PROFESORES:

Sandra Julieta Rueda Rodriguez
Jesus David Borre Ordosgoitia

INTEGRANTES:

Kevin Steven Gamez Abril (201912514)
Sergio Julian Zona Moreno (201914936)

Tabla de contenido

1 Introducción

En el presente documento se planteará solución a los expuesto en el caso 2 de infraestructura computacional. Dentro de este archivo encontrará las gráficas y datos pertinentes de la experimentación de valores en la aplicación diseñada. De igual manera, se presentará una pequeña descripción de la implementación utilizada para cumplir con los requerimientos funcionales.

1.1 Aclaraciones preliminares

-Una recomendación previa para la visualización de este documento es utilizar un zoom de 150%; esto permite visualizar todo con nitidez y no fuerza la vista al lector.
-Todo el código correspondiente se encuentra en el repositorio: https://github.com/kevingamez/Caso_2_Infracomp.git (este repositorio es público por lo que toda la información puede ser consultada sin ningún inconveniente).

2 Algoritmos

2.1 Investigación de los algoritmos y cuáles son utilizados hoy en día

a. MD5:

El algoritmo MD5 es utilizado como una función de codificación o huella digital de una cadena o archivo (MD5Online, 2020). Se encuentra compuesto por una cadena de 32 caracteres hexadecimales y no permite su obtención inversa (es decir, si se codifica un archivo con MD5, es imposible a partir del código criptográfico generado obtener el archivo inicial).

El principal uso del algoritmo MD5 es garantizar integridad de un archivo o cadena, es decir, verificar que el mensaje original no haya sido afectado durante su envío. Sin embargo, MD5 se encuentra en desuso hoy en día; esto se debe a que MD5 tiene inconvenientes con las colisiones que genera (cuando dos o más archivos generan un mismo código Hash) por lo que ciertos atacantes pueden modificar el mensaje original por otro que genere el mismo hash. Además, MD5 presenta vulnerabilidades frente a las preimágenes (es decir, a ataques de fuerza bruta, como los que realizamos en nuestro caso); si bien el gasto computacional es elevado, se denota una mayor facilidad con respecto a otros tipos de algoritmos.

b. SHA-256, SHA-384 y SHA-512:

Los algoritmos de la serie SHA-2 son utilizados medios de codificación o huella digital de una cadena o archivo. Fueron desarrollados por la Agencia Nacional de Seguridad de los Estados Unidos (NSA) y tienen el propósito de generar códigos únicos sobre archivos para evitar la modificación de atacantes. Según el número de bits tienen distintos nombres y complejidades:

- SHA-256 genera cadenas de longitud 64. SHA-256 actualmente se utiliza como direcciones de billeteras dentro de las criptomonedas (especialmente en Bitcoin), esto asegura los datos de los individuos y permite el anonimato y descentralización de este tipo de mercado.
- SHA-384 genera cadenas de longitud. Este algoritmo, al tener un punto intermedio, no tiene implementaciones considerables, no logra ser tan ágil como SHA-256, pero tampoco tan seguro como SHA-512, por lo que no es utilizado ampliamente en el mercado salvo ciertas excepciones particulares.
- SHA-512 genera cadenas de longitud 128. SHA-512 tiene aplicaciones en muchos ámbitos en la actualidad, desde encriptación de contraseñas en sistemas operativos como Unix y Linux, hasta identificar vídeos almacenados en tribunales de justicia (véase el caso de Ruanda).

Todos estos algoritmos NO manejan colisiones, por lo que no existen posibilidades de daño en la integridad de datos sobre el mensaje. Sin embargo, al igual que todos estos tipos de algoritmos, tienen problemas en la preimagen, por lo que es posible obtener el mensaje aplicando fuerza bruta. Sin embargo, es algo inviable computacionalmente. Dado que se deben comprobar 2^128 (para SHA-512 y SHA-384) y 2^64 (para SHA-256) combinaciones posibles para romper estos algoritmos.

