# ESCUELA POLITÉCNICA NACIONAL

## FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

# DESARROLLO DE UNA APLICACIÓN DE USO DIDÁCTICO PARA COMUNICACIÓN SEGURA DE DATOS A TRAVÉS DE LA RED

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y REDES DE INFORMACIÓN**

**JÉSSICA ALEJANDRA JIMÉNEZ LEONES**

**DIRECTOR: Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA**

**Quito, marzo 2018**

# AVAL

Certifico que el presente trabajo fue desarrollado por Jéssica Alejandra Jiménez Leones, bajo mi supervisión.

**Ph.D. Robin Álvarez**
**DIRECTOR DEL TRABAJO DE TITULACIÓN**

# DECLARACIÓN DE AUTORÍA

Yo, Jéssica Alejandra Jiménez Leones, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

JÉSSICA ALEJANDRA JIMÉNEZ LEONES

# DEDICATORIA

Este proyecto de titulación va dedicado a mi abuelita Elvira, quien me formó con su ejemplo de buena cristiana y me apoyó en cada momento.

También se lo dedico a mis padres: Vicelda y René, quienes fueron un apoyo importante tanto en lo profesional como personal a lo largo de mi vida.

A mi tío Abel, quien siempre me ha demostrado que no todo en la vida es fácil, pero se lo puede conseguir con esfuerzo, sacrificio y dedicación.

Y finalmente se lo dedico a mis hijas Violeta y Ariana, demostrándoles que todo lo que se desea conseguir de corazón es posible a pesar de las adversidades, y ser así un ejemplo de superación y dedicación.

Jéssica Alejandra Jiménez Leones

# AGRADECIMIENTO

Agradezco infinitamente a Dios, Padre Celestial por brindarme la vida, por darme la salud para poder culminar esta etapa de mi vida, y al Espíritu Santo que me da el don de la sabiduría y la fortaleza para emprender este proyecto.

A mi madre, quien me ha brindado su apoyo incondicional en todo momento y me da todo su amor, paciencia y dedicación.

A mi abuelita, quien ha sabido guiarme como una madre, me ha apoyado y ha estado conmigo en todo momento.

Gratitud al PhD. Robin Álvarez por brindarme la oportunidad de realizar este proyecto y depositar su confianza en mí. Le agradezco por toda su comprensión, colaboración y entrega durante la elaboración del presente Proyecto de Titulación.

A todos mis familiares y amigos, que han estado junto a mí brindándome su apoyo y motivándome durante este proceso de culminación.

Jéssica Alejandra Jiménez Leones

# ÍNDICE DE CONTENIDO

# ÍNDICE DE FIGURAS

## ÍNDICE DE TABLAS



## ÍNDICE DE SEGMENTOS DE CÓDIGO

# RESUMEN

El presente proyecto de titulación pretende desarrollar una aplicación de uso didáctico en la cual se realiza una comunicación segura de datos por medio de la red, utilizando la herramienta de codificación Matlab. Consta de cuatro capítulos en los cuales se realiza un análisis teórico de la seguridad informática, metodología e implementación de la aplicación, además del análisis de resultados y pruebas de funcionamiento.

En el primer capítulo, se describe de forma teórica los términos generales y servicios de la seguridad informática.

En el segundo capítulo, se pretende describir los requisitos que debe tener la aplicación de uso didáctico, para adaptar los diferentes algoritmos criptográficos (DES, RSA, MD5) en el *software* Matlab, y así implementar los diferentes servicios de seguridad informática.

El tercer capítulo presenta las pruebas de funcionamiento y la comprobación de resultados utilizando el *software* libre CrypTool.

El cuarto capítulo muestra las conclusiones y recomendaciones referidas al presente proyecto de titulación.

Al final, incluye una sección de Anexos en la cual se encuentran códigos de *scripts* utilizados para la implementación los diferentes servicios de seguridad informática, generalidades de Matlab y fundamentos matemáticos del algoritmo RSA.

**PALABRAS CLAVE:** seguridad informática, algoritmos de cifrado/descifrado, servicio de confidencialidad, servicio de integridad, servicio de autenticación, servicio de no repudio.

# ABSTRACT

The present titling project aims to develop a didactic application in which secure data communication is carried out through the network, using the Matlab coding tool. It consists of four chapters in which a theoretical analysis of computer security, methodology and implementation of the application, in addition to the analysis of results and performance tests.

In the first chapter, the general terms and services of computer security are described in a theoretical way.

In the second chapter, we try to describe the requirements that the didactic use application must have, to adapt the different cryptographic algorithms (DES, RSA, MD5) in the Matlab software, and thus implement the different IT security services.

The third chapter presents the functional tests and the checking of results using the free software CrypTool.

The fourth chapter shows the conclusions and recommendations referred to the present titling project.

At the end, it includes a section of Annexes in which are codes of scripts used for the implementation of the different IT security services, generalities of Matlab and mathematical foundations of the RSA algorithm.



**KEYWORDS:** informatic security, encryption / decryption algorithms, confidentiality service, integrity service, authentication service, non-repudiation service.

# 1. INTRODUCCIÓN

La criptografía tiene sus orígenes hace más de 4000 años desde que el hombre empezó a escribir, con la finalidad de mantener protegida información secreta. En la nueva era digital de comunicación la información se transmite a través de internet, y debido a la gran cantidad de usuarios que se comunican a través de este medio, es necesario emplear herramientas automatizadas que permitan proteger la información que se envía y recibe, lo cual requiere desarrollar aplicaciones seguras, basadas en algoritmos matemáticos estandarizados.

En general, todo sistema de comunicaciones debe cumplir con los siguientes requerimientos mínimos de seguridad: confidencialidad, integridad, autenticidad y no repudio. Al acoplar todos estos principios fundamentales se consigue mejorar la confianza al enviar un mensaje, garantizando un nivel adecuado de seguridad en la transferencia de la información.

Actualmente, existen algoritmos modernos que permiten cumplir con estos requerimientos, los cuales se los pueden agrupar en algoritmos simétricos, asimétricos y de *hash* o resumen, los cuales en conjunto generan un sistema de seguridad informático.

Los algoritmos simétricos y asimétricos brindan el servicio de confidencialidad: los datos son cifrados con dichos algoritmos y si la información es interceptada por personas no autorizadas nunca podrían comprender su significado; solamente aquellas personas que conozcan las claves (transmisor y receptor) podrán descifrar el contenido del mensaje transmitido.

La integridad consiste en asegurar que los datos que llegan al receptor no han sido alterados, los algoritmos que permiten cumplir con este objetivo se denominan algoritmos de *hash* y se basan en la obtención de un resumen (secuencia de bits). El destinatario recibe el mensaje y realiza el cálculo del nuevo *hash*, si el *hash* recibido es igual al *hash* calculado entonces el mensaje es auténtico y se asegura que nadie ha cambiado en lo más mínimo el contenido de dicho mensaje.

Finalmente, los servicios de autenticación de emisor y de no repudio se consiguen por medio de la firma digital, la cual se basa en la aplicación de los algoritmos asimétricos sobre el *hash*, lo cifra con la clave privada y cualquier persona que desee conocer la identidad del emisor puede hacerlo usando la clave pública del mismo, de esta manera se garantiza que el emisor es auténtico y además, se cumple con el servicio de no repudio, pues dicho emisor no podrá negar que envió el mensaje.

## 1.1. Objetivos

El objetivo general de este Estudio Técnico es desarrollar una aplicación de uso didáctico para comunicación segura de datos a través de la red.

Los objetivos específicos de este Estudio Técnico son:

- Analizar los conceptos necesarios para el desarrollo del proyecto.
- Diseñar los módulos de la aplicación para el envío seguro de información.
- Implementar una aplicación que integre los algoritmos DES, RSA, MD5 para obtener los cuatro objetivos, como una alternativa de uso didáctico para demostrar el envío seguro de información a través de la red.

## 1.2. Alcance

Con la finalidad de ofrecer una aplicación de uso didáctico, el presente trabajo pretende implementar los siguientes algoritmos: DES, RSA y MD5, los cuales servirán para el envío seguro de información.

Para la implementación de cada algoritmo se utilizará el *software* Matlab para codificar las funciones que implementen los algoritmos, y además, todo lo necesario para implementar la interfaz gráfica de usuario, integrando los cuatro objetivos de seguridad que permitan el envío seguro de información (archivo de texto claro) entre dos computadoras interconectadas a través de la red, como se puede observar en la Figura 1.1.

