Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
Semestre 2023-10

# Caso de Estudio 3 – Seguridad

Santiago Gustavo Ayala Ciendua - 202110734

Juan Esteban Jiménez Benavides – 201922487

Santiago Andrés Ramírez Ramírez – 201910908

Universidad de los Andes, Bogotá, Colombia

Fecha de presentación: mayo 3 de 2023

Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
Semestre 2023-10

## Tabla de contenidos

# Implementación del Prototipo :

- Tabla con los tiempos recopilados:

| | i | n | f | r | a | c | o |
|---|---|---|---|---|---|---|---|
| | Sal: hi | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1T | 256 | 0,01528 | 0,00965 | 0,04050 | 0,28457 | 3,56022 | 109,06120 | 2890,48113 |
| 1T | 512 | 0,01747 | 0,01013 | 0,05075 | 0,41828 | 8,65471 | 224,73801 | 6010,60988 |
| 2T | 256 | 0,01434 | 0,00086 | 0,00623 | 0,41239 | 4,28158 | 95,69531 | 2489,85430 |
| 2T | 512 | 0,01449 | 0,00111 | 0,01368 | 0,55063 | 7,92484 | 206,04880 | 5822,38591 |

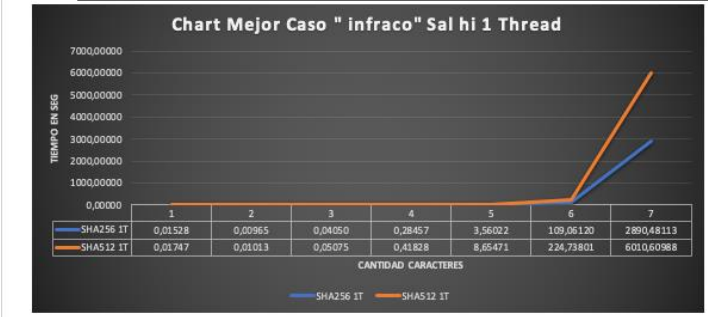
| | z | z | z | z | z | z | z |
|---|---|---|---|---|---|---|---|
| | Sal: fe | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1T | 256 | 0,01593 | 0,01691 | 0,08024 | 0,62101 | 13,02184 | 314,51426 | 8574,23658 |
| 1T | 512 | 0,01998 | 0,01989 | 0,10708 | 1,11952 | 25,29580 | 694,24875 | 18643,35593 |
| 2T | 256 | 0,01717 | 0,00124 | 0,02187 | 0,80000 | 12,79863 | 301,24785 | 8002,57868 |
| 2T | 512 | 0,01876 | 0,00185 | 0,04055 | 1,16576 | 24,20903 | 667,12489 | 18145,32586 |
| | | 1,06151915 | 4,74512123 | 7,73940678 | 20,9688089 | 24,15282786 | | |

| | i | n | f | r | a | c | o |
|---|---|---|---|---|---|---|---|
| | Sal: fe | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1T | 256 | 0,01635 | 0,00878 | 0,04253 | 0,27101 | 4,24219 | 114,68745 | 2889,99845 |
| 1T | 512 | 0,01608 | 0,01059 | 0,05783 | 0,41723 | 8,50823 | 222,74485 | 6005,87414 |
| 2T | 256 | 0,01479 | 0,00080 | 0,00723 | 0,42579 | 4,31977 | 99,14574 | 2455,74858 |
| 2T | 512 | 0,01643 | 0,00105 | 0,01446 | 0,52517 | 8,06649 | 210,68989 | 5884,32585 |

| | z | z | z | z | z | z | z |
|---|---|---|---|---|---|---|---|
| | Sal: hi | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1T | 256 | 0,01637 | 0,01630 | 0,08126 | 0,61264 | 11,19998 | 329,77242 | 8606,77870 |
| 1T | 512 | 0,01746 | 0,02046 | 0,10449 | 1,10852 | 26,26753 | 682,18412 | 18457,69874 |
| 2T | 256 | 0,01609 | 0,00117 | 0,01728 | 0,70656 | 11,24874 | 290,11479 | 7975,23587 |
| 2T | 512 | 0,01890 | 0,00180 | 0,04276 | 1,16770 | 23,91717 | 674,99822 | 18020,75841 |

