Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en Computación

Proyecto 1

Integrantes:

Caballero Montaño Monserrat

Carvente Velasco Carlos Alberto

De La Cruz Flores Hugo Alberto

Miramontes Sarabia Luis Enrique

Criptografía

Profesora: Rocío Aldeco Pérez

Semestre 2020-1

Fecha de entrega

07/10/19

Introducción

La siguiente investigación hace una comparación en el tiempo de ejecución, utilidad y eficiencia entre distintos algoritmos criptográficos de diferentes tipos y realizando diversas operaciones. En primera instancia, se clasificó en tres tipos de algoritmos: algoritmos de clave privada o simétricos, funciones hash o digest y algoritmos de clave asimétrica. Cabe aclarar que se hará una comparación entre tres algoritmos de cada tipo, ya que cada uno cuenta con sus propias características y usos.

Sabemos que cada uno de los tipos de algoritmos previamente mencionados se utilizan para diferentes fines, por lo que aclararemos las comparaciones que se harán. Primero, de los algoritmos simétricos utilizaremos tres por flujo (RC4, AES-CTR, AES-OFB) para tener un marco de comparación entre algoritmos de flujo, dos por bloques (DES y AES), donde serán comparados también contra RSA-OAEP, algoritmo de cifrado asimétrico; estos serán comparados en las operaciones cifrado y descifrado, y se mencionara la eficiencia de cada uno y los casos en los que deberían ser usados. Los siguientes algoritmos serán los tipo Hash los cuales serán MD5, SHA-1 y SHA-2, que al igual que los anteriores se comparará en sus capacidades de ejecución y su eficiencia para proveer seguridad. A continuación, se comparará la efectividad de los algoritmos asimétricos realizando las funciones de firma digital y verificación; para estos algoritmos empleamos RSA-PSS y DSA. Finalmente contrastaremos cada uno de los tipos de algoritmos con los otros para resaltar sus utilidades.

Creemos que el tipo de algoritmo más rápido es el de llave privada por bloque, ya que cifra información en mayor cantidad y no necesita de dos llaves como los algoritmos asimétricos. Por otro lado, tenemos la conjetura de que en los algoritmos tipo hash MD5 es el más rápido, ya que SHA-1 y SHA-2 utilizan un mayor número de operaciones para hacer más seguro el proceso. Por último, pensamos que en los algoritmos asimétricos DSA es la opción más rápida, debido a que sólo se usa para firmas digitales y verificación, en comparación a RSA que se puede utilizar también para cifrado y necesita más recursos de procesamiento para su implementación.

El objetivo principal de este escrito es hacer una comparación entre diferentes algoritmos criptográficos para poder hacer una diferenciación entre cada tipo, poder estudiar sus ventajas y desventajas, y así distinguir el uso apropiado para cada uno dependiendo de sus características y cómo deseemos emplear cada uno.

Para realizar esta investigación utilizamos un equipo MacBook Pro con un procesador Intel Core i7 de cuatro núcleos a 2.2 GHz (Turbo Boost de hasta 3.4 GHz) con 6 MB de caché de nivel 3 compartida. El equipo cuenta con un sistema operativo Mac OS Mojave 10.14.5 y los algoritmos se implementaron en Python utilizando la versión 3.7 con los módulos “Cryptography” y “DES”.

Como parte de la formación de un ingeniero en computación es esencial la aplicación de los conceptos teóricos y el análisis de estos para su futuro empleo en el campo laboral. De igual forma, esta investigación puede servir como base de futuras investigaciones o como marco teórico para el estudio sobre el uso y la eficacia de los algoritmos criptográficos que serán aquí presentados, por lo que está dirigido a un público con conocimientos generales sobre la Criptografía.

Metodología

Para la realización de esta investigación inicialmente se recopiló información de los apuntes dados en la clase de Criptografía por la profesora Rocío Aldeco Pérez y de los resúmenes hechos sobre las lecturas de los diferentes algoritmos que discutimos en estas mismas clases. Además, como complemento de toda la información recopilada hasta ese momento, se realizó una investigación vía electrónica en páginas especializadas en el tema que abordaban información sobre velocidad, eficiencia y uso que se les podría dar a cada algoritmo.

Se diseñó una plataforma de pruebas para poder hacer una comparación de los algoritmos a utilizar. Ésta fue diseñada para sólo cuantificar el tiempo que los algoritmos tardan en sus determinados procesos de cifrado, descifrado, hashing, firma o verificación, y no toma en cuenta el tiempo de instancia de los objetos o generación de llaves, ya que estas no forman parte del algoritmo como tal.

Se realizó un repositorio de GitHub, donde se implementó la recuperación del tiempo de ejecución de cada algoritmo a estudiar utilizando vectores de prueba específicos para cada tipo y se imprimió en formato .csv alrededor de miles pruebas, dependiendo del algoritmo a tratar, de los cuales se obtuvo un promedio del tiempo de cada uno para mostrarlos en las tablas comparativas. De igual forma se realizó la graficación de dichos tiempos utilizando una herramienta y los datos obtenidos previamente.

Resultados / resultados

➔ **Tablas comparativas**

Stream

Algoritmo	Resultados	
	Cifrado	Descifrado
RC4	9.51×10^{-6}	9.94×10^{-6}
AES-CTR	1.02×10^{-5}	1.06×10^{-5}
AES-OFB	1.01×10^{-5}	1.06×10^{-5}

Tabla 1. Promedio de tiempos de ejecución de los algoritmos por flujo en las operaciones de cifrado y descifrado.

Block

Algoritmo	Resultados	
	Cifrado	Descifrado
DES	1.45689×10^{-3}	1.4612×10^{-3}
AES	1.40×10^{-5}	1.49×10^{-5}
RCA-OAEP	1.1596×10^{-4}	1.63516×10^{-3}

Tabla 2. Promedio de tiempos de ejecución de cifrado y descifrado por bloque.

Hash

Algoritmo	Resultados
MD5	2.63×10^{-6}
SHA-1	2.62×10^{-6}
SHA-2	2.64×10^{-6}

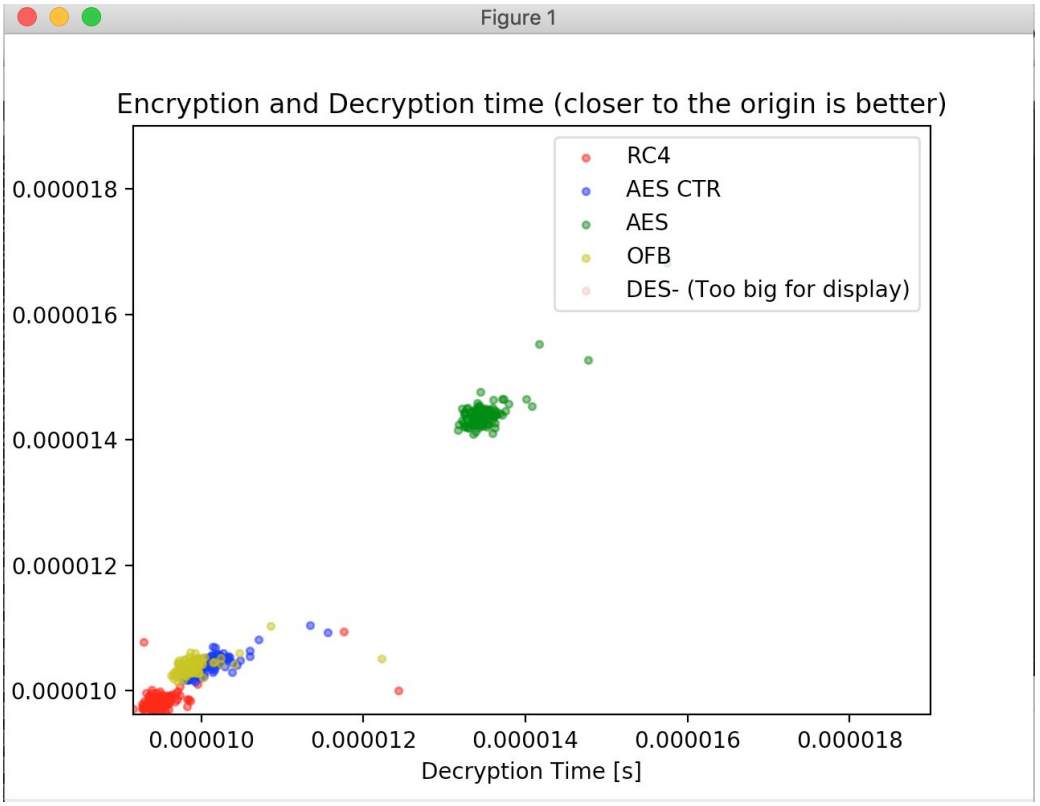
Tabla 3. Promedio de tiempos de ejecución las funciones hash.

Asymmetric key

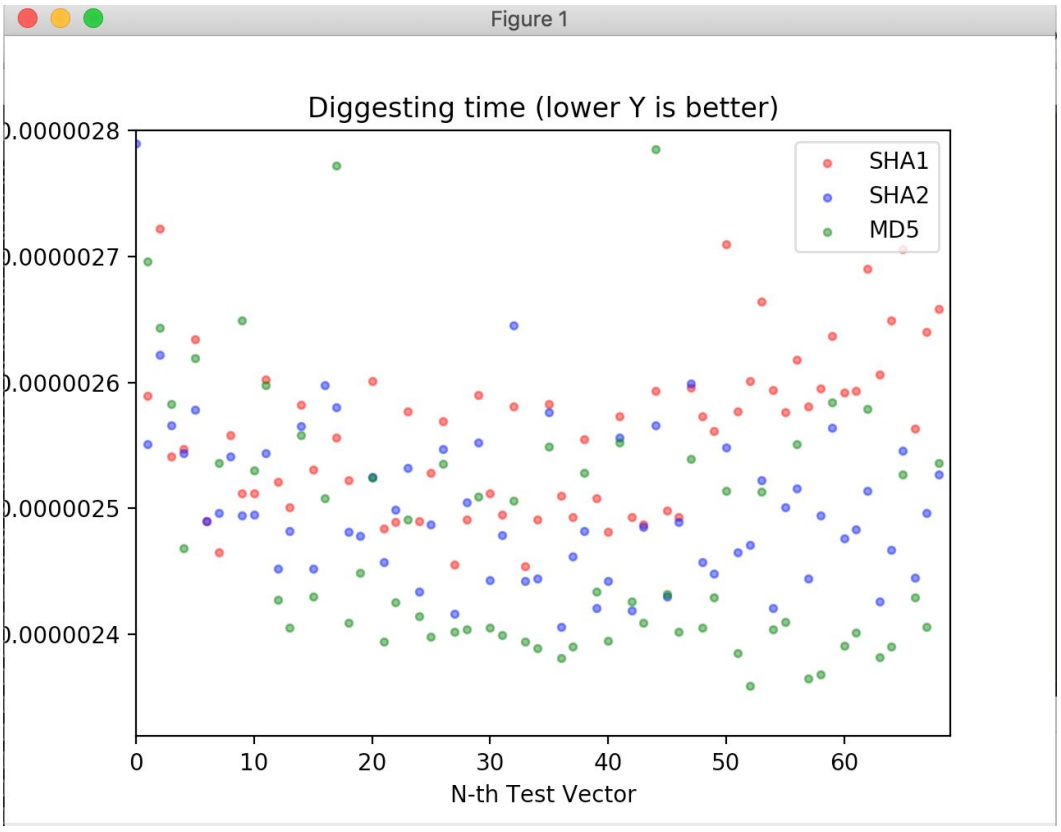
Algoritmo	Resultados	
	Firma	Verificación
RSA-PSS	4.3448×10^{-4}	7.4012×10^{-5}
DSA	1.4473×10^{-3}	1.44947×10^{-3}

Tabla 4. Promedio de tiempos de ejecución las funciones asimétricas realizando las funciones de firma y verificación.

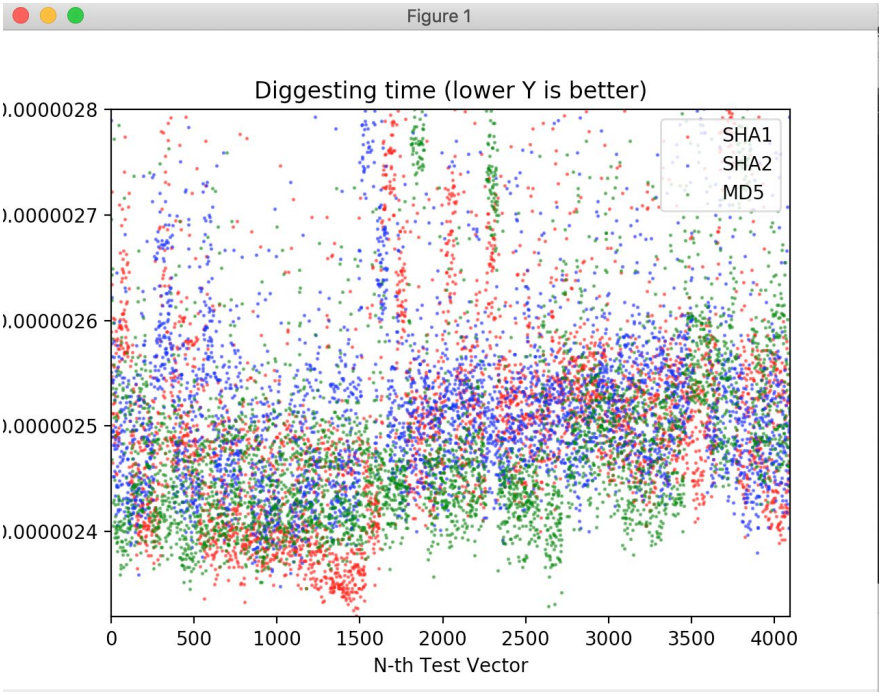
Gráficas comparativas



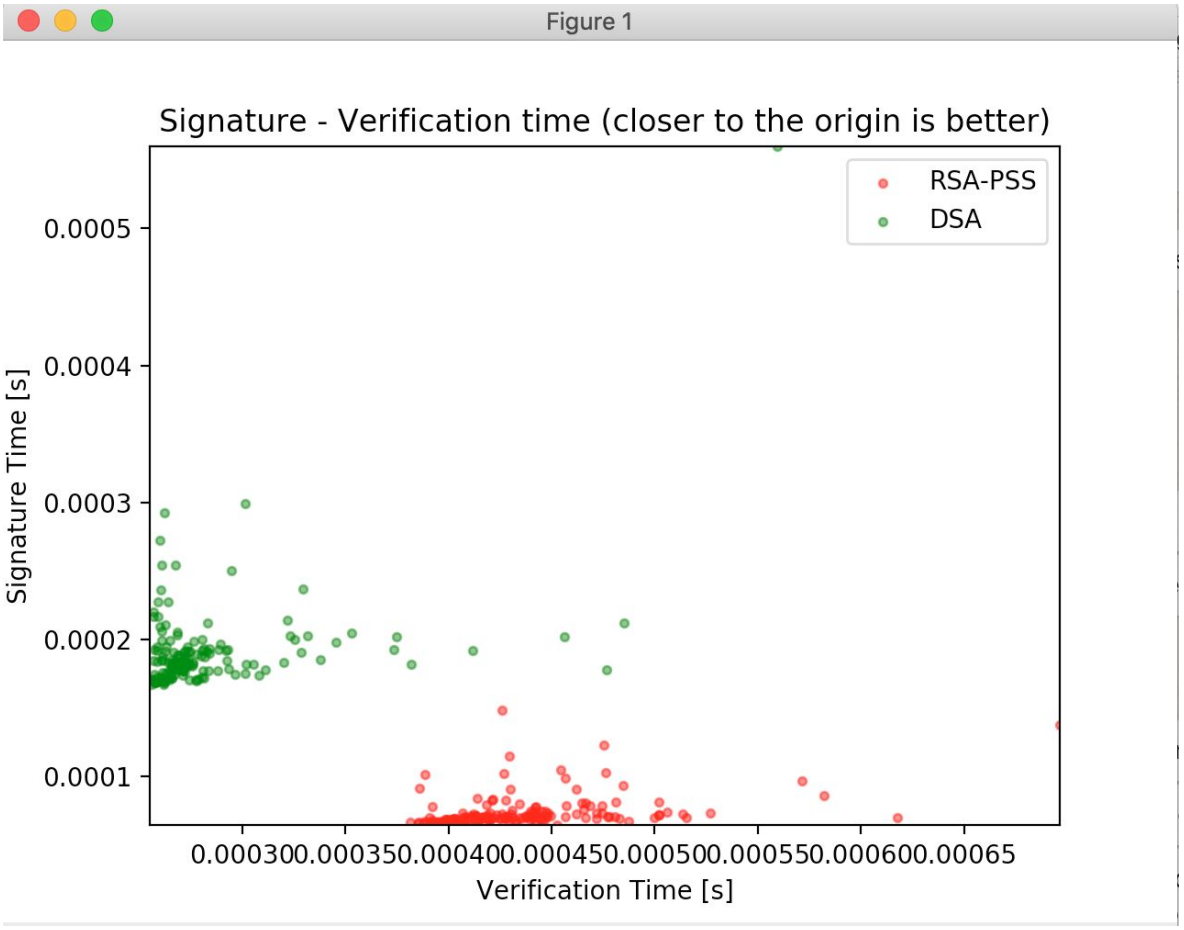
Gráfica 1. Comparación del tiempo de ejecución entre el cifrado y descifrado de los algoritmos criptográficos.



Gráfica 2. Comparación del tiempo de ejecución de las funciones digest con varios vectores de prueba.



Gráfica 3. Comparación del tiempo de ejecución de las funciones digest con mayor cantidad de vectores de prueba.



Gráfica 4. Comparación del tiempo de ejecución entre la verificación y la firma digital de los algoritmos de llave asimétrica.

Discusión

Una vez analizando los resultados descubrimos que para el cifrado por flujo RC4 es más veloz que las versiones CTR y OFB de AES, tanto para el cifrado como para el descifrado, aunque es muy importante destacar que estos últimos son modos de AES, por lo que necesitan de mayores recursos para su ejecución que RC4.

Por otro lado, en los algoritmos por bloques descubrimos que la implementación de AES es mucho más rápida que DES y RCA-OAEP, lo cual nos sorprendió debido a que las operaciones y el tamaño del bloque que procesa DES son mucho más pequeña que en AES. En contraste, era de esperarse que RCA-OAEP fuera el más lento de todos, ya que al tener que hacer operaciones más complejas que las simples utilizadas por DES y AES hace que se necesitan más recursos incluyendo al tiempo.

Para las funciones hash obtuvimos tiempos de ejecución muy similares, por lo que es difícil tomar una decisión. A pesar de esto, consideramos el más rápido a SHA-1, por tener un poco de mejor desempeño en comparación a los otros dos.

Asimismo, nuestros resultados arrojaron que, en cuanto algoritmos de llave simétrica, RSA-PSS es la opción, si queremos que la velocidad al hacer las operaciones sea rápida.

También, gracias a las gráficas, podemos hacer una comparación entre las operaciones de cifrado y descifrado, y firma y validación. Para el primer caso mencionado, podemos ver que ambos tiempos son similares de manera que no nos importa mucho que tipo de operación quisiéramos realizar si necesitamos desempeño. En contraparte, para las segundas operaciones, observamos que la verificación necesita mayor tiempo que la firma, por lo que es un punto importante a tomar en cuenta si necesitamos velocidad.

Por último, mencionaremos que si queremos comparar la velocidad entre los algoritmos de flujo y de bloque obtenemos que los de bloque son más lentos, esto debido a diferentes características que poseen, pero debemos tomar en cuenta el tipo de algoritmo que deseemos utilizar, ya que puede que a veces un por bloque requiera menos tiempo que uno por flujo o viceversa.

Acerca de Python

Python sigue siendo el lenguaje preferido para la ejecución de la criptografía, en particular para el análisis. Python, en sí mismo un lenguaje de programación flexible y creativo, tiene las bibliotecas extendidas, Cryptography y , ezPyCrypto. ezPyCrypto es una API simple, real para criptografía de grado avanzado en Python. Codifica y descifra estadísticas de tamaño aleatorio, como cadenas o archivos, mediante encriptación de clave asimétrica.

Stepic ofrece un módulo Python además de una extensión de línea de comandos para ocultar datos aleatorios dentro de las imágenes. Adapta ligeramente los colores de píxeles en la imagen para almacenar los datos. Estas variaciones son invisibles para los humanos, sin embargo, pueden ser percibidas por los programas.

Python tiene un rendimiento excepcional, su simplicidad de aprendizaje o uso lo convierte en el candidato perfecto para ejecutar nuevos algoritmos o conceptos a medida que se aprenden. Incluso, sin embargo, es simple, sigue siendo lo suficientemente influyente con bibliotecas similares: numpy que le permiten usar matemáticas más radicales, por ejemplo, matrices. De esta manera, podría centrarse más en las matemáticas más las ideas de criptografía en lugar de luchar por un lenguaje de programación que no se conoce demasiado.

Conclusiones y Recomendaciones

La razón por la cual hay tres tipos de técnicas de cifrado es que cada esquema está optimizado para aplicaciones criptográficas específicas. Las funciones de hash, por ejemplo, son adecuadas para garantizar la integridad de los datos porque cualquier cambio realizado en el contenido de un mensaje dará como resultado que el receptor calcule un valor de hash diferente al que el emisor colocó en la transmisión. Dado que es muy poco probable que dos mensajes diferentes produzcan el mismo valor hash, la integridad de los datos se garantiza con un alto grado de confianza.

La criptografía de clave secreta, por otro lado, es ideal para encriptar mensajes, proporcionando privacidad y confidencialidad. El remitente puede generar una clave de sesión por mensaje para cifrar el mensaje; el receptor, por supuesto, necesita la misma clave de sesión para descifrar el mensaje.

El intercambio de claves, por supuesto, es una aplicación clave de la criptografía de clave pública. Los esquemas asimétricos también se pueden usar para la no repudio y la autenticación del usuario; Si el receptor puede obtener la clave de sesión cifrada con la clave privada del remitente, solo este remitente podría haber enviado el mensaje. Teóricamente, la criptografía de clave pública también podría usarse para cifrar mensajes, aunque esto rara vez se hace porque los valores de criptografía de clave secreta generalmente se pueden calcular aproximadamente 1000 veces más rápido que los valores de criptografía de clave pública.