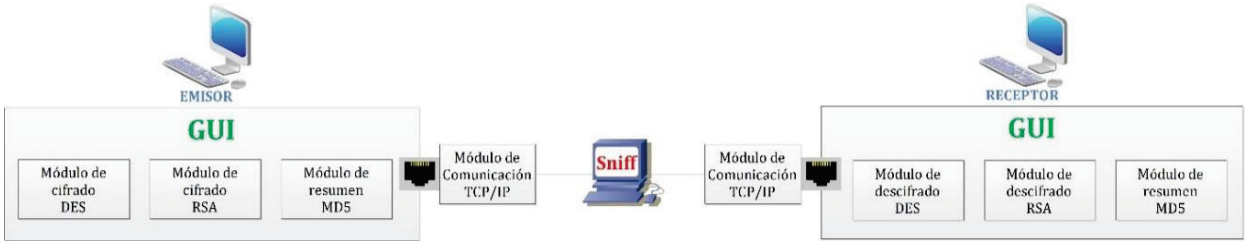


**Figura 1.1.** Diagrama de bloques de la aplicación

En la Figura 1.2 se detalla gráficamente el proceso del sistema completo de seguridad informática, y posteriormente se describe cada etapa.

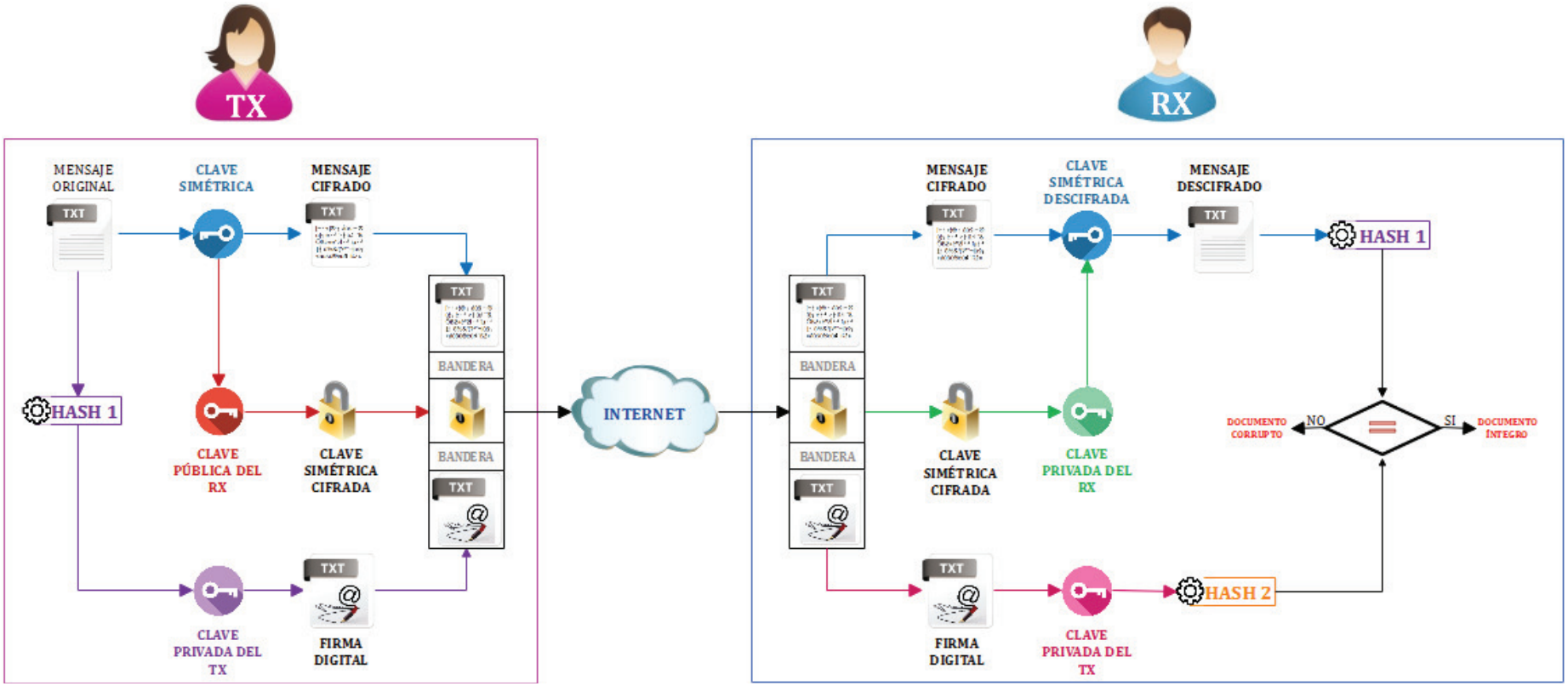


**Figura 1.2.** Diagrama de funcionamiento de la aplicación

La aplicación permitirá recibir como entrada un archivo de texto claro, al que luego se aplicará el algoritmo de cifrado DES empleando una clave simétrica para cifrar el mensaje, este documento cifrado será enviado a través de la red conjuntamente con la firma digital, que es la aplicación del algoritmo RSA al *hash* del documento cifrado simétricamente. Una vez que se reciban ambos documentos, en el receptor se va a realizar el proceso inverso. Por tanto, ya que todo el mundo conoce la clave pública del emisor, será simple recuperar el *hash* generado por dicho transmisor, pero únicamente con la clave privada se podrá descifrar el mensaje. Por otro lado, ya que también llega el mensaje cifrado, se vuelve a aplicar el mismo algoritmo *hash* empleado por el emisor, y este resultado se compararía con el anteriormente obtenido, si ambos son iguales significaría que la firma digital es auténtica.

Para validar que se cumplan los cuatro objetivos de seguridad se comprobará que el texto que envía el emisor se encuentra cifrado haciendo un *sniff* entre el emisor y el receptor, con esto se garantiza la confidencialidad; por otro lado, si los resúmenes son idénticos se garantiza la integridad; mientras que la autenticidad se garantiza con la firma digital junto con la clave privada; y el no repudio se garantiza puesto que tanto el emisor como el receptor no pueden negar que envió/recibió el mensaje, porque tanto el emisor como el receptor tienen pruebas de envío/recepción del mismo.

## 1.1. Marco Teórico

### 1.1.1. Seguridad informática como parte de un sistema de comunicación

En la Figura 1.3 se muestra un sistema de comunicaciones, en el cual, los algoritmos de seguridades constituyen una etapa importante para la transmisión de la información.

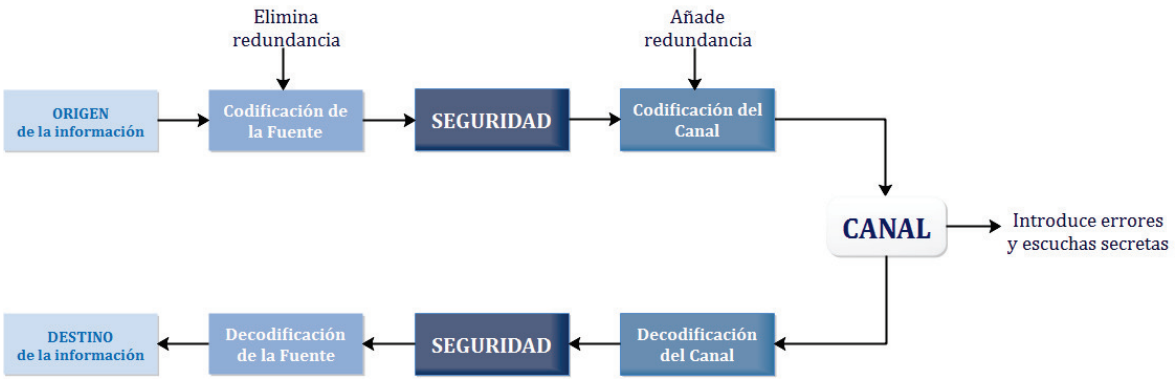


**Figura 1.3.** Etapa de seguridades como parte de un sistema de comunicaciones

Cada etapa cumple una función específica en la comunicación:

- **Origen de la información:** Corresponde a la información que va a ser enviada al destinatario.
- **Codificación de la Fuente:** Elimina toda la información redundante, algunos algoritmos como los códigos de línea o de compresión de la información se encargan de este proceso.
- **Seguridad:** Proporciona los cuatro servicios: confidencialidad, integridad, autenticación y no repudio.
- **Codificación del Canal:** Añade información redundante que permita realizar la corrección de errores, como los códigos convolucionales.
- **Canal:** Corresponde al envío y transporte de la información, dependiendo el tipo de comunicación el canal se considera inseguro si es público.

En el proceso de recepción, los procesos se repiten en orden inverso.

### 1.1.2. Funciones en la seguridad informática

Las funciones o servicios de la seguridad informática permiten que la información se encuentre disponible para los usuarios autorizados y protegidos contra acceso ilícito. En la Figura 1.4 se pueden apreciar dichos servicios y a continuación se detallan cada uno de ellos:


**Figura 1.4.** Objetivos de seguridad informática

#### Confidencialidad

Permite proteger la privacidad de la información, tanto en el acceso a los datos como durante su transferencia mediante el cifrado de la información, de tal manera que solamente los usuarios con autorización legítima consigan acceder e interpretar dicha información [1].

La seguridad de la información de un sistema reside en su clave, por ejemplo, en la Figura 1.5 se puede observar que el emisor utiliza una clave para cifrar el mensaje, la cual debería ser privada, de manera que solamente el destinatario la conozca para poder descifrarlo, así ambos se aseguran que únicamente ellos podrán interpretar el mensaje [2].

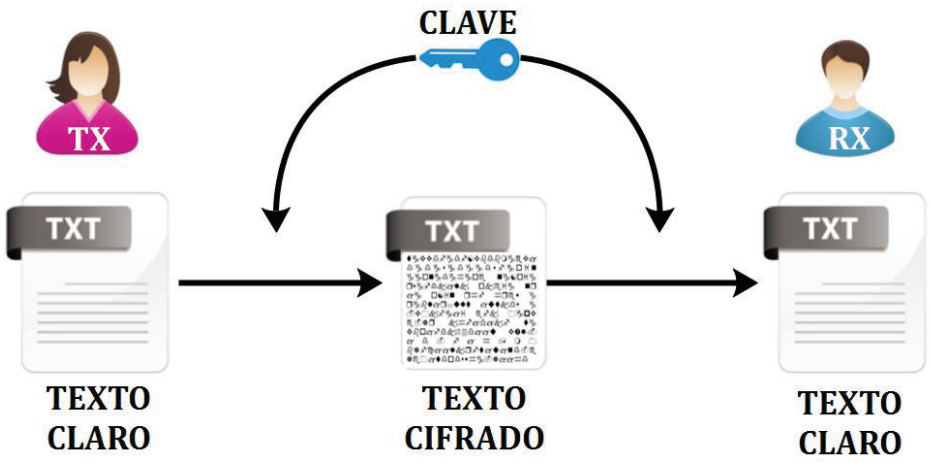


**Figura 1.5.** Servicio de confidencialidad

### Integridad

Es una garantía de que la información almacenada o enviada puede ser modificada o borrada únicamente por el personal autorizado, lo cual certifica que la información que se envía al receptor no ha sido manipulada o alterada y que sea la misma que fue generada y enviada por el emisor. Un ejemplo de violación de la integridad es cuando una persona no autorizada modifica o borra información almacenada o interceptada, ya sea por accidente o malas intenciones [1], tal como se muestra en la Figura 1.6.

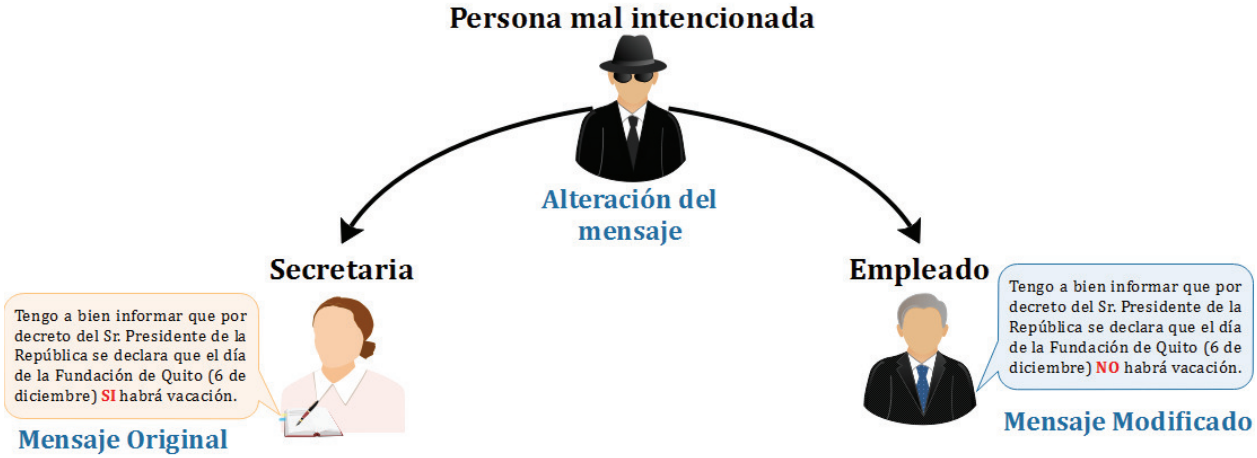


**Figura 1.6.** Ejemplo de integridad

Como se observó en el ejemplo de la Figura 1.6, una persona mal intencionada altera el mensaje original enviado por la secretaria a sus trabajadores, en el cual anuncia a los trabajadores que sí habrá vacación, mientras que el destinatario recibe que no habrá vacación, por lo tanto, el mensaje es diferente del original, ya no es íntegro.

**Autenticación**

Es una manera de asegurar el origen y destino de la información, lo cual garantiza la legitimidad y credibilidad de la persona, verificando que una persona es quien dice ser mediante la comprobación y confirmación de su identidad. Cuando el destinatario recibe el mensaje, él está seguro de la identidad de la persona que lo envió porque únicamente él conoce la clave de cifrado, y no fue una tercera persona, lo cual se conoce como suplantación de identidad. Existen tres maneras de comprobar la autenticidad [3]:

- Algo que el usuario sabe: *password*, clave.
- Algo que el usuario lleva consigo: cédula de identidad, tarjeta magnética.
- Propiedad física: huella dactilar, pupila, voz; o acto involuntario: firma.

Como parte de los sistemas modernos, la autenticidad del emisor se realiza mediante la firma digital. En la Figura 1.7 se muestran dos documentos idénticos, cuyo contenido es un Decreto Presidencial en el cual se anuncia un feriado bancario en todo el país, el primer documento está firmado con autógrafo, mientras que el segundo documento tiene firma digital, la cual tiene la misma validez de una firma manuscrita, pero se considera mucho más segura, pues se genera en base a algoritmos criptográficos modernos.

**No repudio**

También conocido como irrenunciabilidad o irrefutabilidad, se encuentra estrechamente vinculado con la autenticidad de emisor y garantiza probar la participación de las entidades comunicadoras: emisor y receptor, a través de las pruebas de integridad y autenticidad. Un sistema es más seguro si la autenticidad no puede ser refutada después de la comunicación, es decir, que el usuario no esquive su responsabilidad en las acciones desarrolladas durante el intercambio de información. Se pueden distinguir dos tipos [4]:

a) **NRO (No repudio en origen):** Garantiza que el emisor no puede negar el envío del mensaje, ya que el receptor tiene pruebas del envío.

b) **NRR (No repudio en recepción):** El receptor no puede negar la recepción del mensaje, porque el emisor tiene pruebas de recepción del mismo.

**Firma Autógrafa**

Nº228

LENÍN MORENO GARCÉS

PRESIDENTE CONSITUCIONAL DE LA REPÚBLICA

CONSIDERANDO:

Que mediante el Decreto Ejecutivo No. 685 de 11 de marzo de 1999, publicado en el Suplemento del Registro Oficial Nº149 de 16 de marzo de 1999, se declaró el estado de movilización a las instituciones financieras nacionales públicas y privadas, sus entidades off-shore, a las sucursales y agencias de las instituciones financieras extranjeras que operan en el Ecuador, a las compañías administradoras de fondos y fideicomisos, las de arrendamiento mercantil y las emisoras y administradoras de tarjetas de crédito; así como a las personas naturales o jurídicas que mantengan a la fecha deudas y créditos con ellas, que debido al sobreendeudamiento de la deuda externa, la caída del precio del barril de petróleo y desastres naturales irvencibles.

DECRETA:

Que el Banco Central otorgue créditos de estabilidad para antender a personas que se encuentren en proceso de liquidación de activos, en la cual cualquier banca privada podrá entrar en proceso de liquidación.

**En base a este Decreto Ejecutivo, se declara públicamente un feriado bancario de 1 día, con la finalidad de evitar el el retiro masivo de dinero en las instituciones financieras.**

Dado en la ciudad de San Francisco de Quito, Luz de América, a 28 de noviembre de 2017.