2.2 Medición de tiempos y presentación de gráficos

Explicación de la implementación:

Nuestra implementación se basó en el uso de un ataque de diccionario y fuerza bruta para identificar una cadena a partir de Hashes. Dentro de nuestro proyecto, pueden ser observadas 4 clases importantes que son utilizadas para la solución del requerimiento. En primera instancia, encontramos el Main.java, que es donde se ejecuta la aplicación; por medio de esta se llama a Hash.java. Utilizamos una base embebida de Apache Derby para solucionar problemas generales de rendimiento, llenamos esta base de datos como un modelo de ataque de diccionario donde intentamos disminuir a complejidades constantes. El diccionario (el código se encuentra dentro de la clase AtaqueDiccionario.java) utilizado como base son la mayor parte de palabras del español y el inglés, es decir, alrededor de 250.000 palabras que serán encontradas de manera instantánea. Si este ataque falla, se procede a utilizar fuerza bruta calculando todas las combinaciones posibles en (para esto se utiliza la clase Combinaciones.java). Dado que consultamos por orden lexicográfico, el peor caso posible serán todos aquellos Threads concurrentes que tengan terminación en "…zzz…".

Especificaciones de la máquina y métodos de Java utilizados:

Al utilizar el método `System.currentTimeMillis()` se obtiene una implementación para el cálculo de dos momentos en un programa de Java. Como aclaración previa, queremos denotar que la máquina utilizada es una ofrecida por AdmonSis, esta máquina tiene las siguientes características generales:

- Procesador Intel(R) Xeon(R) CPU ES-2640 v3 @ 2.60GHz 2.59 GHz, con 8 núcleos.
- 16 GB de memoria RAM disponible.

Datos obtenidos:

Con base en este PC se presenta la siguiente tabla de resultados obtenidos:

Imágenes anexas de consola:

MD5:

```
Se encontró 1 palabra con el código fbade9e36a3f36d3d676c1b808451dd7
z: fbade9e36a3f36d3d676c1b808451dd7
: fbade9e36a3f36d3d676c1b808451dd7
El proceso de obtención del código tardó: 748 milisegundos
```

```
Se encontró 1 palabra con el código 25ed1bcb423b0b7200f485fc5ff71c8e
zz: 25ed1bcb423b0b7200f485fc5ff71c8e
El proceso de obtención del código tardó: 967 milisegundos
```

```
Se encontró 1 palabra con el código f3abb86bd34cf4d52698f14c0da1dc60
zzz: f3abb86bd34cf4d52698f14c0da1dc60
El proceso de obtención del código tardó: 1244 milisegundos
```

```
Se encontró 1 palabra con el código 02c425157ecd32f259548b33402ff6d3
zzzz: 02c425157ecd32f259548b33402ff6d3
El proceso de obtención del código tardó: 1686 milisegundos
```

```
Se encontró 1 palabra con el código 95ebc3c7b3b9f1d2c40fec14415d3cb8
zzzzz: 95ebc3c7b3b9f1d2c40fec14415d3cb8
El proceso de obtención del código tardó: 15292 milisegundos
```

```
Se encontró 1 palabra con el código f0e8fb430bbdde6ae9c879a518fd895f
zzzzzzz: f0e8fb430bbdde6ae9c879a518fd895f
El proceso de obtención del código tardó: 3604208 milisegundos
```

SHA-256:

```
Se encontró 1 palabra con el código fbade9e36a3f36d3d676c1b808451dd7
z: fbade9e36a3f36d3d676c1b808451dd7
Codigo hash desencriptado: fbade9e36a3f36d3d676c1b808451dd7
El proceso de obtención del código tardó: 460 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código 4a60bf7d4bc1e485744cf7e8d0860524752fca1ce42331be7c439fd23043f151
zz: 4a60bf7d4bc1e485744cf7e8d0860524752fca1ce42331be7c439fd23043f151
Codigo hash desencriptado: 4a60bf7d4bc1e485744cf7e8d0860524752fca1ce42331be7c439fd23043f151
El proceso de obtención del código tardó: 690 milisegundos
```

```
Se encontró 1 palabra con el código 17f165d5a5ba695f27c023a83aa2b3463e23810e360b7517127e90161e
zzz: 17f165d5a5ba695f27c023a83aa2b3463e23810e360b7517127e90161eebabda
Codigo hash desencriptado: 17f165d5a5ba695f27c023a83aa2b3463e23810e360b7517127e90161eebabda
El proceso de obtención del código tardó: 757 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código 2d6ccd34ad7af363159ed4bbe18c0e43c681f606877d9ffc96b62200720d7291
zzzz: 2d6ccd34ad7af363159ed4bbe18c0e43c681f606877d9ffc96b62200720d7291
Codigo hash desencriptado: 2d6ccd34ad7af363159ed4bbe18c0e43c681f606877d9ffc96b62200720d7291
El proceso de obtención del código tardó: 1321 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código 68a55e5b1e43c67f4ef34065a86c4c583f532ae8e3cda7e36cc79b611802ac07
zzzzz: 68a55e5b1e43c67f4ef34065a86c4c583f532ae8e3cda7e36cc79b611802ac07
Codigo hash desencriptado: 68a55e5b1e43c67f4ef34065a86c4c583f532ae8e3cda7e36cc79b611802ac07
El proceso de obtención del código tardó: 13220 milisegundos
```

```
Se encontró 1 palabra con el código 453e41d218e071ccfb2d1c99ce23906a
zzzzzz: 453e41d218e071ccfb2d1c99ce23906a
El proceso de obtención del código tardó: 276194 milisegundos
```

```
Se encontró 1 palabra con el código f0e8fb430bbdde6ae9c879a518fd895f
zzzzzzz: f0e8fb430bbdde6ae9c879a518fd895f
El proceso de obtención del código tardó: 3604208 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código 95fbeb8f769d2c0079d1d11348877da944aaefaba6ecf9f7f7dab6344ece8605
zzzzzz: 95fbeb8f769d2c0079d1d11348877da944aaefaba6ecf9f7f7dab6344ece8605
Codigo hash desencriptado: 95fbeb8f769d2c0079d1d11348877da944aaefaba6ecf9f7f7dab6344ece8605
El proceso de obtención del código tardó: 263917 milisegundos
```

SHA-384:

```
Se encontró 1 palabra con el código c39c06ca383f11c2870c8ea1368e861cee29dde246368c17b6985f7a7d650d86a90aa8bbb176ddbd99f06d490f0495e5
z: c39c06ca383f11c2870c8ea1368e861cee29dde246368c17b6985f7a7d650d86a90aa8bbb176ddbd99f06d490f0495e5
: c39c06ca383f11c2870c8ea1368e861cee29dde246368c17b6985f7a7d650d86a90aa8bbb176ddbd99f06d490f0495e5
El proceso de obtención del código tardó: 779 milisegundos
```

```
Se encontró 1 palabra con el código f3309d55afedce7f60aaa318e029241880c5ecfb7702789f5f1bcae9fbc64a1216f9f5c0789165db48a0f224009f608a
zz: f3309d55afedce7f60aaa318e029241880c5ecfb7702789f5f1bcae9fbc64a1216f9f5c0789165db48a0f224009f608a
El proceso de obtención del código tardó: 366 milisegundos
```

```
Se encontró 1 palabra con el código 135ec8936cbfee7cf6b5b807309dc166ba5a65aba67ab535189152712df59cf4f0b9afcbef7b358f705d692639f62b2e
zzz: 135ec8936cbfee7cf6b5b807309dc166ba5a65aba67ab535189152712df59cf4f0b9afcbef7b358f705d692639f62b2e
El proceso de obtención del código tardó: 332 milisegundos
```

```
Se encontró 1 palabra con el código 02c425157ecd32f259548b33402ff6d3
zzzz: 02c425157ecd32f259548b33402ff6d3
El proceso de obtención del código tardó: 1943 milisegundos
```

```
Se encontró 1 palabra con el código e5036f27f03c63a0ce40d821b79080c6aa8cda5c5406167173498f760623cdd7c850b99982d11f4758a01f252eaeaa6a
zzzzz: e5036f27f03c63a0ce40d821b79080c6aa8cda5c5406167173498f760623cdd7c850b99982d11f4758a01f252eaeaa6a
El proceso de obtención del código tardó: 26524 milisegundos
```

```
Se encontró 1 palabra con el código a97bc16b699a86a18e832220b464efc2c25072fdb81e12c3bacfcc08e8b9af5be9c8aed029149dd93200a93fc633fcb9
zzzzzz: a97bc16b699a86a18e832220b464efc2c25072fdb81e12c3bacfcc08e8b9af5be9c8aed029149dd93200a93fc633fcb9
El proceso de obtención del código tardó: 384163 milisegundos
```

SHA-512:

```
Se encontró 1 palabra con el código 5ae625665f3e0bd0a065ed07a41989e4025b79d13930a2a8c57d6b4325226707d956a082d1e91b4d96a793562df98fd03c9dcf743c9c7b4e3055d4f9f09ba015
z: 5ae625665f3e0bd0a065ed07a41989e4025b79d13930a2a8c57d6b4325226707d956a082d1e91b4d96a793562df98fd03c9dcf743c9c7b4e3055d4f9f09ba015
Codigo hash desencriptado: 5ae625665f3e0bd0a065ed07a41989e4025b79d13930a2a8c57d6b4325226707d956a082d1e91b4d96a793562df98fd03c9dcf743c9c7b4e3055d4f9f09ba015
El proceso de obtención del código tardó: 720 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código aa91a7066b85f39d2224771d2660f202b8d123e9dbbdc22c66638ca9ca625b0de96c678a2a235fbbd0531359f104b7e9dec726094d704fb39360563ef8a0cdfd
zz: aa91a7066b85f39d2224771d2660f202b8d123e9dbbdc22c66638ca9ca625b0de96c678a2a235fbbd0531359f104b7e9dec726094d704fb39360563ef8a0cdfd
Codigo hash desencriptado: aa91a7066b85f39d2224771d2660f202b8d123e9dbbdc22c66638ca9ca625b0de96c678a2a235fbbd0531359f104b7e9dec726094d704fb39360563ef8a0cdfd
El proceso de obtención del código tardó: 849 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código 617e115814675db57bd7fd21b2e5471d612583de968b29bf08b9aec647cbfbc3f90269f41151eaafed0cbdba0d3a0a1db79e0dab03b8f6f1c6af57e43e0426ed
zzz: 617e115814675db57bd7fd21b2e5471d612583de968b29bf08b9aec647cbfbc3f90269f41151eaafed0cbdba0d3a0a1db79e0dab03b8f6f1c6af57e43e0426ed
Codigo hash desencriptado: 617e115814675db57bd7fd21b2e5471d612583de968b29bf08b9aec647cbfbc3f90269f41151eaafed0cbdba0d3a0a1db79e0dab03b8f6f1c6af57e43e0426ed
El proceso de obtención del código tardó: 1122 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código 9152a18e3ece6165d5b226ee685707d92993888dace30f56e2f4b4d2f4058ceacc342eed98e5a95995d88c0e8d1a79c68ff9f520d8abc9250c412d873e97d4f5
zzzz: 9152a18e3ece6165d5b226ee685707d92993888dace30f56e2f4b4d2f4058ceacc342eed98e5a95995d88c0e8d1a79c68ff9f520d8abc9250c412d873e97d4f5
Codigo hash desencriptado: 9152a18e3ece6165d5b226ee685707d92993888dace30f56e2f4b4d2f4058ceacc342eed98e5a95995d88c0e8d1a79c68ff9f520d8abc9250c412d873e97d4f5
El proceso de obtención del código tardó: 1639 milisegundos
```

```
No funcionó el ataque por diccionario. Se procede a utilizar fuerza bruta.
Se encontró 1 palabra con el código de2adab3fb5ff37f39a37695a9ddeae600774d1ca00244651c023eb3fa413db1a9454e771db6fad39363dc6fd1c4e5eac2e53a3d86117bed95017b2394b8e4b4
zzzzz: de2adab3fb5ff37f39a37695a9ddeae600774d1ca00244651c023eb3fa413db1a9454e771db6fad39363dc6fd1c4e5eac2e53a3d86117bed95017b2394b8e4b4
Codigo hash desencriptado: de2adab3fb5ff37f39a37695a9ddeae600774d1ca00244651c023eb3fa413db1a9454e771db6fad39363dc6fd1c4e5eac2e53a3d86117bed95017b2394b8e4b4
El proceso de obtención del código tardó: 15971 milisegundos
```

