Juan José Beltrán Ruiz – 201819446

Santiago Bobadilla Suárez – 201820728

# Caso de Estudio 3 – Seguridad

# Implementación del Prototipo

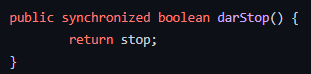
Aparte del código anexado, la implementación completa se encuentra en el siguiente repositorio: <https://github.com/Infracomp-Bobadilla/Caso_3> . Ahí, hay dos carpetas. La primera, y anexada, contiene el proyecto en eclipse donde la clase ‘Main’ permite definir la cadena a probar, el número de ceros a usar y el algoritmo deseado. Así de forma sencilla se puede realizar las pruebas necesarias. La segunda carpeta, contiene la misma implementación, no obstante, está basada en un proyecto de ‘Gradle’ y la clase ‘Main’ esta parametrizada. Esto se debe a que este proyecto se comprimió en un ‘.jar’ con el fin de mejorar los tiempos, y por tanto su ejecución se pudiera hacer desde la consola de comandos ya fuera desde Windows o desde Linux. La carpeta del ‘.jar’ no se encuentra anexada, pero si presente en ‘Github’ con su ultimo ‘Commit’ en las fechas permitidas, por si desean revisar el empaqueta o la ejecución.

Respecto a la implementación, ambas carpetas contienen el mismo código, a excepción de los cambios en la clase ‘Main’. Su funcionamiento se basa en tres clases:

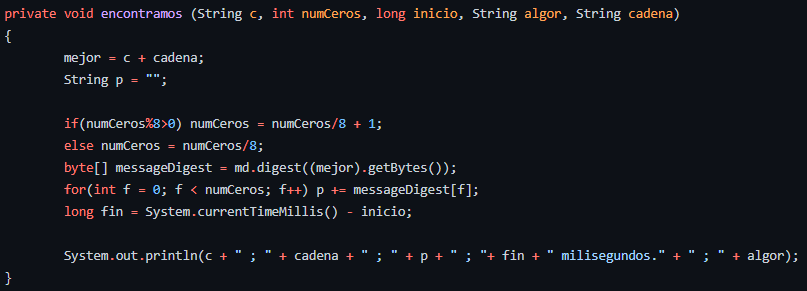
**I) Monitor\_Minado**

Esta clase se usó como un monitor con el fin de controlar la concurrencia de los Threads, y contiene la implementación del algoritmo de minado solicitado. En primera instancia, cuanta con varios atributos: ‘stop’, ‘centinela’ y ‘mejor’ ayudan a la sincronización y concurrencia; ‘md’, y ‘strb’ se proveen para generar el cifrado. El constructor de la clase recibe por parámetro la instancia que permite cifrar con el algoritmo dado. Por último, esta clase cuenta con cuatro métodos, tres auxiliares y uno principal.

* darStop (): Este método retorna la variable booleana ‘stop’ la cual indica a todos los Threads en ejecución si deben o no proceder su búsqueda. En caso de que ya haya una coincidencia válida para el minado esta variable es la que detiene al resto.



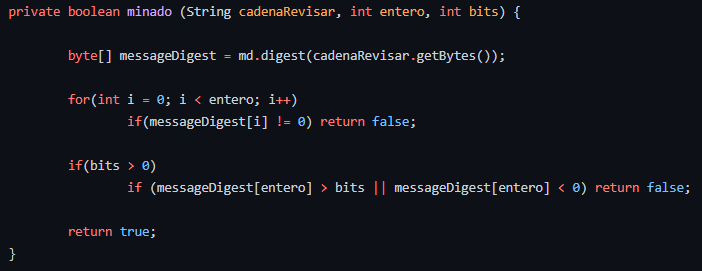
* encontramos (): Este método se encuentra sincronizado sobre el objeto ‘centinela’ y su función es imprimir en consola los resultados de la coincidencia que fue valida. Recibe como parámetros ‘c’, que hace referencia a la combinatoria que dio una coincidencia valida; ‘numCeros’, correspondiente a la cantidad de ceros que se buscaba obtener y ayuda a imprimir con el fin de validar que la coincidencia fue valida; ‘inicio’, el cual contiene la cantidad de tiempo que ha pasado con el fin de registrar con exactitud cuanto se tardo en encontrar la coincidencia; ‘algor’, para reportar el algoritmo; y ‘cadena’, para mostrar la cadena evaluada.



