c0 | 7bf36ae2 | 98badcfe |
| *t* = 2: | 5109913a | ebf3b452 | c045bf05 | 59d148c0 | 7bf36ae2 |
| *t* = 3: | 2c4f6eac | 5109913a | bafced14 | c045bf05 | 59d148c0 |
| *t* = 4: | 33f4ae5b | 2c4f6eac | 9442644e | bafced14 | c045bf05 |
| *t* = 5: | 96b85189 | 33f4ae5b | 0b13dbab | 9442644e | bafced14 |
| *t* = 6: | db04cb58 | 96b85189 | ccfd2b96 | 0b13dbab | 9442644e |
| *t* = 7: | 45833f0f | db04cb58 | 65ae1462 | ccfd2b96 | 0b13dbab |
| *t* = 8: | c565c35e | 45833f0f | 36c132d6 | 65ae1462 | ccfd2b96 |
| *t* = 9: | 6350afda | c565c35e | d160cfc3 | 36c132d6 | 65ae1462 |
| *t* = 10: | 8993ea77 | 6350afda | b15970d7 | d160cfc3 | 36c132d6 |
| *t* = 11: | e19ecaa2 | 8993ea77 | 98d42bf6 | b15970d7 | d160cfc3 |
| *t* = 12: | 8603481e | e19ecaa2 | e264fa9d | 98d42bf6 | b15970d7 |
| *t* = 13: | 32f94a85 | 8603481e | b867b2a8 | e264fa9d | 98d42bf6 |
| *t* = 14: | b2e7a8be | 32f94a85 | a180d207 | b867b2a8 | e264fa9d |
| *t* = 15: | 42637e39 | b2e7a8be | 4cbe52a1 | a180d207 | b867b2a8 |
| *t* = 16: | 6b068048 | 42637e39 | acb9ea2f | 4cbe52a1 | a180d207 |
| *t* = 17: | 426b9c35 | 6b068048 | 5098df8e | acb9ea2f | 4cbe52a1 |
| *t* = 18: | 944b1bd1 | 426b9c35 | 1ac1a012 | 5098df8e | acb9ea2f |
| *t* = 19: | 6c445652 | 944b1bd1 | 509ae70d | 1ac1a012 | 5098df8e |
| *t* = 20: | 95836da5 | 6c445652 | 6512c6f4 | 509ae70d | 1ac1a012 |
| *t* = 21: | 09511177 | 95836da5 | 9b111594 | 6512c6f4 | 509ae70d |
| *t* = 22: | e2b92dc4 | 09511177 | 6560db69 | 9b111594 | 6512c6f4 |
| *t* = 23: | fd224575 | e2b92dc4 | c254445d | 6560db69 | 9b111594 |
| *t* = 24: | eeb82d9a | fd224575 | 38ae4b71 | c254445d | 6560db69 |
| *t* = 25: | 5a142c1a | eeb82d9a | 7f48915d | 38ae4b71 | c254445d |
| *t* = 26: | 2972f7c7 | 5a142c1a | bbae0b66 | 7f48915d | 38ae4b71 |
| *t* = 27: | d526a644 | 2972f7c7 | 96850b06 | bbae0b66 | 7f48915d |
| *t* = 28: | e1122421 | d526a644 | ca5cbdf1 | 96850b06 | bbae0b66 |

| | | | | | |
|---|---|---|---|---|---|
| *t* = 29: | 05b457b2 | e1122421 | 3549a991 | ca5cbdf1 | 96850b06 |
| *t* = 30: | a9c84bec | 05b457b2 | 78448908 | 3549a991 | ca5cbdf1 |
| *t* = 31: | 52e31f60 | a9c84bec | 816d15ec | 78448908 | 3549a991 |
| *t* = 32: | 5af3242c | 52e31f60 | 2a7212fb | 816d15ec | 78448908 |
| *t* = 33: | 31c756a9 | 5af3242c | 14b8c7d8 | 2a7212fb | 816d15ec |
| *t* = 34: | e9ac987c | 31c756a9 | 16bcc90b | 14b8c7d8 | 2a7212fb |
| *t* = 35: | ab7c32ee | e9ac987c | 4c71d5aa | 16bcc90b | 14b8c7d8 |
| *t* = 36: | 5933fc99 | ab7c32ee | 3a6b261f | 4c71d5aa | 16bcc90b |
| *t* = 37: | 43f87ae9 | 5933fc99 | aadf0cbb | 3a6b261f | 4c71d5aa |
| *t* = 38: | 24957f22 | 43f87ae9 | 564cff26 | aadf0cbb | 3a6b261f |
| *t* = 39: | adeb7478 | 24957f22 | 50fe1eba | 564cff26 | aadf0cbb |
| *t* = 40: | d70e5010 | adeb7478 | 89255fc8 | 50fe1eba | 564cff26 |
| *t* = 41: | 79bcfb08 | d70e5010 | 2b7add1e | 89255fc8 | 50fe1eba |
| *t* = 42: | f9bcb8de | 79bcfb08 | 35c39404 | 2b7add1e | 89255fc8 |
| *t* = 43: | 633e9561 | f9bcb8de | 1e6f3ec2 | 35c39404 | 2b7add1e |
| *t* = 44: | 98c1ea64 | 633e9561 | be6f2e37 | 1e6f3ec2 | 35c39404 |
| *t* = 45: | c6ea241e | 98c1ea64 | 58cfa558 | be6f2e37 | 1e6f3ec2 |
| *t* = 46: | a2ad4f02 | c6ea241e | 26307a99 | 58cfa558 | be6f2e37 |
| *t* = 47: | c8a69090 | a2ad4f02 | b1ba8907 | 26307a99 | 58cfa558 |
| *t* = 48: | 88341600 | c8a69090 | a8ab53c0 | b1ba8907 | 26307a99 |
| *t* = 49: | 7e846f58 | 88341600 | 3229a424 | a8ab53c0 | b1ba8907 |
| *t* = 50: | 86e358ba | 7e846f58 | 220d0580 | 3229a424 | a8ab53c0 |
| *t* = 51: | 8d2e76c8 | 86e358ba | 1fa11bd6 | 220d0580 | 3229a424 |
| *t* = 52: | ce892e10 | 8d2e76c8 | a1b8d62e | 1fa11bd6 | 220d0580 |
| *t* = 53: | edea95b1 | ce892e10 | 234b9db2 | a1b8d62e | 1fa11bd6 |
| *t* = 54: | 36d1230a | edea95b1 | 33a24b84 | 234b9db2 | a1b8d62e |
| *t* = 55: | 776c3910 | 36d1230a | 7b7aa56c | 33a24b84 | 234b9db2 |
| *t* = 56: | a681b723 | 776c3910 | 8db448c2 | 7b7aa56c | 33a24b84 |
| *t* = 57: | ac0a794f | a681b723 | 1ddb0e44 | 8db448c2 | 7b7aa56c |
| *t* = 58: | f03d3782 | ac0a794f | e9a06dc8 | 1ddb0e44 | 8db448c2 |
| *t* = 59: | 9ef775c3 | f03d3782 | eb029e53 | e9a06dc8 | 1ddb0e44 |
| *t* = 60: | 36254b13 | 9ef775c3 | bc0f4de0 | eb029e53 | e9a06dc8 |
| *t* = 61: | 4080d4dc | 36254b13 | e7bddd70 | bc0f4de0 | eb029e53 |
| *t* = 62: | 2bfaf7a8 | 4080d4dc | cd8952c4 | e7bddd70 | bc0f4de0 |
| *t* = 63: | 513f9ca0 | 2bfaf7a8 | 10203537 | cd8952c4 | e7bddd70 |
| *t* = 64: | e5895c81 | 513f9ca0 | 0afebdea | 10203537 | cd8952c4 |
| *t* = 65: | 1037d2d5 | e5895c81 | 144fe728 | 0afebdea | 10203537 |
| *t* = 66: | 14a82da9 | 1037d2d5 | 79625720 | 144fe728 | 0afebdea |
| *t* = 67: | 6d17c9fd | 14a82da9 | 440df4b5 | 79625720 | 144fe728 |
| *t* = 68: | 2c7b07bd | 6d17c9fd | 452a0b6a | 440df4b5 | 79625720 |
| *t* = 69: | fdf6efff | 2c7b07bd | 5b45f27f | 452a0b6a | 440df4b5 |
| *t* = 70: | 112b96e3 | fdf6efff | 4b1ec1ef | 5b45f27f | 452a0b6a |
| *t* = 71: | 84065712 | 112b96e3 | ff7dbbff | 4b1ec1ef | 5b45f27f |
| *t* = 72: | ab89fb71 | 84065712 | c44ae5b8 | ff7dbbff | 4b1ec1ef |
| *t* = 73: | c5210e35 | ab89fb71 | a10195c4 | c44ae5b8 | ff7dbbff |
| *t* = 74: | 352d9f4b | c5210e35 | 6ae27edc | a10195c4 | c44ae5b8 |
| *t* = 75: | 1a0e0e0a | 352d9f4b | 7148438d | 6ae27edc | a10195c4 |
| *t* = 76: | d0d47349 | 1a0e0e0a | cd4b67d2 | 7148438d | 6ae27edc |

| | | | | | |
|---|---|---|---|---|---|
| *t* = 77: | ad38620d | d0d47349 | 86838382 | cd4b67d2 | 7148438d |
| *t* = 78: | d3ad7c25 | ad38620d | 74351cd2 | 86838382 | cd4b67d2 |
| *t* = 79: | 8ce34517 | d3ad7c25 | 6b4e1883 | 74351cd2 | 86838382 |

Eso completa el procesamiento del primer bloque de mensaje $M^{(1)}$. El primer valor Hash intermedio, $H^{(1)}$, calculado es:

$H_0^{(1)}$ = 67452301 + 8ce34517 = f4286818
$H_1^{(1)}$ = efcdab89 + d3ad7c25 = c37b27ae
$H_2^{(1)}$ = 98badcfe + 6b4e1883 = 0408f581
$H_3^{(1)}$ = 10325476 + 74351cd2 = 84677148
$H_4^{(1)}$ = c3d2e1f0 + 86838382 = 4a566572.

Las palabras del segundo bloque de mensaje, $M^{(2)}$, son asignadas a las palabras $W_0$, $W_1$,…,$W_{14}$, $W_{15}$.

$W_0$ = 00000000
$W_1$ = 00000000
$W_2$ = 00000000
$W_3$ = 00000000
$W_4$ = 00000000
$W_5$ = 00000000
$W_6$ = 00000000
$W_7$ = 00000000
$W_8$ = 00000000
$W_9$ = 00000000
$W_{10}$ = 00000000
$W_{11}$ = 00000000
$W_{12}$ = 00000000
$W_{13}$ = 00000000
$W_{14}$ = 00000000
$W_{15}$ = 000001c0.

La siguiente tabla muestra los valores hexadecimales de las variables *a, b, c, d* y *e* después de la vuelta *t* del ciclo "*t* = 0 hasta *t* = 79"

| | ***a*** | ***b*** | ***c*** | ***d*** | ***e*** |
|---|---|---|---|---|---|
| *t* = 0: | 2df257e9 | f4286818 | b0dec9eb | 0408f581 | 84677148 |
| *t* = 1: | 4d3dc58f | 2df257e9 | 3d0a1a06 | b0dec9eb | 0408f581 |
| *t* = 2: | c352bb05 | 4d3dc58f | 4b7c95fa | 3d0a1a06 | b0dec9eb |
| *t* = 3: | eef743c6 | c352bb05 | d34f7163 | 4b7c95fa | 3d0a1a06 |
| *t* = 4: | 41e34277 | eef743c6 | 70d4aec1 | d34f7163 | 4b7c95fa |
| *t* = 5: | 5443915c | 41e34277 | bbbdd0f1 | 70d4aec1 | d34f7163 |
| *t* = 6: | e7fa0377 | 5443915c | d078d09d | bbbdd0f1 | 70d4aec1 |
| *t* = 7: | c6946813 | e7fa0377 | 1510e457 | d078d09d | bbbdd0f1 |
| *t* = 8: | fdde1de1 | c6946813 | f9fe80dd | 1510e457 | d078d09d |
| *t* = 9: | b8538aca | fdde1de1 | f1a51a04 | f9fe80dd | 1510e457 |
| *t* = 10: | 6ba94f63 | b8538aca | 7f778778 | f1a51a04 | f9fe80dd |