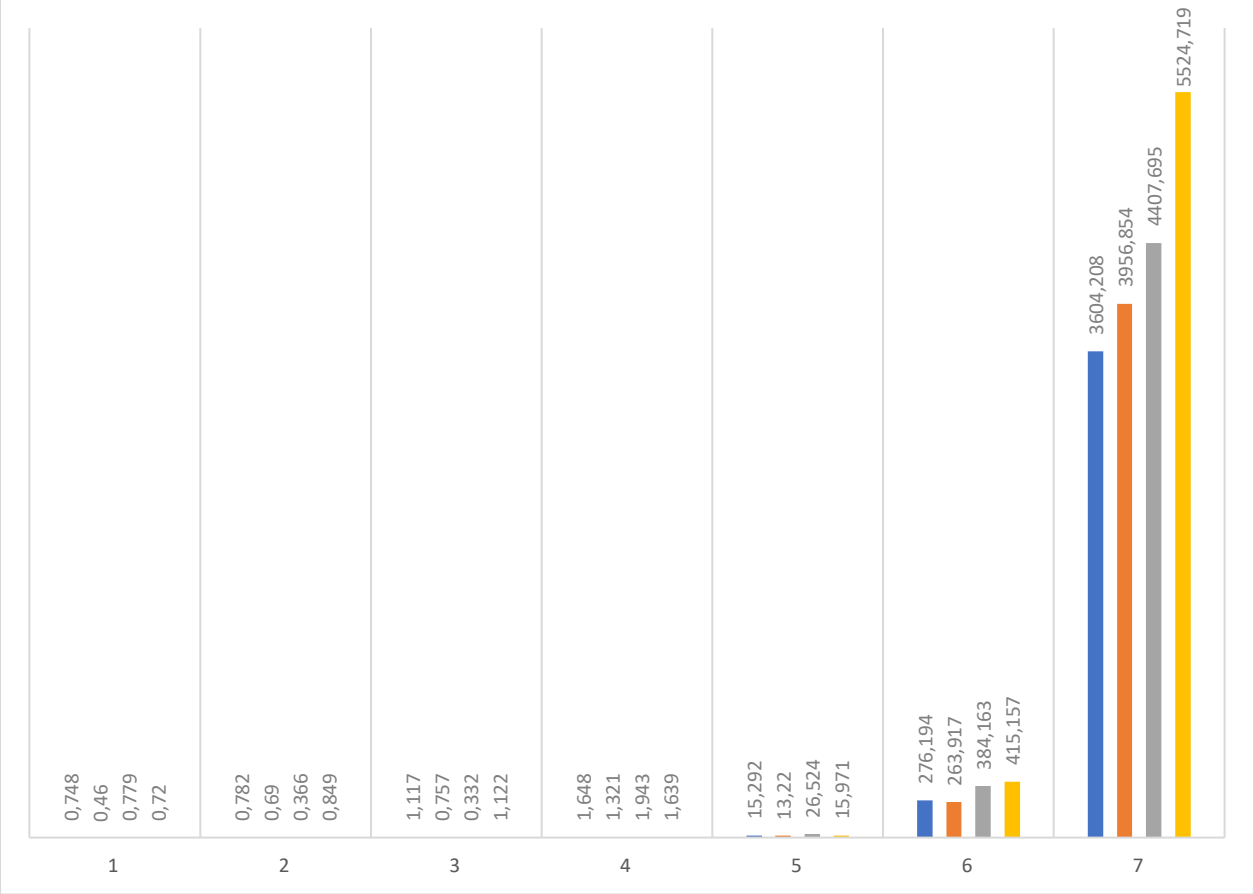
```
Se encontró 1 palabra con el código a97bc16b699a86a18e832220b464efc2c25072fdb81e12c3bacfcc08e8b9af5be9c8aed029149dd93200a93fc633fcb9
zzzzzz: a97bc16b699a86a18e832220b464efc2c25072fdb81e12c3bacfcc08e8b9af5be9c8aed029149dd93200a93fc633fcb9
El proceso de obtención del código tardó: 415157 milisegundos
```

```
Se encontró 1 palabra con el código 1d7f884b782df1d854233d9a46805699763ecdd64fdc444f3a7aa081a67a3fd85369484c76b49e30b77549658bd79f57485fa3c90f04983f70b532fcf43081b8
zzzzzzz: 1d7f884b782df1d854233d9a46805699763ecdd64fdc444f3a7aa081a67a3fd85369484c76b49e30b77549658bd79f57485fa3c90f04983f70b532fcf43081b8
El proceso de obtención del código tardó: 5524719 milisegundos
```

