Experiencias del Hash en la vida informática
Experiences of the Hash in the life computing

Barrera Maxs[1], Santana Luis[1], Pereira Radames[1], María Yahaira Tejedor[2]*
[1]Licenciatura en Desarrollo de Software – Centro Regional de Coclé – Universidad Tecnológica de Panamá,
[2] Facultad de Ingeniería en Sistemas Computacionales – Centro Regional de Coclé – Universidad Tecnológica de Panamá

Resumen Este artículo resalta el proceso de generación de llaves codificadas y, los métodos Hash asociados al mismo y las colisiones que ocurren tras dicha transformación. La tecnología dentro de este contexto está orientada primordialmente a buscar diferentes alternativas para proteger la información almacenada, por ejemplo través de la encriptación de llaves, minimizando el riesgo de que esta pueda alterarse o perderse, aprovechando al máximo el espacio de almacenamiento. Finalmente, se trabajó en el aporte de un nuevo método sobre la base de dos fórmulas, una que genera la llave y otra que se usa en caso de que ocurra una colisión, ya que después del análisis y prueba de algunos de los métodos existentes se evidenciaron vacíos al momento de contrarrestar colisiones, oportunidad que aprovechamos y que sometemos a consideración.

Palabras claves Colisión, conversión, métodos de búsqueda, transformación de llaves, hash, módulo, cadena, ASCII, medio cuadrado.

Abstract This article highlights the process of generating coded keys and, the Hash methods associated and the collisions that occur after transformation. The technology within this context is oriented primarily to look for different alternatives to protect the stored information, for example through the keys encryption, minimizing the risk that it can be altered or lost, taking full advantage of the storage space. Finally, we worked on the contribution of a new method based on two formulas, one that generates the key and the another one that can be use in case of a collision, since the analysis and testing of some of the existing methods evidenced gaps at the moment to countering collisions, opportunity that we take advantage and that we submit to consideration.

Keywords Collision, conversion, search methods, key transform, hash, module, string, ASCII, half square.

* Corresponding author: maria.tejedor@utp.ac.pa

1. Introducción

La intención inicial de este artículo fue expandir nuestra comprensión sobre las funciones Hash y el contexto en donde se aplica, descubriendo que son múltiples los ámbitos en donde impacta esta tecnología, sobre todo aquellos relacionados con la integridad de la información.

Luego surge la inquietud ¿Cómo podemos contribuir en esta área? Es así como decidimos analizar la temática y tratar de encontrar vacíos u oportunidades de mejoras en los métodos que se desarrollan en el artículo, presentando nuevas ideas orientadas a robustecer la seguridad informática.

Resolvimos que la investigación plantearía nuestra posición respecto al Talón de Aquiles de las herramientas Hash en la generación de llaves y resolución de colisiones. Ustedes dirán: "¿Cuál es ese Talón de Aquiles?"; pues bien, es el "Gasto de Memoria" inmerso en las funciones Hash abordadas. Tal escenario nos impulsó a crear el Método del Módulo de la Cadena.

Este documento ha sido organizado de tal forma que en principio se revisan los conceptos básicos del Hash para lograr una mejor comprensión. En la siguiente sección están los métodos importantes para mostrar las colisiones y cómo tratar con ellas. Seguidamente, la aplicación del Hash en la vida informática. En la sección de resultados se plantea nuestro aporte, que consiste en el desarrollo de un método innovador pensando en mejorar las falencias evidenciadas en las pruebas realizadas, buscando ofrecer un complemento más eficiente a lo que ya existe en los métodos estudiados en este documento.

2. Transformación de llaves

El Método de Transformación de llaves nos da la facilidad de encontrar una llave a la cual se le asigna un elemento en específico dentro de un conjunto de elementos.

Se utilizarán cálculos, métodos matemáticos y transformaciones aritméticas para poder asignarle un espacio

o registro, en el cual pueda almacenarse o posicionarse el elemento sin que ocurra ningún tipo de error.

2.1 Características

- Eficacia en el almacenado de los elementos.
- Salvaguardar cualquier tipo de información.
- Evitar errores en la administración del espacio de almacenamiento.
- Proteger los elementos de alteraciones y/o robos.

2.2 Bases fundamentales

Elementos que pertenecen a este método:

- **Hash:** Es el método que genera una llave para almacenar la información, sin que ocurra alguna colisión de llaves o espacios de almacenamiento ya utilizados.
- **Llave:** Es el valor que tendrá la posición o dirección donde se encuentran la información almacenada dentro de un arreglo.

2.3 Bases fundamentales

- **Conversión:** Es una base importante ya que la variable que vamos a utilizar es numérica de tipo entera, en este punto entra la aplicación del ASCII (Sistema de Codificación de Caracteres Alfanuméricos); dicho sistema designa un valor numérico (entero) a cualquier tipo de carácter.

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	SP	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	0
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

Figura 1. Tabla de codificación ASCII [10].

- **Generación:** Es el proceso que genera la llave, aplicando operaciones matemáticas.

$$H(K)= (K \bmod N) +1 \quad (1)$$

- **Compresión:** La llave que fue creada y procesada aún no está apta para usar, por ello se hace la compresión, que es el paso final donde se multiplica nuestra variable por un factor, de modo que éste se convierta en la llave a emplear dentro del arreglo [2].

3. Materiales y métodos

Diferentes autores han considerado los siguientes métodos:

3.1 Hashing por residuo

Consiste en tomar el residuo de la división de la clave entre el número de componentes del arreglo.

La función Hash queda definida por la siguiente fórmula;

$$H(K)= (K \bmod N) +1 \quad (2)$$

Para mejor manejo se pide que N sea un número primo inferior al número total de elementos [5]. Ejemplo:

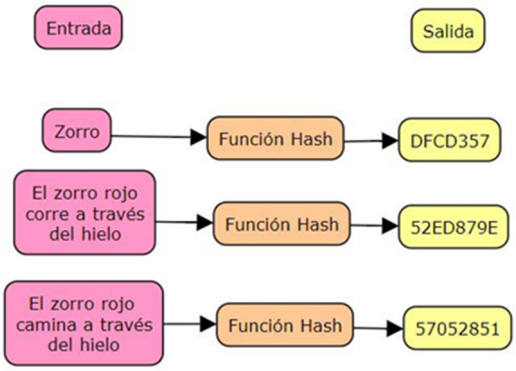


Figura 2. Diagrama del Hashing por Residuo [1].

3.2 Método de medio cuadrado

Se basa en elevar al cuadrado la clave y tomar los dígitos centrales como la ubicación. El número de dígitos a tomar queda determinado por el rango del índice. Sea K la clave del dato a buscar, la función Hash queda con la siguiente fórmula:

$$H(K)=\text{dígitos centrales}(K^2) +1 \quad (3)$$

Esencialmente se toman los dos números que se encuentran en el centro de la cadena ya generada en el proceso de conversión, se suman y luego se elevan al cuadrado, finalmente al resultado anterior se le suma uno "1", generando una llave que se puede aplicar dentro del arreglo [3].

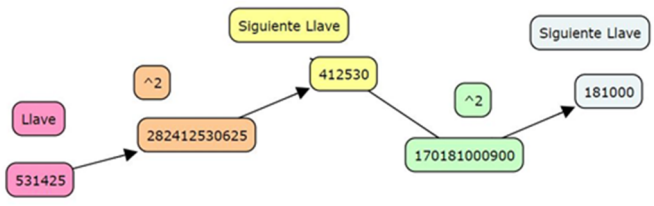


Figura 3. Diagrama del Hashing por Medio Cuadrado [3].

3.3 Caso de estudio

Con la inquietud de aportar como equipo al tema en estudio, trabajamos en la programación de un método en lenguaje JAVA, ya que al probar los métodos que investigamos descubrimos diferentes problemas como:

3.3.1 Hash por residuo:

- La variable N (factor) tenía que ser un número primo inferior al tope, la prueba resultó en algunas llaves grandes que no podían ser utilizadas en arreglos pequeños.
- Generó muchas colisiones.

3.3.2 Hash por medio cuadrado:

- Este método no puede ser implementado en arreglos pequeños.
- Al aplicarlo se necesitó de muchos procesos, ya que consta de especificaciones como: seleccionar dos de los dígitos centrales de la cadena de números. Esto implica que el valor numérico que salía de la conversión tenía que pasarse de nuevo a una variable de tipo *String* (Cadena de caracteres) para poder determinar sólo los dos números que se iban a usar.

Ambos métodos indicaban errores que se evidenciaron en los ensayos, al intentar solucionar una colisión de una colisión; ambos tenían desventajas:

- No contaban de un método propio para evitar las colisiones y hacer búsquedas de llaves.
- Son estáticos, al punto de no poder responder a colisiones continuas.

Considerando la experiencia descrita y todo el análisis teórico planteado, creamos el método que denominamos el Método del Módulo de la Cadena.

