Encripción de Texto a través de Sistemas Distribuidos

Text Encryption through Distributed System

Pineda, Luis D.; Orellana, Marcos; Gutiérrez, Bayron S.; Zambrano-Martínez, Jorge Luis

Luis D. Pineda
eluisdavidpineda@es.uazuay.edu.ec
Universidad del Azuay, Ecuador

Marcos Orellana
marore@uazuay.edu.ec
Universidad del Azuay, Ecuador

Bayron S. Gutiérrez
bgutierrez@es.uazuay.edu.ec
Universidad del Azuay, Ecuador

Jorge Luis Zambrano-Martínez
jorge.zambrano@uazuay.edu.ec
Universidad del Azuay, Ecuador

Latin-American Journal of Computing
Escuela Politécnica Nacional, Ecuador
ISSN: 1390-9266
ISSN-e: 1390-9134
Periodicidad: Semestral
vol. 10, núm. 1, 2023
lajc@epn.edu.ec

Recepción: 10 Agosto 2022
Aprobación: 26 Octubre 2022

URL: http://portal.amelica.org/ameli/journal/602/6023721009/

DOI: https://doi.org/10.5281/zenodo.7503950

Resumen: En los años el crecimiento exponencial de las interconexiones entre dispositivos digitales ha provocado un auge significativo de los incidentes de ciberataques. Esto conllevan a consecuencias desastrosas y graves para una organización, mediante la explotación de vulnerabilidades existentes en tecnologías emergentes. Uno de los mecanismos de defensa más innovadores y efectivos en la actualidad es la criptografía, que se considera un requisito esencial en la comunidad de ciberseguridad para la protección de datos. En este trabajo, implementamos métodos de cifrado en capas, que se ejecutan independientemente en cada máquina de manera distribuida a través del servicio web denominado protocolo de acceso a objetos simples (SOAP), a los datos que se quieren proteger. Para llevarlo a cabo, se creó un escenario experimental mediante un sistema distribuido con máquinas virtuales que albergan los métodos de cifrado comunes como AES-256, César, Blowfish, para enviar datos encriptados desde una máquina origen hacia una máquina destino, y a su vez descifrar los datos encriptados. Los resultados obtenidos demuestran que nuestra propuesta incrementa la seguridad a los datos que son enviados de un origen hacia su destino, dificultando que el atacante obtenga los datos fácilmente. Además, el consumo de recursos computacionales para la ejecución de los algoritmos en el sistema distribuido es mínimo lo cual es adecuado para garantizar su uso en cualquier computador con muy pocos recursos.

Palabras clave: *Criptografía*, *Sistemas Distribuidos*, *SOAP*.

Abstract: In recent years, the exponential growth of interconnections between digital devices has caused a significant rise in cyberattack incidents. This produces severe and disastrous consequences for an organisation by exploiting existing vulnerabilities in emerging technologies. One of the most innovative and effective defence mechanisms today is cryptography, considered an essential requirement in the cybersecurity community for data protection. In this work, we implement encryption methods that are executed independently on each machine in a distributed manner through the web service called Simple Object Access Protocol (SOAP) to protect the data. To perform it, an experimental scenario was created using a distributed system with virtual machines that host standard encryption methods such as AES-256, Caesar, and Blowfish to send encrypted data from a source machine to a destination machine, in turn, decrypt the encrypted data. The results show that our proposal increases the security of the data sent from a source to its destination, making it difficult for the

attacker to obtain the data quickly. In addition, the consumption of computational resources for executing the algorithms in the distributed system is minimal, which is adequate to guarantee its use in any computer with very few resources.

Keywords: *Cryptography*, *Distributed*.

I. INTRODUCCIÓN

Desde sus inicios, la humanidad al ser una especie sociable, ha desarrollado varias formas de comunicarse con otras personas.

Debido a esto, surge la necesidad que cierta parte de la información sea sólo conocida por destinatarios específicos. Esta necesidad de enviar mensajes que únicamente fueran entendidos por los destinatarios, hizo que cada civilización, implementara códigos secretos y métodos de ocultar la información sumamente importante. De esta manera, el mensaje no sea descifrable por aquellas personas que puedan interceptar estos mensajes [1].