En este trabajo gracias los resultados y a la parte de discusión pudimos lograr un mejor entendimiento en cuanto a la naturaleza de cada uno de los algoritmos de clave privada o simétricos, funciones hash o digest y algoritmos de clave asimétrica. De los algoritmos simétricos se utilizaron tres por flujo (RC4, AES-CTR, AES-OFB) para tener un marco de comparación entre algoritmos de flujo, dos por bloques (DES y AES), donde fueron comparados también contra RSA-OAEP, algoritmo de cifrado asimétrico. Todos esos comparados en las operaciones cifrado y descifrado respectivamente. Los algoritmos tipo Hash fueron MD5, SHA-1, SHA-2, donde se utilizan un mayor número de operaciones para hacer más seguro el proceso. Y finalmente la comparación entre los algoritmos de asimetricos DSA contra RSA.

Una parte que debemos tomar en cuenta es para qué queremos utilizar el algoritmo y la cantidad de recursos que tenemos o podemos necesitar. A pesar de descubrir cuales son los algoritmos más rápido no siempre podemos utilizarlos, dependerá de la sensibilidad de la información y de la plataforma o el hardware donde lo estemos utilizando. En muchas ocasiones los algoritmos más rápidos son los que pueden llegar a romperse, por lo que se aumenta la complejidad de estos y con ello también la cantidad de recursos que necesitamos.

En la actualidad, con el poder computacional que tenemos, no importa en gran medida que un algoritmo sea más lento que otro o que necesite más recursos, aunque es de vital importancia tenerlo en cuenta, ya que a la larga el no importarnos puede general problemas mayores a largo plazo.

En teoría, cualquier tipo de encriptación puede romperse con suficiente tiempo, energía y potencia de procesamiento.Las computadoras se vuelven cada vez más rápidas y más eficientes.Entonces nos encontramos en una carrera armamentista entre la capacidad de descifrar el código y el cifrado en sí. Las formas más fuertes de cifrado de hoy serán probablemente ridículamente débiles para mediados de siglo. Afortunadamente, esas mismas herramientas son probablemente la clave para la próxima generación de algoritmos de cifrado imposibles.

Referencias

1. Abdullah, A. M. (2017). Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data. University of Sulaimani.

2. Aldeco, Rocio. *Criptografía: 2020-1*. Universidad Nacional Autónoma de México. CDMX, México. Agosto - Septiembre del 2019.
3. Bravo, S. (2002). *Estudio comparativo de los algoritmos de cifrado de flujo RC4, A5 y SEAL*. CINVESTAV-IPN.
4. Diffie, W., y Hellman, M. E. (1977). *Exhaustive Cryptanalysis of the NBS Data Encryption Standard*. Stanford University: pp. 74-84.
5. Eastlake, A., y Jones P. (2001). *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174.
6. García R., Loatzin G., Cabrera, L., Puente C., y Méndez, V. (2018). *AES como estándar internacional de cifrado*. Instituto Tecnológico de Tlalnepantla.
7. Kasgar, A. K., Dhariwal, M. K., Tantubay, N., y Malviya H. (2013). *A review paper of Message Digest 5 (MD5)*. International Journal of Modern Engineering & Management Research, 1(4), pp. 29-35.
8. Loomis, P. A. (2016). *Una introducción histórica al sistema de criptografía RSA*. Revista Investigación y Tecnología, 4(1).
9. Miramontes, L. E., De La Cruz, H. A., Carvente, C. A., y Caballero M. (2019). *CryptoFirstProject* [Software]. Recuperado de: https://github.com/Tenkoni/CryptoFirstProject
10. Python Cryptographic Authority (2017). *Cryptography* [Software]. Recuperado de: https://cryptography.io/en/latest/community/
11. Rivest, R. L., Shamir, A., y Adleman, L. (1977). *A method for obtaining digital signatures and public-key cryptosystems*. Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge.
12. Wong E. (2019). *DES (Data Encryption Standard)* (Version 1.0.5) [Software]. Recuperado de: https://pypi.org/project/des/
13. Kitty Gupta, The best programming languages for cryptography .(2017), USA.