Informe

Santiago Forero Gutierrez – 202111446
Santiago Tenjo Venegas – 202113965
Jonathan Obed Cabanzo Certuche – 201911749

Simulador de ataque de fuerza bruta
Universidad de los Andes, Bogotá, Colombia
{s.forerog2, s.tenjov, jo.cabanzo }@uniandes.edu.co
Fecha de presentación: abril 29 de 2023

Tabla de contenido

1 Introducción

El presente reporte busca analizar el nivel de protección que ofrecen los algoritmos de Hash SHA-256 y SHA-512, para la protección de contraseñas. A grandes rasgos, el programa prueba distintos valores ($\boldsymbol{V_n}$) en un espacio de búsqueda de entre 1 y 7 caracteres, contando únicamente con combinaciones de las letras de la a-z, una secuencia de dos valores que representa la sal **(S)** y la opción de escoger entre uno o dos threads a usar. El funcionamiento del programa se basa en buscar un valor $\boldsymbol{V_k}$ de los diferentes ($\boldsymbol{V_n}$), tal que su código criptográfico de hash concatenado **S**: H **($\boldsymbol{V_k}$ concat S)** sea igual a un valor de Hash dado**.**

2 Funcionamiento del sistema

El sistema se compone de dos clases principales. La primera de ellas, **EvaluadorProteccion**, tiene la responsabilidad de determinar los parámetros de entrada según el funcionamiento requerido, ya sea con 1 o 2 threads y un algoritmo de Hash SHA-256 o SHA-512. Además, se encarga de determinar si los parámetros serán proporcionados por consola o a través de un archivo de inputs que sigue el formato siguiente:

- Tipo de algoritmo de hash (SHA-256 o SHA-512), hash buscado, sal (Cadena de 2 caracteres), numero de threads (2 o 1)

La segunda clase, **HiloDescifrador**, es una subclase de **Thread**, lo que permite que una o dos instancias de esta se ejecuten de manera concurrente. Esta clase tiene la tarea de explorar el espacio de búsqueda para encontrar la cadena de caracteres $\boldsymbol{V_k}$ concatenada con **S** cuyo valor de hash coincide con el valor de hash deseado.

3 Pruebas realizadas

Se llevaron a cabo diversas pruebas para evaluar la eficiencia del algoritmo y determinar hasta qué punto la complejidad de una contraseña influye en su seguridad. Para analizar la velocidad del algoritmo en cada prueba, se realizaron cambios en los parámetros, como la complejidad de la contraseña, los algoritmos de Hash empleados y el número de threads. A continuación, se presenta una tabla con los resultados obtenidos:

Valor (Contraseña)	Tiempo de ejecución	Algoritmo	Sal	Numero de Threads
s	32 ms	SHA-256	kl	1
s	1 ms	SHA-256	rf	1
s	4 ms	SHA-512	kl	1
s	2 ms	SHA-512	rf	1
s	1 ms	SHA-256	kl	2
s	1 ms	SHA-256	rf	2
s	1 ms	SHA-512	kl	2
s	1 ms	SHA-512	rf	2
z	2 ms	SHA-256	kl	1
z	1 ms	SHA-256	rf	1
z	2 ms	SHA-512	kl	1
z	1 ms	SHA-512	rf	1
z	1 ms	SHA-256	kl	2
z	1 ms	SHA-256	rf	2
z	1 ms	SHA-512	kl	2
z	0 ms	SHA-512	rf	2
vf	11 ms	SHA-256	kl	1
vf	7 ms	SHA-256	rf	1
vf	7 ms	SHA-512	kl	1
vf	6 ms	SHA-512	rf	1
vf	8 ms	SHA-256	kl	2
vf	1 ms	SHA-256	rf	2
vf	4 ms	SHA-512	kl	2
vf	6 ms	SHA-512	rf	2
zz	7 ms	SHA-256	kl	1
zz	2 ms	SHA-256	rf	1
zz	5 ms	SHA-512	kl	1
zz	8 ms	SHA-512	rf	1
zz	1 ms	SHA-256	kl	2
zz	1 ms	SHA-256	rf	2
zz	2 ms	SHA-512	kl	2
zz	2 ms	SHA-512	rf	2
hoj	9 ms	SHA-256	kl	1
hoj	5 ms	SHA-256	rf	1
hoj	7 ms	SHA-512	kl	1
hoj	9 ms	SHA-512	rf	1
hoj	4 ms	SHA-256	kl	2
hoj	9 ms	SHA-256	rf	2
hoj	5 ms	SHA-512	kl	2