PRESIDENTE CONSTITUCIONAL DE LA REPUBLICA

=

Nº228

LENÍN MORENO GARCÉS

PRESIDENTE CONSITUCIONAL DE LA REPÚBLICA

CONSIDERANDO:

Que mediante el Decreto Ejecutivo No. 685 de 11 de marzo de 1999, publicado en el Suplemento del Registro Oficial Nº149 de 16 de marzo de 1999, se declaró el estado de movilización a las instituciones financieras nacionales públicas y privadas, sus entidades off-shore, a las sucursales y agencias de las instituciones financieras extranjeras que operan en el Ecuador, a las compañías administradoras de fondos y fideicomisos, las de arrendamiento mercantil y las emisoras y administradoras de tarjetas de crédito; así como a las personas naturales o jurídicas que mantengan a la fecha deudas y créditos con ellas, que debido al sobreendeudamiento de la deuda externa, la caída del precio del barril de petróleo y desastres naturales irvencibles.

DECRETA:

Que el Banco Central otorgue créditos de estabilidad para antender a personas que se encuentren en proceso de liquidación de activos, en la cual cualquier banca privada podrá entrar en proceso de liquidación.

**En base a este Decreto Ejecutivo, se declara públicamente un feriado bancario de 1 día, con la finalidad de evitar el el retiro masivo de dinero en las instituciones financieras.**

Dado en la ciudad de San Francisco de Quito, Luz de América, a 28 de noviembre de 2017.

(Este documento ha sido firmado electrónicamente por:
LENÍN MORENO GARCÉS - **PRESIDENTE CONSTITUCIONAL DE LA REPÚBLICA**

En la dirección https://documento.electronico/verifica.doc puede ser comprobada la autenticidad de esta copia, mediante el documento electrónico siguiente:
9F3A 4EF2 903D 2D29 DCA1

**Firma Digital**

**Figura 1.7.** Ejemplo del servicio de autenticación de emisor

Este servicio es muy importante para realizar transacciones *online*, debido a la seguridad brindada durante la comunicación. Un ejemplo de este servicio es retirar dinero de la cuenta de ahorros desde un cajero electrónico o banca en línea, quien posteriormente no puede negar que lo hizo [4]. Existen algunos mecanismos de protección de este servicio, entre los principales se encuentran:

- **Servicios de confirmación:** Garantiza el uso seguro de los certificados digitales para la autenticación de personas. Adecuado para servicios electrónicos que involucren el uso de tarjeta de identificación, identificación móvil o un sello electrónico [5].
- **Registros temporales:** Servicio criptográfico de marca de tiempo que permite a las organizaciones o individuos aplicar sellos digitales inviolables a la información digital, proporcionando pruebas a largo plazo de que la información existía en un punto particular de tiempo y no ha sido alterada desde entonces, manteniendo el contenido del archivo en secreto y considerado imposible de falsificar [5].
- **Notarización:** Requiere la certificación notarial de un notario público electrónico para garantizar que el documento no ha sido alterado desde el día que fue creado; además, que es único, identificando cada documento con huellas digitales únicas, generando funciones resumen unidireccionales, sin emplear claves [6].
- **Firmas digitales:** Es información relacionada con el mensaje, adjunta al mismo y que se emplea para autenticar la identidad del emisor. [7].

### 1.1.3. Servicio de Confidencialidad

**Problema de la confidencialidad de la información:** En un intercambio de comunicación entre emisor y receptor, el envío de información a través del canal de comunicaciones puede ser muy inseguro y dicha información puede ser fácilmente interceptada, sobre todo cuando se envían mensajes en texto claro, como se indica en la Figura 1.8.


**Figura 1.8.** Ejemplo de envío/recepción de texto claro por un canal inseguro

### Comunicación TCP/IP

TCP/IP (*Transmission Control Protocol*/*Internet Protocol*) es un conjunto de protocolos que permiten la comunicación entre dispositivos que se encuentren conectados en red. Según la arquitectura TCP/IP, en cada capa se agrega información al paquete de datos, conocido como cabecera, que incluye, entre otros parámetros de control, la dirección IP origen, destino y la información.

### Comunicación TCP/IP en Matlab

En Matlab, TCP/IP utiliza comunicación a través de *sockets*[1] y permite conectarse a servidores remotos para leer y escribir datos. El procedimiento que se debe realizar para el envío de información es el siguiente [8]:

1. Crear la conexión TCP/IP en un servidor,
2. Configurar la conexión si es necesario.
3. Realizar operaciones de lectura y escritura.
4. Borrar y cerrar la conexión.

En la ayuda de Matlab se puede encontrar el comando `tcpclient`, escribiendo `help tcpclient` en la ventana de comandos.

Para comunicarse a través de TCP / IP, primero se debe crear un objeto TCP / IP usando la función `tcpclient`. La sintaxis es:

```
<objname> = tcpclient(Dirección IP, Puerto)
```

La dirección puede ser un nombre de *host* remoto o una dirección IP remota, el puerto debe ser un entero positivo entre 1y 65535.

Para establecer la comunicación entre cliente y servidor permitiendo la transferencia de información binaria y ASCII empleando *Server Sockets* se debe usar la propiedad `NetworkRole`. Se puede usar esta conexión para comunicarse un servidor con un cliente, este comando permite una sola conexión remota, por lo tanto utiliza dos valores `client` y `server`.

En la Tabla 1.1 se describen algunas de las funciones asociadas a la comunicación TCP/IP en Matlab [9].

[1] *Sockets:* Es la constitución de una dirección IP asociada a un puerto. Permiten comunicaciones orientadas a la conexión y no orientadas a la conexión.

**Tabla 1.1.** Funciones de comunicación TCP/IP en Matlab

| Función Matlab | Descripción |
|---|---|
| `fopen` | Conectar el objeto de interfaz |
| `fclose` | Desconectar objeto de interfaz |
| `fread` | Leer datos binarios |
| `fwrite` | Escribir datos binarios |
| `fscanf` | Leer datos y formatearlos como texto |
| `fprintf` | Escribir texto |
| `readasync` | Leer datos de forma asincrónica |
| `stopasync` | Detener operaciones de lectura y escritura asincrónicas |

Se debe considerar que mientras un *socket* en el lado del servidor está esperando una conexión después de `fopen`, el subproceso de Matlab está bloqueado. Para detener `fopen` o dejar de escuchar conexiones, y restablecer el uso de Matlab, se debe escribir Ctrl + C en la línea de comandos de Matlab.

### Captura de paquetes

Como ejemplo se ha considerado enviar de origen a destino un archivo cuyo contenido se encuentra en texto claro, el cual se muestra en la Figura 1.9.

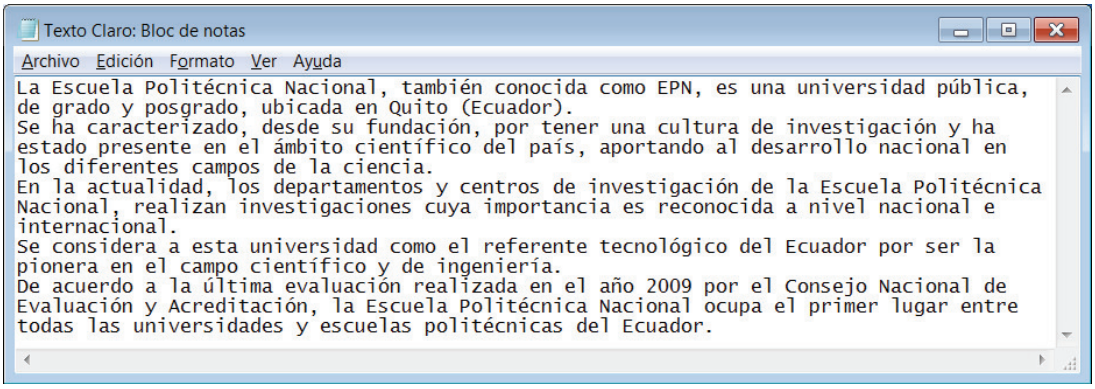


**Figura 1.9.** Información en texto claro a ser enviada de origen a destino

Al enviar un mensaje en texto claro de origen a destino por medio de TCP/IP puede ser interceptado por una entidad externa no autorizada, la cual puede estar capturando los paquetes utilizando cualquier mecanismo de *sniffer*[2]. Empleando el *software* Wireshark se puede capturar el intercambio de paquetes durante la comunicación.

[2] *Sniffer*: Software empleado para escuchar toda la información que pasa a través de una red.

En la Figura 1.10 se puede observar que el programa informático puede escuchar toda la información transmitida, desde la cabecera de la trama Ethernet, paquete IP, datagrama TCP hasta la información de los datos transmitidos.

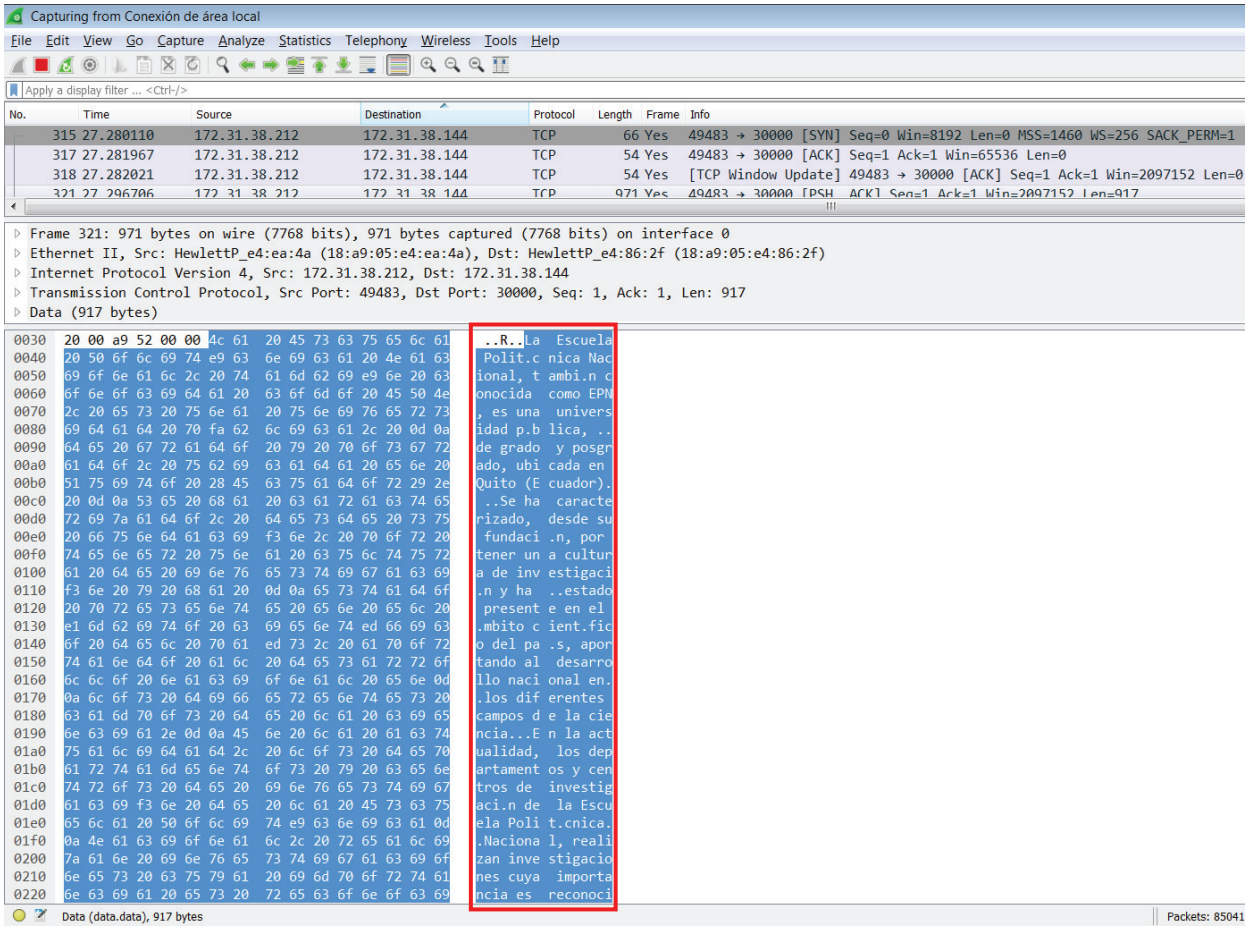


**Figura 1.10.** Captura de paquetes con el *software* Wireshark

Tal como se observa en la Figura 1.10 los datos se muestran tanto en formato ASCII como en hexadecimal; es posible que los caracteres especiales del mensaje se muestren en formato ASCII, sin embargo, se los puede convertir empleando el valor hexadecimal del mismo.

### Solución al problema de la confidencialidad de la información

Para garantizar la confidencialidad de la información, se emplean mecanismos de cifrado.

El procedimiento se describe gráficamente en la Figura 1.11, cuyos elementos agrupados se consideran un criptosistema.

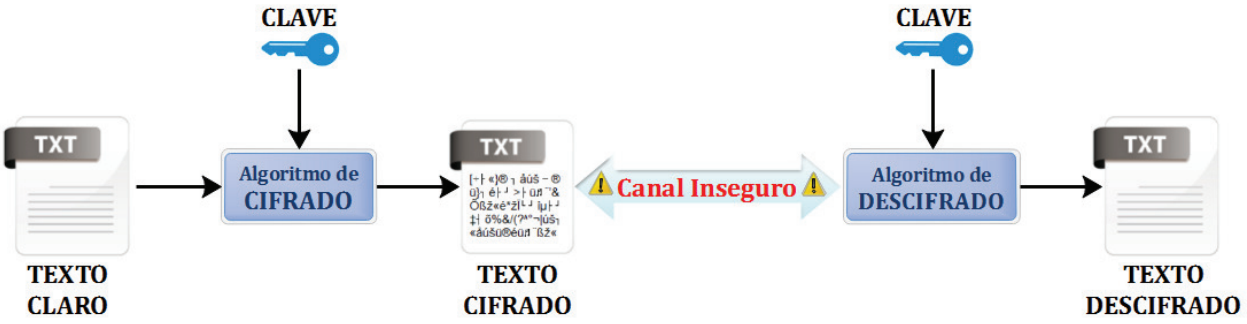


**Figura 1.11.** Criptosistema y elementos constitutivos

A continuación se describe la terminología de la seguridad informática:

- **Criptosistema:** Es un concepto que abarca todos elementos para cifrar la información, el cual se compone de un texto claro, un algoritmo, un texto cifrado, las claves y el texto descifrado. La Figura 1.11 describe de forma gráfica un criptosistema y sus elementos constitutivos:
    - **Texto claro**: Es la información antes de ser cifrada. Si bien se le llama "texto", también pueden ser datos binarios como imágenes, voz digitalizada, video, ejecutables, etc.
    - **Cifrado**: Es el proceso de transformar la información de un texto claro, de tal forma que únicamente pueda ser leída por el receptor autorizado utilizando una clave de cifrado.
    - **Texto cifrado**: Es el texto claro una vez aplicado el proceso de cifrado.
    - **Clave de cifrado**: Es un conjunto de caracteres que se utilizan para cifrar el texto claro y sin el cual no es posible descifrarlo. El mensaje cifrado es posible descifrarlo empleando la misma clave que se usó para cifrarlo o una que tenga alguna relación matemática. Idealmente la clave se debe mantener en secreto y únicamente debe ser conocida por las entidades comunicadoras, excepto la clave pública que la conocen todos.
    - **Algoritmo de cifrado**: Es la técnica matemática que utiliza una o varias claves de cifrado para transformar el texto claro en texto cifrado y viceversa.
    - **Descifrado**: Es el proceso contrario al cifrado. Para poder descifrar la información el receptor realiza el proceso de descifrado junto con la clave para acceder al contenido transmitido.

Históricamente, el servicio de confidencialidad era el único sobre el cual se centraba la seguridad de la información, pero debido a la necesidad de aumentar el nivel de confianza que garantice la transmisión segura de información se desarrollaron varios mecanismos modernos, basados en fundamentos matemáticos, convirtiendo el arte de ocultar la información en ciencia.

### Criptología

El origen de la palabra criptología proviene de los vocablos griegos *krypto* (ocultar) y *logos* (estudio). Es la disciplina que se dedica a estudiar la escritura secreta, desarrollando y mejorando algoritmos para proteger la información [10]. Se divide en dos grandes disciplinas:

1. **Criptografía**

   El origen de la palabra criptografía proviene de los vocablos griegos *κρύπτος* (ocultar) y *γραφη* (escribir), cuyo significado es el arte de la escritura secreta; es decir, es la ciencia que hace uso de un conjunto de métodos y herramientas matemáticas con el objetivo de transformar la información original en texto cifrado, usando para ello dos o más claves, evitando que sea legible para observadores no autorizados y solamente quien conozca la llave pueda conocer el mensaje original.

   El objetivo básico de la criptografía es brindar seguridad a la información, incluyendo la autenticidad de las entidades que se comunican, la confidencialidad e integridad de los mensajes transmitidos y su no repudio [11].

   **Historia de la criptografía:** A lo largo de la historia siempre ha existido la necesidad de proteger la información; así la criptografía tiene sus raíces en la antigua Roma, en el Imperio de César quien creó una representación de criptografía, denominada Cifra de César, utilizada para cifrar mensajes a sus comandantes en el campo. Posteriormente, en la Guerra de los Cien Años en Francia e Inglaterra se introdujo la Cifra Vigenère, conocida como un cifrado polialfabético. Luego el matemático Babbage demostró que dicho cifrado era vulnerable a las técnicas de análisis estadístico [12].

   Años despés, durante la Segunda Guerra Mundial los alemanes utilizaron dispositivos electromecánicos para cifrar mensajes, como la máquina Enigma. Sin embargo, los polacos y británicos lograron romper el cifrado de la máquina Enigma y descifraron gran cantidad de mensajes, lo que sirvió de gran ayuda para los Aliados durante la guerra [12].

En la década de 1970 nació la informática moderna y surgió la necesidad de cifrar el tráfico de datos. En 1976 el Gobierno de Estados Unidos estableció la primera norma de cifrado de datos *Data Encryption Standard* (DES) inventado por IBM y sus posteriores sucesores [12].

Hoy en día, debido al creciente avance tecnológico, la criptografía implica más que un uso militar o político, pues se considera un eje de seguridad informática imprescindible en la actividad diaria de las personas y empresas.

La criptografía moderna se considera una rama de las matemáticas estrechamente vinculada con la informática, la teoría de la información y la seguridad informática. El objetivo básico de la criptografía es brindar seguridad a la información, incluyendo la autenticidad de las entidades que se comunican, la confidencialidad e integridad de los mensajes transmitidos y su no repudio [11].

2. **Criptoanálisis**

   Es la disciplina que implementa y emplea herramientas para probar los algoritmos desarrollados por los criptólogos sin conocer la llave de cifrado, con la finalidad de descubrir las vulnerabilidades de los algoritmos, y así descifrar la información.

Estas dos disciplinas tratan constantemente de mantenerse por delante una de la otra, constituyéndose en un proceso cíclico: mientras los criptógrafos inventan códigos secretos inteligentes, los criptoanalistas descubren vulnerabilidades en los algoritmos para intentar romper estos códigos [11].

A pesar de la robustez de la criptografía ningún método de cifrado se puede considerar irrompible porque siempre hay alguien con la capacidad o los recursos necesarios para romper el código [12].

### Criptografía de clave simétrica

También conocida como cifrado de clave secreta o clave única. Es el cifrado convencional que se basa en algoritmos criptográficos que emplean una única clave para cifrar y descifrar el mensaje, lo que se denomina clave simétrica.

La seguridad de este tipo de cifrado se basa en mantener secreta la clave, de tal manera que solamente el emisor y el receptor deben conocerla para poder cifrar y descifrar el mensaje, lo cual garantiza la confidencialidad de la información.

En la Figura 1.12 se observa que el transmisor cifra la información con la clave simétrica, la cual debe ser enviada a través del canal, posteriormente el receptor descifra la información recibida con la misma clave que empleó el transmisor para cifrar la información, lo cual se convierte en un problema de distribución de claves y vuelve inseguro al canal de comunicación.

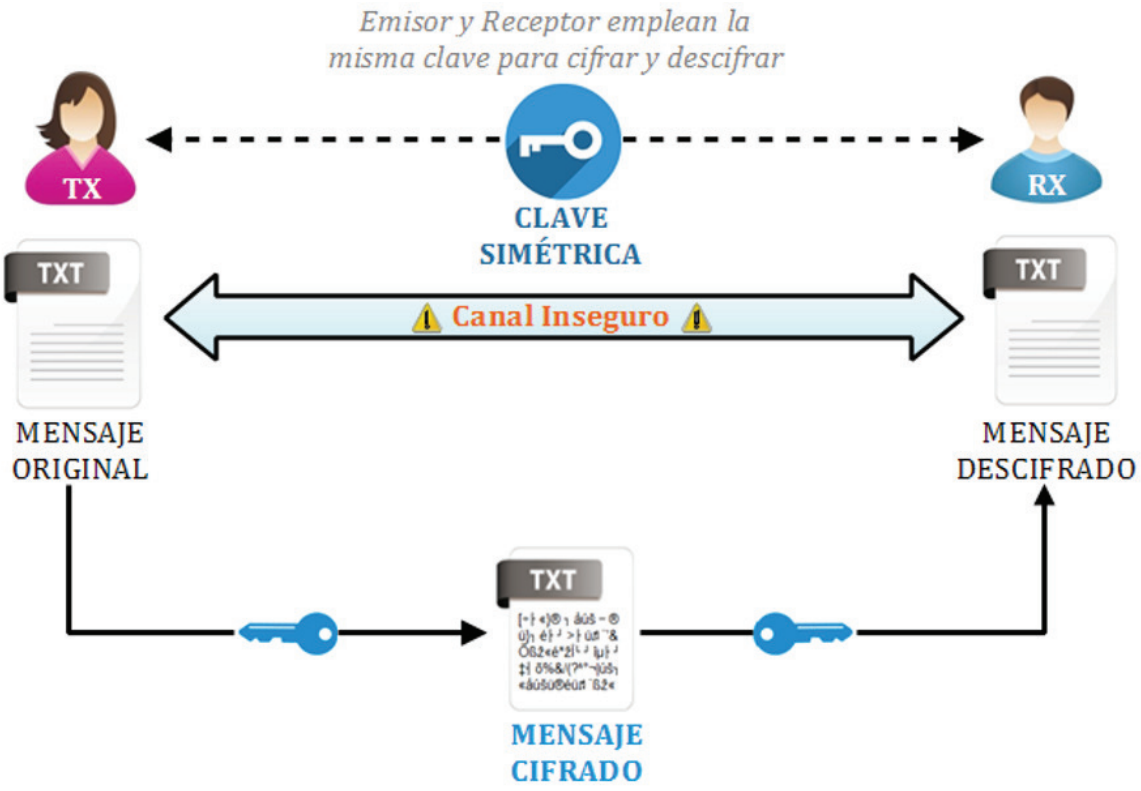


**Figura 1.12** Cifrado Simétrico

### Algoritmos de cifrado simétrico

Existen una gran cantidad de algoritmos de cifrado, sin embargo, a continuación se describen los algoritmos que han sido considerados estándar y son comúnmente utilizados:

- **DES (*Data Encription Standard*)**

  Fue creado por IBM[3], liderado por Horst Feistel en 1973 como un proyecto de investigación criptográfica con el fin de realizar procesos de cifrado sobre *hardware* [13].

[3] IBM (*International Business Machines*): Empresa estadounidense que produce y comercializa hardware y software informático.

En el año 1977 fue declarado como el estándar X9.52 por la ANSI[4] y en 1994 el NIST[5] reconoció a DES para su uso en aplicaciones que manejen información secreta, pero la NSA[6] realizó modificaciones antes de permitir su uso, reduciendo el tamaño de la clave de 128 a 64 bits.

En la Figura 1.13 se puede observar el diagrama de bloques del algoritmo.

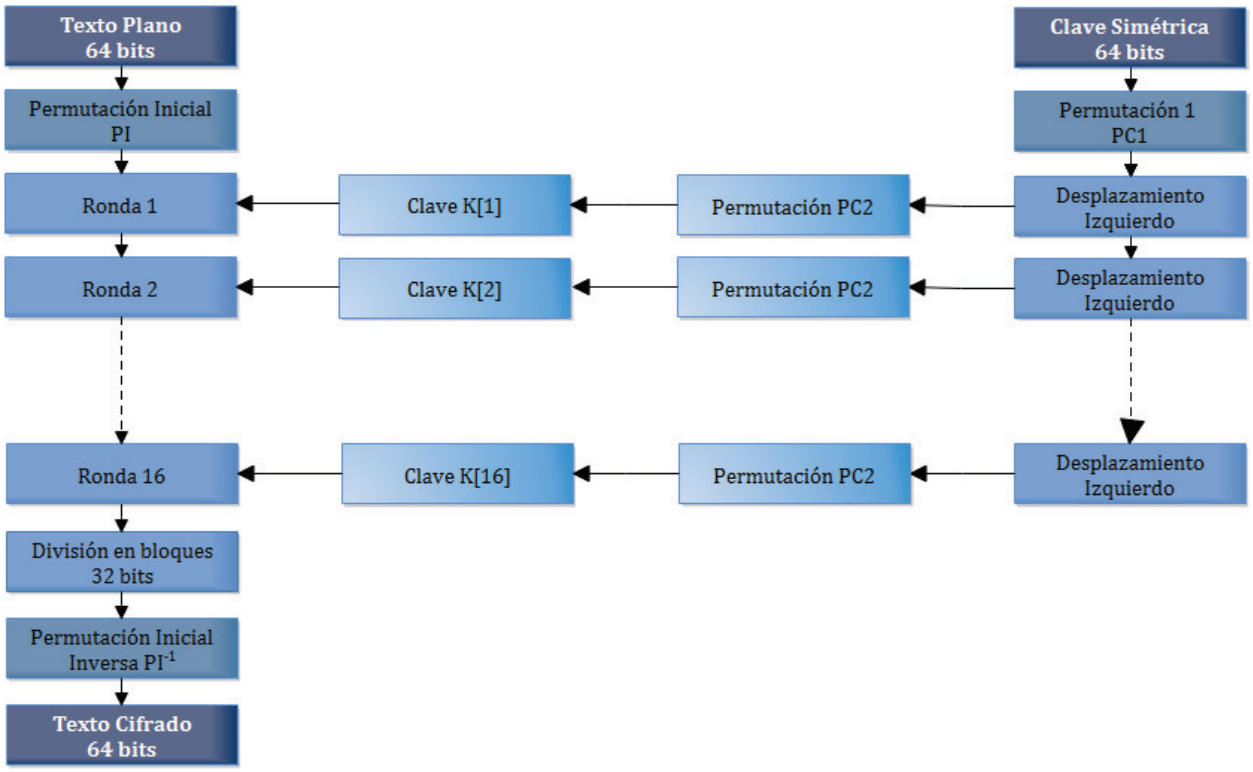


**Figura 1.13.** Diagrama de bloque del algoritmo DES

Para el cifrado DES, el texto plano se divide en bloques de 64 bits (8 Bytes), añadiendo bits de relleno en el último bloque si es necesario. Emplea claves del mismo tamaño y genera un texto cifrado de igual longitud del mensaje original. La estructura general del algoritmo tiene 16 etapas idénticas de procesamiento, denominadas rondas. A cada bloque se le aplica el algoritmo DES, hasta completar de cifrar todo el texto plano.

[4] ANSI (*American National Standards Institute*): Organización que controla el desarrollo de estándares en Estados Unidos.
[5] NIST (*National Institute of Standards and Technology*): Organismo de normalización que se encarga de la creación y mantenimiento de normas de jure, como FIPS (*Federal Information Processing Standard*), que incluye la firma digital estándar y AES.
[6] NSA (*National Security Agency*): Agencia de inteligencia norteamericana que se encarga de la seguridad de la información.

**Ataques al algoritmo DES:** Poco después de que se propusiera DES, dos criptoanalistas argumentaron lo siguiente [14]:

1. El tamaño de la clave es demasiado pequeño, es decir, el algoritmo es vulnerable a los ataques de fuerza bruta.
2. Los criterios de diseño de las cajas S eran secretos y pudo haber existido un ataque analítico que explote las propiedades matemáticas de las cajas S.

A partir de 1997, la empresa RSA Security organizó varios desafíos para romper el algoritmo DES denominados Desafíos DES (*DES Challenge*). La Tabla 1.2 proporciona una descripción general de los ataques propuestos y realizados contra DES en las últimas tres décadas [14].

**Tabla 1.2.** Historia de ataques al algoritmo DES

| Fecha | Ataques implementados |
|---|---|
| 1977 | W. Diffie y M. Hellman proponen la estimación de costos para la máquina de búsqueda de claves. |
| 1990 | E. Biham y A. Shamir proponen criptoanálisis diferencial, que requiere $2^{47}$ textos planos seleccionados. |
| 1993 | M. Wiener propone un diseño de hardware detallado para la máquina de búsqueda de claves con un tiempo promedio de búsqueda de 36 h y un costo estimado de $ 1 000 000. |
| 1993 | M. Matsui propone criptoanálisis lineal, que requiere $2^{43}$ textos cifrados elegidos. |
| Junio 1997 | DES Challenge I rompió DES usando fuerza bruta a través de un sistema distribuido en Internet; tomó 4.5 meses. |
| Febrero 1998 | DES Challenge II-1 rompió DES usando fuerza bruta; a través de un sistema distribuido en Internet; tomó 39 días. |
| Julio 1998 | DES Challenge II-2 rompió DES usando fuerza bruta; *Electronic Frontier Foundation* (EFF) construyó la máquina de búsqueda de claves llamada *Deep Crack* por aproximadamente $ 250,000. El ataque duró 56 h. |
| Enero 1999 | DES Challenge III rompió DES usando fuerza bruta; con aproximadamente 100 000 PCs en un sistema distribuido en Internet combinado con *Deep Crack* en un tiempo total de búsqueda de 22 horas 15 minutos. |
| Abril 2006 | Las universidades de Bochum y Kiel construyeron la máquina de búsqueda de claves COPACOBANA basada en FPGA de bajo costo por aproximadamente $ 10,000. El tiempo promedio de búsqueda es de 7 días. |

Teniendo en cuenta que una clave de 56 bits tiene cerca de 72 mil billones de combinaciones diferentes de números, al parecer DES parecía ser seguro durante mucho tiempo. El aumento en la potencia de procesamiento de las computadoras pronto permitió romper el cifrado DES.

Lo cual afirma que DES se puede romper con relativa facilidad con un exhaustivo ataque de búsqueda de clave, por lo cual DES ya no es adecuado para la mayoría de las aplicaciones que requieran un alto nivel de seguridad, como información militar, gubernamental o bancaria.

A pesar del criptoanálisis muy intensivo durante la vida útil de los DES, los ataques analíticos actuales no son muy eficientes, pues se necesitaría de una gran cantidad de dinero y de PC's distribuidas a lo largo del mundo para descifrar un mensaje, por lo que, fue muy utilizado hasta el año 1999, a partir de lo cual se empezó a emplear 3-DES y posteriormente en octubre fue reemplazado por el algoritmo belga AES o también llamado Rijndael [15].

- **Triple DES / 3DES / TDES**

  Se necesitaba un nuevo estándar para el cifrado de la información, puesto que DES ya no era la solución debido a la gran cantidad de ataques a los que estaba expuesto, por lo cual, muchos años antes de que se rompiera el cifrado DES, los criptógrafos comenzaron a trabajar en su reemplazo. Debido a que gran cantidad de *software* y *hardware* ya se habían codificado con DES, se realizaron cambios en un algoritmo que ya estaba en uso, en lugar de crear uno completamente nuevo. *Triple Data Encryption Standard* (3DES) sirvió como una solución rápida para las vulnerabilidades de DES [15].

  Como su nombre lo indica, Triple DES realiza tres veces el algoritmo DES, pero en lugar de utilizar una clave de 56 bits de la tabla de claves, 3DES realiza el cálculo de DES original tres veces con diferentes claves. Este cálculo más complejo de datos hace que 3DES sea mucho más fuerte que DES, sin embargo, esta mejora tuvo un precio en rendimiento: 3DES tarda más en encriptar y descifrar datos que su predecesor, pero ese es un pequeño precio a pagar por mejorar la seguridad [17].

  La longitud de la clave es de 168 bits [16], la cual se fragmenta en tres partes, tal como se indica en la Figura 1.14.

