Caso de estudio 3: Seguridad

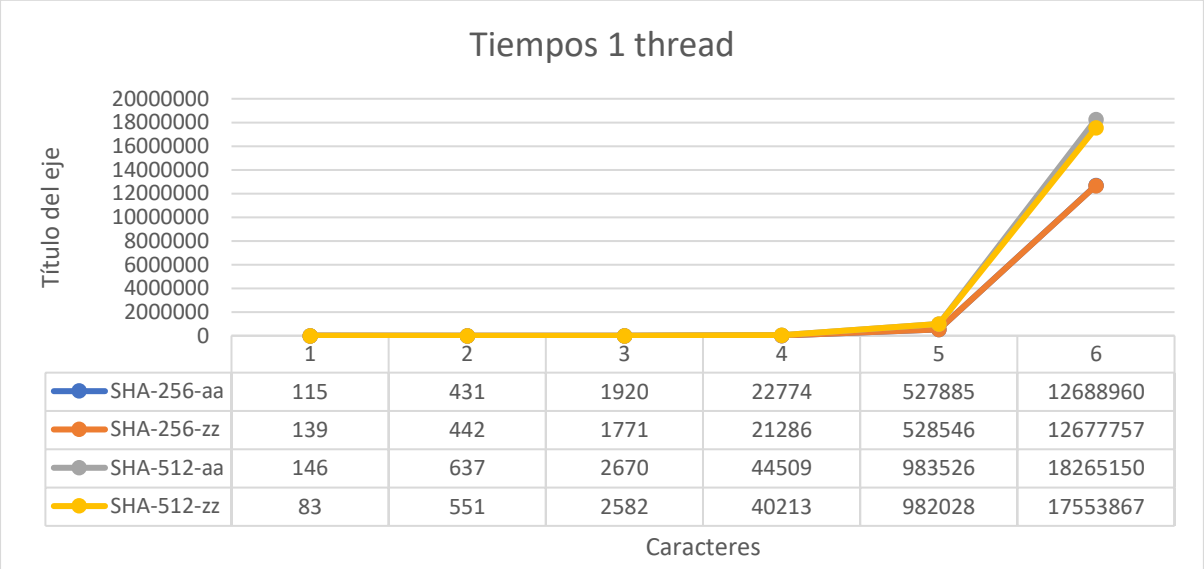
Paola Andrea Campiño, 202020785

Samuel Santa Arias, 202123685

Repositorio: https://github.com/Paolaaaaaa/Caso3_Seguridad.git

1. Programa para una thread:

para los dos algoritmos, sobre cada longitud y las dos sales

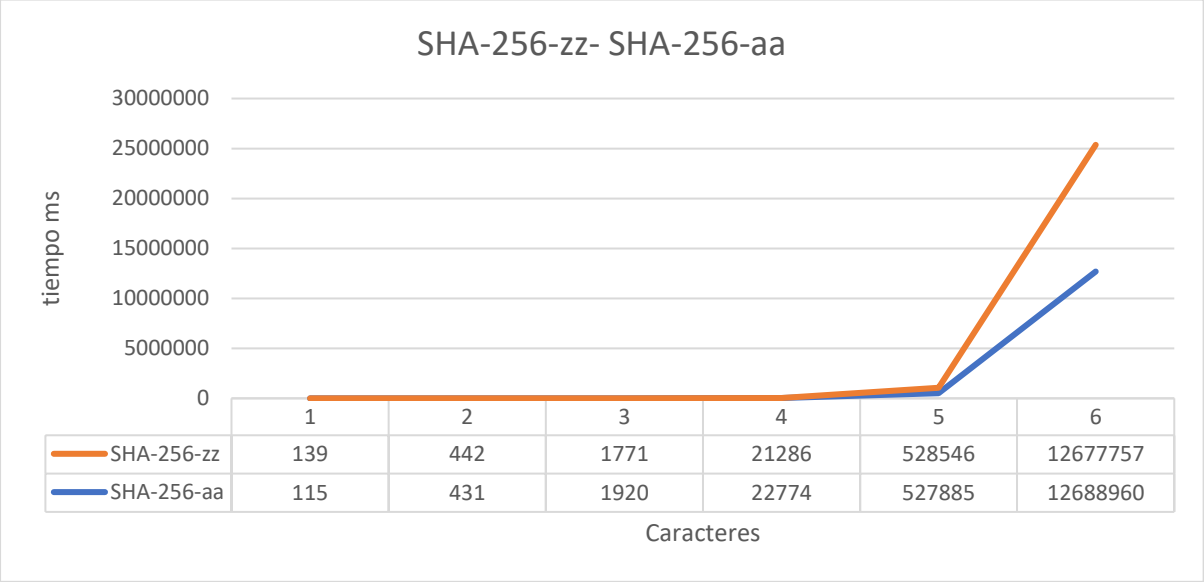


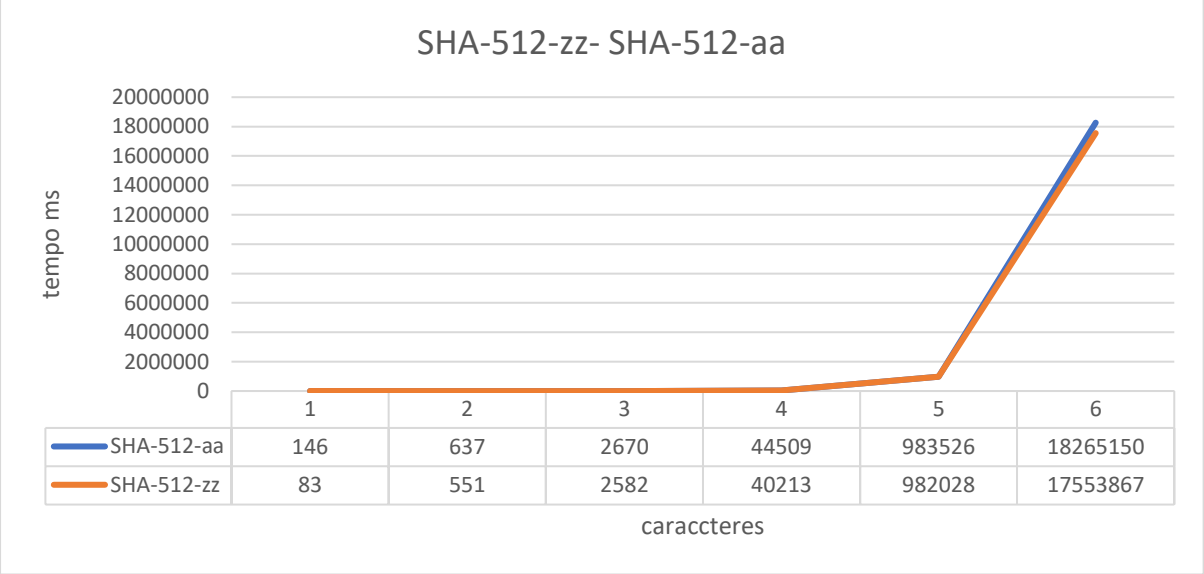
	1	2	3	4	5	6
SHA-256-aa	115	431	1920	22774	527885	12688960
SHA-256-zz	139	442	1771	21286	528546	12677757
SHA-512-aa	146	637	2670	44509	983526	18265150
SHA-512-zz	83	551	2582	40213	982028	17553867

En base a los datos y graficas anteriores podemos notar que hay una clara diferencia entre SHA-256 y SHA-512, ya que los tiempos después de los 5 y 6 caracteres es masiva. Así mismo se puede notar que entre más caracteres tenga el hash mayor seguridad esto no es muy notorio al inicio, pero si cuando se trata de 5-6 caracteres.

Cabe resaltar que no hay mucha diferencia entre las sales, aunque, si se puede ver que la sal “aa” suele tardar un poco menos de tiempo que la sal “zz”.