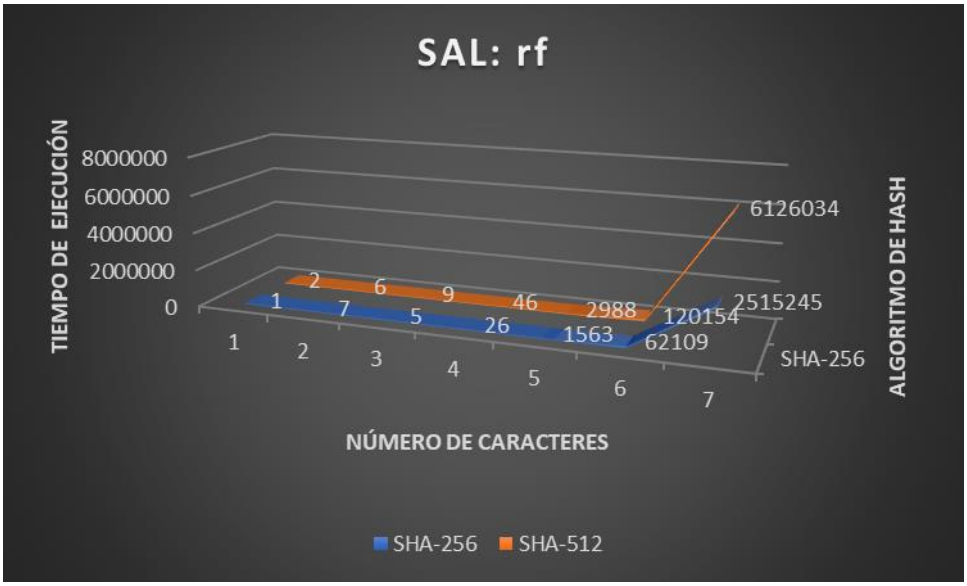
hoj	6 ms	SHA-512	rf	2
zzz	12 ms	SHA-256	kl	1
zzz	18 ms	SHA-256	rf	1
zzz	30 ms	SHA-512	kl	1
zzz	29 ms	SHA-512	rf	1
zzz	12 ms	SHA-256	kl	2
zzz	15 ms	SHA-256	rf	2
zzz	18 ms	SHA-512	kl	2
zzz	18 ms	SHA-512	rf	2
chen	44 ms	SHA-256	kl	1
chen	26 ms	SHA-256	rf	1
chen	40 ms	SHA-512	kl	1
chen	46 ms	SHA-512	rf	1
chen	12 ms	SHA-256	kl	2
chen	14 ms	SHA-256	rf	2
chen	22 ms	SHA-512	kl	2
chen	20 ms	SHA-512	rf	2
zzzz	270 ms	SHA-256	kl	1
zzzz	205 ms	SHA-256	rf	1
zzzz	408 ms	SHA-512	kl	1
zzzz	332 ms	SHA-512	rf	1
zzzz	103 ms	SHA-256	kl	2
zzzz	125 ms	SHA-256	rf	2
zzzz	161 ms	SHA-512	kl	2
zzzz	159 ms	SHA-512	rf	2
infra	1632 ms	SHA-256	kl	1
infra	1563 ms	SHA-256	rf	1
infra	3386 ms	SHA-512	kl	1
infra	2988 ms	SHA-512	rf	1
infra	846 ms	SHA-256	kl	2
infra	874 ms	SHA-256	rf	2
infra	1684 ms	SHA-512	kl	2
infra	1790 ms	SHA-512	rf	2
zzzzz	4463 ms	SHA-256	kl	1
zzzzz	4407 ms	SHA-256	rf	1
zzzzz	8307 ms	SHA-512	kl	1
zzzzz	8373 ms	SHA-512	rf	1
zzzzz	2478 ms	SHA-256	kl	2
zzzzz	2427 ms	SHA-256	rf	2
zzzzz	4666 ms	SHA-512	kl	2