| | | | | | |
|---|---|---|---|---|---|
| *t* = 11: | 43a2792f | 6ba94f63 | ae14e2b2 | 7f778778 | f1a51a04 |
| *t* = 12: | fecd7bbf | 43a2792f | daea53d8 | ae14e2b2 | 7f778778 |
| *t* = 13: | a2604ca8 | fecd7bbf | d0e89e4b | daea53d8 | ae14e2b2 |
| *t* = 14: | 258b0baa | a2604ca8 | ffb35eef | d0e89e4b | daea53d8 |
| *t* = 15: | d9772360 | 258b0baa | 2898132a | ffb35eef | d0e89e4b |
| *t* = 16: | 5507db6e | d9772360 | 8962c2ea | 2898132a | ffb35eef |
| *t* = 17: | a51b58bc | 5507db6e | 365dc8d8 | 8962c2ea | 2898132a |
| *t* = 18: | c2eb709f | a51b58bc | 9541f6db | 365dc8d8 | 8962c2ea |
| *t* = 19: | d8992153 | c2eb709f | 2946d62f | 9541f6db | 365dc8d8 |
| *t* = 20: | 37482f5f | d8992153 | f0badc27 | 2946d62f | 9541f6db |
| *t* = 21: | ee8700bd | 37482f5f | f6264854 | f0badc27 | 2946d62f |
| *t* = 22: | 9ad594b9 | ee8700bd | cdd20bd7 | f6264854 | f0badc27 |
| *t* = 23: | 8fbaa5b9 | 9ad594b9 | 7ba1c02f | cdd20bd7 | f6264854 |
| *t* = 24: | 88fb5867 | 8fbaa5b9 | 66b5652e | 7ba1c02f | cdd20bd7 |
| *t* = 25: | eec50521 | 88fb5867 | 63eea96e | 66b5652e | 7ba1c02f |
| *t* = 26: | 50bce434 | eec50521 | e23ed619 | 63eea96e | 66b5652e |
| *t* = 27: | 5c416daf | 50bce434 | 7bb14148 | e23ed619 | 63eea96e |
| *t* = 28: | 2429be5f | 5c416daf | 142f390d | 7bb14148 | e23ed619 |
| *t* = 29: | 0a2fb108 | 2429be5f | d7105b6b | 142f390d | 7bb14148 |
| *t* = 30: | 17986223 | 0a2fb108 | c90a6f97 | d7105b6b | 142f390d |
| *t* = 31: | 8a4af384 | 17986223 | 028bec42 | c90a6f97 | d7105b6b |
| *t* = 32: | 6b629993 | 8a4af384 | c5e61888 | 028bec42 | c90a6f97 |
| *t* = 33: | f15f04f3 | 6b629993 | 2292bce1 | c5e61888 | 028bec42 |
| *t* = 34: | 295cc25b | f15f04f3 | dad8a664 | 2292bce1 | c5e61888 |
| *t* = 35: | 696da404 | 295cc25b | fc57c13c | dad8a664 | 2292bce1 |
| *t* = 36: | cef5ae12 | 696da404 | ca573096 | fc57c13c | dad8a664 |
| *t* = 37: | 87d5b80c | cef5ae12 | 1a5b6901 | ca573096 | fc57c13c |
| *t* = 38: | 84e2a5f2 | 87d5b80c | b3bd6b84 | 1a5b6901 | ca573096 |
| *t* = 39: | 03bb6310 | 84e2a5f2 | 21f56e03 | b3bd6b84 | 1a5b6901 |
| *t* = 40: | c2d8f75f | 03bb6310 | a138a97c | 21f56e03 | b3bd6b84 |
| *t* = 41: | bfb25768 | c2d8f75f | 00eed8c4 | a138a97c | 21f56e03 |
| *t* = 42: | 28589152 | bfb25768 | f0b63dd7 | 00eed8c4 | a138a97c |
| *t* = 43: | ec1d3d61 | 28589152 | 2fec95da | f0b63dd7 | 00eed8c4 |
| *t* = 44: | 3caed7af | ec1d3d61 | 8a162454 | 2fec95da | f0b63dd7 |
| *t* = 45: | c3d033ea | 3caed7af | 7b074f58 | 8a162454 | 2fec95da |
| *t* = 46: | 7316056a | c3d033ea | cf2bb5eb | 7b074f58 | 8a162454 |
| *t* = 47: | 46f93b68 | 7316056a | b0f40cfa | cf2bb5eb | 7b074f58 |
| *t* = 48: | dc8e7f26 | 46f93b68 | 9cc5815a | b0f40cfa | cf2bb5eb |
| *t* = 49: | 850d411c | dc8e7f26 | 11be4eda | 9cc5815a | b0f40cfa |
| *t* = 50: | 7e4672c0 | 850d411c | b7239fc9 | 11be4eda | 9cc5815a |
| *t* = 51: | 89fbd41d | 7e4672c0 | 21435047 | b7239fc9 | 11be4eda |
| *t* = 52: | 1797e228 | 89fbd41d | 1f919cb0 | 21435047 | b7239fc9 |
| *t* = 53: | 431d65bc | 1797e228 | 627ef507 | 1f919cb0 | 21435047 |
| *t* = 54: | 2bdbb8cb | 431d65bc | 05e5f88a | 627ef507 | 1f919cb0 |
| *t* = 55: | 6da72e7f | 2bdbb8cb | 10c7596f | 05e5f88a | 627ef507 |
| *t* = 56: | a8495a9b | 6da72e7f | caf6ee32 | 10c7596f | 05e5f88a |
| *t* = 57: | e785655a | a8495a9b | db69cb9f | caf6ee32 | 10c7596f |
| *t* = 58: | 5b086c42 | e785655a | ea1256a6 | db69cb9f | caf6ee32 |

| | | | | | |
|---|---|---|---|---|---|
| *t* = 59: | a65818f7 | 5b086c42 | b9e15956 | ea1256a6 | db69cb9f |
| *t* = 60: | 7aab101b | a65818f7 | 96c21b10 | b9e15956 | ea1256a6 |
| *t* = 61: | 93614c9c | 7aab101b | e996063d | 96c21b10 | b9e15956 |
| *t* = 62: | f66d9bf4 | 93614c9c | deaac406 | e996063d | 96c21b10 |
| *t* = 63: | d504902b | f66d9bf4 | 24d85327 | deaac406 | e996063d |
| *t* = 64: | 60a9da62 | d504902b | 3d9b66fd | 24d85327 | deaac406 |
| *t* = 65: | 8b687819 | 60a9da62 | f541240a | 3d9b66fd | 24d85327 |
| *t* = 66: | 083e90c3 | 8b687819 | 982a7698 | f541240a | 3d9b66fd |
| *t* = 67: | f6226bbf | 083e90c3 | 62da1e06 | 982a7698 | f541240a |
| *t* = 68: | 76c0563b | f6226bbf | c20fa430 | 62da1e06 | 982a7698 |
| *t* = 69: | 989dd165 | 76c0563b | fd889aef | c20fa430 | 62da1e06 |
| *t* = 70: | 8b2c7573 | 989dd165 | ddb0158e | fd889aef | c20fa430 |
| *t* = 71: | ae1b8e7b | 8b2c7573 | 66277459 | ddb0158e | fd889aef |
| *t* = 72: | ca1840de | ae1b8e7b | e2cb1d5c | 66277459 | ddb0158e |
| *t* = 73: | 16f3babb | ca1840de | eb86e39e | e2cb1d5c | 66277459 |
| *t* = 74: | d28d83ad | 16f3babb | b2861037 | eb86e39e | e2cb1d5c |
| *t* = 75: | 6bc02dfe | d28d83ad | c5bceeae | b2861037 | eb86e39e |
| *t* = 76: | d3a6e275 | 6bc02dfe | 74a360eb | c5bceeae | b2861037 |
| *t* = 77: | da955482 | d3a6e275 | 9af00b7f | 74a360eb | c5bceeae |
| *t* = 78: | 58c0aac0 | da955482 | 74e9b89d | 9af00b7f | 74a360eb |
| *t* = 79: | 906fd62c | 58c0aac0 | b6a55520 | 74e9b89d | 9af00b7f |

Eso completa el procesamiento del segundo bloque de mensaje $M^{(2)}$. El valor Hash ***final***, $H^{(2)}$, calculado es:

$H_0^{(2)}$ = f4286818 + 906fd62c = 84983e44
$H_1^{(2)}$ = c37b27ae + 58c0aac0 = 1c3bd26e
$H_2^{(2)}$ = 0408f581 + b6a55520 = baae4aa1
$H_3^{(2)}$ = 84677148 + 74e9b89d = f95129e5
$H_4^{(2)}$ = 4a566572 + 9af00b7f = e54670f1

El mensaje digerido resultante es:

84983e44 1c3bd26e baae4aa1 f95129e5 e54670f1

2.3.9 Ejemplo del código del algoritmo SHA-1 en lenguaje C

sha1.h

```
#ifndef _SHA1_H
#define _SHA1_H

#ifndef uint8
#define uint8  unsigned char
#endif

#ifndef uint32
#define uint32 unsigned long int
#endif
```

```
typedef struct
{
    uint32 total[2];
    uint32 state[5];
    uint8 buffer[64];
}
sha1_context;

void sha1_starts( sha1_context *ctx );
void sha1_update( sha1_context *ctx, uint8 *input, uint32 length );
void sha1_finish( sha1_context *ctx, uint8 digest[20] );

#endif /* sha1.h */
```