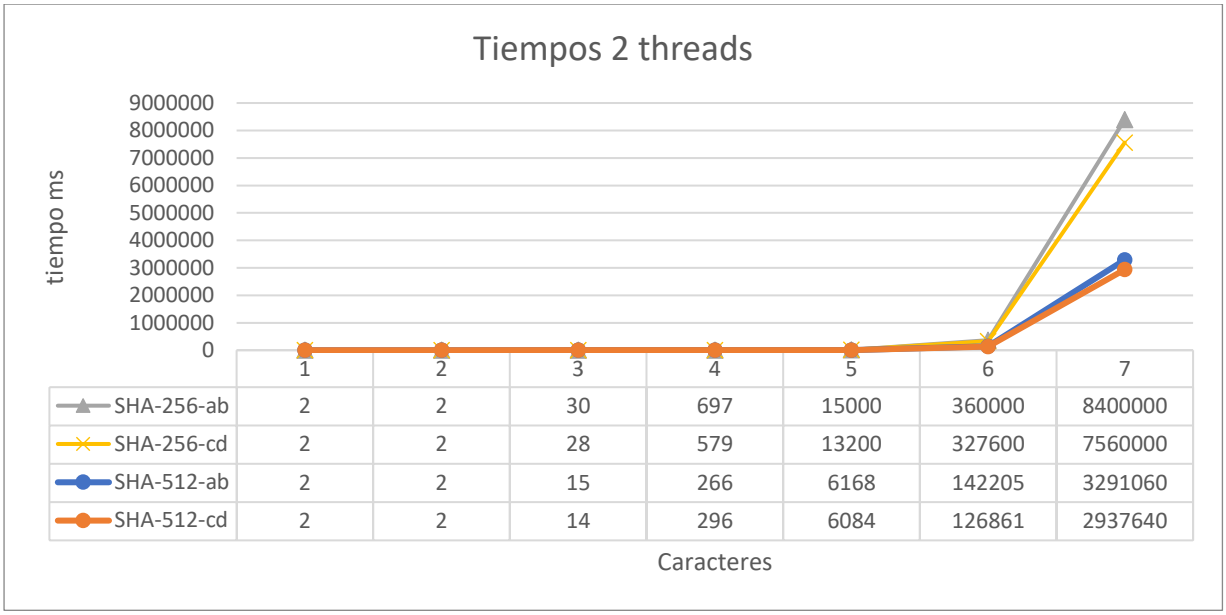
A continuación, los gráficos con mayor detalle:



