COLOMBIA POTENCIA DE LA VIDA

TIC

Funciones **de Hash**

INTEGRADOR-Lección 2 -Unidad 2

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Tiempo de ejecución: 4 horas

PLANTEAMIENTO DE LA SESIÓN	Materiales
En la segunda sesión se estudiarán las funciones de hash y cómo estas tienen una gran importancia en el blockchain al permitir identificar cada bloque de manera inequívoca y convirtiéndolos en registros inmutables para que ningún usuario de la red pueda modificarlo a conveniencia. Se va a hacer énfasis en los primeros pasos para calcular la función SHA-256.	Calculadora de hash https://es.infobyip.com/hashcalculator.php

Desarrollo teórico de la sesión: 2 horas

Algoritmos de hashing

Los algoritmos de hashing son herramientas que convierten datos en valores numéricos más pequeños y de longitud fija, conocidos como valores hash. Estos valores son representaciones numéricas de segmentos de datos y cuando se aplica un algoritmo hash a un párrafo de texto y se realiza un cambio incluso mínimo, el valor hash resultante será diferente. Si el algoritmo hash es seguro desde el punto de vista criptográfico, un cambio en los datos debería generar un valor hash completamente distinto. Por ejemplo, si se altera un solo bit en un mensaje, una función hash segura podría producir un resultado que difiere considerablemente, posiblemente en un 50%. Aunque es posible que diferentes entradas den como resultado el mismo valor hash, encontrar dos entradas distintas con el mismo valor hash es poco factible, esto se llama resistencia a colisiones.

Una función hash es determinista e irreversible, si se da el mismo mensaje de entrada a la función, siempre va a devolver el mismo hash, pero no se hace un proceso inverso en el cual se pueda saber lo que dice el mensaje original únicamente usando el hash resultante. Hay muchas funciones para obtener un hash, entre ellas está la función SHA-256 (Secure Hash Algorithm 256-bit), es la función de hashing que se usa en el blockchain.

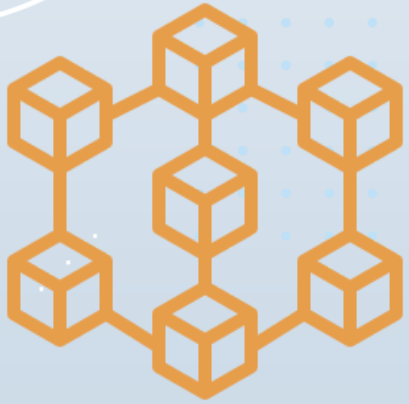
Para validar el funcionamiento de una función hash y cómo un cambio en el mensaje puede alterar el resultado, se va a usar la entrada “Cadena de Bloques” con algunas variaciones y la función SHA-256 para ver el hash resultante.

+En el siguiente enlace se puede acceder a una calculadora de hash:

Entrada (mensaje)	Salida (hash)
Cadena de bloques	19fdd2502ab57f3aef74c02748de2409731c0fb885c7051488e1c3c75ad442da
Cadena de bloques	935c5266596bbb97e9a51471765d8e1a399be5dc31a49cb43c68456b63be252c
Cadena de bloques	127879096b0e5161f9f068a104557df2eb7364bc4a8c117619594932d682c899
Cadena de bloques	6e6340084230b3556b519a0b32e9c7b84eb19c3d4f59494f1a6c5c4627cad08b
Cadena de bloques	19fdd2502ab57f3aef74c02748de2409731c0fb885c7051488e1c3c75ad442da

Tabla 1: Función SHA-256

Como se puede ver en la tabla 1, al cambiar solamente una letra de minúscula a mayúscula obtenemos un hash distinto al de la entrada "Cadena de bloques", y si al final se introduce la primera entrada, retorna el mismo hash de la primera vez en todas las ocasiones. Con esto se aprecia por qué es utilizada en blockchain, al intentar alterar los registros que hay en la cadena el hash va a hacer que los usuarios noten la discrepancia. Por este motivo, el blockchain puede crear una "Cadena Inmutable", y en caso de que alguien altere dicha cadena deliberadamente, se puede rastrear de cuál nodo provino en la red y tomar las medidas necesarias para asegurar la integridad de los registros.