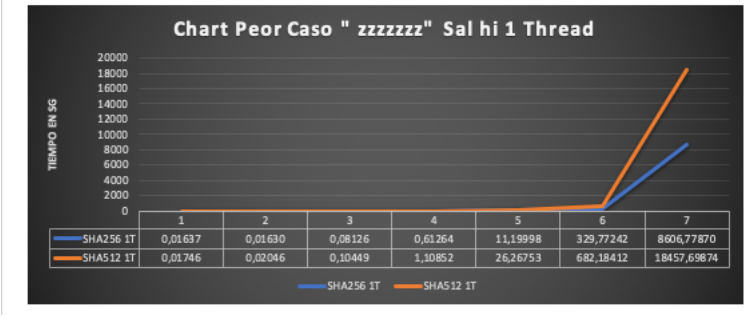
- Gráficas :
    - Gráfica tiempo 1 Thread para los dos algoritmos, sobre cada longitud y las dos sales:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T | 0,01528 | 0,00965 | 0,04050 | 0,28457 | 3,56022 | 109,06120 | 2890,48113 |
| SHA512 1T | 0,01747 | 0,01013 | 0,05075 | 0,41828 | 8,65471 | 224,73801 | 6010,60988 |



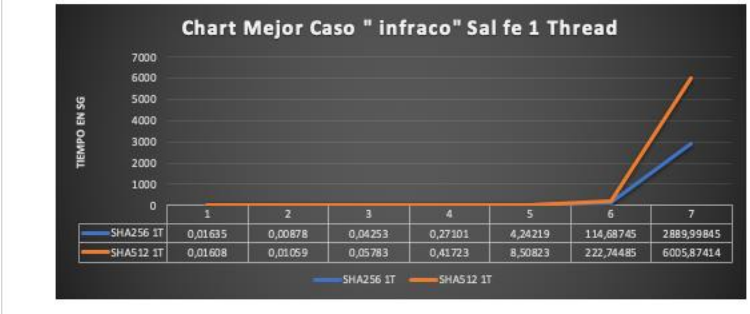

*Grafica 1 Thread en el mejor caso "infraco" sal "hi"*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T | 0,01637 | 0,01630 | 0,08126 | 0,61264 | 11,19998 | 329,77242 | 8606,77870 |
| SHA512 1T | 0,01746 | 0,02046 | 0,10449 | 1,10852 | 26,26753 | 682,18412 | 18457,69874 |




*Grafica 1 Thread en el peor caso ("zzzzzzz") con sal hi*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T | 0,01635 | 0,00878 | 0,04253 | 0,27101 | 4,24219 | 114,68745 | 2889,99845 |
| SHA512 1T | 0,01608 | 0,01059 | 0,05783 | 0,41723 | 8,50823 | 222,74485 | 6005,87414 |



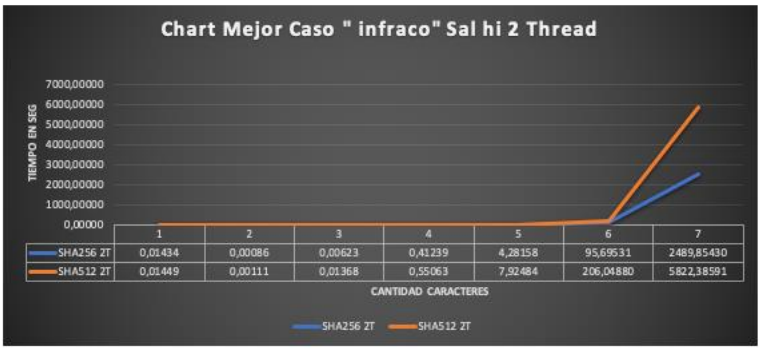

*Grafica 1 Thread en el mejor caso ("infraco") con sal "fe"*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T | 0,01593 | 0,01691 | 0,08024 | 0,62101 | 13,02184 | 314,51426 | 8574,23658 |
| SHA512 1T | 0,01998 | 0,01989 | 0,10708 | 1,11952 | 25,29580 | 694,24875 | 18643,35593 |