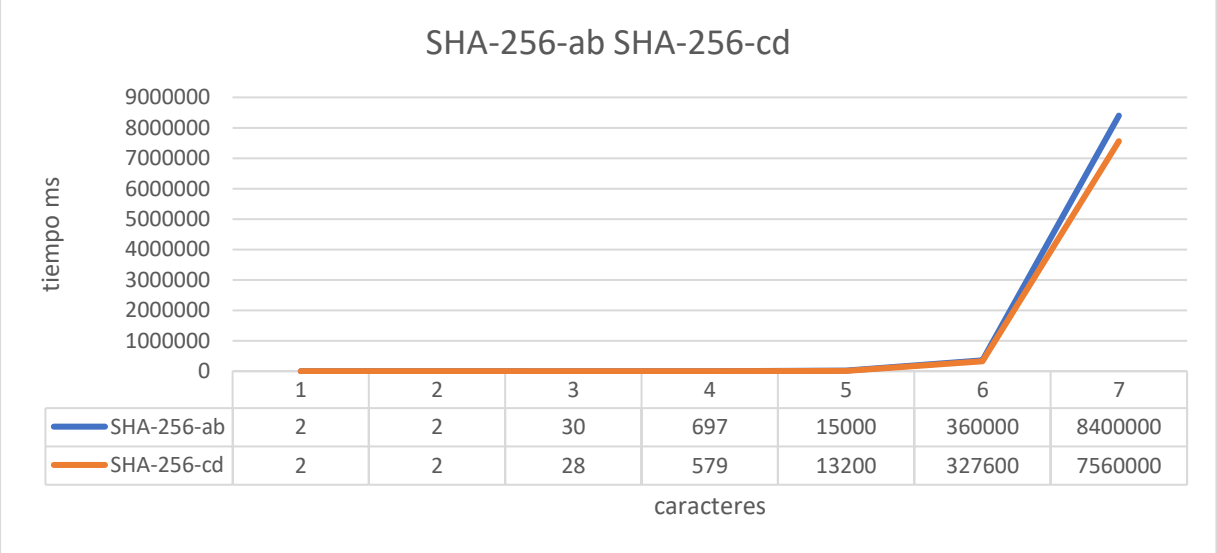


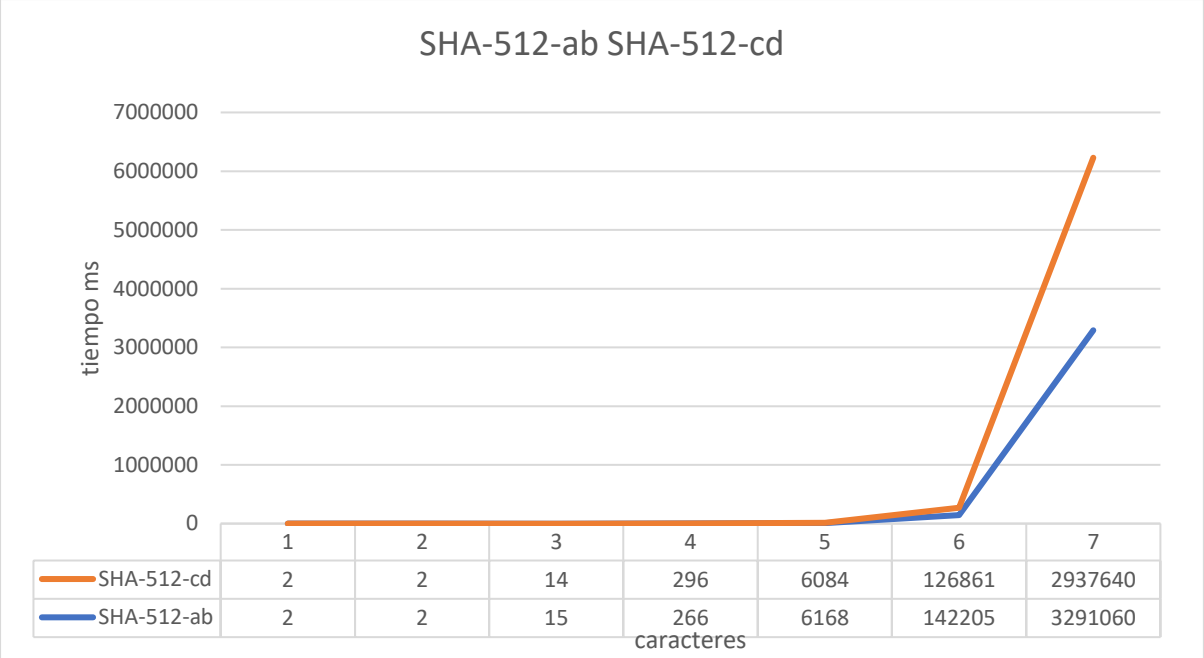
2. Programa para dos threads:

para los dos algoritmos, sobre cada longitud y las dos sales.



En la grafica anterior se puede ver la diferencia entre los diferentes algorithmos y los tiempos a los que responde. Se puede ver que en general SHA-512 es mucho más rápido que SHA-526





3. 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.

Velocidad del procesador: 3.30GHz (ciclos por segundo) = 3.30 millones de ciclos de reloj por segundo.

Conteo de instrucciones:

Generar una lista de Strings = 1 instrucción

For que concatena Strings e identificación de vacíos: 2 instrucciones

Sacar hash (concatenar sal+ pasar a bytes + aplicar digest + pasar a String): 4 instrucciones

Comparar hash (pasar a encontrado + asignación) = 2 instrucciones

Para el ciclo de generación de contraseña (7) = 7 instrucciones

Total de instrucciones: 16 Instrucciones.