zzzzz	4535 ms	SHA-512	rf	2
lluvia	57356 ms	SHA-256	kl	1
lluvia	62109 ms	SHA-256	rf	1
lluvia	120466 ms	SHA-512	kl	1
lluvia	120154 ms	SHA-512	rf	1
lluvia	44049 ms	SHA-256	kl	2
lluvia	39891 ms	SHA-256	rf	2
lluvia	78077 ms	SHA-512	kl	2
lluvia	73018 ms	SHA-512	rf	2
zzzzzz	139410 ms	SHA-256	kl	1
zzzzzz	138009 ms	SHA-256	rf	1
zzzzzz	255459 ms	SHA-512	kl	1
zzzzzz	277265 ms	SHA-512	rf	1
zzzzzz	93702 ms	SHA-256	kl	2
zzzzzz	96552 ms	SHA-256	rf	2
zzzzzz	177099 ms	SHA-512	kl	2
zzzzzz	167414 ms	SHA-512	rf	2
sistema	2557405 ms	SHA-256	kl	1
sistema	2515245 ms	SHA-256	rf	1
sistema	5286286 ms	SHA-512	kl	1
sistema	6126034 ms	SHA-512	rf	1
sistema	1791854 ms	SHA-256	kl	2
sistema	1659909 ms	SHA-256	rf	2
sistema	2901887 ms	SHA-512	kl	2
sistema	2804978 ms	SHA-512	rf	2
zzzzzzz	3472928 ms	SHA-256	kl	1
zzzzzzz	3436617 ms	SHA-256	rf	1
zzzzzzz	6275060 ms	SHA-512	kl	1
zzzzzzz	6330520 ms	SHA-512	rf	1
zzzzzzz	2290289 ms	SHA-256	kl	2
zzzzzzz	2479143 ms	SHA-256	rf	2
zzzzzzz	4281742 ms	SHA-512	kl	2
zzzzzzz	3781438 ms	SHA-512	rf	2

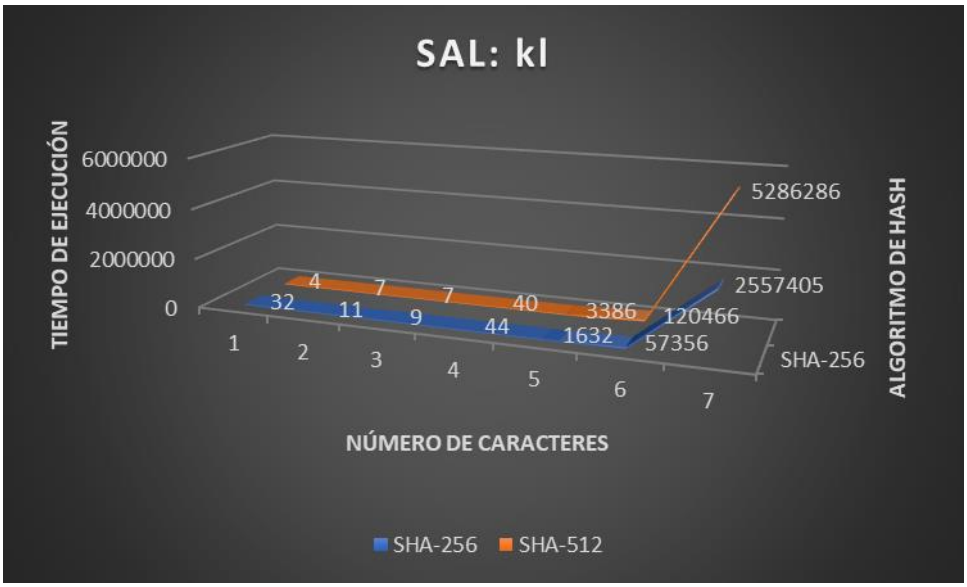
4 Gráficas

Después de analizar los datos mencionados anteriormente y con vías a entender de manera más profunda la complejidad del algoritmo proponemos entender las gráficas independientemente de la cantidad de threads (1 o 2), como se muestra a continuación:

- Se cruzan los datos de la siguiente manera: algoritmo ejecutado x número de caracteres x tiempo de ejecución y una cantidad de **1 thread** se obtienen los siguientes resultados:

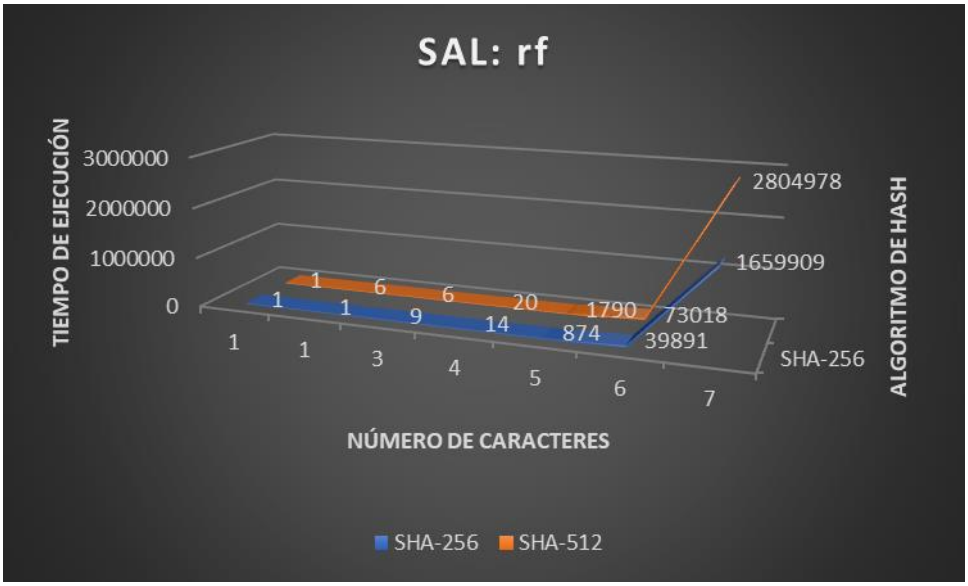


GRAFICA 1: NUMERO DE THREADS: 1, SAL: 'RF'

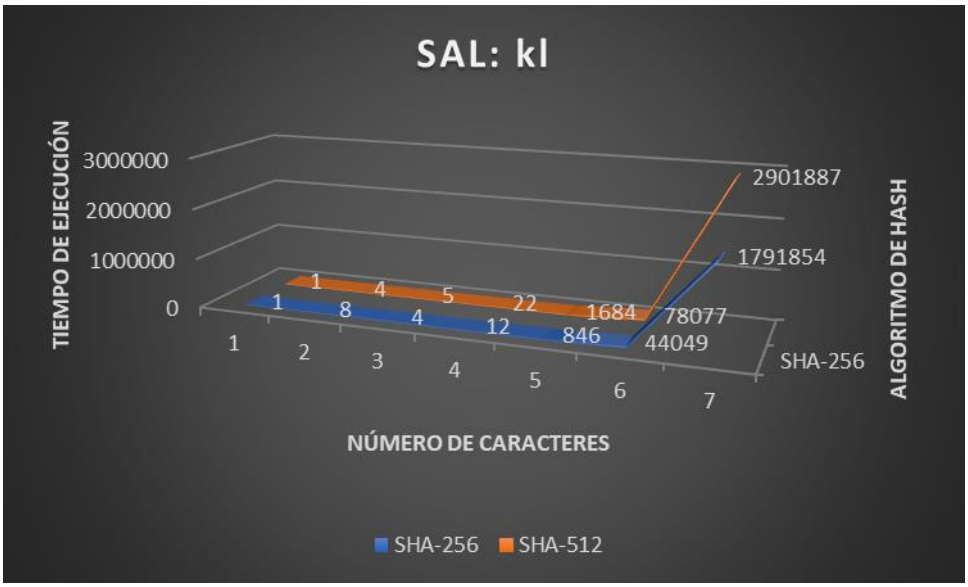


GRAFICA 2: NUMERO DE THREADS: 1, SAL: 'KL'

- De la misma manera se realizan las gráficas para la ejecución del algoritmo con **2 threads** y se obtienen los siguientes resultados:



GRAFICA 3: NUMERO DE THREADS: 1, SAL: 'RF'



GRAFICA 4: NUMERO DE THREADS: 1, SAL: 'KL'

De las gráficas anteriores podemos realizar las siguientes conclusiones: En primera instancia, dado que trabajamos bajo la suposición de que tenemos conocimiento de la **sal**, esta no agrega complejidad al problema, por lo que por cada par de graficas (Para cada thread) el resultado final de tiempos tiende a ser el mismo sin importar la sal.

Al analizar la gráfica 1 que representa la ejecución con un único thread, podemos observar que los tiempos finales de ejecución para cada algoritmo son de 6126034 ms para SHA-512 y 7 caracteres, y de 2515245 ms para SHA-256 y 7 caracteres. Es evidente que se obtiene un tiempo de ejecución mucho más rápido (en promedio un 150% menor) al utilizar el algoritmo SHA-256.

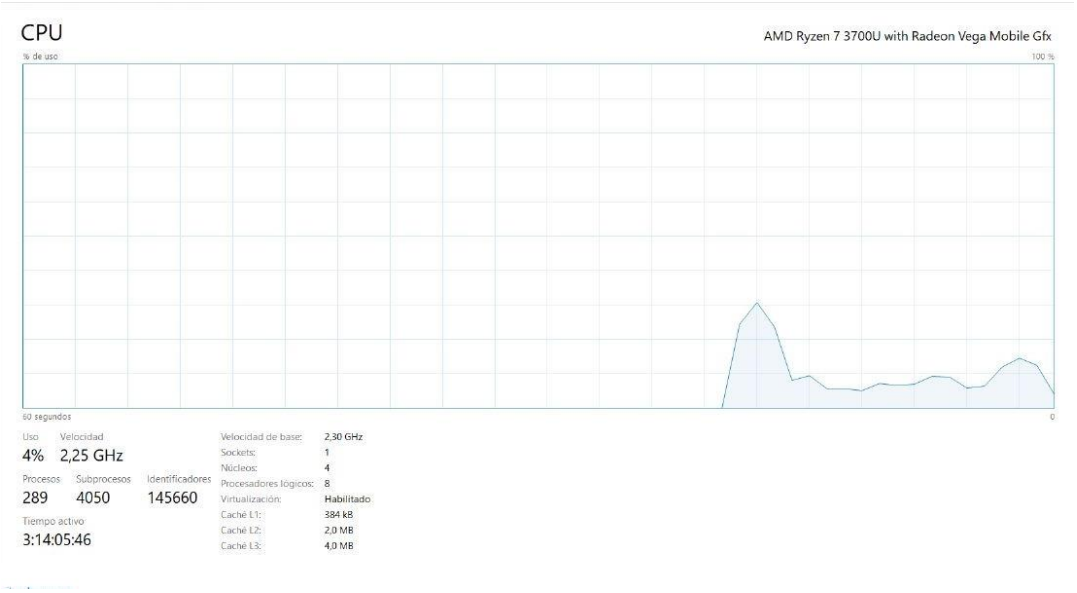
Al examinar una de las dos gráficas que representan la ejecución con 2 threads **(grafica 4)**, se puede observar que los tiempos finales de ejecución para cada algoritmo son de 2901887 ms para SHA-512 y 7 caracteres, y de 1791854 ms para SHA-256 y 7 caracteres. Estos resultados son significativamente más eficientes que los observados anteriormente, ya que se reduce a casi un tercio el tiempo de respuesta. Esto resalta la importancia del uso de multiprogramación en

este tipo de ataques, ya que reduce drásticamente el tiempo en el que una contraseña puede ser descifrada.

En resumen, hemos concluido que el algoritmo SHA-256 es más eficiente que el SHA-512 en términos de tiempo de ejecución para buscar contraseñas, aunque también es menos seguro. Además, hemos observado que implementar varios threads reduce significativamente los tiempos de respuesta, lo que significa que con un mayor número de máquinas y threads, una contraseña con baja complejidad podría ser descifrada rápidamente. Por lo tanto, es crucial usar contraseñas más complejas para aumentar los tiempos de respuesta y mejorar la seguridad de estas.

5 Cálculo de velocidad del procesador

Las pruebas se realizaron en un maquina con las siguientes características:



- Identifique la velocidad de su procesador, y estime cuántos ciclos de procesador tomaría, en promedio, generar y evaluar un valor para determinar si genera o no genera el código buscado. Escriba todos sus cálculos.

La velocidad del procesador es igual 2.30 GHz.

Ahora, la cantidad de ciclos de procesador requeridos depende de muchos factores específicos como la complejidad del programa, el lenguaje de programación, la velocidad del procesador, la cantidad de núcleos, etc. Para el programa actual contamos con la medición de los tiempos de ejecución y como promedio de tiempo usaremos el promedio de las pruebas según se presenten sus características.

Promedio de tiempos de ejecución para SHA-512, con contraseñas de un valor de un caracter, para un único Thread:

$$\frac{4ms + 2ms + 2ms + 1ms}{4\ casos} = 2.25ms \text{ o } 0{,}00225 \text{ segundo}$$

Promedio de tiempos de ejecución para SHA-256, con contraseñas de un valor de un carácter, para un único Thread:

$$\frac{1ms + 2ms + 2ms + 1ms}{4\ casos} = 1.5ms \text{ o } 0{,}0015 \text{ segundos}$$

Seguido a esto, la cantidad de ciclos de procesador estimados para determinar un único valor en el programa sería el resultado de multiplicar la cantidad de milisegundos por la velocidad del procesador y por el equivalente de a su medida en hertzios.

$$0.00225s \cdot 2.30\ GHz \cdot 10^9 = 5.175.000$$

$$0.0015s \cdot 2.30\ GHz \cdot 10^9 = 3.450.000$$

Respectivamente para un único valor en el algoritmo SHA-512 equivale a 5.175.000 ciclos de procesador y para el algoritmo de SHA-256 serian 3.450.000 ciclos de procesador.

6 Cálculo para un programa monothread

- Con base en los cálculos del punto anterior, calculamos cuánto tiempo tomaría un programa monothread, en promedio, para encontrar una contraseña en los siguientes casos:

- A) contraseñas de 8 caracteres, cada carácter puede ser mayúscula, minúscula, número o uno de los siguientes caracteres especiales:.:,;!?(%)\+-/*{}, la sal es una secuencia de 16 bits.
- B) contraseñas de 10 caracteres, cada carácter puede ser mayúscula, minúscula, número o uno de los siguientes carácteres especiales:.:,;!?(%)\+-/*{}, la sal es una secuencia de 16 bits.
- C) contraseñas de 12 caracteres, cada carácter puede ser mayúscula, minúscula, número o uno de los siguientes carácteres especiales:.:,;!?(%)\+-/*{}, la sal es una secuencia de 16 bits.

Para los siguientes casos, contamos con el doble de caracteres usados en el programa, ya que solo se usaron minúsculas equivalentes a 26 caracteres. Aparte, son 10 números y 16 caracteres especiales válidos para este nuevo caso.

$Ct = 10 + 26 + 26 + 16 = 78$

Esto nos deja con un alto número de posibles combinaciones para los casos A,B,C:

A)78^8

B)78^{10}

C)78^{12}

Ahora hallamos los ciclos por combinación requeridos para este nuevo caso según la cantidad de posiciones a evaluar. Tomando como base el caso del punto anterior, que para un único valor se requieren 3.450.000 ciclos de procesador, este lo multiplicamos por la cantidad de caracteres, para tener los ciclos por combinación:

A) $3.450.000\ ciclos_p \cdot 8\ caracteres = 27.600.000$ ciclos por combinación

B) $3.450.000\ ciclos_p \cdot 10\ caracteres = 34.500.000$ ciclos por combinación.