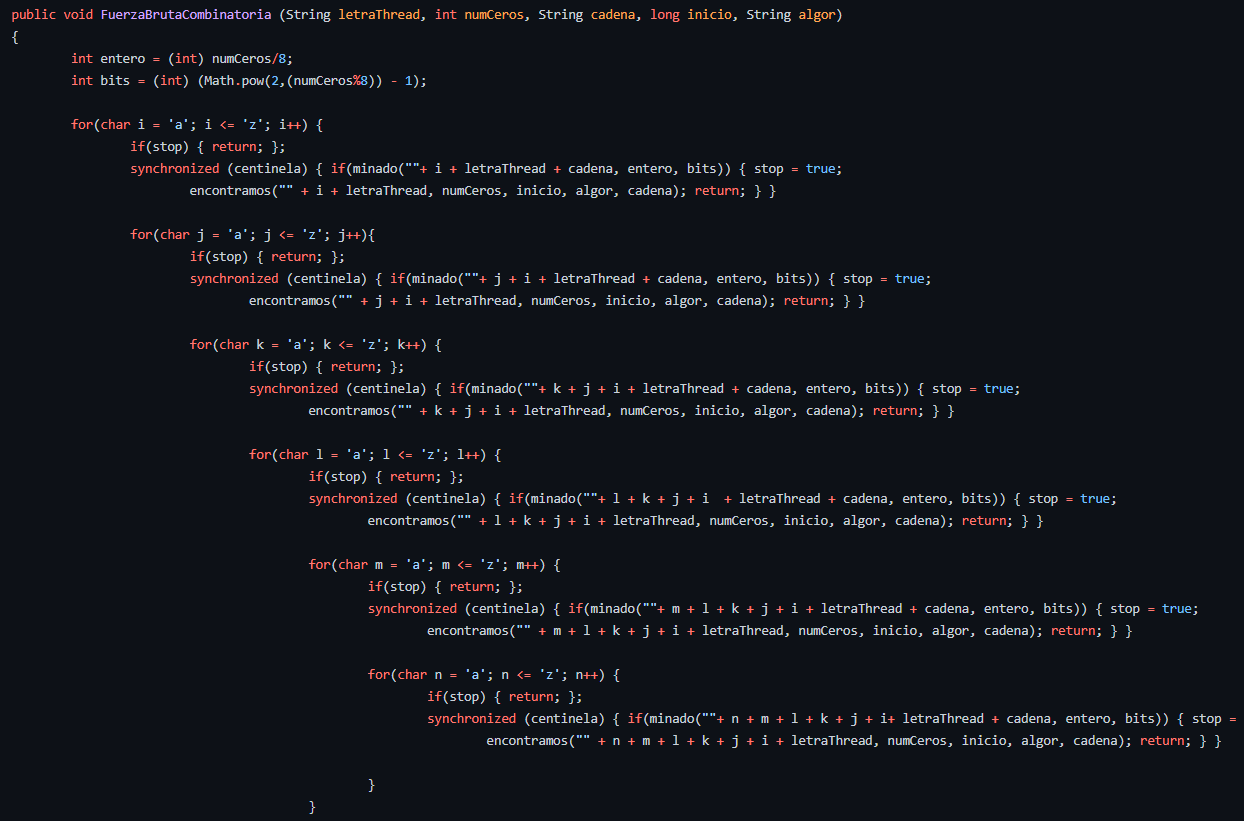
* minado (): Este método se encarga de revisar si la combinación ‘V + cadena’ da un cifrado con la cantidad de ceros a la izquierda solicitada. Con el fin de entender bien la validación, se debe tener en cuenta que un 1 byte contiene 8 bits. Por tanto, si se busca hacer la comparación directamente con los bytes, se debe contemplar ciertas conversiones y propiedades. Primero, si un byte esta en 0, se implica inmediatamente que los 8 bits estuvieron en cero. No existe otra forma de obtener este valor. Segundo, si me piden una cantidad menor a 8 bits en cero, la forma de validarlo en byte se da por medio del máximo valor presente en este. Este máximo valor se calcula como . Así, si el byte en valor es menor a este número (y mayor a cero) se puede afirmar que tiene la cantidad de ceros deseada.