Tiempo entre ejecutar las 16 intrucciones: 1-0 segundo== 1 s =ciclo

= 16 ciclos /clave

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

1. contraseñas de 8 caracteres, cada carácter puede ser mayúscula, minúscula, número uno de los siguientes caracteres especiales: !?(%)\+-/*{}, la sal es una secuencia de 16 bits:
   - Posibilidades= Mayúscula + Minúscula + Números + char_especiales = 26+26+10+16= 78 posibilidades

   - Posibles contraseñas: 78**8 sin considerar la sal

En caso sin saber la sal:

Casos a probar: = 78^8* 2^16

Número de ciclos para descifrar todo = 78^8* 2^16 claves *16 ciclos/clave

Tiempo: 78^8* 2^16*16 /3.3GHz = 43500000000000000 s

=1379375951 años

Casos conociendo la sal:

Casos a probar: = 78^8

Número de ciclos para descifrar = 78^8 * 16= 2.19**16 ciclos

Tiempo en segundos= 2.19**16 ciclos/3.3GHz = 6642978767 s

= 213 años

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

- Posibilidades= Mayúscula + Minúscula + Números + char_especiales = 26+26+10+16= 78 posibilidades

- Posibles contraseñas: 78**10 sin considerar la sal

En caso sin saber la sal:

Casos a probar: = 78^10* 2^16

Número de ciclos para descifrar todo = 78^10* 2^16 claves *16 ciclos/clave

Tiempo: 78^8* 2^16*16 /3.3GHz = 43500000000000000 s

=1379375951 años = 8,52E+09 décadas

Casos conociendo la sal:

Casos a probar: = 78^10

Número de ciclos para descifrar = 78^10* 16= 2.19**16 ciclos

Tiempo en segundos= 2.19**16 ciclos/3.3GHz = 6642978767 s

= 1299378,949 años = 129937,8949 décadas

3. contraseñas de 12 caracteres, cada carácter puede ser mayúscula, minúscula, número o uno de los siguientes carácter es especiales:.:,;!?(%)\+-/*{}, la sal es una secuencia de 16 bits análisis y entendimiento del problema:

- Posibilidades= Mayúscula + Minúscula + Números + char_especiales = 26+26+10+16= 78 posibilidades

- Posibles contraseñas: 78**12 sin considerar la sal

En caso sin saber la sal:

Casos a probar: = 78^12* 2^16

Número de ciclos para descifrar todo = 78^12* 2^16 claves *16 ciclos/clave

Tiempo: 78^8* 2^16*16 /3.3GHz = 1,61E+22 s

=5,18E+14 años = 5,18E+13 décadas

Casos conociendo la sal:

Casos a probar: = 78^12

Número de ciclos para descifrar = 78^12* 16= 2.19**16 ciclos

Tiempo en segundos= 2.19**16 ciclos/3.3GHz = 2,4589E+17 s

= 7905421523 años = 790542152,3 décadas

5. Análisis y Entendimiento del Problema.

4. **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.**

i) De los algorithmos más comunes usados en la actualidad están :
   (1) SHA-256: Es comúnmente usado para autenticación. Sistemas operativos como Unix y Linux lo usan para las contraseñas, adicionalmente criptomonedas como bitcoin los usan para verificar transacciones [3].
   (2) SHA-3: Es el algoritmo criptográfico de hash principalmente implementado en sistemas embebidos y sensores. La implementación de este algoritmo no requiere experticia en criptografía[4].
   (3) Bcrypt: Es un algoritmo que fue diseñado para el almacenamiento de contraseñas de los usuarios. Es principalmente usada en aplicaciones web[5].

ii) Existen varios algoritmos que en la actualidad se consideran obsoletos como lo que pasó con MD5, que fueron diseñados para ser rápidos sin embargo el problema es que para los tiempos de hoy los computadores tienen mayor facilidad de crakear y así mismo se ha detectado que es muy susceptible a colisiones.
   Hay otro algoritmo como lo fue SHA-1 que ante el aumento de capacidad de procesamiento se llegaron a detectar colisiones de hash, por lo que buscadores como Google desde el 2014 advocaron por la depreciación del SHA-1.