2.3 Estimación del peor caso de una llave de 16 caracteres

Algoritmo	Fórmula	Tiempo estimado (años)
MD5	(27^16)/2902262 llaves/seg	883623953,8
SHA-256	(27^16)/2643603 llaves/seg	970080244
SHA-384	(27^16)/2373202 llaves/seg	1080610465
SHA-512	(27^16)/18933722 llaves/seg	1354465127

2.4 Tiempos registrados



2.5 Número de ciclos por segundo por llave

Ciclos
2,19874E+17
2,41387E+17
2,68891E+17
3,37034E+16

3 Análisis de la problemática de la empresa

Dada la intercomunicación entre los tres servidores, se denotan como vulnerabilidades:

- La conexión a internet que permite puertas de entrada a la red y problemas de confidencialidad.
- La constante información para el cambio de llaves genera canales donde pueden ingresar mensajes modificados por atacantes externos, por lo que existen problemas de integridad de datos. Además, no se puede asegurar la autenticación del emisor, ni el no repudio del mismo.

Por lo anterior, los datos que debe manejar el sistema para protección son todos aquellos que corresponden al componente de mensajería e intercomunicación de información. Estos mensajes deben ser encriptados con algoritmos que permitan confidencialidad, y, además, deben enviar de manera paralela un paquete con un código de hash.

4 Conclusión

Se cumplieron con los requerimientos y se comprobó la dificultad de romper algoritmos criptográficos utilizando fuerza bruta.

5 Bibliografía y referencias

- GeeksforGeeks. (2019, 21 febrero). Print all possible strings of length k that can be formed from a set of n characters. https://www.geeksforgeeks.org/print-all-combinations-of-given-length/
- Llanos, J. (2019, 9 abril). MD5: vulnerabilidades y evoluciones (y II) - Think Big Empresas. Think Big. https://empresas.blogthinkbig.com/md5-vulnerabilidades-y-evoluciones-y-ii/
- Stack Overflow. (2010, 12 noviembre). How to securely hash passwords? Information Security Stack Exchange. https://security.stackexchange.com/questions/211/how-to-securely-hash-passwords/31846#31846
- MD5Online. (2018, 17 noviembre). Cifrar MD5. https://md5online.es/cifrar-md5
- Academy, B. (2020, 10 noviembre). 2754. Bit2Me Academy. https://academy.bit2me.com/sha256-algoritmo-bitcoin/