Artículo **Revista de Tecnología Informática**
Septiembre 2017 Vol.1 No.2 41-50

Determinación de parámetros que impiden una implementación eficiente de algoritmos criptográficos en ambiente multiplataforma

GONZÁLEZ-MARRÓN, David†, GAMERO-PLAFOX, Benito, LÓPEZ-MELO, Eduardo y AGUILAR-GÓMEZ, José

Instituto Tecnológico de Pachuca, Felipe Angeles Km. 84.5, Venta Prieta, 42083 Pachuca de Soto, Hgo., México

Recibido Julio 4, 2017; Aceptado Septiembre 7, 2017

Resumen

En este artículo se analizan los problemas que se presentan en el desarrollo de un algoritmo criptográfico simétrico por bloques con un tamaño de llave máxima de 16 caracteres ASCII, que realiza el cifrado de textos de longitud variable en diferentes lenguajes y plataformas, los lenguajes seleccionados para el desarrollo son C++, Java y C#, el algoritmo es probado en los sistemas operativos de Windows y Linux, se analizan los problemas de compatibilidad que se generan al realizar el proceso de sustitución de símbolos, así como los convenientes e inconvenientes que se presentan entre lenguajes. Se analizan los aspectos relativos a especificaciones funcionales requeridas para trabajar en un ambiente de multiplataforma. El algoritmo parte de una especificación general que los desarrolladores deben interpretar para su implementación en cada lenguaje, lo que conlleva cambios significativos en su implementación. Se detalla el desempeño obtenido en cada implementación realizada en los lenguajes de programación utilizados, así como las pruebas utilizadas para verificar el comportamiento del algoritmo bajo diferentes situaciones.

Ingeniería de Software, criptografía, ambiente multiplataforma

Abstract

Throughout this article are analyzed the problems presented in the development of a symmetric cryptographic block algorithm with 16 ASCII characters maximum key, the algorithm realizes the the encryption of variable length texts in different languages and platforms, the languages selected for development are C ++, Java and C #. The algorithm is tested in Windows and Linux operating systems, there are analyzed the compatibility problems generated when performing the process of symbols substitution as well as the conveniences and inconveniences presented between languages. Aspects related to functional specifications required to work in a multiplatform environment are analyzed. The algorithm starts from a general specification that developers must interpret for their implementation in each language, which entails significant changes in its implementation. It is detailed the performance obtained in each implementation performed in the programming languages used, as well as, the tests used to verify the behavior of the algorithm under different situations and the results of the implementation.

Software engineering, criptography, multiplatform environment

Citación: GONZÁLEZ-MARRÓN, David, GAMERO-PLAFOX, Benito, LÓPEZ-MELO, Eduardo y AGUILAR-GÓMEZ, José. Determinación de parámetros que impiden una implementación eficiente de algoritmos criptográficos en ambiente multiplataforma. Revista de Tecnología Informática 2017, 1-2: 41-50

† Investigador contribuyendo como primer autor.

Introducción

Actualmente la cantidad de información que se está generando, está alcanzado niveles muy altos, como puede ser visto en la Figura 1, además de que ésta información es almacenada y transmitida en diversos dispositivos que se encuentran diseminados en el internet.

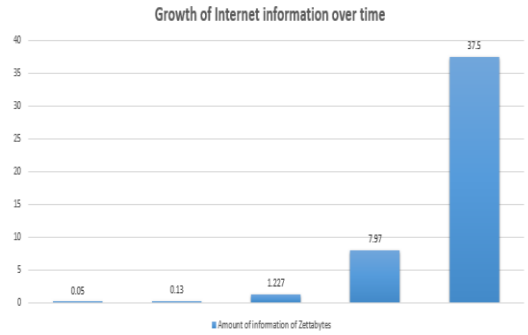

Figura 1 Crecimiento de información en Zbytes

Es por eso que se requiere de uso de mecanismos de protección que preserven las propiedades de Confidencialidad, Integridad y Accesibilidad de la información (CIA), el uso de las técnicas de encriptación nos ayuda a conservar dos de estas propiedades (Confidencialidad e Integridad) de allí la enorme importancia de la encriptación, los desarrolladores de implementaciones de algoritmos de criptografía tienen un reto significativo para hacer algoritmos robustos y rápidos en diversas plataformas. El objetivo del presente trabajo es la identificación de los principales parámetros que pueden llevar a una implementación deficiente de un algoritmo criptográfico.

El análisis de dichos parámetros se ha hecho desde el punto de vista técnico, evaluando el rendimiento de la implementación en diversos lenguajes de programación y bajo diversos sistemas operativos; y desde el punto de vista humano, identificando los retos cuyo origen son deficiencias de ingeniería de software.