En la actualidad, existe una amplia gama de algoritmos que han mejorado el proceso de cifrado y descifrado de datos para diferentes usos. Entre varios ejemplos de procesos de cifrado tenemos algunos muy utilizados en la vida cotidiana, como los mensajes ocultos para un usuario intruso y solo pueden ser observados de punto a punto. El avance de las tecnologías ha posibilitado el crecimiento vertiginoso de los sistemas de información, así como su aplicación sin importar en que campo sea asignado. De esta manera, las telecomunicaciones ofrecen la conectividad entre usuarios sean humanos o máquinas, ubicados en cualquier sitio llegando a conformar una inmensa red intercomunicada. Esto acarrea el desarrollo de centros enfocados en el procesamiento de datos donde se puede desplegar las aplicaciones para llegar a realizar un procesamiento distribuido a una o varias tareas, dependiendo netamente de su configuración. Esta viabilidad ofrece a los usuarios una tolerancia para estructurar de manera eficiente los sistemas de información, y brindar la comodidad de interactuar con otros sistemas distribuidos. Esto ha desembocado una dependencia abismal de los sistemas distribuidos tanto para el procesamiento como para la transmisión de los datos a escala global [2].

Por otra parte, en los sistemas distribuidos, los procesos de transferencia o almacenamiento de información se presentan vulnerabilidades ante cualquier ataque. Con esto en mente, es de suma importancia la protección de la información mediante la conversión de datos de un formato legible a un formato codificado. Este proceso se le conoce como criptografía, que permite transmitir los datos de manera más segura y exclusivamente a las partes interesadas [3]. Consecuentemente, en pro de mejorar e implementar el cifrado de textos, este estudio propone un sistema que permita realizar un cifrado y descifrado de textos en capas de manera independiente a través de la comunicación entre varios computadores utilizando sistemas distribuidos.

Finalmente, este artículo está organizado de la siguiente manera. En la Sección II, se describen los trabajos relacionados con el proceso de cifrado en diferentes métodos en diferentes enfoques presentados por otras investigaciones. La metodología utilizada para llegar a implementar el sistema propuesto y los componentes básicos son analizados en la Sección III. La Sección IV presenta los resultados de la propuesta. Y, por último, en la Sección V, se presentan las conclusiones obtenidas y los trabajos futuros.

II.TRABAJOS RELACIONADOS

Actualmente, los métodos de cifrado presentan una gran variedad de modelos, los cuales son de gran utilidad al momento de cifrar no sólo textos sino gran variedad de archivos. Aunque varios de estos métodos son utilizados, existe una gran cantidad que no llega a ser conocida. Con base a esto, se analizarán algunos trabajos

relacionados donde se utilizan diferentes tipos de métodos de cifrado en diversas áreas. El propósito de la aplicación es cifrar los datos de la forma más segura creando claves siguiendo un protocolo de descifrado para llegar a la información. Así, los autores Dixit et al. [1] analizan algunas técnicas de cifrado hibridas tradicionales y modernas junto con un enfoque cuántico, como Rivest, Shamir y Adleman (RSA) basado en criptografía de curva elíptica (ECC) con la clave variable automática (AVK), estándar de cifrado de datos RSA (DES-RSA), curva cubica singular basada en RSA, Compresión y cifrado de información (JCE), técnica de mapa caótico 3D, Blowfish. Del mismo modo, los autores Mitra et al. [4] estudian un mejor enfoque a los algoritmos de cifrado de datos, resaltando los principales problemas y áreas de aplicación donde estos se adaptan mejor. De este modo, presentan los mejores algoritmos y técnicas para un cifrado y descifrado eficiente de los datos.

Por otro lado, Garcia et al. [5] utilizaron osciladores caóticos de orden fraccionario, y obtuvieron un cifrado más eficaz en relación con los osciladores caóticos de orden entero. El encriptado caótico a través del uso de sistemas enteros evita que una persona sin autorización llegue a descifrar un mensaje previamente cifrado. Esto es debido a que se debe conocer específicamente los parámetros del oscilador que se está utilizando conjuntamente con las condiciones iniciales. Si la persona intrusa llega a utilizar los osciladores dentro de la red, se enfrenta con el problema del desconocimiento de la topología de la red y sus configuraciones.

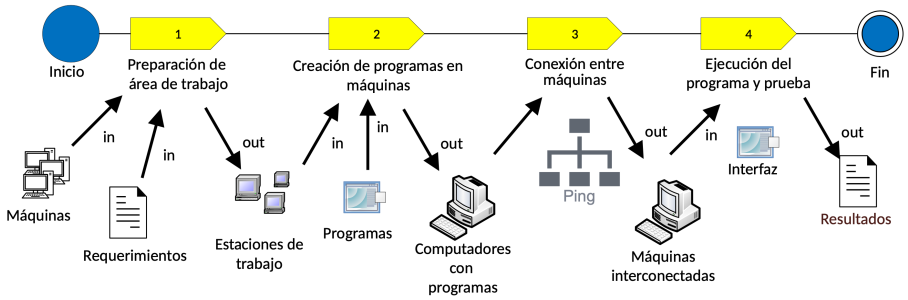

FIG. 1.
Metodología general

Así, en el encriptado caótico, utilizando osciladores de orden fraccionario, existen parámetros desconocidos para un usuario intruso. Estos parámetros contienen valores específicos de los cuales el sistema presenta un comportamiento caótico. Entre los parámetros desconocidos para el intruso está la banda de frecuencia de las señales caóticas. En Salas [6], se comparan métodos de cifrado para proteger datos empresariales como por ejemplo, Gronsfeld, transposición, César, RSA, conjuntamente con herramientas como Tor, Open Puff, OpenSSH. Los métodos de cifrado se aplicaron para las contraseñas de un sitio web, que consta de una llave predefinida en PHP, conjuntamente con hash y otros procesos propios del lenguaje. Los resultados que obtuvieron fueron positivos, ya que implementaron varios métodos de cifrado para obtener un método de encripción más fuerte y robusto. Montenegro [7] estudia la comparación de algoritmos criptográficos en la transferencia de un archivo de un servidor hacia una máquina cliente. Actualmente, hay vulnerabilidades que permiten realizar un ataque dentro y fuera de las organizaciones. Quienes realizan los ataques cibernéticos son hackers que dedican su tiempo a interceptar la información usando conocimientos informáticos con fines criminales. De este modo, se elabora un prototipo de red pública con una topología estrella y concluye que AES es el mejor algoritmo criptográfico en cuanto al tiempo de envío, número de paquetes fraccionados y número de paquetes encriptados, llegando a recomendar su uso en la protección de la información de las organizaciones. Seth et al. [8] implementan una arquitectura novedosa para brindar una mayor seguridad en torno a la computación en la nube. Esta arquitectura se compone de técnicas de fragmentación de datos y cifrado dual que prevén la distribución segura de la información en un entorno de múltiples nubes. Además, se abordan áreas como integridad, seguridad, confidencialidad y autenticación.

Chakroborti et al. [9] proponen un nuevo modelo para mejorar el rendimiento de un servicio web, modificando el principio de seguridad del mensaje SOAP y el anclaje del archivo de lenguaje descriptivo del servicio web (WSDL) para facilitar la autenticación. Los resultados del experimento demuestran mejoras significativas y el tiempo de respuesta con respecto al enfoque tradicional sin comprometer la seguridad tanto en el cifrado como en el descifrado al momento de autenticarse.

Existen varios métodos de encriptación que han dado resultados positivos al cifrar datos. Sin embargo, en este paper se propone implementar algoritmos comunes como el método de César, el estándar de cifrado avanzado 256 (AES-256) y método de cifrado de bloques simétricos Blowfish como un cifrado en capas a través de diferentes máquinas dentro de un sistema distribuido para lograr el objetivo de esta investigación.

III. METODOLOGÍA

En este estudio, se propone la utilización de varios métodos de cifrado. Para representar los métodos en esta implementación utilizó Software Process Engineering Metamodel (SPEM), que está diseñado para detallar procesos y sus componentes, por medio del modelado orientado a objetos basado en lenguaje unificado de modelado (UML) [10]. El diagrama de la presente propuesta está representado en la Figura 1. El enfoque propuesto fue dividido en cuatro tareas: i) preparación del área de trabajo, ii) creación de Programas en cada dispositivo, iii) implementación de la arquitectura experimental propuesta que contiene las conexiones entre los dispositivos y iv) la ejecución del programa con sus respectivas pruebas. La entrada se muestra a la izquierda y la salida a la derecha de cada una de las tareas a manera de artefactos.

