Uso de hash enfocadas en la búsqueda de datos y seguridad informática

Oscar Lopez [1], Tomas Arauz [2]
{Yomar09martinez, Abdiel06arauz}@gmail.com
Profesor Asesor: Ingeniera Tejedor-Morales, María Yahaira.
Universidad Tecnológica de Panamá, C.R. de Coclé, Grupo de investigación SoftSolution Group,
maria.tejedor@utp.ac.pa

Resumen: ***como sabemos la tecnología es algo que avanza constantemente y de forma rápida y la cantidad de información que se genera sobre un tema es enorme, así que surge la necesidad de buscar una forma de almacenar esa información pero de una forma transparente, accesible y segura. En este trabajo de investigación se plantea el análisis del tema de transformación de llave donde estos algoritmos actúan como ayuda para lograr mayor eficiencia y rapidez en la búsqueda de elementos enfocado en el análisis y protección confidencial de datos o sea que además de guardar elementos también tendremos en cuenta la seguridad basándose en la protección de la información. Uno de los problemas más comunes en el mundo del hashing son las colisiones; al momento de ingresar información, veremos cómo se resuelve este problema por cada uno del método. Con una visión más amplia nos enfocaremos en un software (HashCheck) que genera claves utilizando los métodos hash (MD5, SHA 1) más seguros en la actualidad.***

I. Introducción

El estudio realizado se basa principalmente en la rapidez de búsqueda de información, en el cual se puede decir que la transformación de llaves es sinónimo de velocidad al buscar cualquier elemento. El documento aborda el tema a profundidad, presenta los métodos y diversas técnicas empleadas para realizar la búsqueda y cómo es el proceso subyacente. Esto instruye al lector sobre cómo saber si mis archivos son seguros al momento de descargarlos de los sitios web. En la sección 2 veremos concepto de transformación de llaves y formas de uso, luego en la sección 3 se presentan técnicas de cálculo de direcciones empleadas en las actualizaciones de registro. Seguido, en la sección 4 se presentan las comparaciones entre las funciones hash; la sección 5 explica cómo manejar o solventar colisiones, sus ventajas y desventajas. Tomando el concepto de búsqueda veremos cómo funciona el proceso de transformación de llaves en el área de seguridad de datos y dónde puede ser aplicada hoy en día.

Y finalmente en la sección 6 presentamos una forma de saber la originalidad y seguridad en nuestros archivos mediante usos de las técnicas hash.

El tema que nos ocupa es amplio, complejo y profundo; de allí que el documento aspira a despertar en estudiantes de pregrado e investigadores en general la motivación para continuar líneas de investigación y desarrollo que están abiertas; a fin de proseguir con futuros trabajos orientados al desarrollo de aplicaciones que provean soluciones a través de las técnicas y métodos de hash.

II. CONCEPTO DE TRANSFORMACIÒN DE LLAVES Y FORMA DE USO

En conceptos de seguridad ya sea en páginas web o programas creados para empresas en áreas financieras se requiere la protección de información de datos de dichas organizaciones, en la cual podemos ver el uso de funciones hash en un resumen de alto nivel.

Se define la transformación de llave como: “procesamiento de una estructura encargada de representar de forma compacta un archivo o conjunto de datos que normalmente es de mayor tamaño que el hash independientemente del propósito de su uso” [1].

En cuanto a la usabilidad, existen ejemplos y casos donde es importante la protección en la confidencialidad de los sistemas de evaluación de proyectos y estados financieros mediante contraseñas [2]. Vemos una vez más las funciones contextualizadas a la vida real. Ejemplo figura 1.

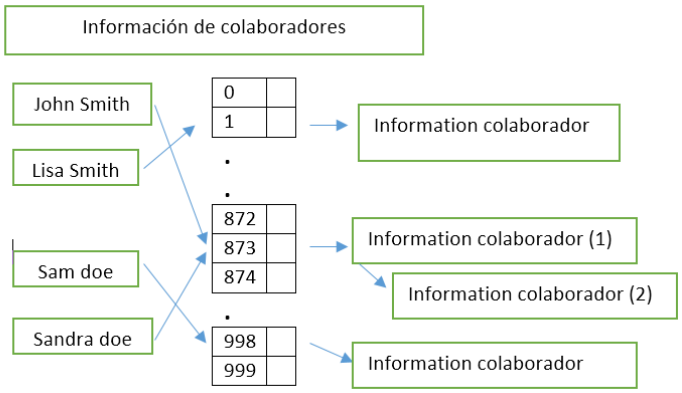

Figura 1. Formas de uso o funcionamiento de transformación de llaves. Donde cada valor hash contiene información de un colaborador de la organización.