sha1.c

```
/*
 *  FIPS-180-1 compliant SHA-1 implementation
 *
 *  Copyright (C) 2001-2003  Christophe Devine
 *
 *  This program is free software; you can redistribute it and/or
modify
 *  it under the terms of the GNU General Public License as published
by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 */

#include <string.h>

#include "sha1.h"

#define GET_UINT32(n,b,i)                       \
{                                               \
    (n) = ( (uint32) (b)[(i)    ] << 24 )       \
        | ( (uint32) (b)[(i) + 1] << 16 )       \
        | ( (uint32) (b)[(i) + 2] <<  8 )       \
        | ( (uint32) (b)[(i) + 3]       );      \
```

```
}

#define PUT_UINT32(n,b,i)                       \
{                                               \
    (b)[(i)    ] = (uint8) ( (n) >> 24 );       \
    (b)[(i) + 1] = (uint8) ( (n) >> 16 );       \
    (b)[(i) + 2] = (uint8) ( (n) >>  8 );       \
    (b)[(i) + 3] = (uint8) ( (n)       );       \
}

void sha1_starts( sha1_context *ctx )
{
    ctx->total[0] = 0;
    ctx->total[1] = 0;

    ctx->state[0] = 0x67452301;
    ctx->state[1] = 0xEFCDAB89;
    ctx->state[2] = 0x98BADCFE;
    ctx->state[3] = 0x10325476;
    ctx->state[4] = 0xC3D2E1F0;
}

void sha1_process( sha1_context *ctx, uint8 data[64] )
{
    uint32 temp, W[16], A, B, C, D, E;

    GET_UINT32( W[0],  data,  0 );
    GET_UINT32( W[1],  data,  4 );
    GET_UINT32( W[2],  data,  8 );
    GET_UINT32( W[3],  data, 12 );
    GET_UINT32( W[4],  data, 16 );
    GET_UINT32( W[5],  data, 20 );
    GET_UINT32( W[6],  data, 24 );
    GET_UINT32( W[7],  data, 28 );
    GET_UINT32( W[8],  data, 32 );
    GET_UINT32( W[9],  data, 36 );
    GET_UINT32( W[10], data, 40 );
    GET_UINT32( W[11], data, 44 );
    GET_UINT32( W[12], data, 48 );
    GET_UINT32( W[13], data, 52 );
    GET_UINT32( W[14], data, 56 );
    GET_UINT32( W[15], data, 60 );

#define S(x,n) ((x << n) | ((x & 0xFFFFFFFF) >> (32 - n)))

#define R(t)                                            \
(                                                       \
    temp = W[(t -  3) & 0x0F] ^ W[(t - 8) & 0x0F] ^     \
           W[(t - 14) & 0x0F] ^ W[ t      & 0x0F],      \
```

```
    ( W[t & 0x0F] = S(temp,1) )                          \
)

#define P(a,b,c,d,e,x)                                   \
{                                                        \
    e += S(a,5) + F(b,c,d) + K + x; b = S(b,30);         \
}

    A = ctx->state[0];
    B = ctx->state[1];
    C = ctx->state[2];
    D = ctx->state[3];
    E = ctx->state[4];

#define F(x,y,z) (z ^ (x & (y ^ z)))
#define K 0x5A827999

    P( A, B, C, D, E, W[0]  );
    P( E, A, B, C, D, W[1]  );
    P( D, E, A, B, C, W[2]  );
    P( C, D, E, A, B, W[3]  );
    P( B, C, D, E, A, W[4]  );
    P( A, B, C, D, E, W[5]  );
    P( E, A, B, C, D, W[6]  );
    P( D, E, A, B, C, W[7]  );
    P( C, D, E, A, B, W[8]  );
    P( B, C, D, E, A, W[9]  );
    P( A, B, C, D, E, W[10] );
    P( E, A, B, C, D, W[11] );
    P( D, E, A, B, C, W[12] );
    P( C, D, E, A, B, W[13] );
    P( B, C, D, E, A, W[14] );
    P( A, B, C, D, E, W[15] );
    P( E, A, B, C, D, R(16) );
    P( D, E, A, B, C, R(17) );
    P( C, D, E, A, B, R(18) );
    P( B, C, D, E, A, R(19) );

#undef K
#undef F

#define F(x,y,z) (x ^ y ^ z)
#define K 0x6ED9EBA1

    P( A, B, C, D, E, R(20) );
    P( E, A, B, C, D, R(21) );
    P( D, E, A, B, C, R(22) );
    P( C, D, E, A, B, R(23) );
    P( B, C, D, E, A, R(24) );
```

```
    P( A, B, C, D, E, R(25) );
    P( E, A, B, C, D, R(26) );
    P( D, E, A, B, C, R(27) );
    P( C, D, E, A, B, R(28) );
    P( B, C, D, E, A, R(29) );
    P( A, B, C, D, E, R(30) );
    P( E, A, B, C, D, R(31) );
    P( D, E, A, B, C, R(32) );
    P( C, D, E, A, B, R(33) );
    P( B, C, D, E, A, R(34) );
    P( A, B, C, D, E, R(35) );
    P( E, A, B, C, D, R(36) );
    P( D, E, A, B, C, R(37) );
    P( C, D, E, A, B, R(38) );
    P( B, C, D, E, A, R(39) );

#undef K
#undef F

#define F(x,y,z) ((x & y) | (z & (x | y)))
#define K 0x8F1BBCDC

    P( A, B, C, D, E, R(40) );
    P( E, A, B, C, D, R(41) );
    P( D, E, A, B, C, R(42) );
    P( C, D, E, A, B, R(43) );
    P( B, C, D, E, A, R(44) );
    P( A, B, C, D, E, R(45) );
    P( E, A, B, C, D, R(46) );
    P( D, E, A, B, C, R(47) );
    P( C, D, E, A, B, R(48) );
    P( B, C, D, E, A, R(49) );
    P( A, B, C, D, E, R(50) );
    P( E, A, B, C, D, R(51) );
    P( D, E, A, B, C, R(52) );
    P( C, D, E, A, B, R(53) );
    P( B, C, D, E, A, R(54) );
    P( A, B, C, D, E, R(55) );
    P( E, A, B, C, D, R(56) );
    P( D, E, A, B, C, R(57) );
    P( C, D, E, A, B, R(58) );
    P( B, C, D, E, A, R(59) );

#undef K
#undef F

#define F(x,y,z) (x ^ y ^ z)
#define K 0xCA62C1D6
```

```
    P( A, B, C, D, E, R(60) );
    P( E, A, B, C, D, R(61) );
    P( D, E, A, B, C, R(62) );
    P( C, D, E, A, B, R(63) );
    P( B, C, D, E, A, R(64) );
    P( A, B, C, D, E, R(65) );
    P( E, A, B, C, D, R(66) );
    P( D, E, A, B, C, R(67) );
    P( C, D, E, A, B, R(68) );
    P( B, C, D, E, A, R(69) );
    P( A, B, C, D, E, R(70) );
    P( E, A, B, C, D, R(71) );
    P( D, E, A, B, C, R(72) );
    P( C, D, E, A, B, R(73) );
    P( B, C, D, E, A, R(74) );
    P( A, B, C, D, E, R(75) );
    P( E, A, B, C, D, R(76) );
    P( D, E, A, B, C, R(77) );
    P( C, D, E, A, B, R(78) );
    P( B, C, D, E, A, R(79) );

#undef K
#undef F

    ctx->state[0] += A;
    ctx->state[1] += B;
    ctx->state[2] += C;
    ctx->state[3] += D;
    ctx->state[4] += E;
}

void sha1_update( sha1_context *ctx, uint8 *input, uint32 length )
{
    uint32 left, fill;

    if( ! length ) return;

    left = ctx->total[0] & 0x3F;
    fill = 64 - left;

    ctx->total[0] += length;
    ctx->total[0] &= 0xFFFFFFFF;

    if( ctx->total[0] < length )
        ctx->total[1]++;

    if( left && length >= fill )
    {
        memcpy( (void *) (ctx->buffer + left),
```

```
                (void *) input, fill );
        sha1_process( ctx, ctx->buffer );
        length -= fill;
        input  += fill;
        left = 0;
    }

    while( length >= 64 )
    {
        sha1_process( ctx, input );
        length -= 64;
        input  += 64;
    }

    if( length )
    {
        memcpy( (void *) (ctx->buffer + left),
                (void *) input, length );
    }
}

static uint8 sha1_padding[64] =
{
 0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};

void sha1_finish( sha1_context *ctx, uint8 digest[20] )
{
    uint32 last, padn;
    uint32 high, low;
    uint8 msglen[8];

    high = ( ctx->total[0] >> 29 )
         | ( ctx->total[1] <<  3 );
    low  = ( ctx->total[0] <<  3 );

    PUT_UINT32( high, msglen, 0 );
    PUT_UINT32( low,  msglen, 4 );

    last = ctx->total[0] & 0x3F;
    padn = ( last < 56 ) ? ( 56 - last ) : ( 120 - last );

    sha1_update( ctx, sha1_padding, padn );
    sha1_update( ctx, msglen, 8 );

    PUT_UINT32( ctx->state[0], digest,  0 );
```

```
    PUT_UINT32( ctx->state[1], digest,  4 );
    PUT_UINT32( ctx->state[2], digest,  8 );
    PUT_UINT32( ctx->state[3], digest, 12 );
    PUT_UINT32( ctx->state[4], digest, 16 );
}

#ifdef TEST

#include <stdlib.h>
#include <stdio.h>

/*
 * those are the standard FIPS-180-1 test vectors
 */

static char *msg[] =
{
    "abc",
    "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq",
    NULL
};

static char *val[] =
{
    "a9993e364706816aba3e25717850c26c9cd0d89d",
    "84983e441c3bd26ebaae4aa1f95129e5e54670f1",
    "34aa973cd4c4daa4f61eeb2bdbad27316534016f"
};

int main( int argc, char *argv[] )
{
    FILE *f;
    int i, j;
    char output[41];
    sha1_context ctx;
    unsigned char buf[1000];
    unsigned char sha1sum[20];

    if( argc < 2 )
    {
        printf( "\n SHA-1 Validation Tests:\n\n" );

        for( i = 0; i < 3; i++ )
        {
            printf( " Test %d ", i + 1 );

            sha1_starts( &ctx );

            if( i < 2 )
```

```
        {
            sha1_update( &ctx, (uint8 *) msg[i],
                         strlen( msg[i] ) );
        }
        else
        {
            memset( buf, 'a', 1000 );

            for( j = 0; j < 1000; j++ )
            {
                sha1_update( &ctx, (uint8 *) buf, 1000 );
            }
        }

        sha1_finish( &ctx, sha1sum );

        for( j = 0; j < 20; j++ )
        {
            sprintf( output + j * 2, "%02x", sha1sum[j] );
        }

        if( memcmp( output, val[i], 40 ) )
        {
            printf( "failed!\n" );
            return( 1 );
        }

        printf( "passed.\n" );
    }

    printf( "\n" );
}
else
{
    if( ! ( f = fopen( argv[1], "rb" ) ) )
    {
        perror( "fopen" );
        return( 1 );
    }

    sha1_starts( &ctx );

    while( ( i = fread( buf, 1, sizeof( buf ), f ) ) > 0 )
    {
        sha1_update( &ctx, buf, i );
    }

    sha1_finish( &ctx, sha1sum );
```

```
        for( j = 0; j < 20; j++ )
        {
            printf( "%02x", sha1sum[j] );
        }

        printf( "  %s\n", argv[1] );
    }

    return( 0 );
}

#endif
```

Por medio del algoritmo ya descrito se realiza la operación para crear las claves del sistema. Este proceso se describe en el siguiente capítulo.

## Capítulo 3 Descripción y operaciones del sistema

### 3.1 Elementos del Sistema

El sistema consta en su forma más básica de los elementos descritos en la figura 5 del capítulo 1.

a. Unidad Central
b. Terminales del sistema
c. Llave
d. Interfaz de control

#### 3.1.1 Unidad Central

Es la entidad encargada de recibir y validar las claves de acceso almacenadas en las llaves y que son ingresadas por medio de las Terminales del Sistema, indicando a estas el resultado de la validación y en su caso generar una nueva clave; también almacena en su memoria principal los valores de las claves generadas para los usuarios ingresados por medio de la Interfaz de control; por último lleva a cabo el registro en bitácora, que es una memoria dentro de la Unidad Central, de los eventos que se presenten. La estructura propuesta para la Unidad Central se muestra en la Figura 6.

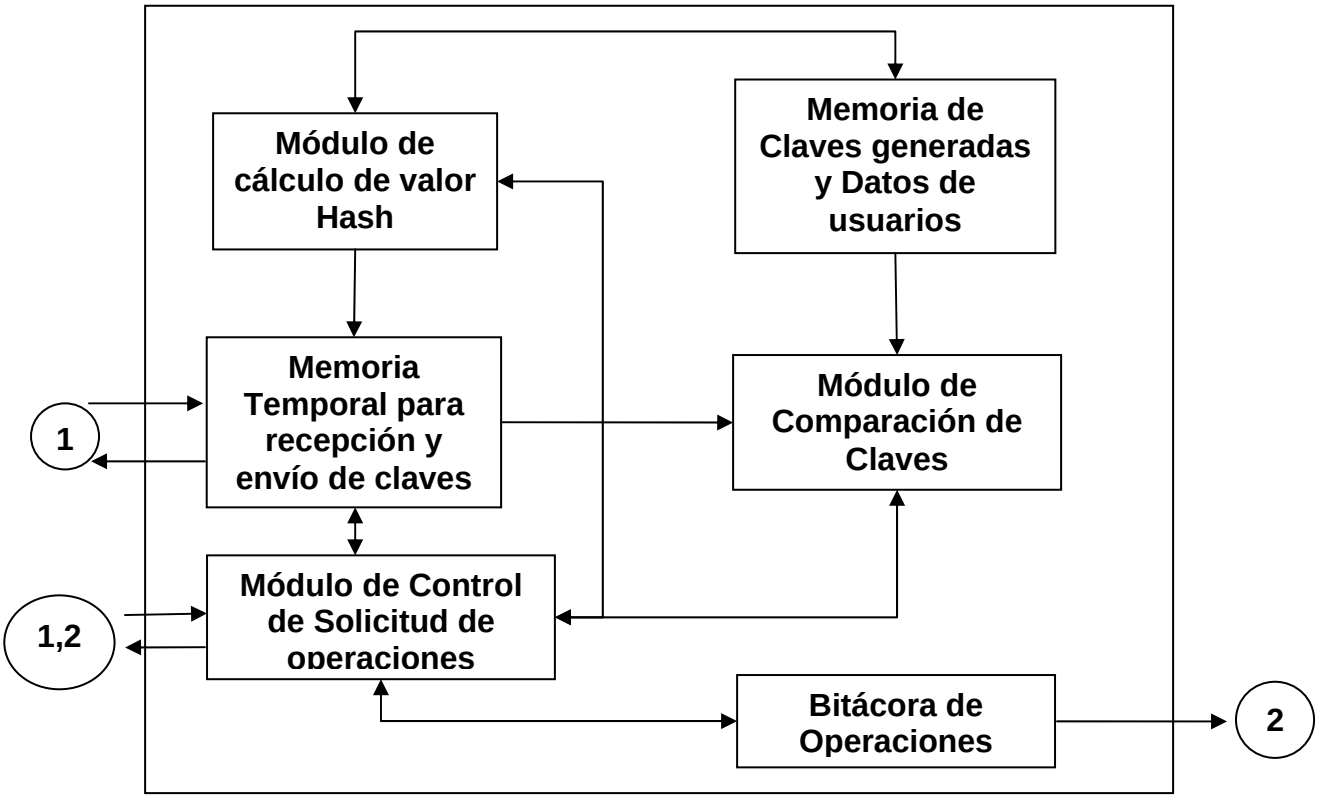


Figura 6. Estructura de la Unidad Central

Las conexiones con el número “1” enlazan a la Unidad Central con las Terminales de Sistema. Las conexiones con el número “2” enlazan a la Unidad Central con la Interfaz de Control.

3.1.2 Terminales de sistema

Se encuentra en cada uno de los accesos que regula el sistema su función primaria es la lectura de las claves almacenadas en las Llaves, transmitir estas claves a la Unidad Central y reaccionar a la validación que este último realice, ya sea realizando una acción de alerta ó almacenando en la llave la nueva clave generada por la Unidad Central; así mismo en base a su programación concederá o negará el acceso. La estructura de las terminales de sistema se describe en la Figura 7.