"La ingeniería de software comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de estos después de que se utiliza" [1] partiendo de esa definición, resulta evidente que la implementación de algoritmos criptográficos en un marco de calidad se verá afectado por aspectos netamente humanos.

Se seleccionaron tres de los lenguajes de programación más conocidos, como son (Java, C# y C++) a fin de elegir el lenguaje más adecuado para el problema de desarrollo de algoritmos de encriptación.

Los lenguajes de programación basados en el lenguaje de programación ANSI C permiten que el software desarrollado sea compatible con diversos Sistemas Operativos. Pero en cuestiones de rendimiento, los lenguajes de programación son totalmente distintos porque algunos se ejecutan bajo el uso de máquinas virtuales, algunos otros lenguajes son compilados o interpretados, incluso las librerías con las que el lenguaje funciona afectan su rendimiento.

La encriptación puede conceptualizarse como una forma de proteger la información a aquellos que no deben tener acceso. El arte de encriptar vio su nacimiento al tiempo que lo hacia la escritura [2], sin embargo, no ha sido hasta épocas más recientes en que la encriptación informática se ha convertido en la forma predilecta de asegurar las transacciones electrónicas.

A lo largo de los años se han desarrollado múltiples algoritmos de cifrado con características específicas que los diferencian. Una de las características más notables en un algoritmo de cifrado es la clave, de hecho, partiendo de la clave es cómo se han podido clasificar los algoritmos en simétricos, aquellos que emplean la misma clave para cifrar y descifrar, y algoritmos asimétricos, aquellos que emplean una clave pública y una privada.

ISSN-2531-2197

El algoritmo seleccionado para realizar esta investigación es de tipo simétrico por bloques y se detalla posteriormente.

Trabajos relacionados

En 2014 Philipp Holtkamp et al [3], realizaron un estudio partiendo del hecho de que el factor humano es el origen de la mayoría de retos en los proyectos de software. Resultado de dicho trabajo se concluyó que todas las competencias de internacionalización son importantes para el desarrollo de software, sin embargo, se ha encontrado que tienen especial importancia en las fases con elementos colaborativos y creativos. En ese trabajo se evaluó el efecto que tienen las deficiencias en las competencias analizadas en la implementación colaborativa de un algoritmo criptográfico.

Encriptación simétrica por bloque

Los algoritmos de cifrado simétrico por bloques se basan en dividir el mensaje en bloques de tamaño fijo y a partir de aquí poder realizar operaciones matemáticas u operaciones lógicas (principalmente sumas utilizando la función XOR) utilizando una clave secreta, con la cual se realizan dichas operaciones. Se recomienda que el tamaño de bloque sea considerablemente grande para poder evitar algún tipo de ataque. Se tiene que verificar que, durante el proceso de cifrado, el tamaño de bloque no cambie, para que este pueda ser reversible y no se tenga problemas a la hora de descifrarlo.

Codificación de caracteres en los lenguajes de programación

Las computadoras manejan internamente toda la información como dígitos binarios por lo que la representación de caracteres se hace mediante números [4]. La codificación de caracteres se basa en establecer una relación uno a uno entre un carácter y un valor numérico.

La evolución de la codificación en los lenguajes de programación se vio impulsada por el surgimiento de los GSD (Global Software Developments) Desarrollos de Software Global, por lo que los lenguajes de programación han empleado diversas codificaciones a lo largo de los años.

El lenguaje Java emplea el estándar Unicode y define el tipo primitivo char como un tipo de dato de 16 bits, con caracteres entre el rango hexadecimal de 0x0000 a 0XFFFF [5].

Visual C# emplea Unicode para almacenar los caracteres y cadenas, sin embargo también puede manejar caracteres en codificación ANSI, que usan un byte para representar un carácter y se limita a 256 caracteres, o ASCII [6].

C++ emplea el código ASCII que proporciona códigos para representar el idioma inglés [7]. Adicionalmente, C++ permite emplear Unicode para aplicaciones internacionales mediante el tipo wchar_t.

Realización de un algoritmo de encriptación por bloque

La especificación original del algoritmo de encriptación requerido se muestra a continuación.

Desarrollar un algoritmo que haga una transposición de valores, utilizando el último elemento como el primero, en base al tamaño seleccionado, una vez realizado esto, hacer una operación de suma con una llave de hasta 16 caracteres (ASCII), la llave será particionada en base al tamaño del bloque seleccionado y en caso de ser una llave menor que el tamaño del bloque deberá ser completada por blancos, los valores válidos a encriptar estarán dados por la tabla ASCII, con un máximo de 255 valores.

ISSN-2531-2197

El algoritmo deberá ser válido para cualquier tamaño de archivo o string. Deberá contar con una interface para hacer el cifrado y descifrado del mensaje y deberán ser programas que puedan ser independientes uno de otro.

Dicha especificación al ser tan genérica dio origen a diversas interpretaciones de parte de los desarrolladores, a continuación, se detallan dos diferentes diagramas de flujo a fin de ver las diferencias que se presentan.

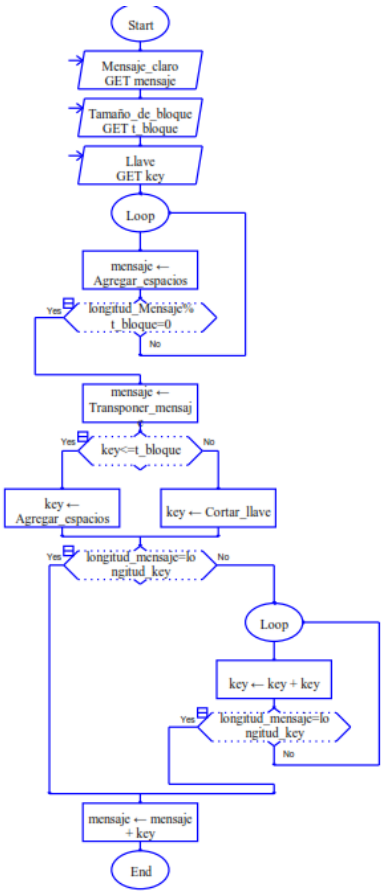

Figura 2 Diseño de algoritmo por programador 1

La primera interpretación que se ilustra en la Figura 2. hace un ajuste al mensaje y a la llave para que su longitud sea múltiplo del tamaño de bloque. Posteriormente se repite la llave hasta que la longitud es igual a la longitud del mensaje, se suma la llave con el mensaje transpuesto y se obtiene el mensaje cifrado.

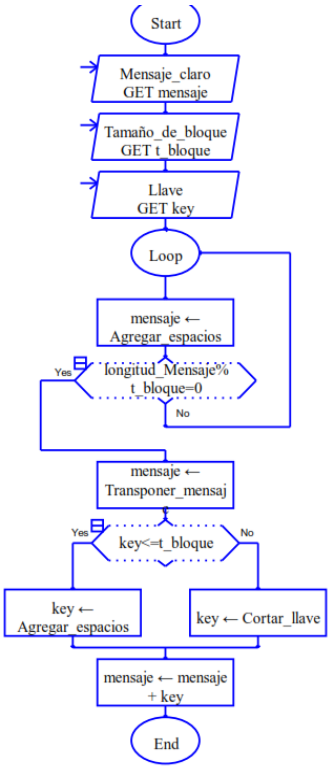

Figura 3 Diseño de algoritmo por programador 2

La segunda interpretación se ilustra en el la Figura 3., la cual consiste en ajustar el mensaje para obtener una longitud de mensaje que sea múltiplo del tamaño de bloque, transponer el mensaje, ajustar la clave según su tamaño respecto al tamaño de bloque y sumar la llave con el mensaje.

Derivado de los problemas de interpretación del algoritmo fue requerido replantear el mismo, resultando el proceso de detalle con el propósito de que las implementaciones fueran equivalentes.

1. Se ajusta el mensaje agregando espacios hasta que su longitud sea un múltiplo del tamaño del bloque.

M = MENSAJE b = 3 K = CLAVE

M	E	N	S	A	J	E		

2. Se divide el mensaje en segmentos del tamaño del bloque.

M	E	N	S	A	J	E		

3. Se transpone cada uno de los segmentos.

N	E	M	J	A	S			E

4. Se unen los segmentos para originar el mensaje.

N	E	M	J	A	S			E

5. Se repite la contraseña hasta que su longitud sea igual a la longitud del mensaje.

C	L	A	V	E	C	L	A	V

6. Se suma el valor decimal del código ASCII vinculado a los caracteres ubicados en la posición n del mensaje y de la clave.

	N_{78}	E_{69}	M_{77}	J_{74}	A_{65}	S_{83}	32	32	E_{69}
+	C_{67}	L_{76}	A_{65}	V_{86}	E_{69}	C_{67}	L_{76}	A_{65}	V_{86}
	145	145	142	160	134	150	108	097	155

C = 145145142160134150108097155

Consideraciones para verificar el correcto funcionamiento del algoritmo en diversos lenguajes

El set de pruebas se basó en pruebas de caja negra, eligiendo los casos de prueba en función de las especificaciones funcionales del software [8], cuyo único interés es determinar si las salidas son correctas en función de las entradas.

Las pruebas realizadas verifican la compatibilidad, robustez y estrés al que se pueden someter las diferentes implementaciones en lenguajes de programación diferentes del algoritmo de cifrado.

En el aspecto de compatibilidad se realizaron pruebas que validan el aspecto funcional del mensaje, que el mensaje cifrado por cualquier implementación pueda ser descifrado por las implementaciones restantes y viceversa.

En el aspecto de robustez se verificó que las diversas implementaciones se comportaran correctamente a valores de entrada incorrectos, evaluando la forma en que el software debería responder

Las pruebas de estrés consistieron en someter los programas implementados en diferentes lenguajes a valores extremos de tamaño de bloque y longitud de mensaje.

Performance de algoritmo en diversos lenguajes

Los aspectos considerados para determinar el desempeño del algoritmo en los diversos lenguajes de programación fueron el tiempo que le toma al software cifrar o descifrar el mensaje, el porcentaje de uso de CPU y la cantidad de memoria RAM.

Se diseñaron un total de 30 pruebas, para los programas desarrollados en los diversos lenguajes que consideraron validaciones de entrada en cada una de las partes componentes, solicitudes de encriptación con incongruencia en parámetros solicitados y pruebas de estrés, en este trabajo se describen las principales pruebas que afectan el performance de los programas. Las pruebas se realizaron en el mismo hardware y sistema operativo, midiendo el consumo de recursos con el software Process Explorer de Mark Russinovich [9].

La metodología de medición del tiempo de ejecución consistió en imprimir la hora en el momento en que se presiona el botón de cifrar/descifrar e imprimir la hora antes de enviar el mensaje cifrado o descifrado a pantalla o archivo.

ISSN-2531-2197

Posteriormente se restó el tiempo inicial al tiempo final, obteniendo así el tiempo de procesamiento.

Resultados Obtenidos

Se muestran mediante gráficas los resultados obtenidos., puntualizando el desempeño en cada lenguaje, explicando el porqué de las diferencias. La prueba número uno se diseñó con el objetivo de someter a las implementaciones del algoritmo de cifrado a lo que se consideró un caso de uso representativo del lenguaje común, sometiendo al software a una prueba de estrés, cifrando un texto extenso de 450 mil caracteres.

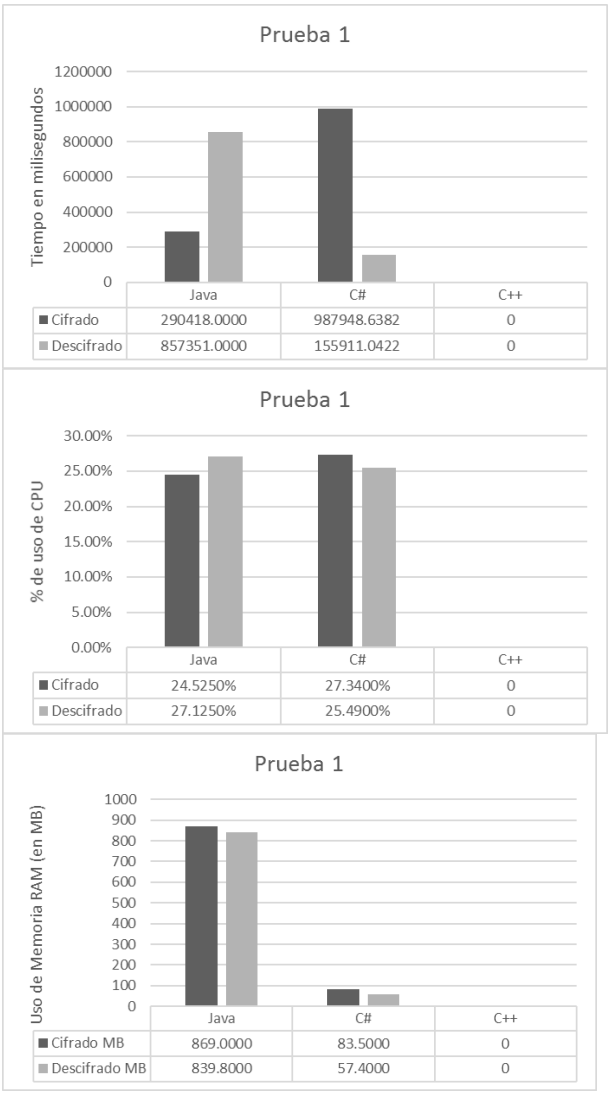

Gráfico 1

Se detectó que el cifrado y descifrado programado en C++ falló debido al deficiente manejo de los caracteres especiales, tales como el salto de línea y la letra "ñ". Verificándose además de que el algoritmo en C++ presentó problemas para manejar el código ASCII extendido.

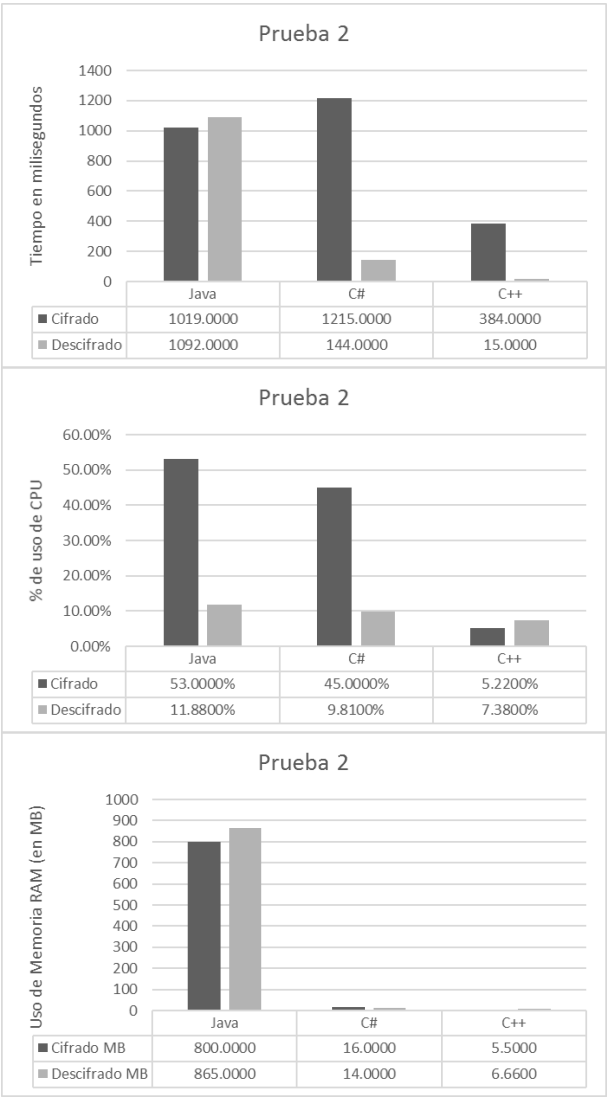

Gráfico 2

La prueba 2, basada en un análisis de rendimiento se basó en encriptar un mensaje pequeño en un bloque de mayor tamaño al del mensaje, se cifró un texto de 45 caracteres y un tamaño de bloque de 10 mil unidades, en esta prueba todos los lenguajes cumplieron la tarea, encontrándose que el programa desarrollado en Java tuvo un consumo mayor en memoria y tiempo de uso del CPU, en comparación con los programas desarrollados en C# y C++.

En la prueba 3 se repite el caso de un mensaje pequeño en un bloque excesivamente mayor, se utilizó un mensaje de 45 caracteres, pero con un tamaño de bloque de 100 mil unidades, con ello se comprobó que el programa en C++ pudo cifrar el mensaje sin problemas, pero surgió el problema en la consola, ya que la consola limita la cantidad de caracteres que pueden ser escritos en esta, por lo tanto, no se pudo llevar a cabo el descifrado en el software desarrollado en C++.

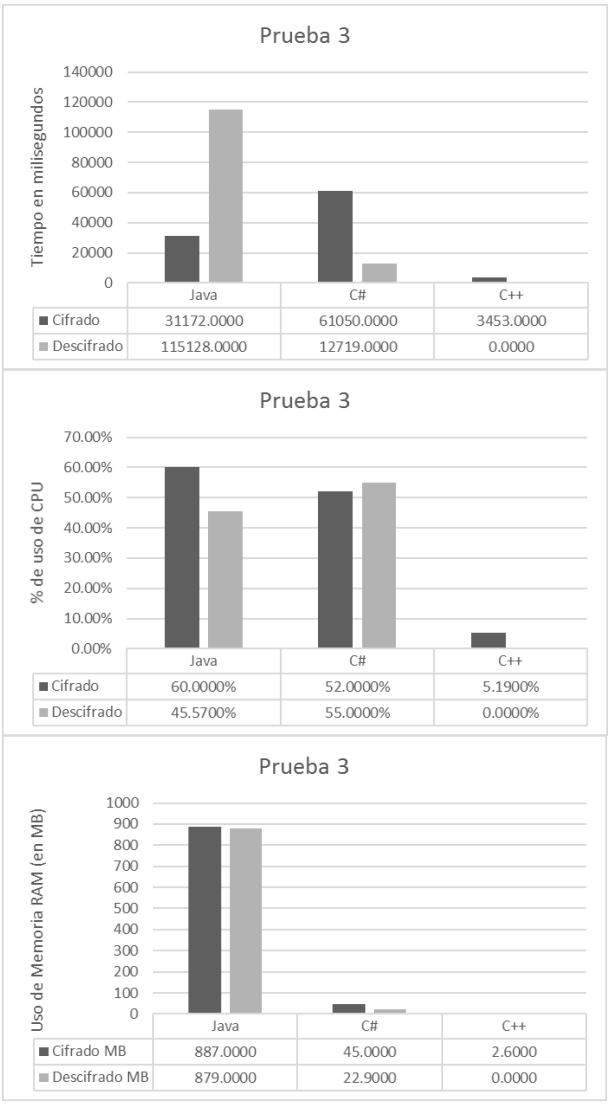

Gráfico 3

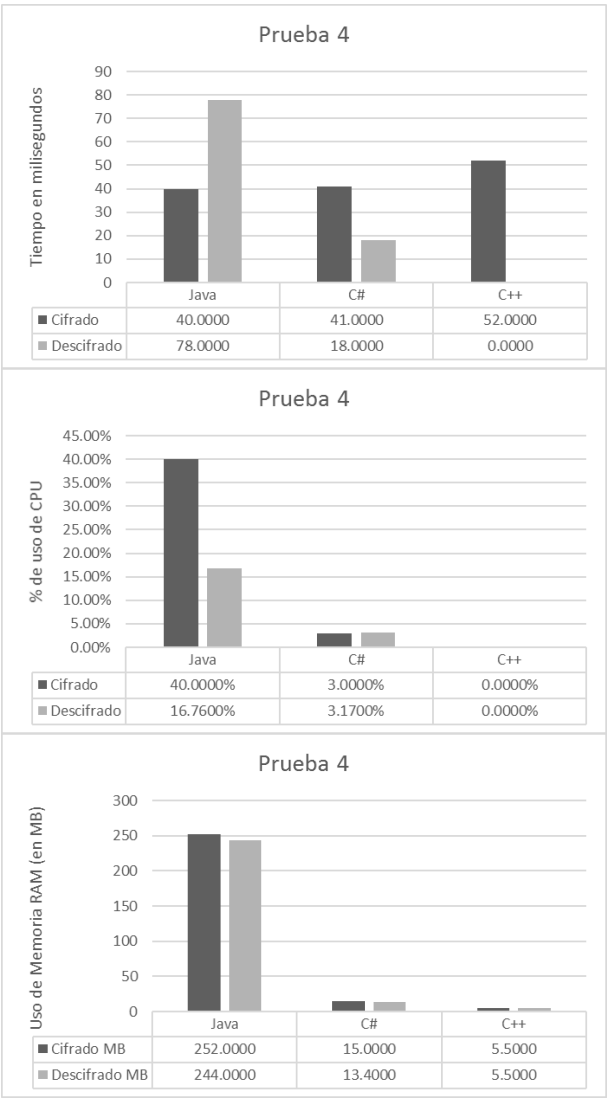

Gráfico 4

ISSN-2531-2197

En la prueba 4, se cifró el mismo mensaje, pero con un tamaño de bloque de mil unidades, y se demostró que el programa en C++ es muy apto en cuestiones de rendimiento, dado que empleó un bajo consumo de CPU y muy poco uso de la Memoria RAM. El diseño de las pruebas número dos, tres y cuatro se hizo de forma que se esperaba que el tiempo de ejecución, consumo de CPU y memoria fuera diez veces mayor en la prueba dos y cien veces mayor en la prueba tres respecto a la prueba cuatro. En la práctica no ha sido posible observar una relación entre los tiempos y consumo de las diversas pruebas por lo que se asume que existen factores cuyo nivel relevancia es más significativa para el desempeño.