III. TÉCNICAS DE CÁLCULO DE DIRECCIONES EMPLEADAS EN LAS ACTUALIZACIONES DE REGISTRO

Básicamente las técnicas de cálculo de dirección son distintos campos de alternativa que se eligen con el objetivo de evitar las distintas colisiones que se dan en el proceso de almacenamientos de los datos dentro de un array .
Ahora nos preguntaremos qué significado le damos a una colisión. Podemos definir como colisiones a las veces en que un determinado dato es colocado o enviado a la misma posición de un array donde ya se encuentra o ya existe un elemento.
Como podemos observar una colisión y las técnicas de cálculo de dirección son conceptos ligados porque si se produce una colisión de elementos es necesario utilizar unas de las técnicas de cálculo de dirección. A continuación se explica qué son, en qué consisten, cuál es su funcionamiento, entre otros detalles.

A. *Hashing por residuo enfocado en la localización de registros*

$$H(K)=K+1 \qquad \textbf{(1)}$$

En hashing por residuo lo que hacemos es tomar los 2 últimos valores de la clave o el dato y ese sería la ubicación donde se guarda dicho elemento en el array.
¿Pero qué pasaría si se llega a producir una colisión?
En este caso tomando, la ventaja de utilizar hashing es que las claves son asignadas con valores enteros positivos, esto permite que en una situación donde se llegue a presentar distintas colisiones, lo que ocurre es que se toma la clave o el dato y se le suma un valor 1; de esta forma cada vez que exista colisión al irle sumando un 1 a la clave lo que sucede es que se va recorriendo el array hasta encontrar una posición libre, ejemplo en la figura 2.

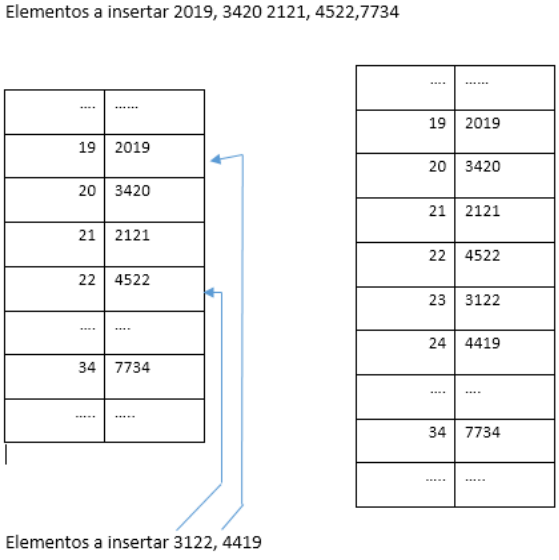

Figura 2. Se ingresan valores al arreglo y ocurre colisiones. Se insertan cinco valores correctamente, al inserta el valor 3122 ocurre la primera colisión; ya que, la posición 22 se encuentra ocupada por el valor 4522 en la parte izquierda se busca una posición libre para ingresar los valores colisionados.

B. *Hashing por cuadrado medio*

$$H(K) = \textit{digitos\_centrales}\ (K^2). \qquad \textbf{(2)}$$

Hashing por cuadrado medio, también denominado mitad del cuadrado, es la técnica que consiste en tomar la clave o valor hash y elevar dicho valor al cuadrado. Entonces dependiendo del valor resultante se toman los valores que se ubican en el centro y ese sería la ubicación donde se guardará el elemento dentro de un array [3].
Este proceso se realiza para cada vez que se produce una colisión elevarlo al cuadrado para alejarse lo más que se pueda de donde se produjo dicha colisión. Ejemplo en la tabla1.