C) $3.450.000\ ciclos_p\ \cdot\ 12\ caracteres\ =\ 41.400.000$ ciclos por combinación

Luego lo multiplicamos por el total de combinaciones, 78^8

A) $78^8\ \cdot\ 27.600.000\ ciclos_{comb} = 37.815.156.630.854.553.600.000$

B) $78^{10} \cdot 34.500.000\ ciclos_{comb} = 287.584.266.177.648.880.128.000.000$

C) $78^{12} \cdot 41.400.000\ ciclos_{comb} = 2.099.595.210.509.778.944.038.502.400.000$

Por último, para obtener el tiempo estimado, se debe dividir los ciclos de procesador entre la velocidad del procesador.

A) $\frac{37.815.156.630.854.553.600.000}{2.30GHz}$= 16441372448197632000000 s

B) $\frac{287.584.266.177.648.880.128.000.000}{2.30GHz}$= 125036637468542991360000000 s

C) $\frac{2.099.595.210.509.778.944.038.502.400.000}{2.30GHz}$=912867482830338671321088000000s

Los resultados aquí equivaldrían a aproximadamente a 5.210.063.522.000 siglos para el caso A, 39.622.533.085.000.000 siglos para el caso B y 289.276.189.550.000.000.000 siglos para el caso C. Esto contando con las capacidades de cómputo mencionadas y los tiempos medidos.

7 Análisis y entendimiento del problema

Este algoritmo representa una situación común en la que la seguridad de un sistema se ve comprometida y un atacante obtiene información de los datos de acceso de los usuarios, como los hashes de las contraseñas y sus sales correspondientes. Aunque el atacante aún no tenga acceso directo a las contraseñas de los usuarios, puede utilizar un algoritmo como el que hemos desarrollado para generar cadenas que combinen diferentes caracteres y sal específica de cada usuario, esperando encontrar una cadena cuyo hash coincida con el del usuario. Esto le permitiría descifrar la contraseña y suplantar al usuario, lo que le daría acceso a todas sus acciones y datos privados.

Para los usuarios, es fundamental generar contraseñas lo más complejas posibles para ampliar el espacio de búsqueda de los atacantes, llegando a un punto donde el tiempo de ejecución sea inmanejable, incluso por años. Además, los desarrolladores de aplicaciones deben priorizar el uso de algoritmos de hash más seguros que requieran un mayor tiempo de respuesta para generar los hashes. Estos dos aspectos son vitales para garantizar la seguridad de los datos de los usuarios y de la empresa desarrolladora.

1. Busque información adicional sobre los algoritmos de generación de códigos criptográficos de hash: (i) cuáles se usan hoy día y en qué contexto y (ii) por qué dejamos de usar aquellos que se consideran obsoletos

Dentro de los algoritmos de generación de códigos criptográficos de hash usados hoy en día, encontramos algoritmos robustos con enfoques y usos distintos.

- Blake2 es alta mente usado por su alta velocidad, seguridad y simplicidad, esto último lo hace un algoritmo eficiente en termino de memoria para sistemas con recursos limitados, también es resistente a los ataques de colisión y preimagen. Su uso se encuentra en el estándar de autenticación de OpensSSL o en sistemas de anonimacion de tráfico Tor.
- Whirlpool es un algoritmo de hash criptográfico, es usado para calcular resúmenes criptográficos y se encuentra principalmente en sistemas de seguridad, debido a que cuenta con una longitud de salida de 512 bits, lo que le permite ser resistente a ataques de fuerza bruta. Su uso se encuentra en bibliotecas criptográficas como OpenSSL o Bouncy Castle y en sistema de archivos cifrados.

Como podemos observar, estos algoritmos cuentan con un diseño para resistir ataques lo que los hace más seguros y confiables frente a otros que aún no cuentan con esto. El tener una medida de protección contra ataques evita que el algoritmo sea obsoleto y a su vez permite encontrar usos según las capacidades del mismo. Un algoritmo se vuelve obsoleto cuando logran vulnerarlo y por ende representan un riesgo a la integridad de los datos.

2. La tecnología bitcoin usa un procedimiento conocido como "mining" que también depende de la existencia de algoritmos criptográficos de hash que no tienen funciones inversas. Describa en qué consiste el proceso de mining.