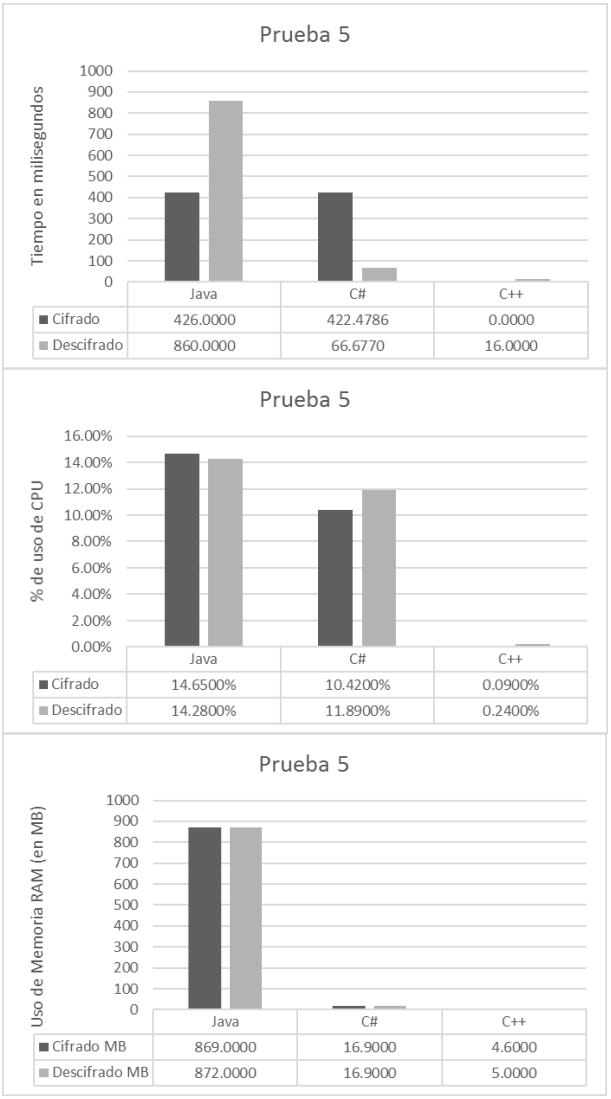

Gráfico 5

La prueba número cinco consistió en cifrar un mensaje de mil caracteres con un tamaño de bloque de 10 mil unidades. Se corroboró el excelente manejo de recursos de la implementación de C++. Además, se constató que el consumo de memoria para los procesos de cifrado y descifrado es muy similar dentro del mismo lenguaje.

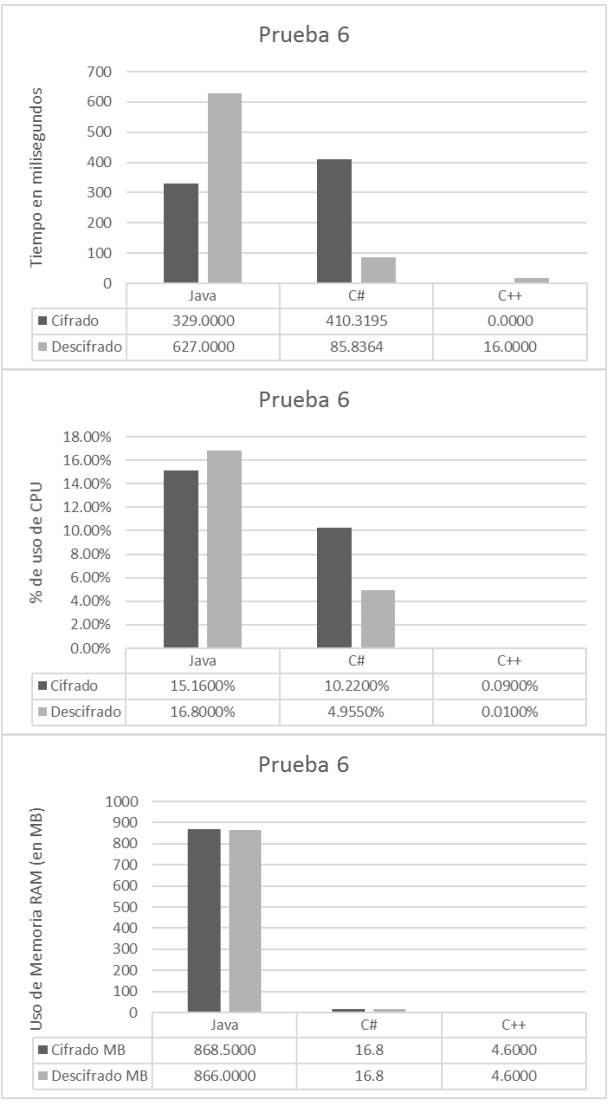

Gráfico 6

La prueba seis consistió en invertir los valores de longitud del mensaje y tamaño de bloque de la prueba cinco, se empleó la misma contraseña y se observó un rendimiento muy similar entre ambas pruebas. A continuación, se contrastan los resultados de la prueba número cinco y seis para conocer las similitudes y diferencias de ambas pruebas.