Tabla1. Funcionamiento por cuadrado medio.

Hashing por cuadrado medio		
llaves	**Llave ^2**	**Posiciones**
11155	124 434 025	**434**
23466	550 653 156	**653**
26711	713 477 521	**477**

C. *Hashing por pliegue*

$$H\ (K) = ((d1...di) + (d2...dj). \qquad \textbf{(3)}$$

Técnica de separación porque básicamente se basa en realizar separaciones entre los valores de la clave, considerando que la clave esté separada de forma correcta. Cada separación debe tener los mismos dígitos con excepción de los últimos dígitos ya que estos pueden tener la misma cantidad de dígitos o menos, después de separarlos sumar o multiplicar, al final el resultado obtenido será la posición o índice. Ejemplo en la Tabla2.

Tabla2. Ejemplo de separación de la clave por técnicas de pliegue

Llaves	**Separación y suma**	**Posiciones**
8149	81 + 49	130
6236	62 + 36	90
5533	55 + 33	88

Adicional a las formas de hashing vistas, existe otra muy utilizada conocida como encadenamiento enlazado que básicamente consiste en ir creando una lista enlazada mediante cada colisión que se produzca cada vez que se inserte elemento al array [4], podemos ver un ejemplo en la figura 3.

Realizando un resumen de estas técnicas, podemos expresar que su funcionamiento en aspectos de seguridad va dirigido a la búsqueda de texto escrito, ya sea el nombre completo de un usuario, realizando una comparación entre el nombre y muchos nombres dentro de una base de datos. Es sumamente importante la veracidad de información [5], o sea que los datos clave de los colaboradores de una organización como su identificación de deben ser correctos. Imaginémonos que soy cliente de un banco y necesito realizar un depósito, vemos que se necesita la búsqueda del número de cuenta, se necesita una búsqueda rápida y sin errores, podemos ver una vez más el uso de las técnicas de cálculo de dirección.

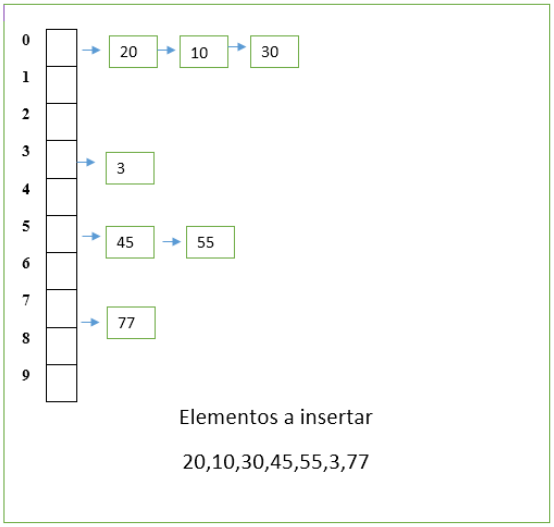

Figura3. Mediante cada colisión se va creado una lista enlazada. El tamaño de la lista enlazada dependerá de la cantidad de elementos que colisionaron al momento de ingresar elementos.

IV. COMPARACIÓN ENTRE LAS FUNCIONES HASH

Las funciones hash son usadas en el ámbito de seguridad, con el objetivo de proteger la confidencialidad de una contraseña; ya que mediante estas funciones de resumen, los datos están protegidos ante cualquier intento de robo de información.

Tabla3. Comparación entre las funciones

Funciones de resumen	
Hashing por residuo	Se toman los 2 últimos valores de la firma digital.
Hashing por cuadrado medio	Se toma el valor de la firma digital y se eleva al cuadrado luego eligen los valores del centro del resultado.
Hashing por pliegue	Esta técnica consiste en la separación de la firma digital.

V. MANEJO DE COLISIONES

A. Área de Desbordamiento:

Consiste en almacenar los datos que han colisionado en una lista aparte o secundaria, el único problema es que guarda espacio automáticamente para un acumulador o apuntador a su lista de colisión o hash [6].

5.2 Sobrecarga progresiva encadenada:

Es otra manera fácil y simple de resolver el tema de colisiones, lo que hace este método es crear una lista ligada a cada registro que sea igual a otro de la lista original, de manera que podamos movernos entre aquellos que colisionan. Como se muestra en la figura 4.

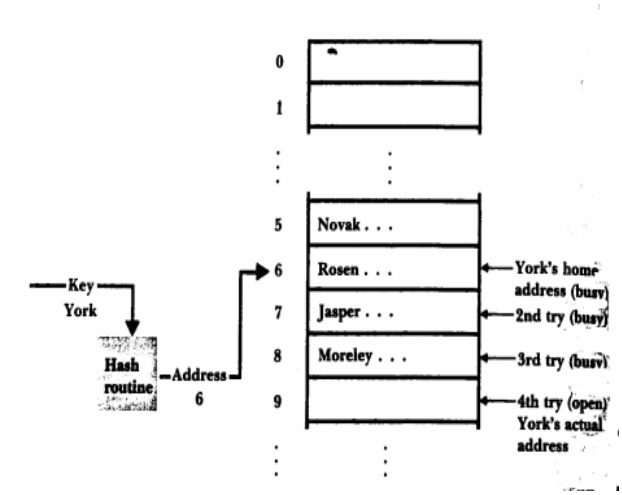

Figura 4. Ejemplo de sobrecarga progresiva encadenada [6].Creación de lista ligadas a registros con información similar.

Double Hashing: es otro método efectivo al igual que los anteriormente mencionados, aunque también tiene fallas.
Consiste únicamente en crear otro hash, con un código distinto para controlar mejor la colisión. La característica de este método es que no elimina la colisión en sí, sino que ayuda a localizar de manera más rápida algún registro en un espacio donde pueden encontrarse más colisiones [7], [8].

B. Sobrecarga Progresiva:

Es el más sencillo de todos los algoritmos de manejo de colisiones.
El algoritmo dice que cuando insertamos un dato en un registro y a su vez insertamos el registro, obtenemos una dirección con la función de hashing. Si la dirección está vacía, sin datos, insertamos ahí nuestro registro. En caso contrario, se busca la primera dirección que le siga y que esté disponible y se inserta ahí.

Cabe destacar que el archivo debe tener un número fijo de direcciones posibles, con lo cual, cuando se llega al máximo de direcciones debe comenzar a buscar espacios disponibles desde el inicio del archivo, y si no hay, pues hubo un error y hay que empezar con el código para insertar el registro desde el principio.

Home address	Actual address	Data	Address of next synonym
20	20	Adams . . .	22
21	21	Bates . . .	23
20	22	Cole . . .	25
21	23	Dean . . .	−1
24	24	Evans . . .	−1
20	25	Flint . . .	−1

Figura5. Ejemplo de Sobrecarga Progresiva [6].

VI. ORIGINALIDAD Y SEGURIDAD DE ARCHIVOS

Actualmente la inseguridad en la red es un problema que afecta a todos los que disponemos de la web para las actividades cotidianas. Mediante un estudio e investigación encontramos la manera de cómo estar seguros de lo que descargamos de la web, por ejemplo, en alguna oportunidad has decidido descargar algo de la web, digamos que quieres descargar una ISO de Ubuntu, junto al enlace de descarga te aparecen unas series de letras y números (un hash).

A. ¿Para qué nos puede servir?

La firma digital del archivo puede ayudar a verificar si el archivo que descargado, en este caso el ISO de Ubuntu, es el mismo que estaba en la web y que no fue corrompido durante la descarga, inclusive para comprobar que ningún atacante lo ha modificado durante la descarga hacia el computador.
Lo bueno de esto es que para verificar un archivo es algo muy fácil de hacer y rápido.
Los usuarios de Mac y Linux para generar valores hash no es necesario instalar nada en el ordenador. Para Windows se debe tener un pequeño software llamado HashCheck, es libre y está disponible para descargar [9].
Este software añade una ventana extra a la ventana de propiedades. En esa pestaña se encuentran el hash MD5 y SHA1 del archivo. Sólo hay que compararlos con el que da al descargar el archivo para verificar que todo está bien, ejemplo demostrado en la figura (6).