Conversión a hexadecimal:

El resultado al aplicar la función SHA-256 es una cadena de 256 bits de longitud expresada en sesenta y cuatro (64) caracteres hexadecimales o base dieciséis (16). Para procesar el mensaje de entrada se debe realizar la conversión a formato hexadecimal. Este proceso sustituye cada carácter ASCII por su número hexadecimal equivalente. Por ejemplo, tomando la cadena w = “Hola mundo”, se realizan las conversiones como se muestra en la figura 2:

+ info

Char		*Dec*		*Hex*
H	$\Rightarrow$	72	$\Rightarrow$	48
o	$\Rightarrow$	111	$\Rightarrow$	$6f$
l	$\Rightarrow$	108	$\Rightarrow$	$6c$
a	$\Rightarrow$	97	$\Rightarrow$	61
	$\Rightarrow$	32	$\Rightarrow$	20
m	$\Rightarrow$	109	$\Rightarrow$	$6d$
u	$\Rightarrow$	117	$\Rightarrow$	75
n	$\Rightarrow$	110	$\Rightarrow$	$6e$
d	$\Rightarrow$	100	$\Rightarrow$	64
o	$\Rightarrow$	111	$\Rightarrow$	$6f$

Figura 2: Conversión de entrada a hexadecimal.

Como se ve en la **figura 2**, se aplica el proceso de conversión de ASCII a hexadecimal y se concatena el mensaje de salida.

$$Hola\ mundo = 486f6c61206d756e646f$$

Figura 2.1: Mensaje en hexadecimal

El mensaje hexadecimal en la **figura 2.1** se debe reservar a parte para usarlo posteriormente en otras funciones.

Cálculo de la longitud M

Se debe hacer el cálculo de la longitud del mensaje de entrada para hallar el hash. El primer paso es calcular la longitud en bits. Consiste en multiplicar el número total de caracteres ASCII de la entrada por 8, debido a que cada carácter equivale a 8 bits.

$$\underset{Hola\ mundo}{M} = \underset{8 \times 10}{Bits \times Chars} = \underset{80}{Dec} \Rightarrow \underset{50}{Hex} \quad \underset{|M|_{16} = 50}{Longitud}$$

Figura 2.2: Calculo de longitud M.

+ info

Como se aprecia en la **figura 2.2**, se multiplica por 8 cada carácter en el mensaje para obtener su valor decimal, incluyendo el espacio (“ “). Luego, se obtiene el equivalente en hexadecimal. Este dato se debe guardar al igual que el mensaje en hexadecimal para operaciones en futuras funciones.

Construcción de la variable W_t:

Esta variable es un array de 64 elementos que contienen palabras hexadecimales de 32 bits. Su longitud es de 2048 bits o 256 bytes y se obtiene mediante una función recursiva que se define por intervalos como se muestra en la figura 2.3:

$$W_t = \begin{cases} M_i & si \quad 0 \leq i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & si \quad 16 \leq i < 64 \end{cases}$$

Figura 2.3: Función recursiva de la variable Wt.

En la **figura 2.3** aparecen las funciones σ0 y σ1, las cuales realizan operaciones lógicas de compresión como se muestra en la figura 2.4:

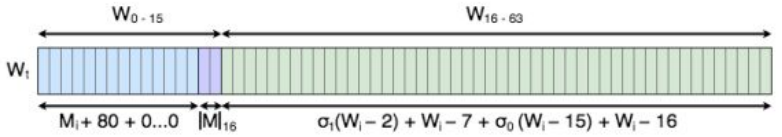
$$\sigma_0(x) = ROT\ R^7(x) \oplus ROT\ R^{18}(x) \oplus SHR^3(x)$$
$$\sigma_1(x) = ROT\ R^{17}(x) \oplus ROT\ R^{19}(x) \oplus SHR^{10}(x)$$

Figura 2.3: Función recursiva de la variable Wt.

+ info

En la **figura 2.4** aparecen algunos algoritmos ROT, los cuales consisten en una operación de rotación de bits en la variable x. Su funcionamiento es el mismo que algoritmos como ROT5 y ROT13, pero aplicado a bits.

Retomando la **figura 2.3**, se muestra que el primer intervalo abarca los primeros 16 registros, desde W0 hasta W15. El segundo intervalo abarca los 48 registros restantes, desde W16 hasta W65. Ambos intervalos cuentan con reglas propias en la aplicación de los cálculos.



En la figura 2.5 se puede apreciar gráficamente los intervalos de la función Wt explicada anteriormente.

Figura 2.5: Representación gráfica de la función Wt .