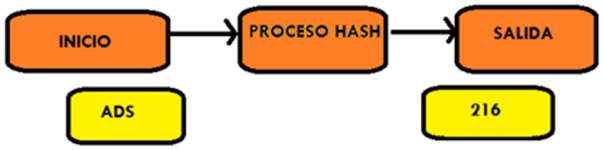


Figura 4. Gráfica del funcionamiento Método del Módulo de la Cadena.

En resumen el programa trabaja de la siguiente manera:
Paso1: Se ingresa la cadena que se desea guardar.
Paso2: Se hace una sumatoria de códigos ASCII de cada carácter de la cadena para que al final quede una sola variable.
Paso3: Generar la llave mediante la fórmula:
llave = ((Sumatoria ASCII / ASCII del Carácter de la Cadena) mod Tope) +1

Básicamente consiste en:

- Tomar el valor resultante de la sumatoria ASCII.
- Dividir el resultado anterior entre el valor ASCII del primer carácter y tomar el cociente.
- Dividir el cociente entre el tope o tamaño total del arreglo.
- Extraer solo el módulo de la operación anterior y sumarle uno "1".

Paso Auxiliar*:* si se produjera una colisión después de la búsqueda de la llave y si está ocupado el registro, se pasa a la fórmula auxiliar que se muestra:

sumaASCII= sumaASCII + (llave^2) + ((ASCIIcaracter(i))^3)

Nota: este ciclo se repetirá hasta que ya no ocurra ninguna colisión. Además, se incluye una variable contador que tiene como tope el tamaño de la cadena que se ingresó para que no resulte ningún error.
Paso 4*:* almacenamiento del elemento en el registro que indica la llave.

Concretamente, las fórmulas del Método del Módulo de la Cadena son las siguientes:

llave= ((Sumatoria ASCII / ASCII del Carácter de la Cadena) mod Tope) +1 (4)

sumaASCII= sumaASCII + (llave^2) + ((ASCIIcaracter(i))^3) (5)

El siguiente Diagrama de Nassi-Shneiderman ilustra el Método del Módulo de la Cadena:

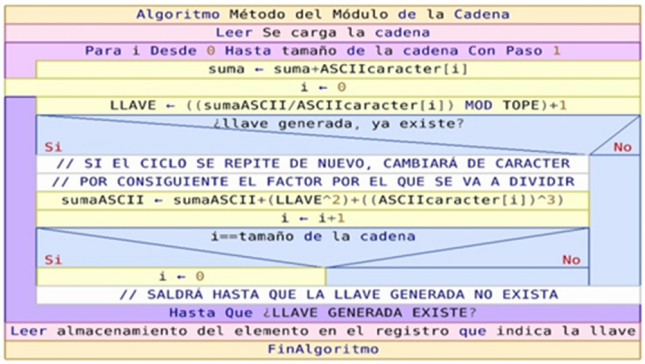


Figura 5. Diagrama de Nassi-Shneiderman del Método del Módulo de la Cadena.

4. Colisiones

Errores causados por la asignación de múltiples de datos que van dirigidos a una misma llave.

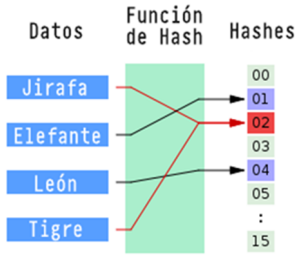


Figura 6. Diagrama donde se ejecuta una colisión [1].

En la imagen anterior se observa que el dato Jirafa es insertado y se le asigna una llave (02).

Al momento de insertar el dato Tigre se le sitúa en la misma llave (02); esto provoca la pérdida de datos.

4.1 Métodos para evitar colisiones

- **Propagar los registros:** Distribuyendo de manera aleatoria los registros podemos evitar "agrupaciones" de llaves que produzcan las mismas direcciones.
- **Usar memoria adicional:** Dispone de almacenar un espacio de direcciones mayor que el número de registros a usar, de modo que, si vamos a insertar 100 registros, tener un espacio de 500 direcciones, evita errores de almacenamiento.
- **Colocar registros adicionales en una dirección:** A diferencia de los anteriores donde cada dirección almacena únicamente un registro, como si fueran casilleros; este método coloca los registros que colisionan en un único registro, de manera que al hacer la búsqueda debemos recuperar el casillero entero y entonces buscar el registro deseado.

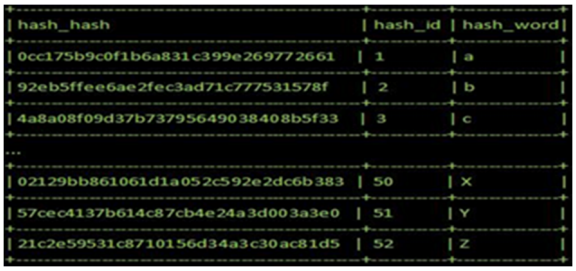
4.2 Comparación de los métodos para búsqueda de registros

- Búsqueda Secuencial: Busca un elemento comparándolos secuencialmente con cada uno de los elementos existentes en algún tipo de arreglo, hasta que lo encuentre. Se utiliza ya cuando el vector no se encuentra ordenado o no se pueda ordenar debido a ciertas condiciones [3].
- Búsqueda Binaria: Se divide y se compara con el elemento central, si el elemento es menor se busca en la izquierda y si es mayor hacia la derecha. El Arreglo debe estar ordenado para poder trabajar este método [3].

5. Usos de Hash

- Cuidar la confidencialidad de las contraseñas [8].
- Respaldar la seguridad de los datos.
- Autentificación de los documentos, textos, imágenes entre otros tipos de archivos [6].
- Este método se utiliza en informática forense para confirmar la autenticidad de los datos extraídos con los datos a procesar. Esto nos daba la seguridad que los datos no habían sido alterados.
- Las funciones Hash están inmersas en el protocolo Bitcoin en dos importantes áreas: direcciones y minería, relacionadas directamente con la generación de las claves públicas y privadas usadas en esta tecnología.
- Para descifrar datos almacenados en medios volátiles o RAM, utilizábamos Rainbows Tables o tablas conjuntas de Hash para acceder a esos datos.

Las Rainbows Tables son un bloque de tablas Hash que asemejan la combinación de una clave de datos cifrados.



hash_hash	hash_id	hash_word
0cc175b9c0f1b6a831c399e269772661	1	a
92eb5ffee6ae2fec3ad71c777531578f	2	b
4a8a08f09d37b73795649038408b5f33	3	c
...		
02129bb861061d1a052c592e2dc6b383	50	X
57cec4137b614c87cb4e24a3d003a3e0	51	Y
21c2e59531c8710156d34a3c30ac81d5	52	Z

Figura 7. Uso de una Rainbow Table.

6. Resultados

El equipo tomó la información y los datos que recabó en esta investigación y los llevó a un plano físico con limitaciones, como un espacio de almacenamiento de cien registros. Como resultado se comprobó que uno de los métodos (Hash Medio Cuadrado) que investigamos no podría ser utilizado en estos casos ya que las llaves que generaban estaban fuera del rango del arreglo.

El otro método (Hash por Residuo) evidenció resultados buenos, llaves precisas, fáciles de manejar y utilizar.

Sin embargo, los dos métodos tenían un pequeño problema que no solucionaban: si ellos al generar una llave podrían ocasionar una colisión no tendrían métodos para contrarrestar este problema.

Es la oportunidad de innovar, por lo cual nosotros decidimos desarrollar un método que denominamos "**El Método del Módulo de la Cadena**" el cual contiene dos fórmulas:

- Fórmula Generadora de Llaves (llave= ((sumaASCII / ASCIIcaracter(i)) mod TOPE) +1): esta fórmula es capaz

de generar llaves de forma precisa teniendo como entrada solo la cadena que se va a procesar. De manera automática se ejecutan los cálculos que crea la llave.

- Fórmula Auxiliar Contrarrestradora de Colisiones (sumaASCII=sumaASCII + (llave^2) + (ASCIIcaracter(i)^3)): es utilizada para los problemas de colisiones que pueden ocurrir después de la generación de una llave.

7. Conclusiones

La información genera conocimiento y este es a su vez poder. En la medida que la información esté asegurada mediante técnicas Hash como las contempladas en este documento, las empresas obtendrán competitividad al ofrecerles a sus clientes la garantía de la protección de la información. Sin embargo, ser conscientes de que algunas técnicas revelan algunos vacíos también es importante para discriminar entre los mecanismos Hash más eficientes en base a la particular confidencialidad que se demanda.

Casos reales en informática forense demuestran cómo los datos, principalmente cuentas de banco, son atacados en secreto, cometiendo delitos informáticos donde se modifican documentos, imágenes, tablas con fines deshonestos. En este contexto cobran importancia los métodos que evitan colisiones y salvaguardan información.