Esta afirmación se hace con el fin de facilitar la búsqueda cuando se pide una cantidad de ceros la cual no es múltiplo de 8, como lo fue por ejemplo 20, 28 y 36. En este orden de ideas, y con esto en mente, los parámetros que recibe este método son ‘cadenaRevisar’ el cual su nombre dice todo; ‘entero’, en cual son las posiciones iniciales donde el valor del byte debe estar en 0; y ‘bits’ el cual corresponde al valor máximo que puede presentar el ultimo byte a evaluar. El método devuelve ‘true’ si toda la cadena fue valida, y ‘false’ si una de estas condiciones ya explicadas no se cumple. Cabe resaltar que este método, está igualmente sincronizado sobre el objeto ‘centinela’ con el fin de evitar problemas de concurrencia.



* FuerzaBrutaCombinatoria (): Este método se encarga de generar todo el espacio de combinaciones posible. El cual para este caso es por cada Thread. Al contar con 26 Threads, uno por cada letra, se contempla todas las posibilidades del espacio . La forma en que se arman los diferentes V es por medio de ciclos anidados. Cada ciclo va de ‘a’ hasta ‘z’ y siempre se va concatenando con el ciclo anterior. Además, en cada ciclo se contemplan varias operaciones. Uno, se valida si se puede romper la creación de combinaciones en caso de que ya haya una valida. Dos, sincronizado bajo el objeto centinela, se revisa si la ‘V + Cadena’ es valido o no. Si sí lo es, se da la orden de parar la ejecución de los otros Threads, se imprime el resultado y se detiene igualmente el Thread que hizo la coincidencia.



Por último, cabe resaltar que como nunca se especifica en la rúbrica, ni en el enunciado el orden en que se debe concatenar, se hace la unión en orden ‘V + Cadena’.

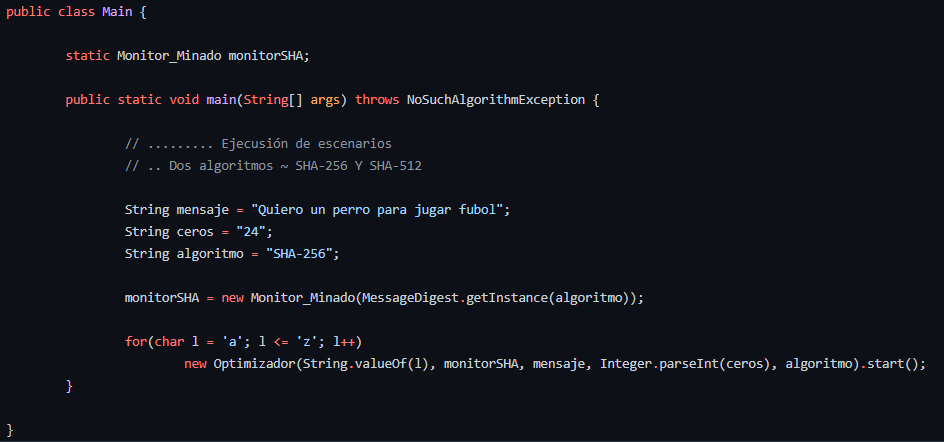
**II) Optimizador**

La clase optimizador funciona representando los Threads. Estos cuentan con los siguientes atributos: ‘letra’, la cual es de la ‘a’ a la ‘z’; ‘monitor’, sobre el cual va a ejecutarse; y ‘cadena’, ‘numCeros’, y ‘algoritmo’ que son la cadena que se esta buscando, cuantos ceros busca y que algoritmo esta revisando y permite tanto hacer configuraciones personalidades como imprimir los resultados de cuando hubo o no coincidencia de forma precisa. El único método con el que cuenta en su run () y hace dos cosas. Uno, ejecutar el método de FuerzaBrutaCombinatoria de la clase Monitor\_Minado. Dos, registrar el tiempo que le tardo cubrir todo el espacio de búsqueda dado que no hubo coincidencias.



**III) Main**

Esta clase cuenta con el método principal el cual permite correr el programa. Puede estar parametrizado por medio de Strings predefinidos que facilitan correrlo, o como argumentos que recibe al correr desde consola. En ambos casos, se procede a crear el monitor de los Threads y a inicializar 26 Threads, uno por letra desde la ‘a’ hasta la ‘z’.



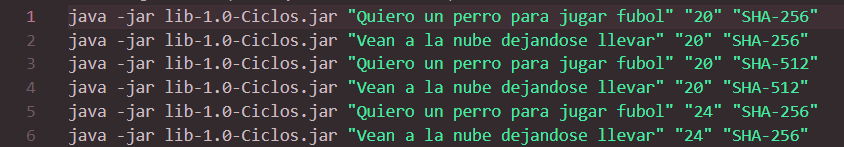
En dado caso de ejecutar desde consola se seguía la siguiente sintaxis y siempre se imprime de la siguiente forma la coincidencia:



En caso de no encontrar, cada Thread muestra un reporte de la siguiente forma:



Por último, para ejecutar varios escenarios en secuencia desde consola se uso un archivo de tipo ‘.bat’ para Windows o ‘.bash’ para Linux. La estructura del ‘.bat’ fue:

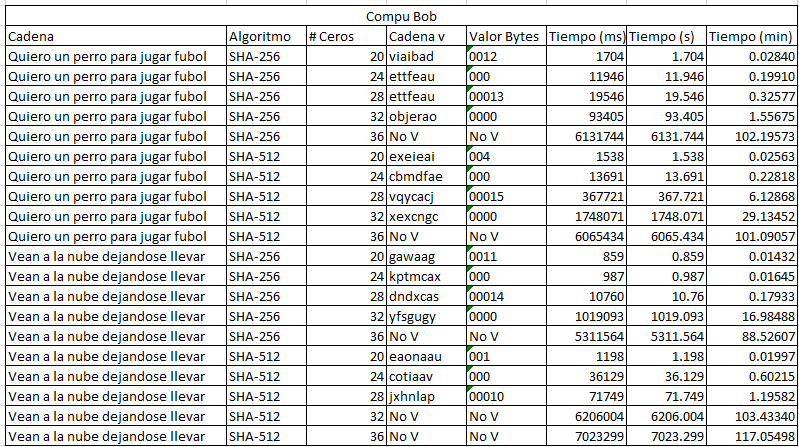


## Corra su programa en diferentes escenarios construidos a partir de las variaciones en los datos de entrada: para cada algoritmo, sobre dos cadenas diferentes y para número de ceros 20, 24, 28, 32, y 36 y construya una tabla con los datos recopilados.

Después de hacer las respectivas pruebas en ambos computadores de forma local, pudimos recopilar dos tiempos diferentes para cada escenario de prueba. También cabe mencionar que la máquina Linux ofrecía un rendimiento considerablemente menor por razones que desconocemos, por lo cual decidimos no incluir dichos tiempos ya que concluimos que iban a dañar los valores. A continuación, se muestran dos tablas, una para cada computador de los integrantes del equipo. Pudimos determinar adicionalmente que la frecuencia de reloj de ambos computadores Windows es de 1.8Ghz al máximo, y para correr las pruebas se pusieron los competidores en dicho rendimiento. Dicho esto, se procede a mostrar las tablas:

Tabla

Descripción generada automáticamente



El orden de las columnas es el siguiente: cadena, algoritmo de encriptado, número de ceros, cadena que resuelve el problema o en su defecto “No V” para indicar que no se encontró ninguna cadena y se tuvo que recorrer el espacio de búsqueda completo, el valor de los Bytes que respaldan la cadena encontrada, el tiempo en milisegundos, tiempo en segundos y finalmente tiempo en minutos. Esta última columna es importante a medida que crece el número de ceros buscado.

## Construya una gráfica que muestre los tiempos registrados para los dos algoritmos usados de acuerdo con el número de ceros buscados. Use los datos del punto anterior.

En este proceso de construcción de tablas, vimos que era necesario hacer varias segmentaciones con el fin de comparar adecuadamente cada algoritmo y máquina utilizada. A continuación, se muestran las gráficas junto con una pequeña explicación de los resultados:

* Gráfica 1: todos los tiempos presentados en la tabla en un gráfico de dispersión:

Es claro en este gráfico que los tiempos de 20 a 28 ceros son muy pequeños, por lo cual no logran distinguirse sus diferencias, sin embargo, esta gráfica es valiosa porque permite entender que en promedio el algoritmo SHA-256 toma un tiempo menor al SHA-512.

* Gráfica 2: tiempos totales del algoritmo SHA-256 en dispersión:

En esta gráfica buscamos diferenciar mejor los tiempos según la cadena, el numero de ceros buscado y el computador utilizado. Se nota que los tiempos para 36 cero crecen bastante dado que no hay cadena que cumpla la condición y por tanto se debe evaluar todo el espacio de búsqueda.

* Gráfica 3: tiempos totales diferenciados para el algoritmo SHA-512:

En este caso es importante notar que los tiempos crecen considerablemente, esto dado que el algoritmo SHA-512 no encuentra una condición para la cadena de la nube en 32 ceros, y que para la cadena del perro terminan varios Threads antes de encontrar el resultado esperado. Es por esto que los tiempos comienzan a subir desde los 32 ceros.

Ahora, con el fin de diferenciar claramente el desempeño entre ceros, se insertan gráficos de barras para saber el tiempo específico que toman las diferentes máquinas, cadenas y algoritmos.

* Gráfico 4: tiempos para 20 ceros en segundos

En este gráfico vemos una ligera volatilidad en los tiempos, la cual atribuimos al manejo de Threads, lo que hace desconozcamos cómo se asignan en el procesador y de la misma manera en qué posición se encuentra el Thread que efectivamente encuentra la cadena.

* Gráfico 5: tiempos para 24 ceros en segundos:

En este caso vemos la misma volatilidad entre los tiempos, pero se puede ver que los tiempos entre máquinas son similares. De la misma manera, conocemos que esta volatilidad se da por la cantidad de procesos que se tienen abiertos a la hora de correr el programa, lo cual implica que haya más procesos compitiendo por los recursos del procesador.

* Gráfico 6: tiempos para 28 ceros en segundos:

Nuevamente, se puede ver que los tiempos de la máquina “Bob” son un poco más altos, dado que en esta se tenían más procesos abiertos, como las clases en “Zoom” y algunos archivos de Excel en ejecución, mientras que las ejecuciones de la máquina “JJ” se hicieron a altas horas de la noche sin programas adicionales corriendo

* Gráfico 7: tiempos para 32 ceros en minutos:

En este caso comienzan a subir los tiempos porque para 32 ceros el algoritmo 512 no encuentra la cadena de la nube, pero se conserva el comportamiento visto anteriormente.

* Gráfico 8: tiempos para 36 ceros en minutos:

En esta cantidad de ceros sabemos que para todos los casos se debe recorrer completamente el espacio de búsqueda, y sabemos que los tiempos son diferentes porque desconocemos la asignación de recursos que hace el sistema operativo.

## Identifique la velocidad de su procesador, y estime cuántos ciclos de procesador toma, en promedio, generar y evaluar un valor para determinar si cumple o no con la condición buscada. Escriba todos sus cálculos.

Para evaluar correctamente el número de casos evaluados, se tomaron los casos en los que se debía recorrer todo el espacio de búsqueda y de esta manera saber que debíamos evaluar combinaciones posibles. Adicionalmente, determinamos que los relojes de ambas máquinas son de 1.8GHz. Con esto, se procedió a hacer los cálculos respectivos:

Luego, se muestran los casos totales que se deben recorrer en el espacio de búsqueda:

En este momento, se fragmentan los tiempos de ambos computadores. Para hacer una estimación que contemple todos los casos se toma el tiempo promedio de las observaciones para 36 ceros, lo cual significa que se recorrió todo el espacio de búsqueda:

Con esto, se puede conocer el tiempo promedio que necesita un caso en cada computador para ser evaluado:

Finalmente, se debe dividir este tiempo promedio por caso por el tiempo que toma un reloj:

Los resultados se consignaron en esta tabla, de igual manera el Excel con los procesos y gráficos se encuentra anexada junto con el informe:

Tabla

Descripción generada automáticamente

# Análisis y Entendimiento del Problema.

## 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.

Con el fin de responder ambos sub literales es necesario empezar por el II y un par de definiciones. Un algoritmo de generación de códigos criptográficos se considera obsoleto cuando su uso y la longitud de clave es permitido pero el usuario debe aceptar algunos riesgos. Estos riesgos se deben a que, con la capacidad actual de los computadores y las habilidades tecnológicas, alguien ya pudo romper la función de seguridad. (Acosta, 2016) Es decir, aquello que se envía haciendo uso de esos algoritmos no se considera seguro, y por tanto si bien se puede, no es recomendable usarlo. En ese orden de ideas, se empieza a resaltar que los primeros algoritmos que se fueron desarrollando en general ya se consideran obsoletos. (Vazquez, 2018)

Por ejemplo, la función Comprobación de Redundancia Cíclica planteada por Wesley Peterson en 1961 fue revolucionaria en su momento, pero actualmente usarla no garantiza de ninguna manera seguridad. Otro ejemplo es la MD2 creada por Ronald Rivest, en el año de 1989, o su posterior actualización MD5 la cual se pudo romper/crakear en 1996. (Me, s.f.)

Actualmente, aquellas que sí se consideran seguras son principalmente la criptografía creada por el proyecto europeo RIPE en 1992, o por la NSA en 1193. (Me, s.f.) Específicamente resaltando las versiones RIPEMD-160 y SHA-256, las cuales se consideran extremadamente completas, incluso, a tal punto que harían falta miles de años usando supercomputadoras para romper su función de seguridad. (Luz, 2016) Existen otros algoritmos igualmente aceptados, con una complejidad teórica menor; al igual, que muchos otros que ya se consideran obsoletos o poco recomendados. Los principales son:

**Aceptados**

1. scrypt
2. bcrypt
3. Argon2
4. sha512crypt
5. sha256crypt
6. PBKDF2

**Obsoletos**

1. MD5
2. md5crypt
3. UNIX crypt (3)
4. SHA-1/2/3
5. Skein
6. BLAKE2
7. Any general-purpose hashing function.
8. Any encryption algorithm.
9. Your own design.
10. Plaintext

(Toponce, 2016)

Los principales temas en los cuales se usan los Hash son:

* Asegurar la integridad de la información
* Firma digital
* Blockchain
* Estructuras de datos
* Gestión de los identificadores y contraseñas

(Vazquez, 2018) (GUTIÉRREZ, 2013)

## La tecnología blockchain se construyó a partir de la propuesta de bitcoin. Busque información sobre blockchain y presente un caso de uso en el que sería razonable incorporar la tecnología blockchain (diferente a bitcoin). Justifique su respuesta: use argumentos concretos (considere las preguntas: ¿qué problemas resuelve blockchain en el caso presentado? y ¿cómo los resuelve?).