Los primeros 16 registros de W_t:

En el primer intervalo de Wt, los 16 primeros registros se reservan para almacenar el mensaje de entrada M en formato hexadecimal, dividido en bloques de 32 bits. Este mensaje tiene una longitud de 512 bits. La longitud del mensaje puede variar, incluso siendo una cadena vacía. Independientemente de la longitud, se utiliza su representación hexadecimal y se divide en palabras de 32 bits de izquierda a derecha. Si algún bloque no ocupa los 32 bits completos, los bits restantes se dejan vacíos para completarlos más adelante.

$$Hola\ mundo = \overbrace{486f6c61}^{32\ bits}\ \overbrace{206d756e}^{32\ bits}\ \overbrace{646f}^{32\ bits}$$

Figura 2.6: Representación del mensaje en bits.

Como se muestra en la figura 2.6, se forman bloques de 32 bits con el mensaje de entrada, en caso de que alguno de los bloques no ocupe esta cantidad, simplemente se debe dejar vacío.

Luego, se toma un bit que representa el número “1” en decimal o base 10, es decir 00000001, este se debe desplazar al bit más alto del byte, con lo que se obtiene 10000000 y al final se calcula el valor hexadecimal, como se muestra en la figura 2.7:

$$10000000_2 = 80_{16}$$

Figura 2.7: Cálculo del valor hexadecimal.

Sin importar la longitud de la cadena hexadecimal de la entrada, se añade 80 por la derecha, como se muestra en la figura 2.8:

$$486f6c61 + 206d756e + 646f + 80$$

Figura 2.8: Agregación del 80 en la cadena hexadecimal.

Después, se debe añadir una cantidad de bits con valor 0 hasta llegar a 448 bits en total. 448 bits es la longitud que abarca todos los intervalos desde W_0 hasta W_{13}.

+ info

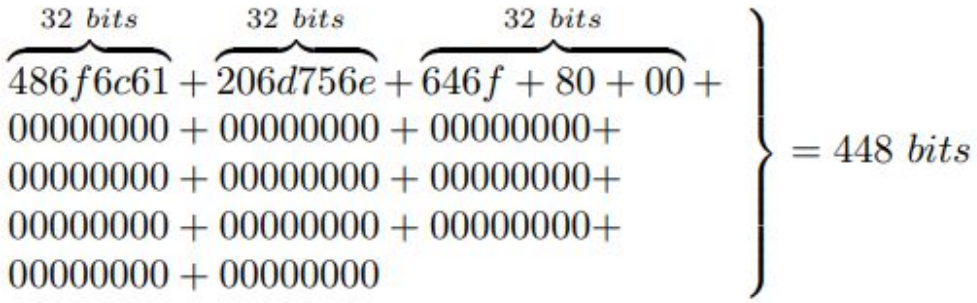


Figura 2.9: Agregación de bits con valor 0.

Como se muestra en la figura 2.9, se toma el resultado anterior y se rellena con los bits de valor 0 para alcanzar la longitud necesaria de la función.

Ahora solo se deben rellenar los últimos dos bloques de 32 bits desde W14 y W15 con la longitud del mensaje de entrada en hexadecimal, se deben poner bits de valor 0 por la izquierda para alcanzar 64 bits de longitud. Este es el dato que se obtuvo en el cálculo de longitud de M, para la cadena “Hola mundo” es 50.

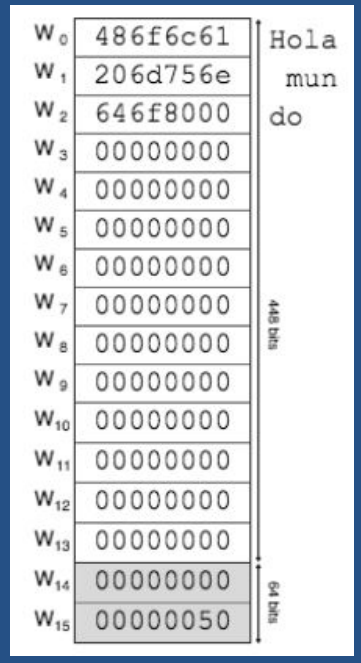


Figura 2.10: Relleno de las primeras 16 posiciones.

En la **figura 2.10** se observa el resultado de las primeras 16 posiciones combinando los resultados hexadecimales y los rellenos que se hacen con bits para alcanzar la longitud necesaria.

Constante K_t:

Esta se compone de 64 palabras hexadecimales. Cada palabra representa los primeros 32 bits en hexadecimal de la parte fraccionaria de la raíz cúbica de los primeros 64 números primos, desde el 2 hasta el 311.
A partir de este punto, se empiezan los cálculos de las 64 palabras de Kt, el cálculo de las 8 palabras iniciales, y las rondas criptográficas, las cuales introducen funciones y cálculos más avanzados los cuales no se van a dictar en esta lección por su alta complejidad.

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira