Artículo **Revista de Sistemas Computacionales y TIC's**
Diciembre 2017 Vol.3 No.10 42-49

Análisis Comparativo de Cifrado Asimétrico algoritmos RSA y ElGamal

TOMÁS-MARIANO, Víctor Tomás*†, ESCUDERO-CISNEROS, José y HERNÁNDEZ HERNÁNDEZ, Iván

Recibido Octubre 27, 2017; Aceptado Noviembre 21, 2017

Resumen

Se analiza el proceso de cifrado asimétrico en distintos tipos de archivos en dos modelos, cliente servidor y modo "localhost", el primero permite transferir archivos encriptados en red con el uso de sockets y multiprocesamiento, el segundo, admite encriptar archivos desde una computadora de uso personal; se implementan los algoritmos RSA y ElGamal con números de distinta longitud en dígitos, para ello se utiliza la clase BigInteger de Java. La creación, duración y destrucción de claves se gestionan en el tiempo que permanece abierta una sesión de trabajo, las claves tienen diferente longitud en bits que exigen los algoritmos para su implementación. Los resultados en archivos encriptados permite observar un incremento en su tamaño de hasta 400% con RSA y hasta un 1800% con ElGamal. El tiempo en encriptar es mayor con RSA que con ElGamal si se aplica el proceso de encriptación a un mismo archivo. Si bien existen en la literatura varias implementaciones de los algoritmos aquí usados, en la mayoría de estos sólo se enfocan a un tipo de archivo, en esta propuesta se utilizan los formatos de archivos más conocidos, archivos con extensión: pdf, docs, xls, ppt, txt e imágenes.

Criptografía asimétrica, RSA, ElGamal

Abstract

The asymmetric cipher process is analysed in different types of files in two models, client server and "localhost", the first one allows to transfer encrypted files in network with the use of sockets and threads, the second, allows to encrypt files from a computer of personal; we implement the RSA and ElGamal algorithms with numbers of different length in digits, using the Java BigInteger class. The creation, duration and destruction of keys are managed in the time that a work session lasts, the keys have different length in bits that require the algorithms for their implementation. The results in encrypted files observe an increase in size up to 400% with RSA and up to 1800% with ElGamal. The time to encrypt is greater with RSA than with ElGamal if the encryption process is applied to the same file. Although there are several implementations of the algorithms used in the literature, most of them only focus on one type of file, in this proposal we use the most popular file formats, files with extension: pdf, docs, xls, ppt, txt and images.

Asymmetric cipher, RSA, ElGama

Citación: TOMÁS-MARIANO, Víctor Tomás, ESCUDERO-CISNEROS, José y HERNÁNDEZ HERNÁNDEZ, Iván. Análisis Comparativo de Cifrado Asimétrico algoritmos RSA y ElGamal. Revista de Sistemas Computacionales y TIC'S 2017, 3-10: 42-49

* Correspondencia al Autor (Correo Electrónico: victor_tomas@uaeh.edu.mx)
† Investigador contribuyendo como primer autor.

©ECORFAN-Spain

Introducción

El manejo de la información es muy importante en la actualidad, fundamentalmente en el envío de esta por canales inseguros como lo es el internet o una red local dentro de una empresa u organización, ya que está expuesta a terceros que pueden captar el contenido de un archivo (mensaje), y quedar exhibida a personas no autorizadas. La información pasa por distintas etapas y en cada una de ellas se debe tener cuidado, por ejemplo: *creación, envío y recepción* ya que son en estás en las que puede sufrir algún tipo de alteración (Álvarez M., Pérez G.P.P, 2004).

El manejo de la información principalmente se debe de cumplir dos o más de los siguientes principios: la *confidencialidad* asegura que solo las personas o procesos autorizados tienen acceso a la información. La *integridad* asegura que los datos no han sido modificados durante la transmisión. La *autenticidad* que asegura que la información se debe comprobar de quien lo envía es quien dice ser, es decir, es posible usar firmas digitales para cumplir este principio (Aguilar M.E., 2012, Ibarra Q.R. et all, 2001).

En este sentido la criptografía es un pilar fundamental en la seguridad de información, no solo en los archivos sino también en el medio de transporte ya permite ocultar el contenido de un mensaje con la aplicación de diferentes técnicas de cifrado, entre ellas está el cifrado asimétrico que utiliza distintas claves, por lo menos dos, para cifrar y descifrar el contenido de un archivo, en particular en este trabajo se implementa el algoritmo de cifrado RSA y el algoritmo ElGamal.

En la sección I se mencionan los conceptos de criptografía y los trabajos relacionados con el presente trabajo, en la sección II se describen las fases de los algoritmos RSA y ElGamal, en la sección III de las herramientas de software utilizadas y por último en la sección de resultados se describen los hallazgos y de las posibles mejoras a la aplicación.

Fundamentos de Criptografía

Existen en la literatura varias definiciones de criptografía, en este artículo se usa la propuesta por (Ramio A.J, Sánchez P.D.J., 2016) que la define como "*aquella ciencia que hace uso de métodos y herramientas matemáticas con el objeto principal de cifrar, y por tanto proteger, un mensaje o archivo por medio de un algoritmo, usando para ello dos o más claves, con lo que se logra en algunos casos la confidencialidad, en otros la autenticidad, o bien ambas simultáneamente*". Esta definición no es exacta para criptografía simétrica, en la que se utiliza una misma clave para cifrar y descifrar un mensaje, sin embargo, si corresponde con la criptografía asimétrica, en la que se generan dos claves diferentes, una para el proceso de cifrado y otra clave para el descifrado, pública y privada respectivamente; la seguridad de este tipo de criptografía radica en mantener en secreto la clave privada o que ésta sea muy difícil de obtener a partir del conocimiento de la clave pública y que computacionalmente tiene un costo muy alto en intentar conocer el valor de la clave privada. Entre los algoritmos más importantes que existen en la literatura para criptografía asimétrica podemos encontrar el RSA y ElGamal (Bishop D., 2003), entre otros.

A continuación se mencionan algunos trabajos relacionados con los algoritmos ElGamal y RSA.

ISSN-2444-5002
ECORFAN® Todos los derechos reservados

Una implementación y análisis de los algoritmos ElGamal y RSA se desarrolla por Sharma (2014) en la que realiza pruebas para comparar el tiempo de cifrado y descifrado para cumplir con las bases de seguridad de comunicación en red. También describe las fases de cada algoritmo así como el funcionamiento con ejemplos sencillos.

Satvika (2016) propone la generación de claves mediante la combinación de los algoritmos RSA y ElGamal. El esquema propuesto reduce el tiempo de cifrado comparado con el algoritmo RSA. Utiliza números primos de 256 bits lo que supone pérdida de seguridad; sin embargo, la clave pública y clave privada no puede conocerse de manera directa porque se basan en el logaritmo discreto lo que les da más fortaleza.

Otro trabajo interesante es el de (Sen y Siva 2013) en la que realizan un reporte técnico sobre el análisis de los algoritmos ElGamal y RSA, se hace una comparación entre ambos algoritmos: funcionamiento, fundamentos matemáticos, ejemplos con valores numéricos pequeños, aplicaciones, código fuente de módulos de programación que utiliza cada algoritmo así como una explicación detallada.

Con respecto a los esquemas de criptografía basados en logaritmos discretos en ElGamal (1985) se propone un esquema de su funcionamiento, las ventajas que posee así como los fundamentos matemáticos que deben cumplir para obtener resultados satisfactorios.

Metodología

Se implementan dos algoritmos de criptografía asimétrica RSA y ElGamal para el cifrado-descifrado de archivos, en ambos algoritmos se usan números primos de distinta longitud.

El proceso de cifrado se aplica en archivos que contienen básicamente caracteres del código ASCII, sin embargo, también se aplica a archivos con diferente formato, como lo son las imágenes, archivos con extensión pdf, xls y doc, obteniendo buenos resultados en el cifrado.

A continuación se mencionan las fases del funcionamiento de los algoritmos RSA y ElGamal

Fundamentos de Algoritmo RSA

El algoritmo RSA se basa en el manejo de números primos de gran tamaño, se recomienda 1024 bits o más, es decir, usar números primos de más de cien dígitos de longitud, aplica pruebas de primalidad para comprobar si un número generado es primo o no lo es, existen varios algoritmos, entre ellos el de Fermat, pero en la práctica se suele utilizar el de Miller-Rabin (Bishop D., 2033, Richard J., 2005) entre otros, debido a que ofrece mejores resultados en las pruebas de primalidad. La utilización de números primos es debido a que son muy difíciles de factorizar, y el hecho de que cumpla la prueba de primalidad asegura que no tiene factores.

A continuación se enumeran los pasos que sigue este algoritmo (Bishop D., 2003):

Generación de claves, basado en la exponenciación modular, el algoritmo RSA funciona de la siguiente forma:

1. Generar dos números primos lo suficientemente grandes: p y q (entre 100 y 300 dígitos de longitud.)

2. Calcular el valor de $n = p * q$ y $ø(n) = (p - 1) * (q - 1)$.

3. Seleccionar un número entero e tal que $1 <= e <= ø(n)$, de modo que cumpla con: $mcd\ (e, ø(n))=1$.

ISSN-2444-5002
ECORFAN® Todos los derechos reservados

4. Calcular *d*, de tal manera que cumpla $e*d \equiv 1 \ (mod \ ø(n))$ con $1<= d < ø(n)$.

5. Publicar la pareja *(n, e)* que es la clave pública, el valor de la clave privada *d*, y los valores *p*, *q* y *ø(n)* deben permanecer en secreto.

La robustez del algoritmo RSA radica en el tamaño de *p* y *q*, se recomienda un tamaño mínimo de 1024 bits, a mayor longitud más seguro será el algoritmo ya que se obtiene una longitud de n que duplica los valores de p y q (Satvika, 2016). Sin embargo, en la realización de los cálculos de cifrado y/o descifrado se usan valores de distinta longitud en la exponenciación modular lo que lo hace más lento. Al obtener un valor de n lo bastante grande también resulta más complejo descomponer en sus factores. Otro elemento importante es el tamaño de las claves, si se manejan longitudes mayores a 100 dígitos al realizar la multiplicación con números de esta longitud se obtienen valores mayores a 200 dígitos (Satvika, 2016). Se sugiere consultar (Bishop D., 2003 y Aguilar M.E., 2012) para ejemplos de este algoritmo.

Fundamentos del Algoritmo ElGamal

Este algoritmo inicialmente fue usado para firmas digitales, más tarde se modificó para cifrar y descifrar mensajes (Satvika, 2016). Es similar al intercambio de claves con Deffie-Hellman, los pasos necesarios para cifrar un mensaje son: (Bishop D., 2003):

1. Se genera un numero primo p y un generador *g modulo p*, *p* debe ser de al menos de 1024 bits de longitud.

2. Seleccionar un numero a tal que 1< a <p-1, y calcular el residuo r:
   $r \equiv g^a \ (mod \ p) \quad (0 \leq r \ < p)$

3. Hacer público los valores p, g y r. La clave privada es a.

4. Suponga ahora que se desea enviar el mensaje P, con $0 \leq P < p$.

5. El emisor genera un numero aleatorio *k* tal que $1<=k<=p-2$, es importante elegir un valor diferente para cada mensaje; se calculan los siguientes valores:

$c \equiv g^k \ (mod \ p) \quad (0 \leq c < p)$
$d \equiv Pr^k \equiv P(g^a)^k \ (mod \ p) \ (0 \leq d < p)$

6. El texto cifrado es el par de valores c y d; esto es: C= (c, d)

7. Para descifrar el mensaje C, se calcula el inverso de c^amodulo p; esto es:
   $z \equiv (c^a)' \ (mod \ p) \ \ (0 \leq z < p)$

8. Para recuperar el texto original se calcula:

$P \equiv zd \ (mod \ p) \ (0 \leq P < p)$

La robustez de este algoritmo radica en la dificultad de calcular el logaritmo discreto. El problema de logaritmo discreto es, si *p* es un número primo, g y y un número entero cualquiera, buscar x tal que $g^x = y \ (mod \ p)$ lo cual resulta muy complicado encontrar (Satvika 2016, Sharma A., 2014) (Marfia A.M., 2015). Se recomienda consultar (Vasirani S.R.) para ejemplos del funcionamiento de este algoritmo.

El algoritmo ElGamal es más seguro comparado con el algoritmo RSA ya que es más complejo al cifrar texto, también es más lento en el proceso de cifrado y descifrado porque genera más de una clave pública (Sharma, 2014).

El manejo de la claves juega un papel fundamental, se deben generar de una manera aleatoria. Al respecto, en criptografía asimétrica, es el propio algoritmo que genera las claves para el proceso de cifrado-descifrado.

ISSN-2444-5002
ECORFAN® Todos los derechos reservados

Lo más común es generar las claves de manera automática debido a que son más seguras, estas claves se crean y destruyen de manera automática, su durabilidad es por tiempo limitado, segundos o milésimas de segundos, algunos protocolos de comunicación las utilizan para el intercambio de información a través de un canal inseguro, ofrecen la posibilidad de cifrar el medio de transporte, en algunos casos, cifrar el contenido o mensaje que viaja por el canal.

Herramientas de programación

BigInteger

Es una clase de Java que proporciona la posibilidad de manejar números enteros de gran longitud, manejo de operaciones como la multiplicación, división, suma, resta, comparación, etc. mismas que se pueden hacer con los datos primitivos como el tipo de dato Integer. Tiene integrado métodos sobre aritmética modular, máximo común divisor, generación de números primos, pruebas de primalidad, exponenciación, cálculo de inversos, entre otros (Demo Source and Support, 2017). En la figura 1, se muestran algunos resultados de números generados con distintas longitud en bits.

Longitud en Bits	Número aleatorio probablemente primo con la clase BigInteger
256	898402865429232609682269648299642774036187389532602614743605419008198227990 21
256	871528915207582766559389353459849280456127867914883165160676764643550196528539
512	745689624413208713137421917677329410598335890055107635406920858668972966972866809227104033651814868434663786328675492722194952635628322780766620130842306 3
512	122574250521863292936663333618134448091623831665532874223669765902622972190399140110458775470066694122685495042711058152860930738429004583463452700871358 47
1024	980240313794529327384246970704389485819304977209326788578576825303322644995550040632191337104768222833433402148699322885589148156233374656561261839446544563409639561637528699335948897671170954907518545593768447884648051107449706574090186520502977048042893253240270910042239585881850120003490157278577337918 23
1024	104985243449914717964776269820499996494218797418791357619743026126104411811103070258008854397388264765537490697848035265941484685149600824274424535627170650149705127421041457067890666014851709836054388654793656082134515215232831538558427830774263689700334589108631572126715415537078739817748209954565771203299 1

Figura 1 Números de distinta longitud generados con BigInteger

Fuente: creación propia

Sockets y Threads

Los sockets en java son una herramienta que permite conectar dos computadoras en modo localhost o en red, maneja un conjunto de métodos para el envío de archivos y/o mensajes a través de un canal, utiliza el protocolo TCP/IP para la comunicación entre dos nodos con la dirección IP y un puerto. Tiene métodos para utilizar distintos tipos de flujo entre los nodos. Permite configurar un nodo como servidor o cliente según sea el caso, o intercambiar los roles en tiempo de ejecución. En la figura 2 se muestra la conexión en red de pruebas de trasmisión y recepción de archivos.

Por otra parte, los threads (hilos) permiten simular programación concurrente, es decir, se puede tener varios procesos ejecutándose al mismo tiempo, ofreciendo distintos servicios en tiempo de ejecución. Un thread puede lanzar un servidor y dejarlo en espera de conexiones de clientes, a su vez, por cada cliente conectado se crea otro thread que da servicio a un solo cliente, lo que permite atender varios clientes al mismo tiempo. Los threads tienen distintos status que se pueden modificar a conveniencia del programador para poder lanzar (run), dormir (sleep), interrumpir o pararlos e incluso sincronizar procesos. Combinado con los sockets ofrecen una poderosa herramienta para desarrollar aplicaciones servidor-cliente en entornos java (Deitel and Deitel, 2008, pags. 925-1035).

Resultados

El algoritmo RSA genera dos claves, una pública misma que se utiliza para cifrar y una privada para recuperar el mensaje original, el tamaño de claves es de 1024 bits. Para el desarrollo de esta aplicación se usa la plataforma Netbeans versión 8.1.

ISSN-2444-5002
ECORFAN® Todos los derechos reservados

Figura 2 Pruebas de transmisión en red

Fuente: creación propia

Para el entorno cliente-servidor, la comunicación se realiza por medio de sockets; para mantener el enlace o sesión de trabajo, se usan threads.

Se adapta el algoritmo RSA en la parte cliente y en la parte servidor, es decir ambas computadoras están a la espera de información (mensajes) para iniciar el proceso de cifrado. El cliente como el servidor generan dinámicamente sus claves pública y privada, la clave pública es compartida para que la parte emisora cifre el mensaje y lo transmita. La clave privada se usa por el receptor para descifrar el mensaje. En la figura 3 se muestran los números cifrados con RSA con una longitud de clave de 512 bits.

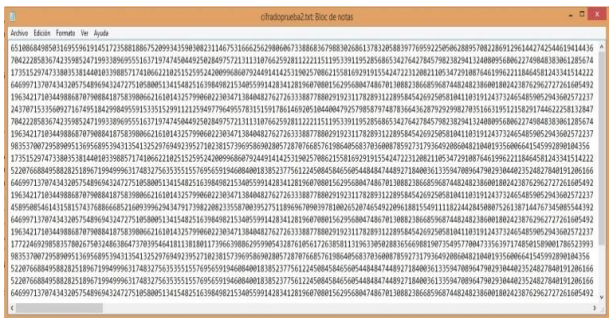

Figura 3 Pruebas de cifrado con RSA

En la tabla 1 se muestran los resultados obtenidos con pruebas realizadas con el algoritmo RSA con longitud de claves de 10 dígitos.

TOrig	TCif	Ms	TDes	Ms
2 KB	261 KB	6984	2 KB	3824
5 KB	678 KB	110889	5 KB	5230
10 KB	1630 KB	102089	10 KB	4825

Tabla 1 Tiempos de cifrado algoritmo RSA (Ms=Milisegundos)

La aplicación permite cifrar/descifrar archivos de Word, Power Point, Excel, también pdf, bloc de notas e imágenes. Los archivos obtenidos mediante el proceso de cifrado sufren un aumento aproximadamente en promedio del 400% sin embargo hay ocasiones que se puede incrementar hasta en un 800%, esto se debe al tamaño de las claves y manejo de números enteros de distinta longitud, otro factor que se ve afectado por el tamaño de las claves es el tiempo que tarda en cifrar y descifrar un archivo.

En la tabla 2 se muestran los resultados obtenidos en pruebas realizadas con archivos de texto con longitud de clave de 512 bits.

El algoritmo ElGamal cifra y descifra información de tipo texto o imagen, implementa dos claves, una pública que es conocida por todo mundo y una privada que solo la posee el receptor, el tamaño de claves es de 1024 bits. La aplicación tiene dos opciones de funcionamiento para el envío de archivos y un chat.

En el envió de archivos la aplicación muestra el contenido de un directorio, el usuario elije un fichero, lo cifra, genera la clave y almacena el archivo cifrado y clave asociada en una carpeta del disco local C, en esta fase puede cifrar tantos archivos como se desee. En la segunda fase, consiste en enviar el archivo cifrado junto con la clave asociada del archivo al cliente o clientes conectados al servidor, del lado del cliente se crea una carpeta con el contenido de archivos cifrados recibidos y es el usuario cliente quien decide qué archivos descifrar.

ECORFAN® Todos los derechos reservados

En la figura 4 se muestra una prueba de cifrado con una clave de 256 bits con algoritmo ElGamal.

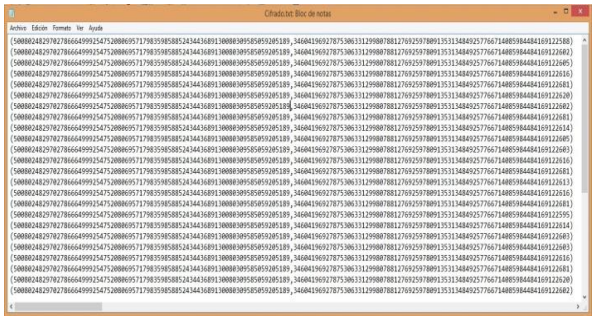

Figura 4 Pruebas de cifrado con ElGamal

Por otro lado, en la función de chat todos los mensajes que son enviados a través de la red pasan por el algoritmo ElGamal, se cifran se envían a través de la red a sus respectivos clientes, los clientes reciben el mensaje y son descifrados en tiempo real.

TOrig	TCif	Ms	TDes	Ms
2 KB	270 KB	17735	2 KB	7858
5 KB	729 KB	42829	5 KB	21552
11 KB	1521 KB	113202	11 KB	51422

Tabla 2 Prueba con ElGamal con longitud de 512 bits (Ms=Milisegundos)

El incremento en tamaño con respecto al cifrado ElGamal varía dependiendo de la longitud de los números generadores y de la clave, ver tabla 2.

Sin embargo, el incremento del tamaño del archivo encriptado está en función del tamaño de longitud en bits elegido para los números primos o la clave, lo que influye de manera significativa en el proceso de cifrado-descifrado ya que a mayor longitud en bits, mayor tiempo en ambos procesos. En la tabla 3 se muestran los tiempos al menjar números de 1024 bits.