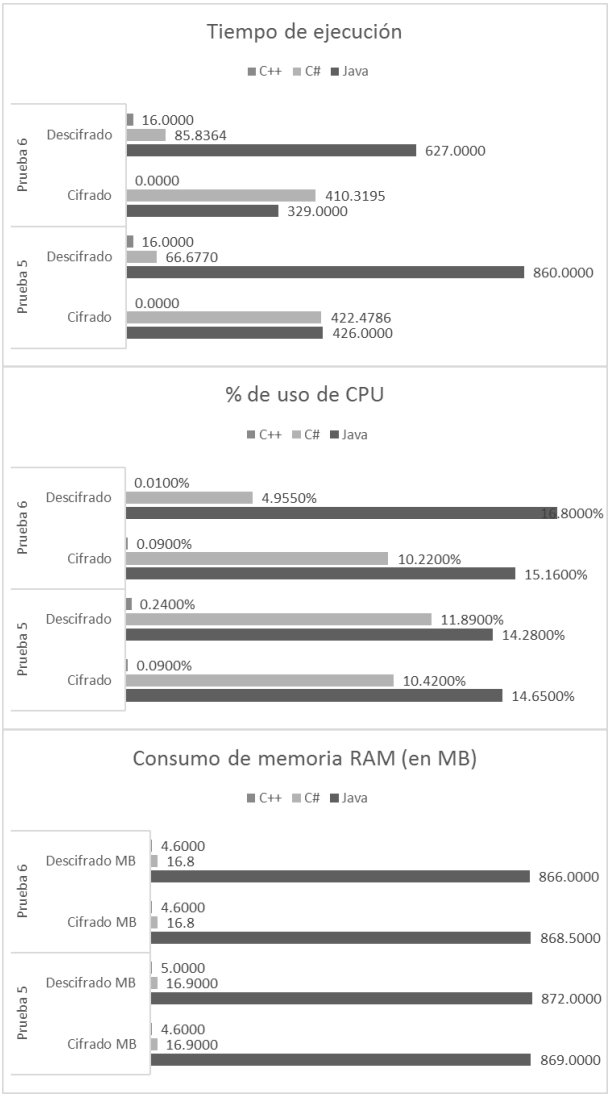

Gráfico 6

Al comparar los resultados de tiempo de ejecución y uso de CPU no se observa alguna similitud aparente por lo que se corrobora lo observado en el set de pruebas dos, tres y cuatro. Por otra parte, el consumo de memoria RAM es muy similar en el caso de los tres lenguajes de programación por lo que se asume que mientras la cantidad de información a cifrar o descifrar sea similar y no existan factores que puedan perjudicar el rendimiento, se obtendrá un consumo de memoria similar.

Conclusiones

Durante el desarrollo del software de cifrado, diferentes experiencias fueron obtenidas requeridas para su realización.

Se requiere una especificación a detalle del algoritmo, como pudo ser visto en el proceso a detalle de desarrollo del algoritmo especificado, principalmente si es un software que será realizado por diferentes programadores en diferentes lenguajes y plataformas, porque esto puede generar una implementación deficiente

Existen detalles técnicos de cada lenguaje que deben ser considerados por los desarrolladores de código, en este caso se debieron realizar juntas entre los diferentes programadores a fin de solucionar problemas existentes en cada desarrollo, como se recomienda en [10].

El funcionamiento de los algoritmos debió ser comprobado entre los diferentes programas desarrollados usando el concepto de sistemas débilmente acoplados, utilizando para esto archivos de paso donde se almacenaban los textos planos y cifrados para verificar su correcto funcionamiento

La transformación de datos al ser generada por un proceso aritmético, causó problemas debiéndose restringir los valores obtenidos a un rango definido que no afectara la reconversión de datos
Pudo ser constatada una mejor eficiencia en el lenguaje C++ con respecto a los otros lenguajes, lo que era de esperarse, sin embargo, las interfaces de los otros lenguajes son claramente superiores, facilitando su desarrollo

ISSN-2531-2197

En general el proceso de cifrado y descifrado es muy similar en condiciones normales debido a que es un procedimiento matemático inverso, la selección inapropiada del lenguaje de programación es un factor importante en el desarrollo de software, pues la implementación de un algoritmo criptográfico requiere de un lenguaje de programación exacto, con un consumo que no afecte el rendimiento del equipo y tenga un amplio soporte de los caracteres necesarios para comunicar mensajes en diferentes idiomas.

Referencias

Sommerville, I., & Alfonso Galipienso, M. (2005). Ingeniería del software. Madrid: Pearson Educación.

Orozco, G., & Nuñez, J. (2017). Introducción a la Criptografía. Recuperado de http://patux.net/downloads/crypto/crypto.pdf

Holtkamp, P., Jokinen, J., & Pawlowski, J. (2015). Soft competency requirements in requirements engineering, software design, implementation, and testing. Journal Of Systems And Software, 101, 136-146. http://dx.doi.org/10.1016/j.jss.2014.12.010

Garrido A (2006). Fundamentos de programación en C++. Madrid: Delta Publicaciones.

Unicode (The Java™ Tutorials > Internationalization > Working with Text). (2017). Docs.oracle.com. Recuperado 10 Agosto 2017, Sitio web: https://docs.oracle.com/javase/tutorial/i18n/text/unicode.html

Ceballos Sierra F. (2007). Microsoft C#. Madrid: Ra-Ma.

Bronson, G., Borse, G., & Velázquez Arellano, J. (2007). C++ para ingeniería y ciencias. México: Thomson.

Beizer, B. (1990). Software testing techniques. London: International Thomson computer Press.

Process Explorer. (2017). Docs.microsoft.com. Recuperado 10 de Agosto de 2017, from https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer

Tuya, J., Ramos Román, I., & Dolado Cosín, J. (2007). Técnicas cuantitativas para la gestión en la ingeniería del software. Oleiros, La Coruña: Netbiblo.