*Grafica para el peor caso "zzzzzzz" con sal "fe" en 1 Thread*

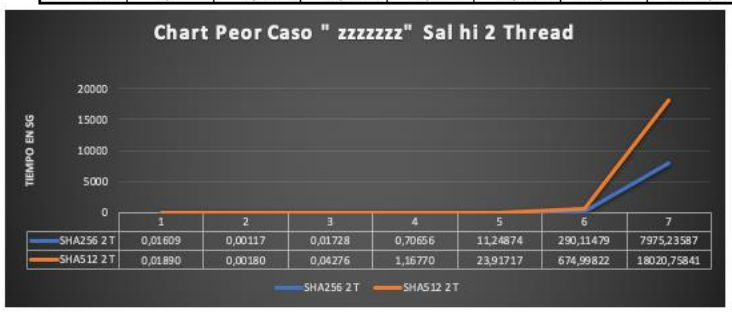
Gráfica tiempo 2 Thread para los dos algoritmos, sobre cada longitud y las dos sales:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2T | 0,01434 | 0,00086 | 0,00623 | 0,41239 | 4,28158 | 95,69531 | 2489,85430 |
| SHA512 2T | 0,01449 | 0,00111 | 0,01368 | 0,55063 | 7,92484 | 206,04880 | 5822,38591 |



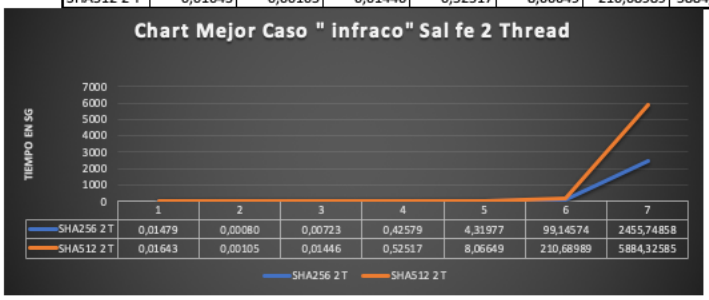

*Grafica 2 Threads Mejor caso ("infraco") con sal hi*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2 T | 0,01609 | 0,00117 | 0,01728 | 0,70656 | 11,24874 | 290,11479 | 7975,23587 |
| SHA512 2 T | 0,01890 | 0,00180 | 0,04276 | 1,16770 | 23,91717 | 674,99822 | 18020,75841 |



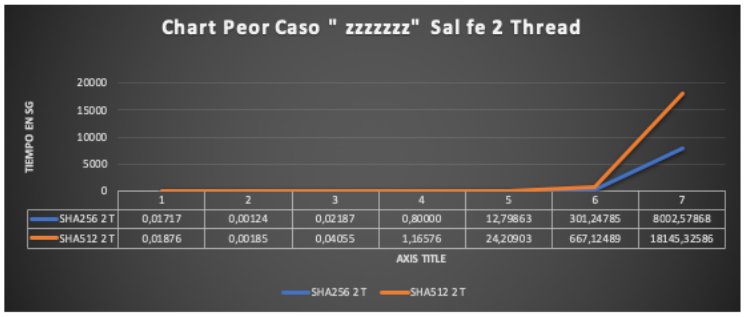

*Grafica 2 Threads Peor caso (“zzzzzzz”) sal hi*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2 T | 0,01479 | 0,00080 | 0,00723 | 0,42579 | 4,31977 | 99,14574 | 2455,74858 |
| SHA512 2 T | 0,01643 | 0,00105 | 0,01446 | 0,52517 | 8,06649 | 210,68989 | 5884,32585 |




*Grafica 2 Threads mejor caso (“infraco”) sal fe*

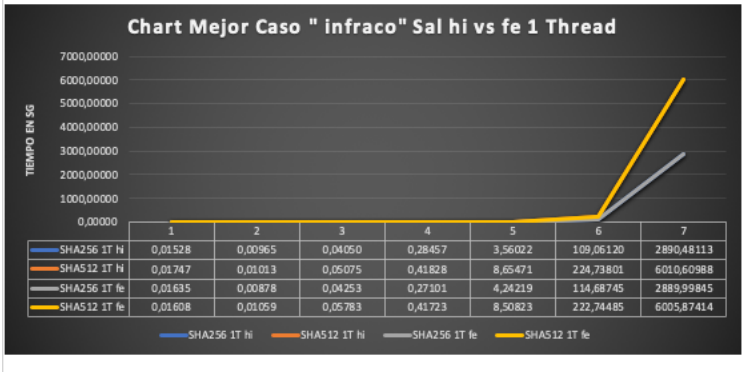
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2 T | 0,01717 | 0,00124 | 0,02187 | 0,80000 | 12,79863 | 301,24785 | 8002,57868 |
| SHA512 2 T | 0,01876 | 0,00185 | 0,04055 | 1,16576 | 24,20903 | 667,12489 | 18145,32586 |




*Grafica 2 Threads peor caso (“zzzzzzz”) sal fe*

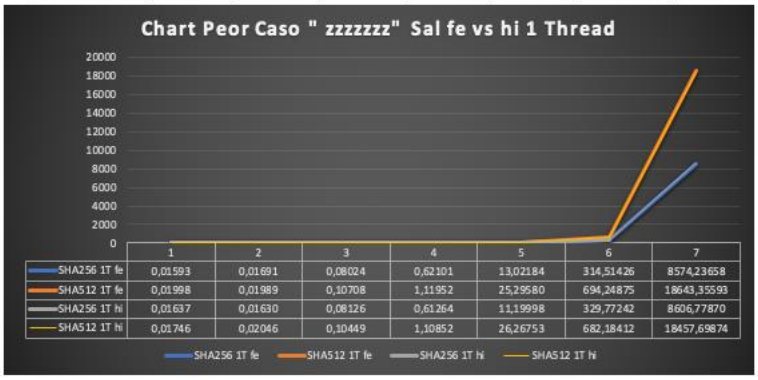
Graficas comparativas :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T hi | 0,01528 | 0,00965 | 0,04050 | 0,28457 | 3,56022 | 109,06120 | 2890,48113 |
| SHA512 1T hi | 0,01747 | 0,01013 | 0,05075 | 0,41828 | 8,65471 | 224,73801 | 6010,60988 |
| SHA256 1T fe | 0,01635 | 0,00878 | 0,04253 | 0,27101 | 4,24219 | 114,68745 | 2889,99845 |
| SHA512 1T fe | 0,01608 | 0,01059 | 0,05783 | 0,41723 | 8,50823 | 222,74485 | 6005,87414 |



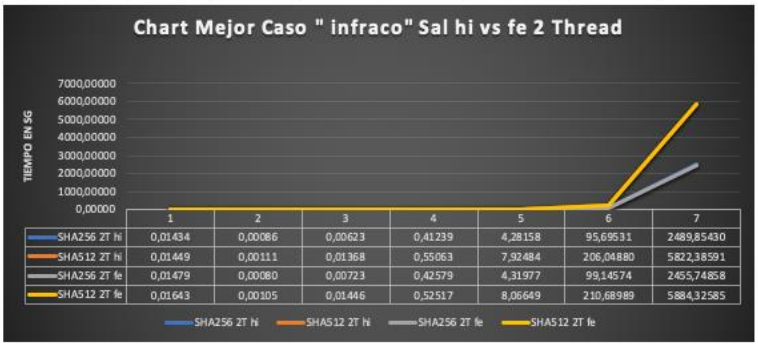

*Grafica Mejor caso sal hi vs fe en 1 Thread*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T fe | 0,01593 | 0,01691 | 0,08024 | 0,62101 | 13,02184 | 314,51426 | 8574,23658 |
| SHA512 1T fe | 0,01998 | 0,01989 | 0,10708 | 1,11952 | 25,29580 | 694,24875 | 18643,35593 |
| SHA256 1T hi | 0,01637 | 0,01630 | 0,08126 | 0,61264 | 11,19998 | 329,77242 | 8606,77870 |
| SHA512 1T hi | 0,01746 | 0,02046 | 0,10449 | 1,10852 | 26,26753 | 682,18412 | 18457,69874 |



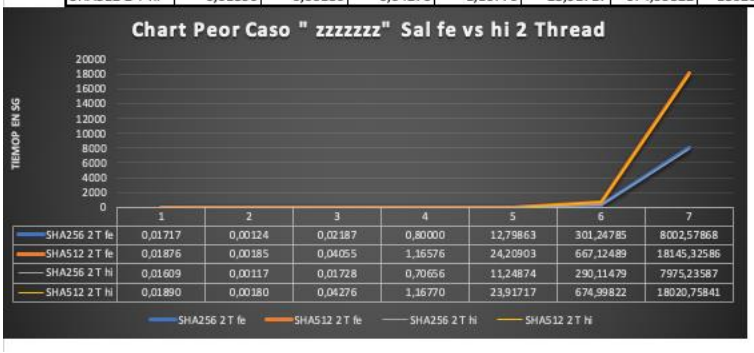

*Grafica Peor caso sal fe vs hi en 1 Thread*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2T hi | 0,01434 | 0,00086 | 0,00623 | 0,41239 | 4,28158 | 95,69531 | 2489,85430 |
| SHA512 2T hi | 0,01449 | 0,00111 | 0,01368 | 0,55063 | 7,92484 | 206,04880 | 5822,38591 |
| SHA256 2T fe | 0,01479 | 0,00080 | 0,00723 | 0,42579 | 4,31977 | 99,14574 | 2455,74858 |
| SHA512 2T fe | 0,01643 | 0,00105 | 0,01446 | 0,52517 | 8,06649 | 210,68989 | 5884,32585 |



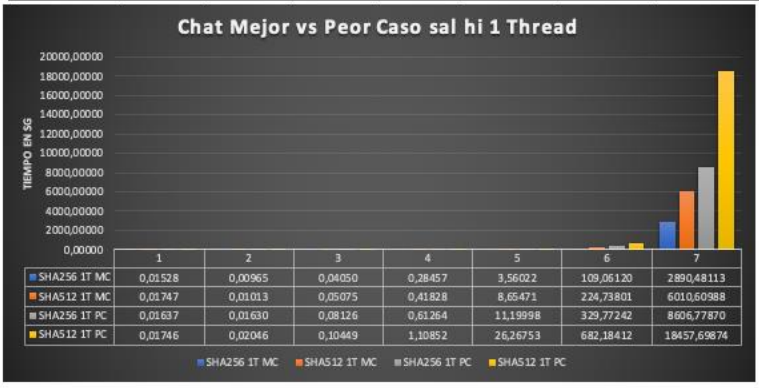

*Grafica Mejor caso sal hi vs fe 2 Threads*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2 T fe | 0,01717 | 0,00124 | 0,02187 | 0,80000 | 12,79863 | 301,24785 | 8002,57868 |
| SHA512 2 T fe | 0,01876 | 0,00185 | 0,04055 | 1,16576 | 24,20903 | 667,12489 | 18145,32586 |
| SHA256 2 T hi | 0,01609 | 0,00117 | 0,01728 | 0,70656 | 11,24874 | 290,11479 | 7975,23587 |
| SHA512 2 T hi | 0,01890 | 0,00180 | 0,04276 | 1,16770 | 23,91717 | 674,99822 | 18020,75841 |



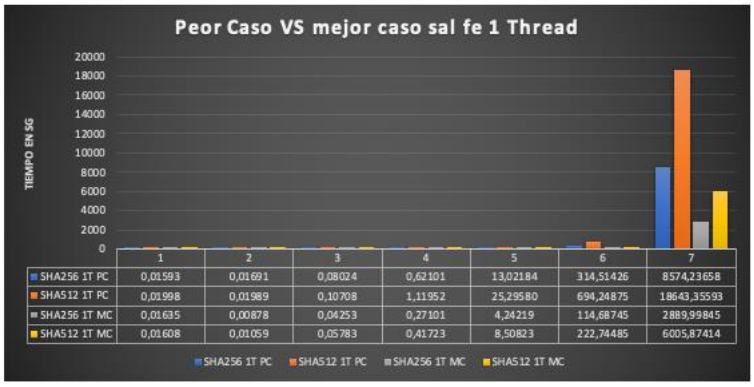

*Grafica Peor caso sal fe VS hi en 2 Threads*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T MC | 0,01528 | 0,00965 | 0,04050 | 0,28457 | 3,56022 | 109,06120 | 2890,48113 |
| SHA512 1T MC | 0,01747 | 0,01013 | 0,05075 | 0,41828 | 8,65471 | 224,73801 | 6010,60988 |
| SHA256 1T PC | 0,01637 | 0,01630 | 0,08126 | 0,61264 | 11,19998 | 329,77242 | 8606,77870 |
| SHA512 1T PC | 0,01746 | 0,02046 | 0,10449 | 1,10852 | 26,26753 | 682,18412 | 18457,69874 |



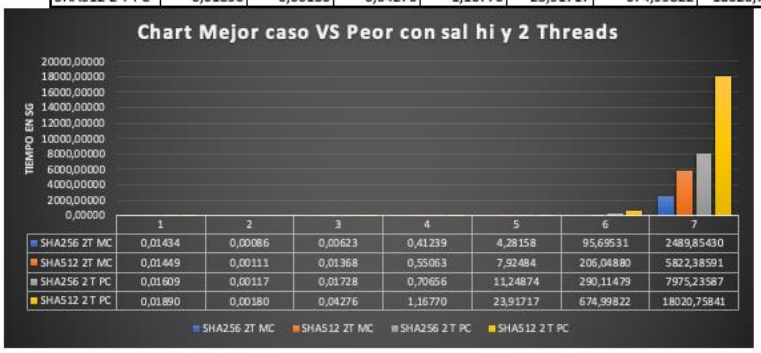

*Grafica Mejor caso vs Peor caso sal hi 1 Thread*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 1T PC | 0,01593 | 0,01691 | 0,08024 | 0,62101 | 13,02184 | 314,51426 | 8574,23658 |
| SHA512 1T PC | 0,01998 | 0,01989 | 0,10708 | 1,11952 | 25,29580 | 694,24875 | 18643,35593 |
| SHA256 1T MC | 0,01635 | 0,00878 | 0,04253 | 0,27101 | 4,24219 | 114,68745 | 2889,99845 |
| SHA512 1T MC | 0,01608 | 0,01059 | 0,05783 | 0,41723 | 8,50823 | 222,74485 | 6005,87414 |




*Grafica Peor caso VS Mejor caso sal fe 1 Thread*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2T MC | 0,01434 | 0,00086 | 0,00623 | 0,41239 | 4,28158 | 95,69531 | 2489,85430 |
| SHA512 2T MC | 0,01449 | 0,00111 | 0,01368 | 0,55063 | 7,92484 | 206,04880 | 5822,38591 |
| SHA256 2 T PC | 0,01609 | 0,00117 | 0,01728 | 0,70656 | 11,24874 | 290,11479 | 7975,23587 |
| SHA512 2 T PC | 0,01890 | 0,00180 | 0,04276 | 1,16770 | 23,91717 | 674,99822 | 18020,75841 |




*Grafica Mejor caso VS Peor caso sal hi 2 Threads*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SHA256 2 T PC | 0,01717 | 0,00124 | 0,02187 | 0,80000 | 12,79863 | 301,24785 | 8002,57868 |
| SHA512 2 T PC | 0,01876 | 0,00185 | 0,04055 | 1,16576 | 24,20903 | 667,12489 | 18145,32586 |
| SHA256 2 T MC | 0,01479 | 0,00080 | 0,00723 | 0,42579 | 4,31977 | 99,14574 | 2455,74858 |
| SHA512 2 T MC | 0,01643 | 0,00105 | 0,01446 | 0,52517 | 8,06649 | 210,68989 | 5884,32585 |


*Grafica Peor caso VS Mejor Caso sal fe 2 Threads*

- Velocidad Procesador y estimación :
    - 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.
    - El computador de las pruebas cuenta con un Ryzen 5 5600x, 32 GB de RAM a 3,7 GHz.
    - Cálculos con 6 letras 1 thread:

$$Tiempo\ para\ zzzzzz\ monothread\ SHA256 = 329{,}77s$$

$$Espacio\ de\ busqueda\ = 26^6 = 308\,915\,776$$

$$T\ promedio\ por\ verificación = \frac{329{,}77}{308.915.776} = 0.00000106\ s = 1{,}06ns$$

$$Tiempo\ por\ ciclo = \frac{1}{3.7GHz} = 0{,}27ns$$

$$Numero\ de\ ciclos \frac{1{,}06}{0{,}27} = 3{,}9 = \ 4\ ciclos$$

- Cálculos programa Monothread :

Tiempo que tomaría un programa monothread, en promedio, para encontrar una contraseña en los siguientes casos:

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

$$Espacio\ de\ busqueda\ caracteres = 78^8$$

$$Espacio\ de\ busqueda\ con\ sal = 78^8 * 2^{16}$$

$$Ciclos\ x\ clave = 4$$

$$Ciclos\ para\ descifrar\ todo = \ 78^8 * 2^{16} * 4$$

$$Ciclos\ x\ segundo = 3.7\ GTz = \ 3.700.000.000\ cps$$

$$Tiempo = \ (78^8 * 2^{16} * 4\ ) / \ 3.700.000.000 \ = 97.072.232.862s$$
$$= 3078{,}14\ años$$

- Ccontraseñas 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 :

$$Espacio\ de\ busqueda\ caracteres = 78^{10}$$

$$Espacio\ de\ busqueda\ con\ sal = 78^{10} * 2^{16}$$

$$Ciclos\ x\ clave = 4$$

$$Ciclos\ para\ descifrar\ todo = \ 78^{10} * 2^{16} * 4$$

$$Ciclos\ x\ segundo = 3.7\ GTz = \ 3.700.000.000\ cps$$

$$Tiempo = \ (78^{10} * 2^{16} * 4\ ) / \ 3.700.000.000 \ = 590587454730698s$$
$$= 187274{,}06\ siglos$$

- 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 :

$$Espacio\ de\ busqueda\ caracteres = 78^{12}$$

$$Espacio\ de\ busqueda\ con\ sal = 78^{12} * 2^{16}$$

$$Ciclos\ x\ clave = 4$$

$$Ciclos\ para\ descifrar\ todo = \ 78^{12} * 2^{16} * 4$$

$$Ciclos\ x\ segundo = 3.7\ GTz = \ 3.700.000.000\ cps$$

$$Tiempo = (78^{12} * 2^{16} * 4)/ 3.700.000.000 = 3{,}59313E\text{^}18s$$
$$= 113937{,}54\ millones\ de\ años$$

## Análisis y Entendimiento del Problema

- ### Información adicional sobre algortimos :

Hoy en día algunos algoritmos de cifrado usados son:

- SHA-2: Se utiliza en metodos de seguridad como SSH:

Es una familia de funciones hash criptográficas que incluye SHA-224, SHA-256, SHA-384 y SHA-512... Y se utilizan en aplicaciones de seguridad,para abordar temas como la autenticación, la integridad de datos y la protección de la privacidad.

- SHA-256 Se utiliza en metodos de seguridad SSL y TLS:

Se utiliza en aplicaciones de seguridad como SSL y TLS que son protocolos de seguridad para la transmisión de datos en la web, en particular SHA-256 se utiliza para garantizar la integridad de los datos transmitidos y para autenticar la identidad de los servidores web.

- SHA-512 Se utiliza para el hashing de correos y la verificación de registros digitales :

Se emplea en aplicaciones que requieren una alta seguridad, como la protección de contraseñas y la autenticación de mensajes. Por ejemplo, muchos sistemas de gestión de contraseñas utilizan SHA-512 para almacenar las contraseñas de manera segura.

Dejamos de usar los algoritmos obsoletos, cuando se pueden generar colisiones intencionalmente, esto significa que dos entradas diferentes, produzcan el mismo código de hash. Esto es preocupante ya que representa un riesgo en la seguridad del programa ante ataques que busquen la manera de pasar archivos por otros o falsificar contraseñas.

Del mismo modo, existen razones por las cuales se dejan de usar algoritmos obsoletos, como la vulnerabilidad ante ataques criptográficos avanzados, como los ataques de fuerza bruta, que pueden descifrar claves criptográficas débiles. En este orden de ideas, estos “algoritmos obsoletos” también pueden ser vulnerables a otros tipos de ataques, como los ataques de timing, los ataques de canal lateral y los ataques de inyección de

código. Finalmente, la evolución de la tecnología y el aumento de la potencia de procesamiento de los ordenadores pueden hacer que los algoritmos de cifrado obsoletos sean menos eficaces en la protección de los datos. Por lo tanto, el cambio en los algoritmos de cifrado, se debe a la necesidad de tener mayor seguridad, eficiencia y rendimiento.