TOrig	TCif	Ms	TDes	Ms
2 KB	229 KB	26674	2 KB	11020
5 KB	1428 KB	74789	5 KB	27958
11 KB	2970 KB	130863	11 KB	62853

Tabla 3 Prueba con ElGamal longitud de 1024 bits (Ms=Milisegundos)

Conclusiones

En ambos algoritmos se obtienen resultados satisfactorios para el cifrado y descifrado de información, se recupera de forma legible el mensaje original del archivo lo que refleja en buen funcionamiento de los algoritmos.

El algoritmo RSA ofrece buenos resultados de cifrado, sin embargo, incrementa el tamaño del archivo de una manera considerable con respecto al tamaño original del archivo.

Con el algoritmo ElGamal los resultados fueron muy satisfactorios, al encriptar un archivo hay un incremento mayor conrespecto al RSA, este algoritmo utiliza dos números cifrados por lo tanto requiere más memoria para su almacenamiento en disco duro con respecto al RSA y utilizando la misma longitud de claves.

En relación al chat, se logra mantener una conversación cifrada en ambos extremos de la red, esto es debido a que por lo regular en una conversación de chat se escriben pocos caracteres lo que hace que el proceso de cifrado y descifrado sea imperceptible en tiempo real.

En ambos algoritmos RSA y ElGamal utilizan números primos de gran longitud, esto hace que los archivos cifrados sufran un cambio considerable en su tamaño, sin embargo, cuando el tiempo de cifrado no importa, se obtienen excelentes resultados.

ISSN-2444-5002
ECORFAN® Todos los derechos reservados

Una posible mejora es implementar técnicas de programación paralela para acelerar el tiempo del proceso de cifrado-descifrado, por ejemplo en GPU.

El tiempo de cifrado y descifrado no solo está en función del tamaño de los números primos sino también del tipo de flujo de datos que se abre desde y hacialos canales origen y destino a los que se envía la información.

Las pruebas se realizaron en Laptops con características similares por ejemplo en una lenovo de 1TB de disco duro 4 GB de RAM y un procesador Pentium N3540 de una velocidad de 2.16GHz de 4 núcleos físicos.

Referencias

Aguilar M. E. (2012), Sistema tutorial de Fundamentos de Criptografía, Tesis ingeniería en computación, UNAM.

Álvarez M., Pérez G.P.P (2004), Seguridad informática para empresas y particulares, McGraw-Hill, ISBN: 84-481-4008-7.

Bishop D. (2003), Introduction to Cryptography with applets.

Deitel P.J. and Deitel H.M., (2008), Como programar en Java, 7ma. Edición, Pearson Educación, Capítulo 23 y Capítulo 24.

Demo Source and Support, (2017) Use BigInteger: BigInteger « Data Type « Java Tutorial, Recuperado de http://www.java2s.com/Tutorial/Java/0040__Data-Type/UseBigInteger.htm.

Elgamal T., (Julio, 1985), "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472.

Ibarra Q. R., Serrano L. M. A. (2001), Teoría de la información y encriptamiento de datos, IPN.

Marfia, A. M., 2015, Obtenido de http://www.mate.unlp.edu.ar/~demetrio/Monografias/Materias/EA/33.%20Criptografia%20-%20Marfia%20&%20Martinez%20Lopez.pdf , consultado el 05/05/2017.

Ramió A.J., Sánchez P. D. J. (2016), Introducción a la seguridad en informática y criptografía clásica, Universidad Politécnica de Madrid, curso en línea, http://www.criptored.upm.es/crypt4you/temas/criptografiaclasica/leccion2.html, consultado: 26/05/2017.

Richard J. (2005), Matemáticas Discretas, Pearson Education, ISBN: 970-26-0637-3, México.

Satvika I. (Octubre, 2016), Key Generation Algorithm Design Combination of RSA and ElGamal Algorithm, 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Yagyacarta, Indonesia.

Sen J., Siva K., (2013), CryptoSystems RSA and ElGamal: A Technical Report, Summer Research Fellowship-2013, National Institute of Science Technology, Odisha, India.

Sharma A., Attri J., Devi A., Sharma P., (Octubre, 2014), Implementation and Analysis of RSA and ElGamal Algorithm, Asian Journal of Advanced Basic Sciences, vol. 2, No. 3, pp.125-129.

Vasirani, R. H., Fundamentos Matemáticos de la Criptografía. Seguridad Informática, 31. Obtenido de http://www.ia.urjc.es/cms/sites/default/files/userfiles/file/SEG-I/2012/s2xFundamentosMatematicos.pdf

ECORFAN® Todos los derechos reservados