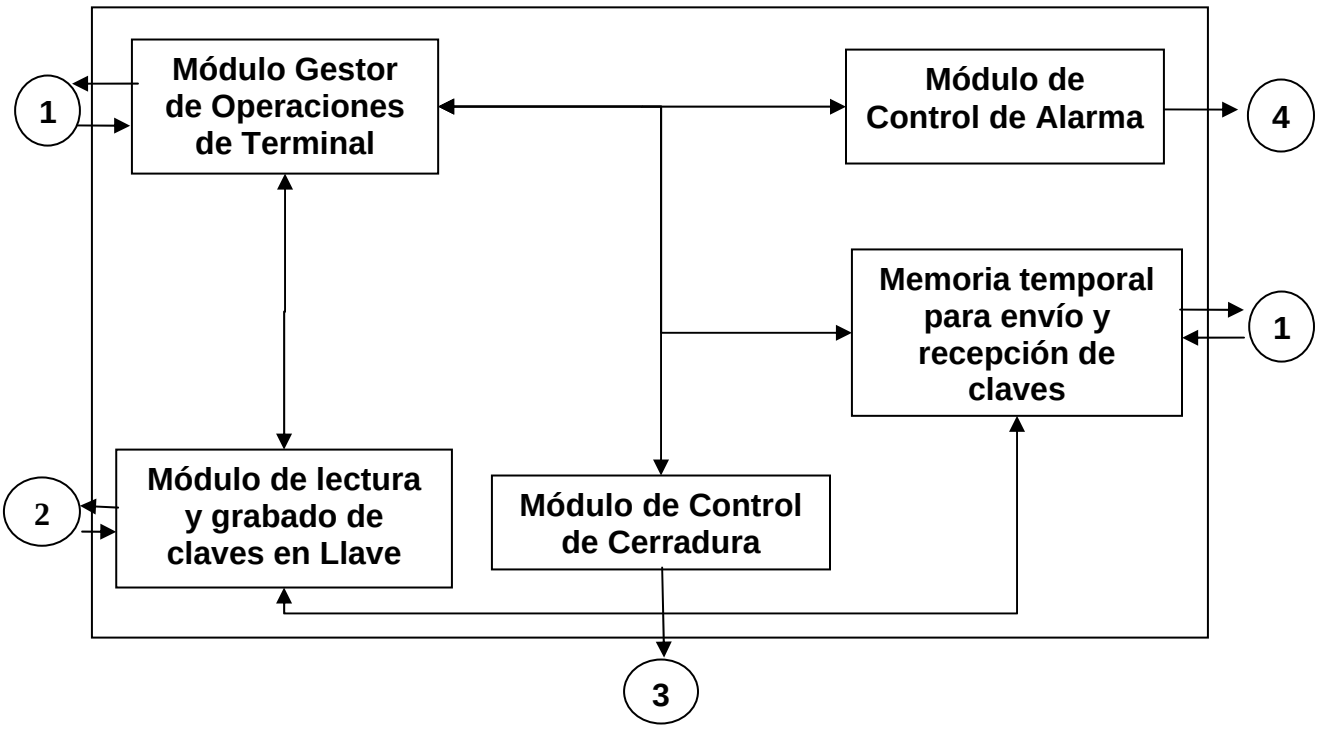


Figura 7. Estructura de las Terminales de Sistema

Las conexiones con el número “1” enlazan a la Terminal de Sistema con la Unidad Central. La conexión con el número “2” enlaza a la Terminal de Sistema con la Llave. La conexión con el número “3” enlaza a la Terminal de Sistema con el dispositivo de apertura de cerradura. La conexión con el número “4” enlaza a la Terminal de Sistema con el dispositivo de alarma si este se encuentra presente.

3.1.3 Llave

Es la unidad de almacenamiento para las claves generadas por la Unidad Central, solo interactúan con esté directamente en las funciones de gestión de usuarios; interactúan indirectamente con la Unidad Central por medio de las Terminales de sistema cuando se realizan las operaciones de Lectura de claves y generación de claves. Es una memoria de tipo no volátil con capacidad limitada.

3.1.4 Interfaz de control

Por medio de este dispositivo se realizan las operaciones de gestión de usuarios y la solicitud de bitácoras del sistema. Se complementa con una computadora personal en la cual por medio de una interfaz gráfica y la conexión de un puerto se administra el funcionamiento de la Unidad Central, así también se solicita la información de los eventos que éste ha generado. La misma conexión de puerto cuenta con la función de grabar en la Llave la clave generada por una operación de alta de usuario. Como se describe en la Figura 8.

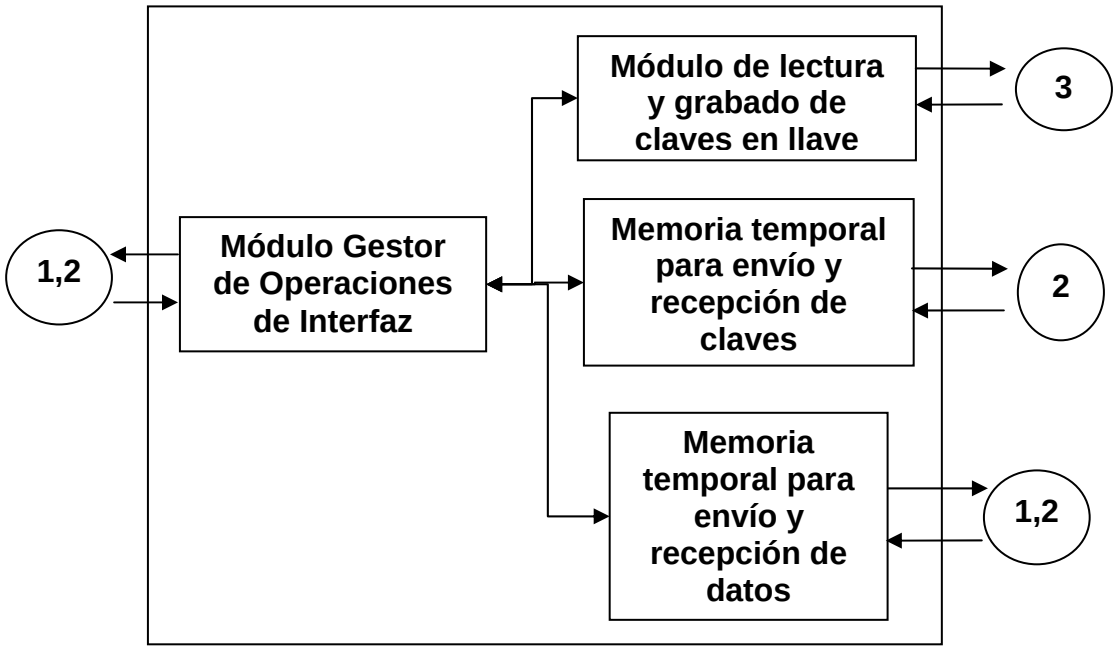


Figura 8. Estructura de la Interfaz de control

Las conexiones con el número “1” enlazan a la Interfaz de Control con la computadora en la que se instala la interfaz gráfica. Las conexiones con el número “2” enlazan a la Interfaz de Control con la Unidad Central. La conexión con el número “3” enlaza a la Interfaz de Control con la Llave.

3.2 Funciones del Sistema

El sistema lleva a cabo las siguientes funciones:

a. Generación de claves
b. Lectura de claves
c. Validación de las claves
d. Acciones de acceso
e. Acciones de alerta
f. Monitoreo de las terminales
g. Registro de Bitácora

h. Gestión de usuarios

3.2.1 Generación de claves

Esta función se lleva a cabo en dos momentos de operación del sistema en primer lugar cuando se da de alta un nuevo usuario, y se le asigna una llave que almacenara la clave correspondiente al usuario, también en cada acceso válido que se ejecute en una Terminal de sistema.

3.2.1.1Elementos para generar claves

1. Nombre del Usuario
2. Apellido Paterno del Usuario
3. Apellido Materno del Usuario
4. Edad del Usuario
5. Departamento al que pertenece el Usuario
6. Hora en la que se realiza la operación
7. Número de llave

3.2.1.2 Proceso para generar una clave

a. El sistema toma al azar tres de los primeros cinco elementos para generar claves y los concatena con el sexto.

b. Esta cadena se somete al algoritmo de digestión SHA-1 (véase capitulo 2), resultando en un valor Hash de 160-bit.

Este valor Hash es almacenado en la Unidad Central, relacionándolo con el Número de llave. Así mismo será enviado a una Terminal de Sistema concatenando primero el Numero de Llave, siempre y cuando esta clave se haya generado por una operación de Lectura y de Validación, la única excepción a esto es cuando se realicen operaciones de Gestión de Usuarios en la Interfaz de Control. Por último se agregará este evento a la bitácora refiriendo el Número de llave para el que se creo una clave, pero no se agregara el valor Hash de la clave.

3.2.2 Lectura de claves

La función de Lectura de claves es realizada por las Terminales del Sistema, almacenada temporalmente en estas para ser enviada a la Unidad Central y que sea validada por ésta.

3.2.3 Validación de las claves

La Unidad Central realiza esta función, con la cual compara la clave enviada tras una función de Lectura de claves en una Terminal de Sistema con la que se tiene almacenada.

De encontrarse que son idénticas se realizarán las siguientes funciones:

1. Se generará una nueva clave de acuerdo al procedimiento antes descrito

2. Se Indicará a la Terminal que autorice el acceso
3. Se agregará el evento de Generación de clave y autorización de acceso en la bitácora de sistema.

En caso de que la clave enviada por la terminal y la clave almacenada sea diferente en cualquier medida se realizarán las siguientes funciones:

1. Agregar el evento a bitácora
2. En caso de que cuente con un dispositivo de alarma centralizado activarlo
3. Enviar una señal de alerta a la Terminal en la que se solicitó el acceso.

3.2.4 Acciones de Acceso

Esta función cubre dos aspectos, que la Terminal de Sistema almacene en la llave del usuario que solicitó el acceso y que fue validado aprobatoriamente la nueva clave que envió la Unidad Central y que permita el acceso al área en la que se solicitó.

3.2.5 Acciones de Alerta

El sistema es capaz de comportarse de manera diferente en cada Terminal de Sistema cuando se presenta una situación de alerta, gracias a que estas acciones no son controladas por la Unidad Central, cada Terminal de Sistema es programada individualmente y esto será lo que determine su proceder cuando la Unidad Central le manda una señal de alerta. Estas acciones son:

1. Negar el acceso y activar la alarma de la Terminal
2. Permitir el acceso, borrar la clave almacenada en la llave sin activar la alarma de la Terminal.

El criterio para imponer cualquiera de estas dos acciones es determinado por las políticas que el propietario del sistema desee imponer de acuerdo a lo sensible que considere sus bienes.

3.2.6 Monitoreo de terminales

El sistema central mantiene una conexión directa con las terminales del sistema en caso de que una terminal deje de funcionar la Unidad Central detecta esta situación y activa el dispositivo de alarma centralizado.

3.2.7 Registro de Bitácora

Cada vez que una Terminal de Sistema o la Interfaz de Control interactúan con la Unidad Central ésta va almacenando en su memoria los eventos hasta que la Interfaz de Control le solicita esta bitácora, de esta manera no se satura la memoria de la unidad central.

3.2.8 Gestión de Usuarios

Las funciones de Gestión de Usuarios son tres y se realizan en la Interfaz de Control, estas

son:

1. Alta de usuarios, se ingresan en el sistema los cinco datos del usuario y se genera una clave para este usuario, por último se almacena esta Clave en una Llave.
2. Cambio de los datos de usuario, por esta función se actualizan los datos de un usuario, ya sea su edad o el departamento al que esta adscrito.
3. Baja de usuarios, con esta acción se borra el registro completo del usuario lo que incluye sus datos personales y la Clave que haya estado asociada en ese momento.

Estas son las operaciones regulares del sistema propuesto, en el siguiente capítulo se describe las posibles situaciones que pueden poner en riesgo el correcto funcionamiento del sistema.

## Capítulo 4 Posibles escenarios para el sistema

Por su naturaleza un sistema de control de acceso está sometido no solo a ataques directos sino también indirectos y situaciones de emergencia, a continuación se plantean algunos de estos sucesos y la forma en la que el sistema responderá a ellos.

### 4.1 Falla en la energía eléctrica

El sistema provee de energía eléctrica a cada cerradura, y este puede estar conectado a un sistema de respaldo el cual garantice su funcionamiento por un periodo prolongado de tiempo. En caso de que se suspenda totalmente el suministro eléctrico cada Terminal se puede configurar de manera que se queden abiertas o se cierren hasta que se restaure la energía.

### 4.2 Pérdida, extravío o falla de una llave

Es cierto que en algún momento una de las llaves se puede ver comprometida ya sea por pérdida o robo de la misma, además de que puede ser que fallen, en ese caso el Usuario debe de contactar al administrador del sistema para que su registro y la clave de dicho usuario sean borrados del sistema, posteriormente cuando se vuelva a acreditar al usuario se volverá ingresar su registro y se generará una nueva clave que se almacenará en una nueva llave.

### 4.3 Presencia de intrusos

Cuando personas no autorizadas tratan de interactuar con el sistema si no poseen una llave con una clave válida se presentará una situación de alarma en la cual el uso ilegal del sistema se puede considerar en los siguientes dos escenarios.