[1] Freda Anthony (2023).*¿Qué es el algoritmo de hashing MD5 y como funciona?,* Academy . Recuperado de: https://www.avast.com/es-es/c-md5-hashing-algorithm#:~:text=El%20MD5%20se%20usa%20principalmente,uso%20principal%20es%20la%20autenticaci%C3%B3n.

[2]Fisher (2023). *SHA-1:What it is & how it's used for data verification*, lifewire. Recuperado de: https://www.lifewire.com/what-is-sha-1-2626011#toc-history-and-vulnerabilities-of-the-sha-hash-function

[3] N-able(2019). *SHA-256 Algorithm overview*, N-able. Recuperado de: https://www.n-able.com/blog/sha-256-encryption#:~:text=SHA%2D256%20is%20used%20in,SHA%2D256%20for%20verifying%20transactions.

[4]. Scott jones.(2019). What you need to know about SHA-3 for embedded system security, Electronic Design. Recuperado de: https://www.electronicdesign.com/technologies/embedded/article/21808025/what-you-need-to-know-about-sha3-for-embedded-system-security

[5] Arias Dan .(2021).*Hashing in Action: Understanding bcrypt, auth0* .Recuperado de: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/

5. **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" o minería es el proceso de validación y registro de transacciones en la cadena de bloques de Bitcoin. La minería también es el proceso de creación de nuevos bitcoins al resolver un problema criptográfico complejo, y es una parte integral del funcionamiento del sistema Bitcoin.

El proceso de minería comienza cuando los nodos de la red Bitcoin recopilan transacciones que aún no se han incluido en un bloque. Los nodos luego transmiten estas transacciones a los mineros. Los mineros compiten por resolver un problema criptográfico complejo, que esencialmente consiste en encontrar un número aleatorio llamado "nonce" que, cuando se agrega a los datos del bloque, da como resultado un hash que cumple con ciertos requisitos. El requisito principal es que el hash debe ser menor que un valor específico, que se conoce como el "objetivo".

El proceso de minería es intensivo en recursos computacionales y energía, ya que los mineros deben realizar cálculos repetidos hasta encontrar un hash que cumpla con los requisitos. Una vez que un minero encuentra un hash que cumple con los requisitos, se lo comunica a los demás nodos de la red. Los nodos validan el bloque y lo agregan a la cadena de bloques, y el minero que encontró el hash válido recibe una recompensa en forma de nuevos bitcoins.

Cada bloque en la cadena de bloques de Bitcoin contiene un registro de transacciones verificadas y un hash que vincula ese bloque con el bloque anterior. Esto crea una cadena de bloques interconectados que garantiza que cada transacción sea válida y no se haya duplicado. La complejidad del problema criptográfico utilizado en la minería de Bitcoin se ajusta automáticamente para mantener una tasa de producción de nuevos bloques constante, independientemente de la cantidad de poder de procesamiento dedicado a la minería en la red.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Recuperado de https://bitcoin.org/bitcoin.pdf

6. 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

(i) El problema de seguridad asociado con el uso de Rainbow Tables es que permite a un atacante precomputar los hashes de una gran cantidad de posibles contraseñas y almacenarlos en una tabla, lo que le permite encontrar fácilmente la contraseña original correspondiente a un hash dado en cuestión de segundos. Esto es particularmente peligroso cuando se trata de contraseñas débiles, como las que contienen sólo letras minúsculas o números.

(ii) La sal es una cadena aleatoria que se añade a la contraseña antes de aplicar la función hash, lo que hace que cada hash sea único, incluso si la misma contraseña se ha utilizado varias veces. Esto significa que un atacante tendría que crear una tabla separada para cada posible sal, lo que hace que el ataque sea mucho más difícil y costoso en términos de recursos de computación. Por lo tanto, la sal ayuda a mitigar el problema de seguridad asociado con el uso de Rainbow Tables y aumenta la seguridad de las contraseñas almacenadas.

Ferguson, N., Schneier, B., & Kohno, T. (2010). Cryptography engineering: design principles and practical applications. John Wiley & Sons.