El proceso de mining para bitcoin se basa en el uso del algoritmo criptográfico de hash SHA-256, este proceso se caracteriza por el uso de bloques que representa una única pieza de código que funciona como una estructura de control y que se encarga de descifrar el hash por métodos de fuerza bruta.

Los bloques se categorizan según su estado:

- Genesis – Es el equivalente al bloque de inicio de la cadena de bloques.
- Valido - Que hace referencia al bloque que ya fue agregado a la cadena.
- Huerfano – Aquel bloque que no ha sido agregado debido a que no completa aun la prueba.

Cada bloque cuenta con un Header, que guarda el hash anterior por esto es que si se cambia un valor del bloque el hash cambiara completamente y por ende todos los bloques siguientes también; un Body, que es donde se guardan los datos de la transacción realizada; y al final una condición para validar el bloque siguiente, para el caso de bitcoin se busca un hash que inicie con 30 ceros al inicio.

Con esto ya claro, solo aquella máquina que encuentre primero el hash tiene la oportunidad de agregar el bloque al registro general, esto crea cadenas grandes de bloques donde para compartir el bloque y hacer valida la transacción se requiere que el bloque se enlace con la cadena de mayor longitud. Esta simple regla garantiza la integridad de los datos en los bloques ya que, para poder cambiar los datos sin ser detectado, se necesitaría la capacidad de cómputo de la mitad más uno de toda la red de blockchain activa en ese momento.

3. La adición de la sal se relaciona con el problema asociado al uso de Rainbow Tables. (i) Describa cuál es el problema de seguridad asociado, (ii) Describa cómo ayuda la sal a resolver o mitigar ese problema.

El problema de Rainbow Tables se trata de almacenar los valores de hash pre calculados para todas las posibles entradas de una longitud determinada en una tabla, esto reduce el tiempo y la complejidad,

La Sal para este caso funciona como un identificador único que se concatena con la contraseña antes de aplicar la función de hash, esto hace que la búsqueda de la tabla se vuela exponencialmente mucho más costosa y que cada clave cifrada reciba un valor de hash distinto.

8 Conclusiones

A modo de cierre, el objetivo principal del programa fue demostrar la implementación de una búsqueda de valores hash a través de un espacio de búsqueda determinado, utilizando técnicas de programación estructurada y sin dependencias externas.

El programa cumplió con los requisitos establecidos y logró encontrar la cadena V y la sal S que permitieron generar el código C utilizando el algoritmo SHA256 o SHA512, y a su vez reportó el tiempo que tomó la búsqueda realizando un total de 116 pruebas.

Sin embargo, los códigos criptográficos de hash no son adecuados para garantizar la confidencialidad de los datos, ya que es posible realizar ataques de fuerza bruta o de diccionario para obtener la cadena original a partir del hash, tal es el caso de nuestro programa.

Con todo esto claro, diríamos que las distintas aplicaciones de códigos criptográficos de hash si son esenciales para garantizar la integridad de los datos, pero no son suficientes para garantizar la confidencialidad. Es por esto necesario implementar medidas de seguridad adicionales y evaluar constantemente el esquema de seguridad para garantizar la protección efectiva de los datos confidenciales en los sistemas informáticos, sin olvidar la importancia de conocer también los tiempos de respuesta o alternativas a los códigos criptográficos aquí vistos.

Referencias

- *BLAKE2*. (s. f.). Recuperado 2 de mayo de 2023, de https://www.blake2.net/

- Sant, H. (2023, febrero 15). *Cómo protegerse de un ataque de mesa arcoíris*. Geekflare. https://geekflare.com/es/rainbow-table-attack/

- *¿Qué son los bloques en la tecnología Blockchain?* Blog ITDO - Agencia de desarrollo Web, APPs y Marketing en Barcelona. https://www.itdo.com/blog/que-son-los-bloques-en-la-tecnologia-blockchain/

- *Whirlpool—Algoritmo resumen (hash)—CCN-STIC 401*. (s. f.). Recuperado 2 de mayo de 2023, de https://www.dit.upm.es/~pepe/401/index.html#!7685