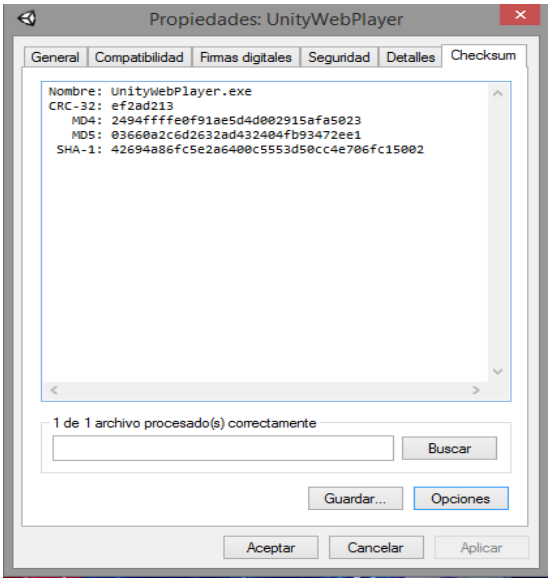

Figura 6. Captura de pantalla de la ventana de hashCheck.

VII. CONCLUSIÓN

La investigación ha conceptualizado alrededor del hash, presentado procesos diversos para encontrar métodos que resuelvan el problema de colisiones, los cuales favorecen a la hora de buscar un elemento con grandes dimensiones. Cada técnica ayuda a evaluar este inconveniente y a mejorarlo. Se han planteado ejemplos documentados en las referencias que destacan la aplicación de las técnicas hash en estructuras de datos como arrays. Particularidades en ciertas técnicas, como hashing por residuo, exige que el arrays tiene que ser de gran tamaño por si existe colisión, ya que tendría que ir buscando una posición vacía; de lo contrario se presenta un desbordamiento de datos. La evaluación de las técnicas considerando ventajas y desventajas con relación a la confiabilidad de la información ha sido ejemplificada, sobretodo en

organizaciones que demandan alto nivel de seguridad de la información. La complejidad para manejar estas operaciones requiere el uso de las técnicas de hash para robustecer los servicios y ofertar a usuarios y clientes transacciones seguras y plataformas garantes de la confidencialidad de la información.

Un nuevo reto está por delante, el desarrollo de aplicaciones en donde se concretice toda la abstracción del hash en soluciones tendientes a fortalecer una de las áreas más polémicas como lo es la seguridad informática.

RECONOCIMIENTOS

Agradecer a la Biblioteca del Centro Regional de Coclé, Universidad Tecnológica de Panamá, por facilitar Fuentes de Información, acceso a consultas Digitales, que fueron de gran ayuda para llevar a cabo el documento.

REFERENCIAS

[1] Byton S. Gottfried ¨Programación en C¨ Departamento de informática y autonómica Facultad de Ciencias Físicas Universidad Complutense de Madrid, 1996.

[2] Nassir Sapag Chain, Rinaldo Sapag Chain, "Preparación y evaluación de proyectos" 3ª. edicion, 1997.

[3] Alfred V. Aho, Jeffrey D Ullman y John E. Hopcroft, "Estructura de datos y algoritmos",1998.

[4] Javier Ceballos "Enciclopedia de Microsoft Visual C#" 2ª. Edición, 2003.

[5] Gonzalo Álvarez Marañón, Pedro Pablo Pérez García, "Seguridad informática para empresas y particulares" ,2004.

[6]Martin sierra, Antonio J. "Java 2 programador certificado" 4ª. Edición*

[7] Drozdek, A. "Estructura de Datos y Algoritmos en Java". México: Thomson, 2007.

[8] Luis Joyanes Aguilar, Ignacio zahonero Martínez "Estructura de datos con java",2008

[9] HashCheck shell Extension Online. Available: http://code.kliu.org/hashcheck/ [Accessed: 30- Jun- 2017].

[10] El origen de los algoritmos hash. Online Available http://ict.udlap.mx/people/carlos/is215/ir09.html [Accessed: 30- Jun- 2017].

[11] Luis Joyanes Aguilar, Lucas Sánchez García, Ignacio zahonero Martínez "Estructura de datos en C++",2007.