- Descripcion Mining:

El proceso de mining consiste en verificar transacciones del sistema de blockchain, por medio de entrar números en su mayoría aleatorios a un hash y que el resultado cuadre con la transacción que tiene el blockchain. Si coincide el código de hash la transacción se vuelve tuya, obtienes un número de bitcoins, luego se genera un nuevo código de hash a adivinar. Y el proceso se repite indefinidamente añadiendo más y más bloques al blockchain a medida que los códigos son encontrados.

Además de verificar transacciones y crear hashes, se encarga de velar por la integridad del sistema blockchain, gracias al consenso descentralizado, donde los nodos de la red verifican y validan transacciones y bloques creados por "mineros". Cada vez que se agrega un nuevo bloque a la cadena de bloques, la cadena se actualiza y válida con los bloques anteriores. Este proceso de consenso es lo que garantiza que ninguna transacción pueda ser manipulada o eliminada sin ser detectada, lo que a su vez garantiza la confianza y la transparencia de la red Bitcoin.

- Rainbow Tables :

El problema de seguridad asociado es un ataque de fuerza bruta, esto ya que las rainbow tables contienen una tabla de claves y su hash correspondiente en algún cifrado y lo que se hace es comparar este hash con hashes en una base de datos comprometida para averiguar a que clave corresponde cuál hash en la base de datos. Usar una sal ayuda a solucionar el problema, porque la sal agrega nuevos valores a la clave original para que de esta forma al pasar por el algoritmo de hash su resultado sea distinto, de esta forma aun así dos claves tengan la misma palabra sus hashes serán distintos y por ende no se podría usar una rainbow table, ya que ninguna clave es el texto original sino una combinación del texto original con la sal.

Es así como el proceso de descifrado se vuelve lento y costoso para los atacantes, ya que se debe aplicar el proceso de hashing de manera individual para cada posible combinación de clave y sal (teniendo en cuenta que la sal aumenta la complejidad de la clave original) lo que hace que los atacantes necesiten más tiempo y recursos para descifrarla, y deriva en un aumento en la seguridad del sistema en general.

## Referencias:


- Antonopoulos, A. M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Inc.
- Clowes, J. (2013). Salting Your Password: Best Practices. Retrieved from https://crackstation.net/hashing-security.htm
- European Union Agency for Cybersecurity (ENISA). (2018). Cryptographic algorithms and protocols. Technical report. Retrieved from https://www.enisa.europa.eu/publications/cryptographic-algorithms-and-protocols-2018
- Ferguson, N., Schneier, B., & Kohno, T. (2010). Cryptography engineering: design principles and practical applications. John Wiley & Sons.
- European Union Agency for Cybersecurity (ENISA). (2018). Cryptographic algorithms and protocols. Technical report. Retrieved from https://www.enisa.europa.eu/publications/cryptographic-algorithms-and-protocols-2018
- What is Bitcoin Mining? (In Plain English) (99Bitcoins). (2019). Retrieved from What is Bitcoin Mining? (In Plain English)
- How does bitcoin mining work (Euny Hong) (2022) Investopedia. Retrieved from https://www.investopedia.com/tech/how-does-bitcoin-mining-work/
- Rainbow Table Password Attack – Que es y como te proteges de ella (En Giles) (2020) Retrieve from https://www.cyclonis.com/es/rainbow-table-password-attack-que-es-y-como-te-proteges-de-ella/


Contenidos del curso


- Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1997). Handbook of applied cryptography. CRC press.
- National Institute of Standards and Technology (NIST). (2019). Security of cryptographic algorithms and modules. Special Publication 800-131A. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
- National Institute of Standards and Technology (NIST). (2021). Cryptographic standards and guidelines. Special Publication 800-175B. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175Br1.pdf

- Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf
- Stallings, W. (2017). Cryptography and network security: principles and practice. Pearson.

Thomas, M., & Cross, C. (2018). Cybersecurity and applied mathematics. CRC Press.