Opinamos que por el método Binario se maneja mejor la búsqueda de datos, por el hecho de ser fácil y da la confianza de que no tendremos problemas al comparar elementos del arreglo. Sin embargo, los datos almacenados tienen que estar ordenados, lo cual puede asumirse como desventaja.

En cambio, la búsqueda secuencial no necesita estar ordenada, pero requiere un mayor tiempo de búsqueda en muchos casos; debido a que tiene que analizar todo el arreglo para poder encontrar el elemento.

Finalmente, con el método del Hash el problema que ocurre es que si se dan casos de valores o datos muy largos puede ser un poco inestable, sin mencionar el problema de las colisiones que presentan.

Por otro lado, el Método de Transformación de Llaves considera un enfoque de seguridad de datos, pues no solo comprueba ficheros, si no también verifica que la integridad no haya sido modificada.

Al aplicar todos estos conocimientos nos percatamos de que el Hash es necesario para ver errores de almacenamiento y colisiones de información, por ello sugerimos que el Hash se implemente en todo tipo de procesos que manipulen información y así evitar que sean plagiados o almacenados de una forma inadecuada en la que se pueda perder información.
Como trabajos futuros se tiene contemplado mejorar el código desarrollado en Java para el Método del Módulo de la cadena y programarlo en otros lenguajes, además de probarlo en diversos DBMS.

También deseamos explorar la relación de Hash con el soporte para sistemas embebidos y de clúster. Y siendo más audaces, sería interesante la posibilidad de implementar versiones más recientes del kernel (4.9.9), que incluye el soporte a OrangeFS y USB 3.1.

En las secciones y subsecciones no deben utilizar más de tres (3) niveles de títulos. Otros títulos (subsecciones) deben tener una fuente de 10pts excepto los títulos de primer nivel, que son de 14pts. La letra inicial de cada palabra en el título debe ser en mayúscula excepto para las palabras cortas.

- Primer nivel de título: un título en el nivel 1 debe estar justificado a la izquierda, fuente Times New Roman, tamaño 14pts y enumerado con números arábigos seguido por un punto, ejemplo: ver el título “3. Estilo de página” de este documento. Los títulos “Agradecimientos” y “Referencias” no deben ser enumerados.
- Segundo y tercer nivel de título: un título en el nivel 2 y 3 deben estar justificado a la izquierda y enumerado con números arábigos. Por ejemplo, ver el título “3.3Títulos de las secciones” o “3.4.1 Título de figura”.

REFERENCIAS

[1] Osvaldo Cairo, Silvia Guardati,”Estructura de Datos”, tercera edición, fecha de consulta: 2/05/2017.

[2] N. Wirth 1985, "Algorithms and Data Structures", Oberon version: August 2004), fecha de consulta: 3/05/2017.

[3] Ana Castro, “función Hash”, URL: https://es.slideshare.net/analivecastrovargas/funcion-Hash-metodos-de-divisi , 2014, fecha de consulta: 3/05/2017.

[4] Autor Anónimo “Búsqueda por Hash”, URL: https://es.slideshare.net/sykrayo/busqueda-por-Hash, 2011.

[5] Edgar Blake, “Hash”, URL: https://prezi.com/p4rvhh_hmf9b/Hash-residuo-de-division/, 2012, fecha de consulta: 12/05/2017.

[6] Diego García Ordás, Laura Fernández Robles, María Teresa García Ordás, Óscar García-Olalla, Enrique Alegre Gutiérrez, “Comprobar modificación de Imagenes”, URL: http://www.asasec.eu/docs/results/result3/ASASEC_Articulo_Encontrar_imagen_modificada.pdf, fecha de consulta: 7/05/2017.

[7] Jesper M Johansson, Josh D. Benaloh, “Password protection’’,URL: https://www.google.com/patents/US7602910, 2009, fecha de consulta: 8/05/2017.

[8] Anyi Bernal, “Algoritmos de Búsqueda Hash”, URL:https://prezi.com/ghlikcvrek1o/algoritmos-de-busqueda-Hash, Fecha de creación: 18/05/2016, Fecha de consulta: 6/06/2017.

[9] Paulaila Toledo, “Búsqueda mediante transformación de claves”, URL: https://paulailatoledo.wordpress.com/2012/06/15/busqueda-mediante-transformacion-de-claves-Hashing/, fecha de consulta: 6/6/2017.

[10] Autor Anónimo, “El Código ASCII”, URL: http://www.elcodigoascii.com.ar/, fecha de consulta: 1/9/2017.