Después de hacer una exhaustiva investigación se pudo encontrar que un caso de uso para la tecnología blockchain está comenzando en la industria bancaria china. Este medio se enfrenta actualmente a diferentes presiones, las cuales están relacionadas con un decaimiento en las ventas y un aumento en el riesgo de la economía, lo cual ha hecho que se necesite un cambio en los procesos de desarrollo tecnológico. Inicialmente se da por los nuevos modelos tecnológicos por el boom de las criptomonedas y las finanzas por internet, lo cual está representando un gran reto para los modelos bancarios tradicionales. Es por esto que, los modelos bancarios necesitan nuevos modelos para asegurar procesos de confiabilidad, adaptabilidad e innovación. (Guo & Liang, 2016).

Blockchain se vuelve determinante en esta problemática dado que actualmente se usan modelos de análisis de datos en los que se enriquecen los diferentes escenarios de inversión, para darle más alternativas a los clientes y de esta manera ser más atractivos. El problema de implementar esta medida es que la mayoría de los escenarios que se han corrido ya están implementados, haciendo que sea muy complejo el proceso de innovación. Para resolver esto, Blockchain digitaliza el mercado de activos y una transferencia de valor con seguridad punto a punto, lo cual puede reconstruir la infraestructura financiera (Guo & Liang, 2016). Todo esto resulta en una mejora en los procesos de eficiencia de remuneración y compensación por activos financieros mientras reduce de igual forma los costos posteriores a las transacciones. Cabe mencionar finalmente que dentro del mercado financiero, la criptomoneda BAT ya ha comenzado a implementar el modelo de Blockchain así como bitcoin, por lo que los modelos tradicionales bancarios no deben demorar en tomar esta decisión. (Guo & Liang, 2016)

# Bibliografía

Acosta, D. (08 de 08 de 2016). *PCI DSS*. Obtenido de ¿Qué algoritmos criptográficos se deben emplear para cumplir con PCI DSS?: https://www.pcihispano.com/que-algoritmos-criptograficos-se-deben-emplear-para-cumplir-con-pci-dss/

Guo, Y., & Liang, C. (2016). *Blockchain application and outlook in the banking industry.* Obtenido de Springer Link: https://doi.org/10.1186/s40854-016-0034-9

GUTIÉRREZ, P. (15 de 01 de 2013). *Genbeta*. Obtenido de ¿Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales: https://www.genbeta.com/desarrollo/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales

Luz, S. D. (09 de 06 de 2016). *Redes Zone*. Obtenido de Descubre los mejores y más seguros algoritmos de hash, y conoce los que no son recomendables hoy en día: https://www.redeszone.net/2016/07/09/descubre-los-mejores-mas-seguros-algoritmos-hash-conoce-los-no-recomendables-hoy-dia/

Me, A. B. (s.f.). *Academy Bit 2 Me* . Obtenido de ¿Qué es un hash?: https://academy.bit2me.com/que-es-hash/

Toponce, A. (2016). *PThree.org*. Obtenido de Let's Talk Password Hashing: https://pthree.org/2016/06/28/lets-talk-password-hashing/

Vazquez, A. (22 de 02 de 2018). *CYSAE LEGAL*. Obtenido de ¿Qué son las funciones Hash y para que se utilizan?: https://www.cysae.com/funciones-hash-cadena-bloques-blockchain/