A. PREPARACIÓN DEL ÁREA DE TRABAJO

En esta tarea, que se observa en la Figura 2, se presentan los pasos que fueron realizados para la preparación del área de trabajo.

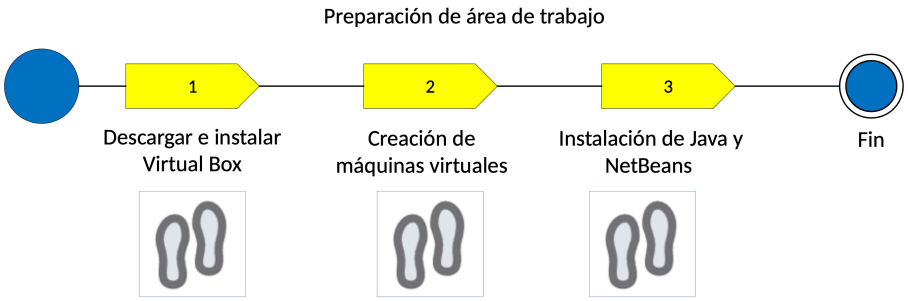

FIG. 2.
Preparación del área de trabajo

Para preparar el área de trabajo se instaló un software para realizar la virtualización de Sistemas Operativos (SO) dentro de una maquina local denominada VirtualBox [11]. Luego, se crearon las máquinas virtuales con el mínimo de características:4 GB de memoria RAM, 8 GB de almacenamiento de disco, con un procesador de un solo núcleo e interconectadas en su propia red virtual izada. Como siguiente paso, se instaló el SO de Microsoft Windows 7 en cada máquina virtual creada.

Una vez que fueron creadas las máquinas virtuales, se precedió a instalar en cada computador un entorno de desarrollo integrado (IDE) libre, llamado Netbeans, conjuntamente con el kit de desarrollo de Java (JDK) en versión 8, para terminar con la preparación del área de trabajo.

B. CREACIÓN DE PROGRAMA

En esta tarea, se realizó la creación de los programas en las diferentes máquinas que van a consumir el servicio web, como se muestra en la Figura 3.

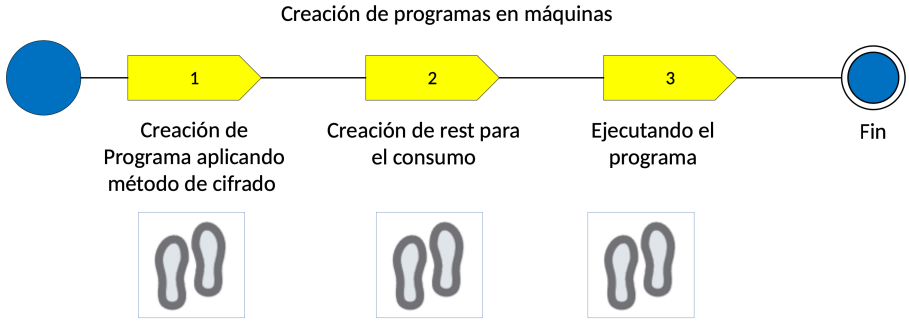

FIG. 3.
Creación del programa

La implementación de cada método de cifrado escogido se realizó en cada una de las computadoras virtuales usando el lenguaje de programación Java. En la primera máquina, se utilizó el método de César. Este método posiciona cada letra en un determinado número de espacios en el alfabeto, ya que es un tipo de cifrado por sustitución, es decir una letra del texto original es sustituido por otra letra que se encuentra en un número fijo de posiciones más posterior en el alfabeto [12].

Para el método de cifrado/descifrado de César, se tuvo que obtener inicialmente el módulo del recorrido que se va a realizar. En este caso, cada vez que se llegó a la letra "Z" se volvió a recorrer a la letra "A", y se sumó la respuesta de la longitud a la nueva posición. Después, se sacó su cadena de texto (string) y se acumuló para devolver la nueva cadena rotada. Además, se comparó si es una letra mayúscula para saber cuál es la posición desde donde se inicia a rotar, como se muestra en el Algoritmo 1.

Algoritmo 1*Cifrado/Descifrado 1*
Requiere originalString, numRotation
Salida stringConverted

1. *iniMayus = 97*
2. *iniMin = 65*
3. *lenAlphabet = 26*
4. **Para** *each character # originalString***hacer**
5. *currentChar ← character*
6. **Si** *currentChar # alphabet***Luego**
7. stringConverted ← currentChar
8. **Fin Si**
9. *currentCharASCII # currentChar*
10. *isMayus.bool ← currentChar*
11. *newPosAlphabet* ← (currentCharASCII − (**Si** *isMayus.bool* **entonces** *iniMayus* **En todo caso***iniMinus) + numRotation) mod lenAlphabet*
12. **Si** *newPosAlphabet< 0* **Luego**
13. n*ewPosAlphabet # lenAlphabet*
14. **Fin Si**
15. *newPosCharASCII* ← (**Si** *isMayus.bool entonces iniMayus* **En todo caso** *iniMinus*) + *newPosAlphabe*t

16. *stringConverted # (character # newPosCharASCII)*
17. **Fin Para**

Algoritmo 2*Cifrado/descifrado AES-256*
Requiere secretKeyAES, saltAES, secretKeyTemp, data, mode
Salida stringConverted

1. keySpecific # PBEKEYSPEC(secretKeyAES # Char,saltAES # Bytes
2. *secretKeyTemp #GENERATESECRET(keySpecific)*
3. **Proceso** *GETAESENCRYPT-DECRYPT(data, mode)*
4. *secretKey # SECRETKEYSPEC(secretKeyTemp EncodedAES)*
5. *ivParameterSpec # IVPARAMETERSPEC(byte[16])*
6. **Si** *mode is encrypt* **Luego**
7. *cipher # INIT(Encrypt Mode, secretKey, ivParameterSpec)*
8. *cipherFinal # DOFINAL(data # Bytes(UTF - 8))*
9. *stringConverted # ENCODER. ENCODETOSTRING (cipherFinal)*
10. **En todo caso Si***mode is decrypt* **Luego**
11. *cipher # INIT(Decrypt Mode, secretKey, ivParameterSpec)*
12. *decodeData # DECODER. DECODE(data)*
13. *stringConverted # DOFINAL(decodeData)*
14. **En todo caso**
15. *stringConverted # NULL*
16. **Fin Si**
17. **Retorno** *stringConverted*
18. **Fin Proceso**

En la siguiente máquina virtual, se implementó el Algoritmo 2, donde se observa los dos primeros parámetros que son la llave (secretKeyAES) y la sal o bytes aleatorios (saltAES). Estos parámetros deben ser los mismos tanto para cifrar como para descifrar. En este caso, se utilizaron los mismos parámetros para todos los cifrados, los cuales deben ser generados aleatoriamente y con al menos 25 dígitos para cada cifrado, con la finalidad de incrementar la seguridad de la información. La instancia de SecretFactory funciona inicialmente con PBKDF2, que es una función de derivación de claves, y es bastante usado para reducir la vulnerabilidad deataques por fuerza bruta [13]. La instancia de PBEKeySpec es cifrado basado en contraseña, donde se tiene que iterar un n número de veces dentro de la instancia, otorgando una clave secreta temporalmente cifrada [14]. El método getAESEncrypt – Decrypt contiene dos variables de entrada que son los datos y el mode donde se da a elegir si se desea cifrar o descifrar. Por un lado, está el cifrado basado en AES-256 que generó un SecretKeySpec en AES con la clave temporal cifrada. La variable cipher fue generada en AES por bloques CBC y con una llave de tipo PKCS5Padding. Después, se inició la variable cipher con secretKey y en Encrypt Mode. Finalmente se codificó y se devolvió en Base64 a una cadena con bytes en formato UTF-8. Por otra parte, está el descifrado que solamente se utilizó para cambiar la variable cipher por DECRY PT MODE, para posteriormente decodificar de Base64 a cadena, con lo cual obtenemos como resultado la cadena descifrada.

En la última máquina virtual, se implementó el método de cifrado Blowfish, como se muestra en el Algoritmo 3.

Algoritmo 3*Cifrado/descifrado Blowfish.*
Requiere saltBf, data, mode
Salida stringConverted

1. **Proceso** *GETBFENCRYPT-DECRYPT(data, mode)*
2. *byteData # saltBf # Bytes*

3. *secretKey # SECRETKEYSPEC(byteData, "Blowfish")*
4. *cipher # GETINSTANCE("Blowfish")*
5. **Si** *mode is encrypt***Luego**
6. *cipher # INIT(Encrypt_Mode,secretKey)*
7. *cipherFinal # DOFINAL(data # Bytes)*
8. *stringConverted # ENCODER. ENCODE(cipherFinal)*
9. **En todo caso Si***mode is decrypt***Luego**
10. *cipher # INIT(Decrypt Mode, secretKey)*
11. *decodeData # DECODER. DECODE(data)*
12. *stringConverted #DOFINAL(decodeData)*
13. **En todo caso**
14. *stringConverted # NULL*
15. **Fin Si**
16. **Retorno** *string Converted*
17. **Fin proceso**

Para cifrar la información con el algoritmo Blowfish, se requiere de dos procesos: i) procesamiento previo de claves y ii) cifrado de información. Así, como ocurrió con el Algoritmo 2, este algoritmo contiene una parte de cifrado y otra parte de descifrado en el método getBfEncrypt−Decrypt que posee como entrada los datos y su propio modo (cifrado-descifrado). En la parte del cifrado, se procedió a encriptar con Blowfish, y se generó un SecretKeySpec con la clave temporal cifrada. Luego, se cifró cipher en Blowfish y se inició con la secretKey conjuntamente con ENCRY PT MODE. Por último, se codificó y se devolvió en Base64 a una cadena con bytes en formato UTF-8. En cambio, para descifrar se cambió en la variable cipher por DECRY PT MODE y se decodificó de Base64 a cadena, con lo cual obtenemos como resultado la cadena sin cifrado.

C. IMPLEMENTACIÓN DE LA ARQUITECTURA EXPERIMENTAL PROPUESTA

Después de haber creado los algoritmos en cada computador virtual conjuntamente con el servicio SOAP, se realizó la desactivación del corta fuegos de las tres máquinas para que no presente alguna restricción en su interconexión. Luego, se obtuvo las direcciones IP de las máquinas virtuales: 102.38.236.100, 102.38.236.101, 102.38.236.102 dentro de la red 102.38.128.0/17 para comprobar la conectividad entre estas. En la Figura 4 se presenta el escenario del experimento.

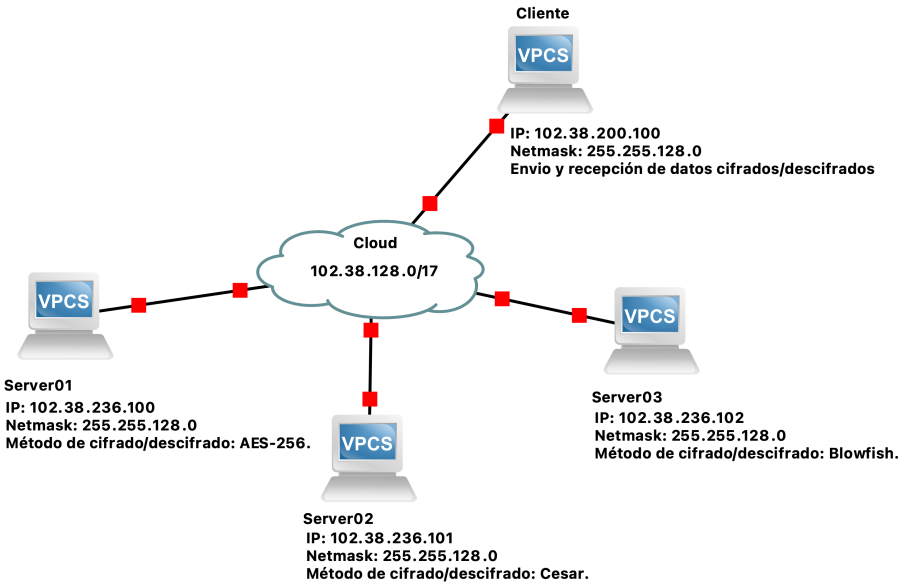

FIG. 4.
Escenario experimental y conexión del sistema distribuido

Y, por último, en esta fase de la metodología correspondiente a la ejecución del programa, se realizó las debidas pruebas concernientes a la propuesta que hemos realizado, así como se muestra en la Figura 5.

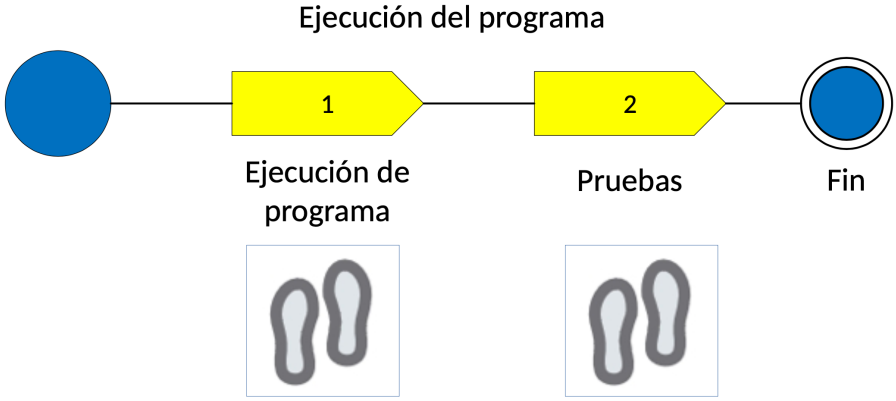

FIG. 5.
Ejecución y pruebas del Programa

IV. RESULTADOS

Para demostrar el funcionamiento de nuestra propuesta, se utilizaron algunos ejemplos de texto, desde un solo carácter hasta palabras largas en español. El programa principal recibe como entrada una cadena de texto que se desea cifrar, seguido de dos guiones bajos para diferenciar entre el texto y el orden de las máquinas que realizan el cifrado, como por ejemplo <<Esternocleidooccipitomastoideos M2M1M3>>.

Consecuentemente, el sistema procede a encriptar la cadena de texto *Esternocleidooccipitomastoideos* en el orden del cifrado que es la máquina 2 con el algoritmo AES-256, la máquina 1 con el algoritmo César y la máquina 3 con el algoritmo Blowfish. Así, una vez ingresada la cadena a encriptar, se invoca el servicio SOAP para enviar la cadena a las máquinas correspondientes que contienen los algoritmos de cifrado. Esto depende del orden en que se invocan los algoritmos, por lo que, siguiendo el ejemplo anterior, primero se ejecutó el algoritmo AES-256, luego, la cadena de texto cifrada es enviada hacia el computador que tiene el

algoritmo César y por último la cadena de texto con los dos algoritmos previos integrados es enviada hacia el último algoritmo Blowfish. Para volver al texto original, se procede en forma inversa, es decir, desde la última máquina que contiene la cadena de texto con los tres algoritmos de cifrado incluidos, ejecutando los algoritmos en orden inverso hasta obtener como resultado la cadena de texto original.

Con respecto al desempeño de la arquitectura propuesta, desde el punto de la complejidad algorítmica, se observó que es bastante efectiva y robusta durante el cifrado/descifrado de cadenas de texto. El tiempo de procesador fue mínimo al utilizar los tres algoritmos en cada computador, teniendo lecturas entre 95 a 97 milisegundos en tiempo de cifrado, y entre 95 a 100 milisegundos en tiempo de descifrado. En cuanto al uso de memoria RAM, se obtuvieron valores entre 3,08 a 3,12 Megabytes en tiempo de cifrado, y entre 3,05 a 3,09 Megabytes en tiempo de descifrado. Los resultados obtenidos dependen del orden de ejecución de los algoritmos establecidos en los computadores.

V. CONCLUSIONES

En este trabajo, se exploran diferentes algoritmos descifrado utilizando el lenguaje de programación Java. Además, se implementa el servicio SOAP conjuntamente con los servicios web para la conexión de diferentes máquinas virtuales interconectadas. Cada computador virtual tiene la funcionalidad de ejecutar el cifrado/descifrado de una cadena de texto y recibir su respectiva respuesta. Los algoritmos descifrados utilizados como AES-256, César y Blowfish trabajan de manera conjunta y llegan ser un gran aporte para sistemas modernos. Esto se debe a la forma en como encriptan la información, mejorando los resultados al pasar el mensaje por diferentes máquinas de la red.

Consecuentemente, para un atacante cibernético, la dificultad de interceptar o descifrar un mensaje se incrementa sustancialmente debido a que los algoritmos de cifrado están distribuidos en diferentes computadores.

Con respecto al consumo de recursos, se observó que tanto el uso de procesador como de memoria es mínimo, incluso utilizando computadores con prestaciones bajas como los que se emplearon en los experimentos.

En los trabajos futuros se emplearán más algoritmos descifrado extendiendo a arquitecturas de red más complejas, que nos permitirá evaluar los tiempos de cifrado/descifrado, y la mejora de seguridad de información sensible en una empresa. Otro trabajo futuro es emplear arquitecturas de sistemas distribuidos con algoritmos de cifrado más complejos en microordenadores, y dispositivos de Internet de las Cosas(IoT).

Agradecimientos

Agradecemos al Vicerrectorado de Investigaciones de la Universidad del Azuay por el apoyo financiero y académico, así como a todo el personal de la Escuela de Ingeniería de Ciencias de la Computación, y del Laboratorio de Investigación y Desarrollo en Informática - LIDI.

REFERENCIAS

P. Dixit, A.K.Gupta, M.C.Trivedi and V. K. Yadav, “Traditional and Hybrid Encryption Techniques: A Survey,” in Networking communication and data knowledge engineering. Springer, 2018, pp. 239–248.

[2] A. Tanenbaum and M. van Steen, Distributed Systems. CreateSpace Independent Publishing Platform, 2017.

[3] P. Matta, M. Arora, and D. Sharma, “A Comparative Survey on Data Encryption Techniques: Big Data Perspective,” Materials Today: Proceedings, vol. 46, pp. 11 035–11 039, 2021.

[4] S. Mitra and D. Das, "A Critical Study on the Applications of Run Length Encoding Techniques in Combined Encoding Schemes." International Journal of Advanced Research in Computer Science, vol. 8, no. 5, 2017.

[5] O. García Sepúlveda, "Encriptado de Datos con Osciladores Caóticos de Orden Fraccionario," Ph.D. dissertation, Universidad Aut´onoma de Nuevo León, 2015.

[6] J. R. Tirado Salas. (2017) Encriptado de Datos para Proteger Información de las Empresas. [Accedido 06 diciembre 2022]. [Online]. Available: https://bit. ly/3utyTls

[7] D. Montenegro Torres. (2020) Algoritmos de Encriptación de Archivos para la Transferencia en Mensajería Instantánea. [Accedido 06 diciembre 2022]. [Online]. Available: https://bit.ly/3FwW1WD

[8] B. Seth, S. Dalal, V. Jaglan, D.-N. Le, S. Mohan, and G. Srivastava, "Integrating Encryption Techniques for Secure Data Storage in the Cloud," Transactions on Emerging Telecommunications Technologies, vol. 33, no. 4, p. e4108, 2022.

[9] D. Chakroborti and S. S. Nath, "Web Service Performance Enhancement for Portable Devices Modifying SOAP Security Principle," in 2017 20th International Conference of Computer and Information Technology (ICCIT). IEEE, 2017, pp. 1–7.

[10] R. Bendraou, B. Combemale, X. Cr´egut, and M.-P. Gervais, "Definition of an Executable SPEM 2.0," in 14th Asia-Pacific Software Engineering Conference (APSEC'07). IEEE, 2007, pp. 390–397.

[11] K. Srinivasa and A. Singh, "Hands-On Guide to Virtual Box," in Design and Use of Virtualization Technology in Cloud Computing. IGI Global, 2018, pp. 194–207.

[12] J. W. Ríos Taipe. (2017) Sistema de Encriptación para Optimizar el Proceso de Desarrollo de Software de una Empresa de Lima. [Accedido 06 diciembre 2022]. [Online]. Available: https://bit.ly/3h6noO2

[13] A. Visconti and F. Gorla, "Exploiting an HMAC-SHA-1 Optimization to Speed up PBKDF2," IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 4, pp. 775–781, 2018.

[14] M. Hazhirpasand, M. Ghafari, and O. Nierstrasz, "Cryptoexplorer: An Interactive Web Platform supporting Secure Use of Cryptography APIs," in 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2020, pp. 632–636.