1. Uso de una llave falsa
   Aunque las claves que son generadas por el sistema tienen la misma longitud no son fijas ya que se renuevan en cada ingreso válido para cada usuario, de esta manera es muy improbable que se pueda crear una llave valida aunque se conozca el procedimiento para crear las claves de acceso ya que se tienen que conocer cuales fueron los tres de los cinco datos de usuario y la hora del sistema cuando se creo la clave, si no cuenta con estos datos la clave que se genere no concordará con la almacenada en el sistema y se presentará una situación de alarma.
2. Violación de una Terminal
   El daño físico a una terminal siempre es una posibilidad para ingresar a las instalaciones que el sistema controla, es por esto que cada Terminal es monitoreada por la Unidad Central por medio del cable de conexión, si la terminal deja de funcionar o el cable es cortado se reflejará en el estado de la conexión, de esta manera quedará asentado en la bitácora del sistema y se emitirá la señal de alarma en la Unidad Central.

### 4.4 Falla de la Unidad Central

Como ya se dijo, anteriormente cada Terminal depende totalmente de la Unidad Central es

por esto que si esta falla todo el sistema se verá afectado, en principio no se podrán validar los accesos, e inclusive las Terminales pueden presentar una falla en su suministro de energía lo que se planteo anteriormente. En el caso de que la energía no se vea interrumpida pero la Unidad Central no desarrolle su labor, se tiene contemplado la creación de una “llave maestra” para todas las Terminales, esta clave estará almacenada en todas ellas. Su uso no pasa desapercibido, como esta clave tiene que ser estática en casi todo momento es la más comprometida de todas las claves, es por esto que su uso manda una señal de alarma a la Unidad Central así cuando se haga uso de ella ilegalmente esto será registrado en la bitácora y se activará la señal de alarma de la Unidad Central.

Una vez descritas en este capítulo y el anterior las operaciones del sistema y su forma de reaccionar ante situaciones en las que es posible que falle se realiza la propuesta de su estructura física y la forma en la que sus componentes realizan sus operaciones específicas.

## Capítulo 5 Funcionamiento de la Unidad Central y Terminal del sistema

A continuación se describe la relación de los elementos que componen a los dispositivos del sistema así como los pasos que siguen cada uno para llevar a cabo su función.

5.1 Unidad Central

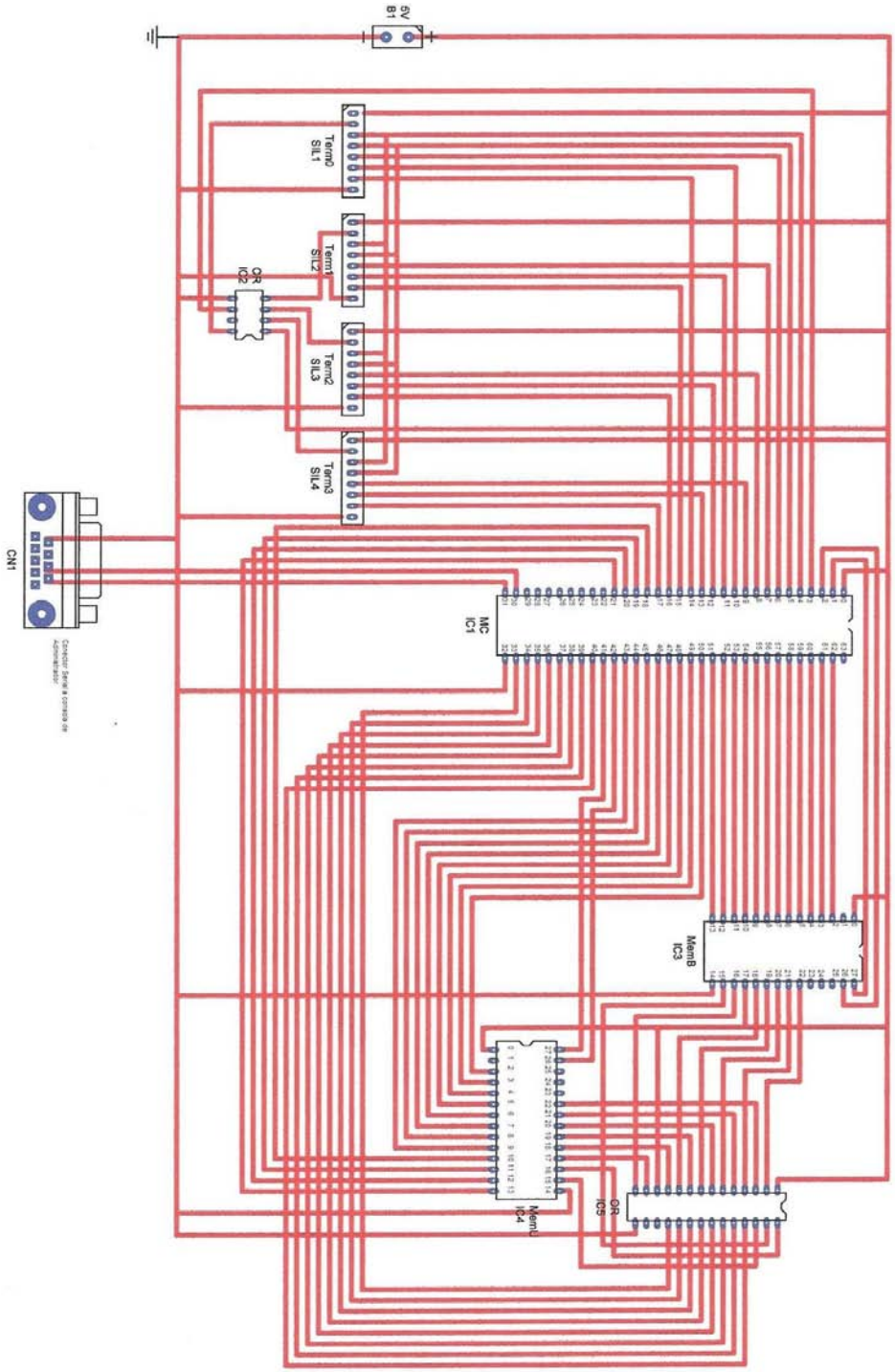


Figura 9. Diseño interno de la Unidad Central

Donde:

- MC es el *Módulo de Control* de la Unidad Central, es un circuito programable, donde los pines 1 y 2 habilitan las operaciones de lectura o escritura en la memoria de Bitácora; se conecta a cada una de las terminales por medio del pin 3 al pin 6 para la transmisión de datos, los cuales son almacenados en una memoria temporal interna tanto para su recepción como su envío de manera serial; el pin 7 es la señal de emergencia, y es común para todas las terminales; el pin 8 es la señal de alerta y también es común a todas las terminales; del pin 9 al pin 12 es la señal de envío de clave para cada una de las terminales; del pin 13 al 16 corresponden a la señal de inicio de envío de clave para cada terminal; del pin 17 al 20 son los conectores para la señal de fin de envío de clave para cada terminal; los pines 20 a 24 direccionan la memoria de Datos de Usuario; los pines 30 y 31 sirven para la comunicación de la Consola del Administrador; los pines 33 al 40 sirven de entrada paralela de datos para las memorias de la Bitácora y de Datos de Usuario, las cuales para transmitir esta información están conectadas al Modulo de Control por medio de una compuerta OR; los pines 41 y 42 habilitan las operaciones de lectura o escritura en la memoria de Datos de Usuario; los pines 43 al 50 envían información en paralelo a la memoria de Datos de Usuario; los pines 51 a 54 direccionan la memoria de Bitácora; los pines del 55 al 62 envían información en paralelo a la memoria de Bitácora; el pin 0 es Vcc y el 32 es conexión a tierra.
- MemB es la *Memoria de Bitácora*, es una memoria no volátil, de la cual sus pines 2 al 9 reciben información en paralelo desde el Módulo de Control; los pines 10 al 13 son para el direccionamiento de las operaciones de lectura y escritura; los pines 15 al 22 envían información en paralelo al Módulo de Control por medio de una compuerta OR; los pines 26 y 27 habilitan las operaciones de lectura o escritura; el pin 0 es Vcc y el pin 14 es conexión a tierra.
- MemU es la *Memoria de Datos de Usuario*, es una memoria no volátil, de la cual sus pines 2 al 9 reciben información en paralelo desde el Módulo de Control; los pines 10 al 13 son para el redireccionamiento de las operaciones de lectura y escritura; los pines 15 al 22 envían información en paralelo al Módulo de Control por medio de una compuerta OR; los pines 26 y 27 habilitan las operaciones de lectura o escritura; el pin 0 es Vcc y el pin 14 es conexión a tierra.
- Tem0, Tem1, Tem2 y Tem3 son los conectores de las terminales, se propone que sean conexiones RJ-45 para poder utilizar cable tipo par trenzado, el orden de los pines es como sigue 0 Vcc, 1 Datos, 2 Señal de Emergencia, 3 Señal de Alerta, 4 Señal de Envío de Clave, 5 Señal de Inicio de envío de clave, 6 Señal de Fin de clave y 7 conexión a tierra.

Como ya se ha explicado con anterioridad la Unidad Central verifica y genera las nuevas claves, además de monitorear a las Terminales del sistema, para este fin realiza diferentes operaciones las cuales se explican a continuación.

5.1.1 Generación de Claves

1. Recibir señal de inicio de generación de Claves (interna al Módulo de Control)
2. Selección aleatoria de los campos de datos de usuario almacenados en la Memoria de datos de usuario
3. Lectura de los campos de usuario seleccionados, direccionandolos desde la Memoria

de datos de usuario.

4. Concatenación de la hora a la cadena de los campos de usuario
5. Cálculo SHA-1 de la cadena de los campos de usuario y la hora del sistema
6. Almacenamiento de esta clave en la memoria principal relacionándola con el Número de Llave
7. Emitir señal de final de generación de clave (interna al Módulo de Control)
8. Anexar entrada en la bitácora del sistema, la entrada consiste de la hora de la solicitud, Terminal de Sistema que la solicito y Número de clave que fue modificado.

Este operación se describe en la Figura 10a y en la Figura 10b.

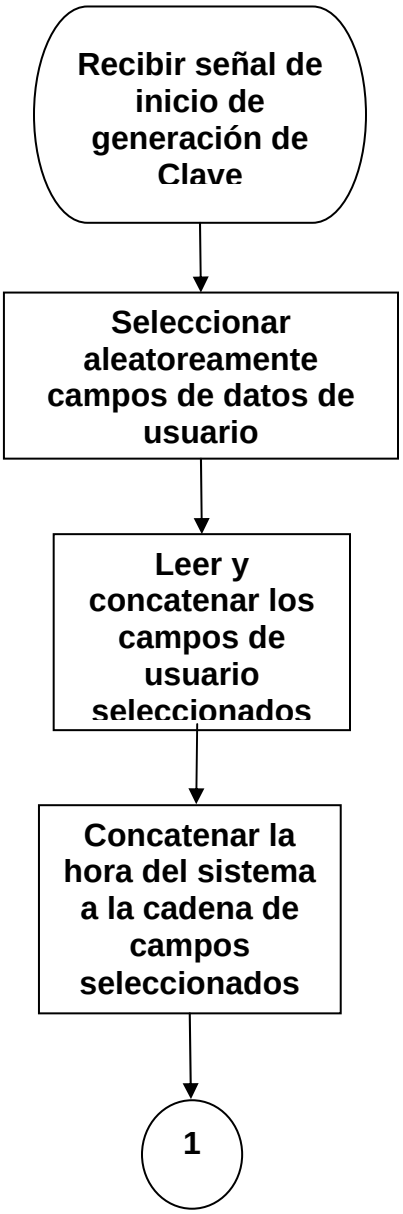


Figura 10a Generación de Claves (inicio)

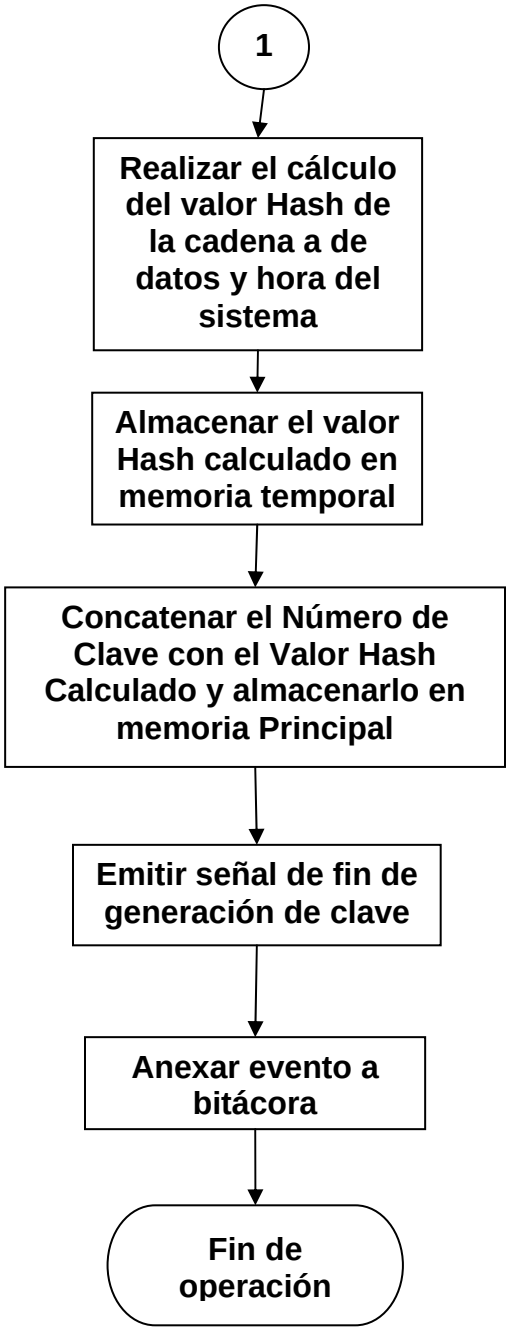


Figura 10b Generación de Claves (fin)

5.1.2 Comparación de Claves

1. Recibir señal de inicio de comparación de clave (interna en el Módulo de Control)
2. Enviar a terminal señal de envío de clave
3. Recibir señal de inicio de envío de clave desde la terminal de sistema
4. Almacenar clave en memoria temporal de la Unidad Central
5. Determinar Número de Clave

6. Comparar clave en memoria temporal con clave de memoria de datos de usuario de acuerdo al índice de clave
7. De acuerdo al resultado se elige una de las siguientes dos opciones
   A. Iniciar proceso de generación de clave
   B. Anexar entrada en bitácora del sistema y emitir señal de alerta a la terminal del sistema

Esta operación se describe en la Figura 11.

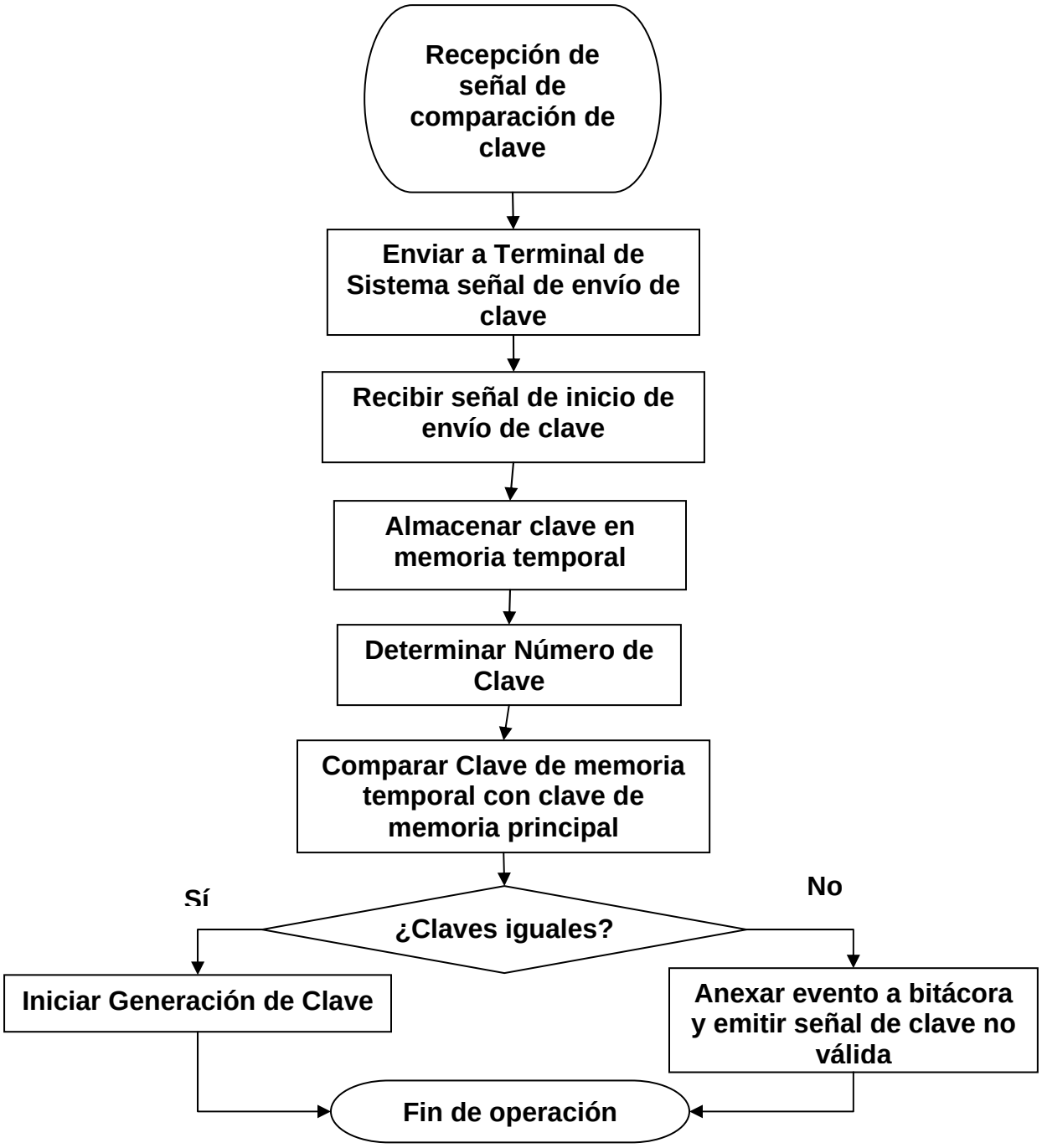


Figura 11 Operación de Comparación de Clave

5.1.3 Alta de usuario

1. Recibir señal de Alta de usuario de la consola de administrador
2. Enviar señal de envío de datos de usuario a la consola de administrador
3. Recibir datos en memoria temporal a través del convertidor serie a paralelo del módulo de control
4. Almacenar datos de usuario en memoria de datos de usuario
5. Anexar entrada en bitácora del sistema
6. Iniciar operación de generación de clave

Esta operación se describe en la Figura 12.

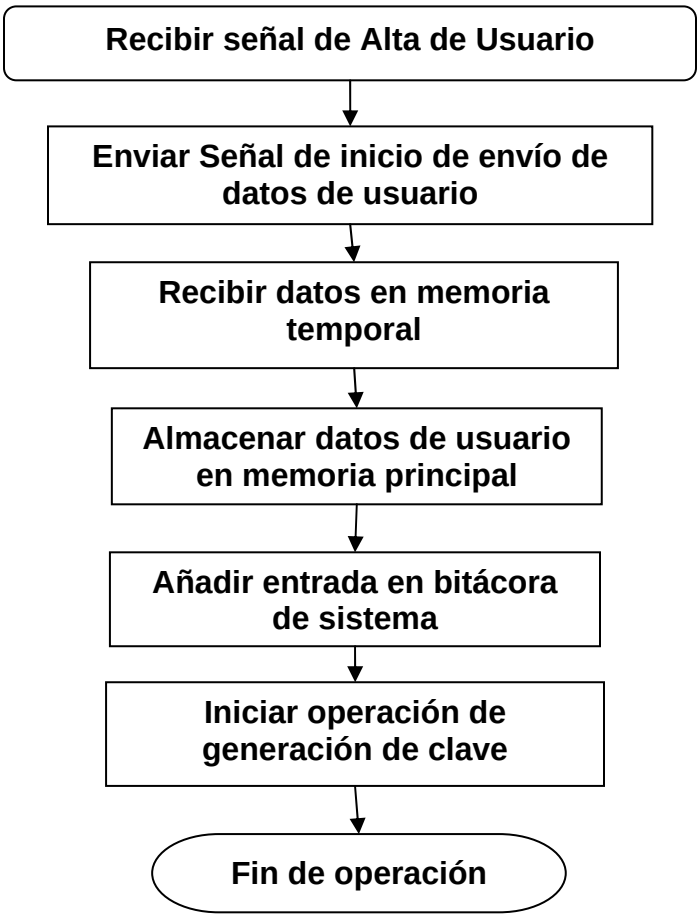


Figura 12 Operación de alta de usuario.

5.1.4 Baja de usuario

1. Envío de señal de baja de usuario desde la consola de administrador
2. Localizar Numero de Clave en memoria de datos de usuario
3. Desplazarse al Número de Clave posterior
4. Copiar desde el registro actual hasta el final de registros a memoria temporal
5. Desplazarse al Número de Clave anterior
6. Borrar desde el índice actual hasta el final de registros

7. Copiar a memoria principal el contenido de la memoria temporal
8. Agregar entrada en bitácora del sistema
9. Enviar señal de final de operación

La descripción de esta operación se encuentra en la Figura 13.

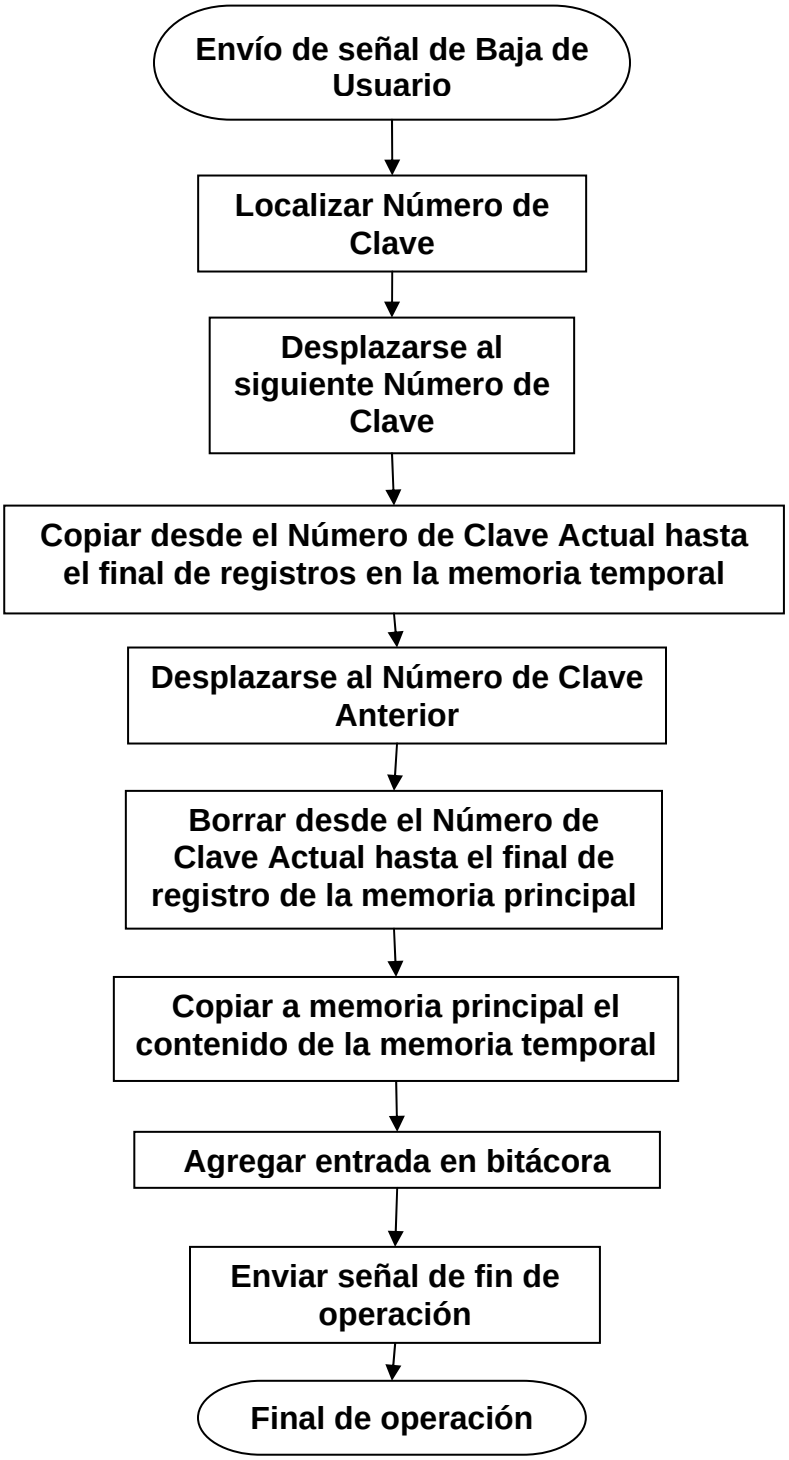


Figura 13 Operación de Baja de Usuario

5.1.5 Actualizar datos de usuario

1. Enviar señal de actualizar datos de usuario desde la consola de administrador
2. Realizar operación de baja de usuario
3. Recibir señal de final de operación (interna del Módulo de Control)
4. Realizar operación de alta de usuario
5. Recibir señal de final de operación (interna del Módulo de Control)
6. Enviar señal de final de operación actualización de datos de usuario (interna de Módulo de Control)

La descripción de esta operación esta en la Figura 14.

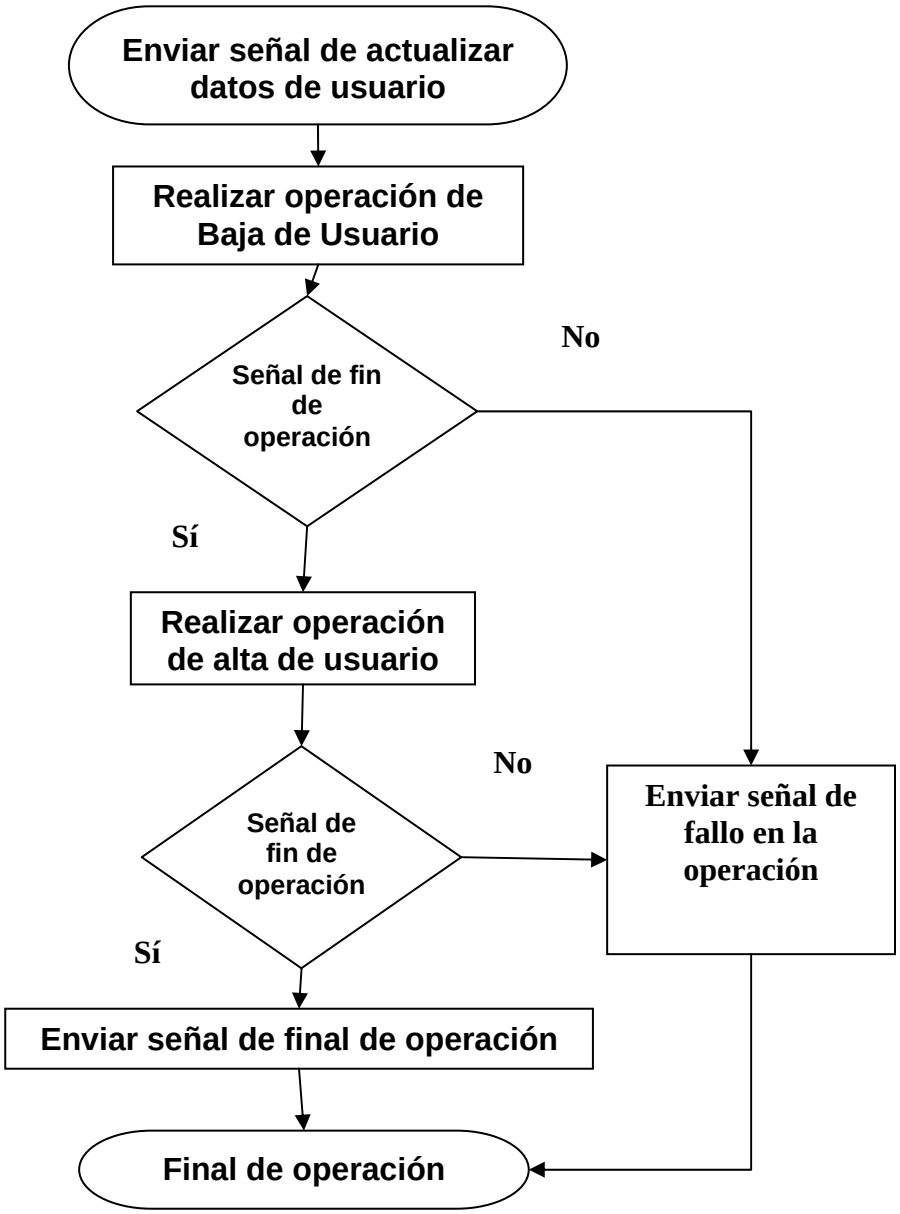


Figura 14 Operación de actualización de datos de usuario

5.2 Terminales del Sistema

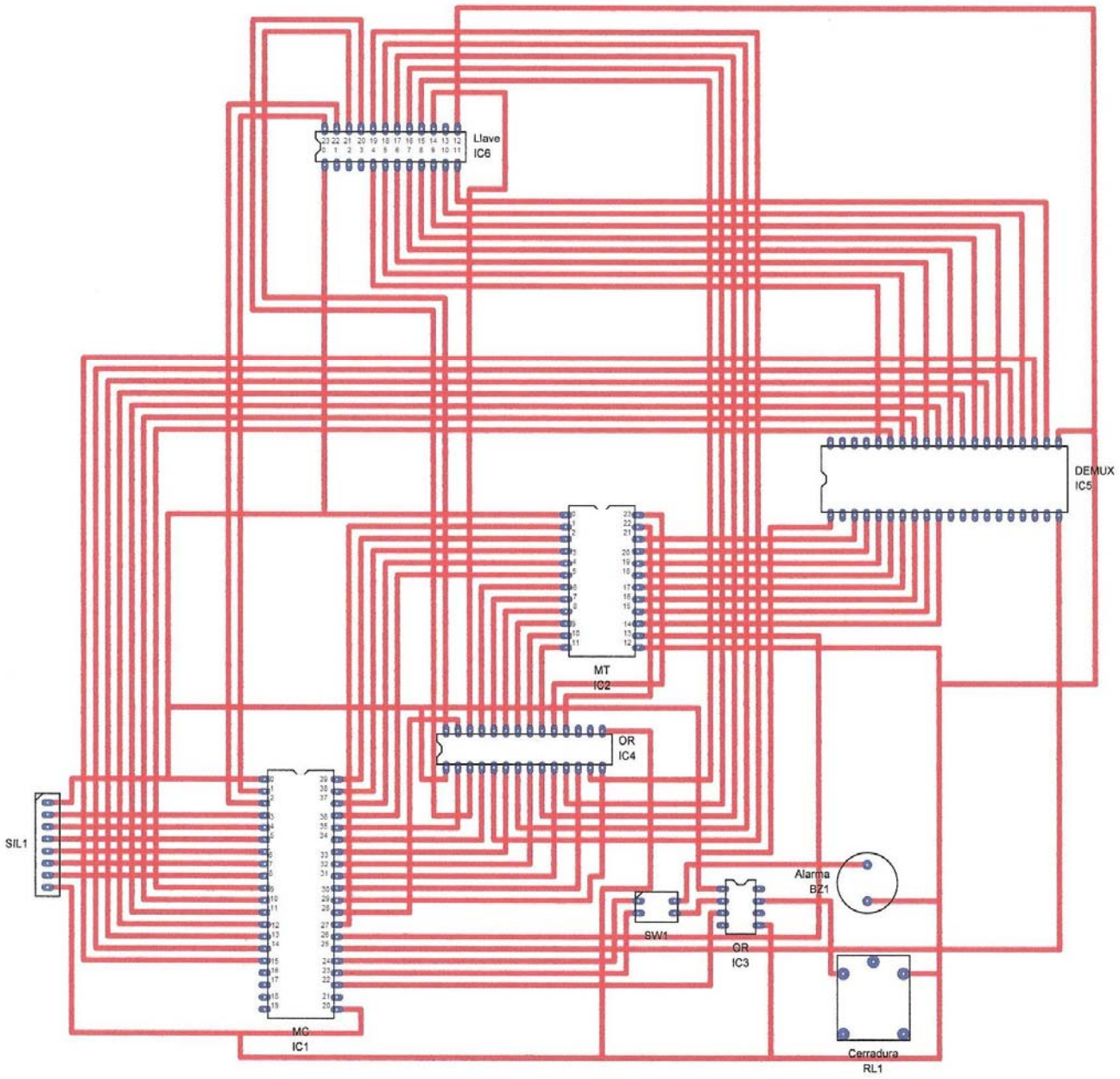


Figura 15 Diseño Interno de las Terminales de Sistema

Donde:

- MC es el *Módulo de Control*, es un circuito programable, del cual sus pines 1 y 2 habilitan las operaciones de lectura y escritura en la Llave conectada a la terminal; 3 es para enviar y recibir datos en serie en relación a la Unidad Central, el pin 4 es para la Señal de Emergencia; el pin 5 es para la Señal de Alerta; el pin 6 es para la señal de Envío de Clave; el pin 7 es para la señal de Inicio de envío de clave; el pin 8 es para la señal de Fin de Clave, del pin 9 al pin 15 reciben información en paralelo desde la Memoria Temporal por medio de un Demultiplexor; el pin 22 es la señal de

apertura de cerradura; el pin 23 acciona la apertura de la cerradura en caso de una señal de emergencia siempre y cuando el switch de configuración este en la posición de paso; el pin 24 activa la alarma de la Terminal en el caso de que se reciba la señal de alerta siempre y cuando el switch de configuración este en la posición de paso; el pin 25 indica al Demultiplexor si debe de enviar la información de la Memoria Temporal al Módulo de Control o a la Llave conectada a la Terminal; los pines 26 y 27 indican a la Memoria Temporal el modo de operación en lectura o escritura; los pines 28 al 35 envían datos en paralelo a la Memoria Temporal por medio de una compuerta OR; los pines 36 a 39 direccionan la memoria temporal durante las operaciones de lectura y escritura; el pin 0 es Vcc y el pin 20 es conexión a tierra.

- MT es la *Memoria Temporal*, es una memoria volátil, en la cual los pines 1 y 13 indican el modo de operación en lectura o escritura; del pin 2 al 5 direccionan a la memoria durante las operaciones de lectura y escritura los pines 6 al 11, 22 y 23 reciben datos en forma paralela desde el Módulo de Control o la Llave que este conectada a la Terminal; los pines 14 a 21 envían datos en paralelo al Módulo de Control o a la Llave conectada a la Terminal por medio de un Demultiplexor; el pin 0 es Vcc y el pin 12 es conexión a tierra.
- Llave es la terminal para que las *Llaves* puedan interactuar con el sistema donde los pines 4 al 11 reciben información en paralelo desde la Memoria Temporal a través de un Demultiplexor, los pines 14 al 21 envían datos en paralelo a la Memoria Temporal a través de una compuerta OR; los pines 22 y 23 indican a la llave si debe realizar una operación de lectura o una operación de escritura; el pin 0 es Vcc y el pin 12 es conexión a tierra.
- Sil1 es el conector que enlaza a la Terminal con la Unidad Central es un conector RJ-45 y el medio de enlace es un cable de par trenzado en el cual el pin 0 es Vcc, el 1 es transmisión y recepción de datos en serie, el pin 2 es la Señal de Emergencia, el pin 3 es la Señal de Alerta, el pin 4 es la señal de Envío de Clave, el pin 5 es la Señal de Inicio de Envío de Clave, el pin 6 es la señal de Fin de Envío de Clave y el pin 7 es la conexión a tierra.

Además de los componentes ya descritos en la figura se proponen una compuerta OR para el correcto funcionamiento de la señal de Emergencia, un switch para determinar en cada terminal el modo de funcionamiento durante las señales de Emergencia y Alerta, un buzzer para activarse durante la señal de Alerta y un Relay para accionar el pasador de la cerradura.

Cada una de las terminales aunque dependa constantemente de la Unidad Central lleva a cabo procesos para la validación y permitir el acceso.

### 5.2.1 Lectura de Clave

1. Llave es insertada en Terminal de Sistema
2. Copiar contenido de la llave en memoria temporal de la terminal de sistema
3. Envío de señal de solicitud de envío de clave a Unidad Central
4. Recepción de señal de envío de clave desde la Unidad Central
5. Envío de señal de inicio de envío de clave hacía la Unidad Central
6. Envío de clave por medio del convertidor paralelo – serial del Módulo de Control de la Terminal de Sistema

7. Envío de señal de fin de clave hacía la Unidad Central

La operación de Lectura de Clave se representa en la Figura 16

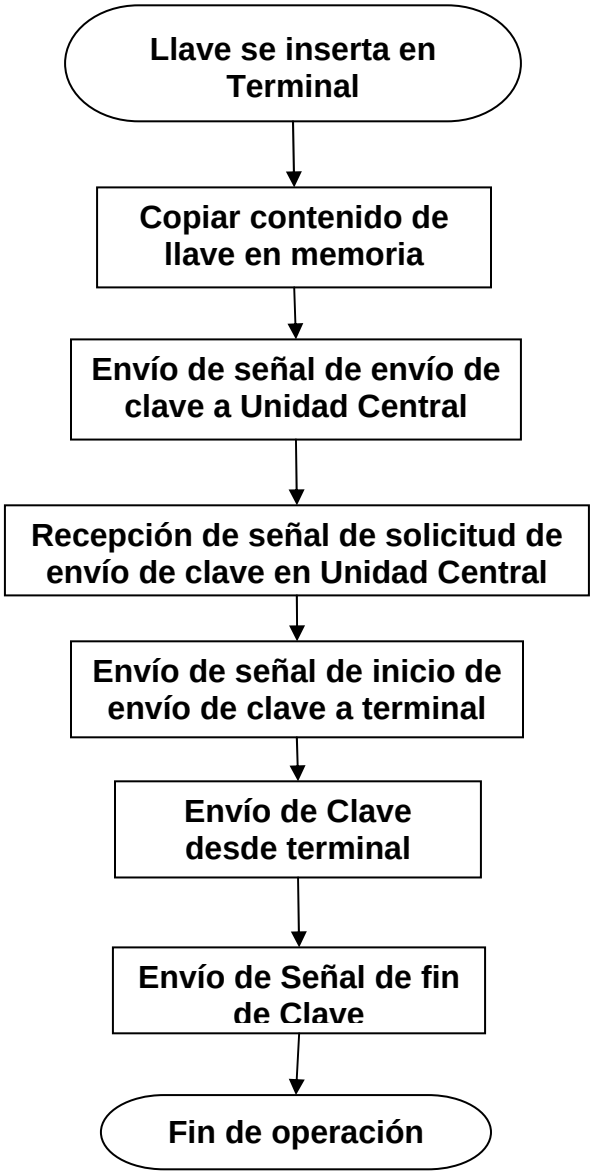


Figura 16 Operación de lectura de clave

5.2.2 Almacenamiento de clave

1. Recepción de señal de actualización de clave desde la Unidad Central
2. Envío de señal de recepción de clave hacía la Unidad Central
3. Recepción de señal de inicio de clave desde la Unidad Central
4. Almacenar clave en memoria temporal de la Terminal de Sistema
5. Recepción de señal de fin de clave desde la Unidad Central
6. Almacenar clave en llave direccionando lectura de la Memoria Temporal y escritura en

el conector de la Llave
7. Realizar operación de acceso permitido

Se describe la operación de almacenamiento de clave en la Figura 17

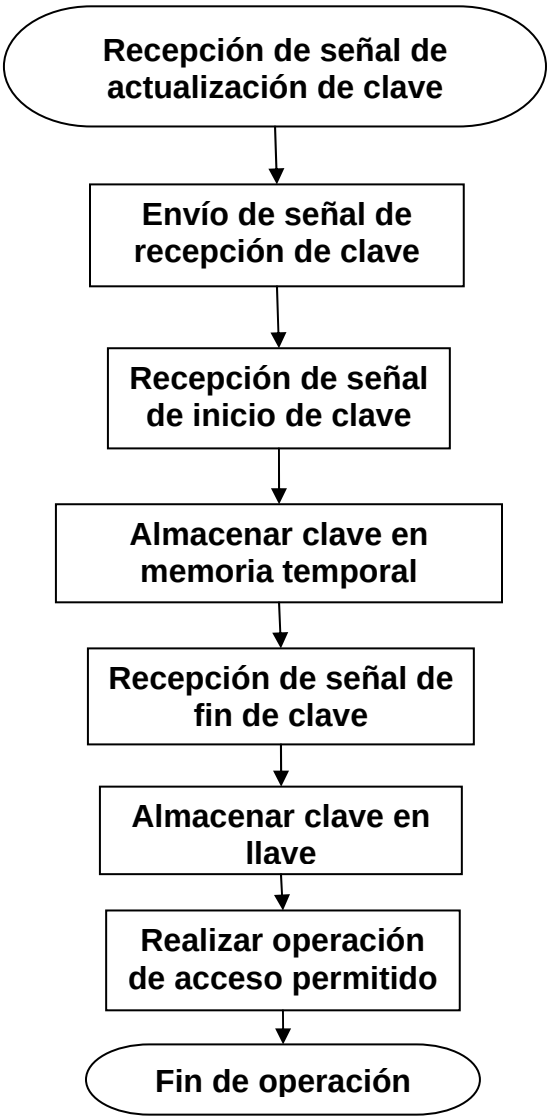


Figura 17 Operación de Almacenamiento de Clave

5.2.3 Acceso permitido

1. Recibir Señal de Acceso permitido (interna al Módulo de Control de la Terminal)
2. Activar mecanismo de apertura

La Figura 18 muestra la operación de acceso permitido

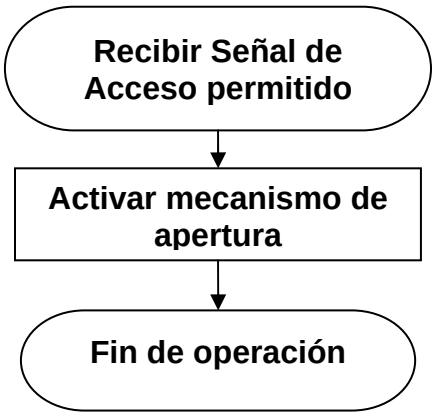


Figura 18 Operación de acceso permitido

5.3 Opciones especiales de las terminales de sistema

Señal de clave no valida

1. Recibir señal de alerta desde la Unidad Central
2. De acuerdo a la programación de la terminal se tomará una de las siguientes dos opciones, lo que se llama modo silencioso
   A. Simular operación de acceso permitido al activar el mecanismo de apertura
   B. Activar señal de alarma hasta que la unidad central deje de emitir la señal de clave no valida

Esta operación se representa en la Figura 19.

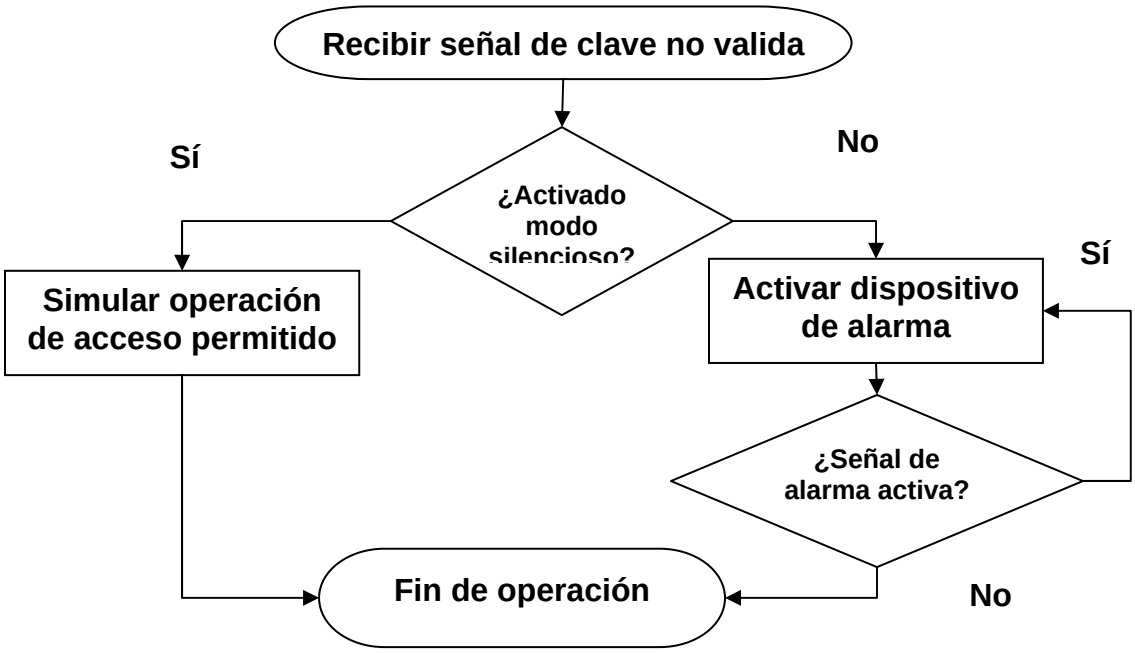


Figura 19 Operación con señal de clave no valida

Señal de emergencia

1. Recibir señal de emergencia
2. De acuerdo a la programación de la terminal se tomará una de las siguientes dos opciones, esto se llama modo de protección
   A. Iniciar y mantener activación de mecanismo de apertura
   B. Activar señal de alarma hasta que la unidad central deje de emitir la señal de emergencia

La Figura 20 muestra el procedimiento de la operación de señal de emergencia.

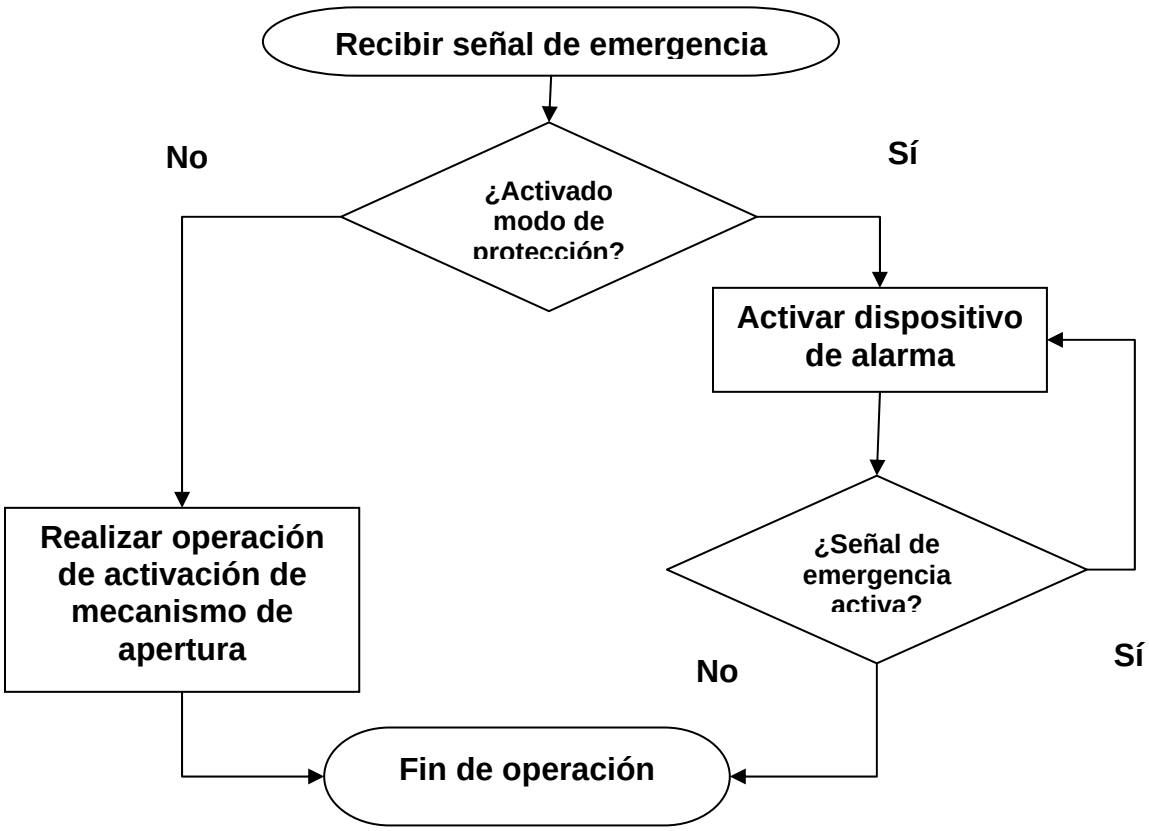


Figura 20 Operación con señal de emergencia

Es así que el sistema cumple con su función de permitir el acceso a las entidades que poseen una Llave en la que esta almacenada una clave válida y reacciona a los eventos de emergencia, manteniendo un registro de cada uno de estos eventos.

## Conclusiones

El desarrollo de este sistema tiene la posibilidad de aumentar sus capacidades gracias a la característica de poder programar los módulos de control de los elementos que lo conforman. Así, en este trabajo se ha propuesto una estructura física que pueda acomodar a estos módulos y les permita crecer en opciones conforme se requiera, así mismo, se ha propuesto el empleo de un algoritmo de resumen para la creación de las claves de acceso para que este sistema pueda ser avalado por un estándar de un organismo internacional como lo es el NIST (National.Institute of Standards and Technology.

Las posibles mejorías que puede tener este sistema es el permitir accesos por tiempo y por área, de tal manera que un usuario solo pueda tener acceso en las terminales que le corresponden a su función y en los tiempos en los que debe de realizarlas, agregando también, los eventos correspondientes en la bitácora.

Una mejoría para las terminales puede ser el agregar un botón de pánico de tal manera que un usuario pueda mandar una señal de alerta a la Unidad Central en caso de que lo necesite.

**Bibliografía**

**[1] Syscom. Catálogo [En línea] http://www.syscom.com.mx/Productos/Monitoreo_Seguridad/lectoras_biometricas.htm [Consulta: Septiembre de 2006].**

**[2] Syscom. Catálogo [En línea] http://www.syscom.com.mx/Productos/Monitoreo_Seguridad/lectoras_tarjetas_teclados.htm [Consulta: Septiembre de 2006].**

**[3] Syscom. Catálogo [En línea] http://www.insseg.com.mx/Productos/Monitoreo_Seguridad/lectoras_tarjetas_teclados.htm [Consulta: Septiembre de 2006]**

**[4] Manuel José Lucena López. Criptografía y Seguridad en Computadoras. Segunda Edición. Universidad de Jaén. Septiembre de 1999: 25, 26**

**[5] National Institute of Standards and Technology (NIST). Federal Information Processing Standards Publication 180-2 (+ Change Notice to include SHA-224), FIPS PUB 180-2. Agosto de 2002: 4-10, 12-13, 15-18, 25-32.**