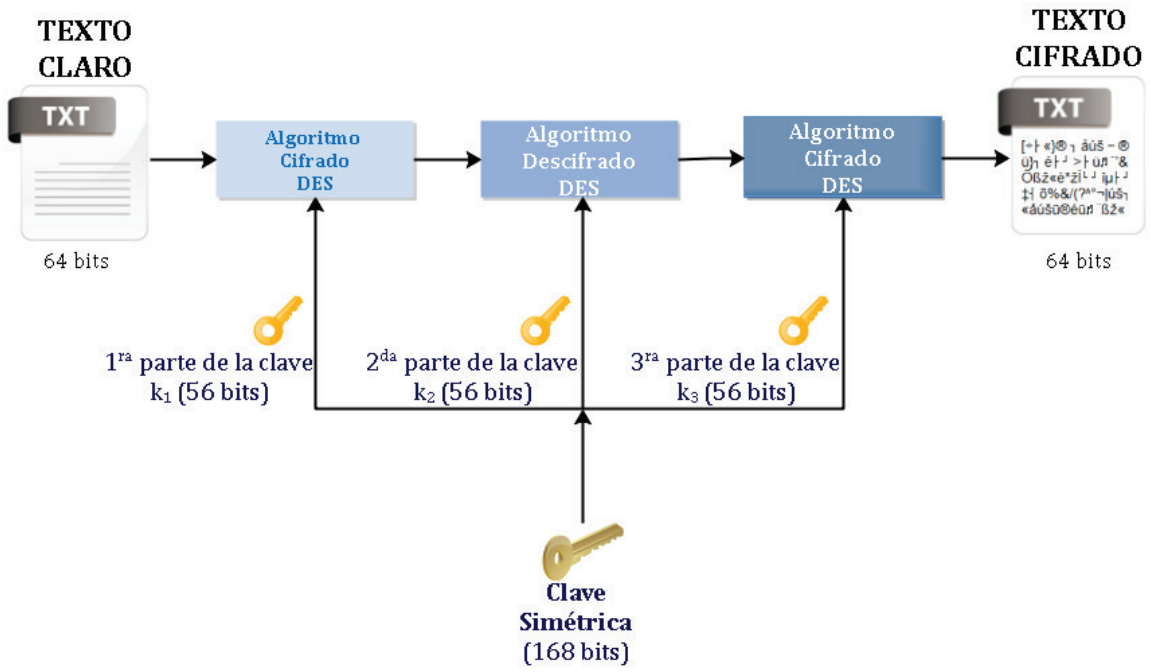


**Figura 1.14.** Diagrama de bloques del algoritmo simétrico 3DES

- **AES (*Advanced Encryption Standard*)**

  En 1997, NIST presentó otra solicitud al público para un nuevo estándar de cifrado de bloque simétrico capaz de soportar claves de 128, 192 y 256 bits, hubieron cinco finalistas, pero en octubre del año 2000 NIST eligió Rijndael como nuevo estándar de cifrado del siglo XXI, desarrollado por Joan Daemen y Vincent Rijmen, debido a sus mejoras en seguridad, eficiencia, rendimiento y flexibilidad. Esto se debe principalmente a su gran capacidad de cifrado de grandes gamas de bloques de texto y tamaños de clave de 128, 192 y 256 bits. Por lo tanto, es más fuerte y más rápido que 3DES y consume menos procesamiento y memoria [15].

  AES-256 es el estándar de cifrado simétrico más ampliamente utilizado debido a que es uno de los únicos algoritmos comerciales validados con la suficiente protección para proteger la información [15]. NIST recomienda su uso hasta el año 2020.

  En la Figura 1.15 se describe gráficamente el diagrama de bloques del algoritmo de AES.

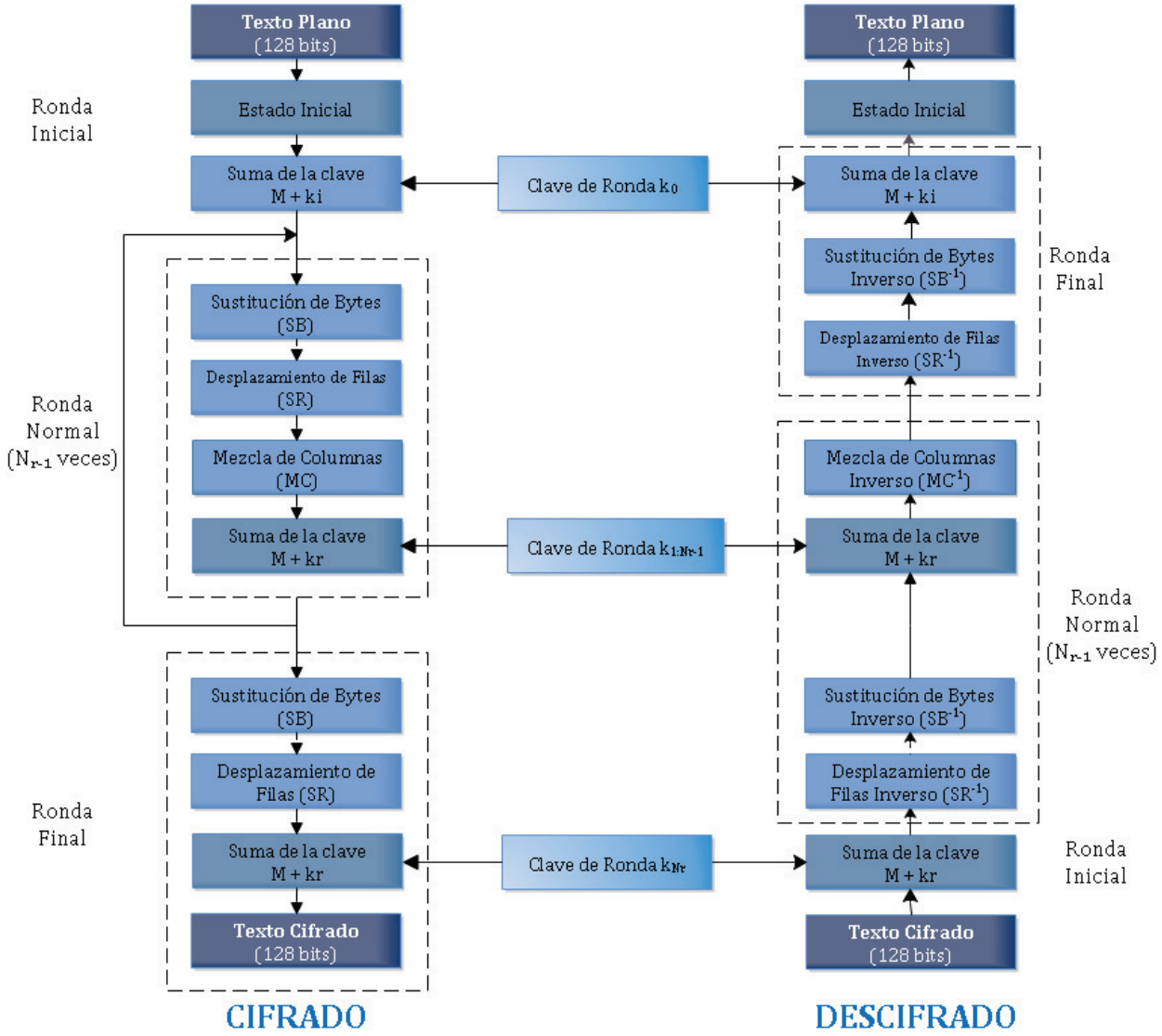


**Figura 1.15.** Diagrama de bloques del algoritmo simétrico AES

En la Tabla 1.3 se indica el tiempo estimado requerido para encontrar la clave empleando una búsqueda exhaustiva.

**Tabla 1.3.** Tiempo promedio requerido para búsqueda exhaustiva de claves

| Tamaño de la clave (bits) | Número de combinaciones de clave posibles | Tiempo promedio requerido para encontrar la clave |
|---|---|---|
| 32 | $2^{32}=4.3x10^{9}$ | 2.5 milisegundos |
| 56 | $2^{56}=7.2x10^{16}$ | 10 horas |
| 128 | $2^{128}=3.4x10^{38}$ | 5.4 $x10^{18}$ años |
| 168 | $2^{168}=3.7x10^{50}$ | 5.9 $x10^{30}$ años |

En la Tabla 1.3 se puede observar que mientras mayor sea el tamaño de la clave, el algoritmo se considera más seguro, es por este motivo que hasta el momento AES no ha presentado ninguna vulnerabilidad.

En la Tabla 1.4 se realiza una comparación de los algoritmos de cifrado simétricos.

**Tabla 1.4.** Comparación con los algoritmos de cifrado simétricos

| Algoritmo | Tamaño de la clave (bits) | Tamaño del bloque de datos (bits) | Número de iteraciones o rondas |
|---|---|---|---|
| DES | 56 | 64 | 16 |
| 3DES | 112 | 64 | 48 |
| AES | 128 | 128 | 10 |
| | 192 | | 12 |
| | 256 | | 14 |

**Distribución de clave simétrica:** Para garantizar la confidencialidad de la información la clave simétrica debe mantenerse en total secreto, almacenándose en un lugar seguro y que sea de fácil identificación.

**Ventajas:**

- La misma clave se emplea para cifrar y descifrar el mensaje.
- El cifrado simétrico es seguro si la clave no se envía en texto claro al receptor.
- El texto cifrado es compacto, mantiene el tamaño del texto original.
- Consume pocos recursos computacionales, por lo cual es rápido.

**Desventajas:**

- La distribución de claves es vulnerable a interceptación, pues la única forma de transferir la clave es en texto claro.
- Al transferir la clave usando dispositivos de almacenamiento el emisor y el receptor deberían reunirse para intercambiarla, lo cual es un gran problema si se encuentran geográficamente a una distancia considerable uno del otro.
- Para que el intercambio de información se realice, el emisor y el receptor deben establecer la conexión previamente.

- Al emplear la misma clave para cifrar y descifrar la información, si un atacante intercepta esta información es posible descifrar el mensaje, lo cual representa un alto riesgo de violación informática.
- La misión del emisor y receptor es mantener la clave en secreto. Si cae en manos equivocadas ya no se podría considerar que la comunicación es segura y se debe generar una nueva clave.
- Para mejorar el nivel de seguridad se debe generar una clave por cada comunicación a establecer, por lo cual no es escalable para poblaciones con alta densidad de personas: para que n personas se comuniquen entre sí se requieren $n(n-1)/2$ claves.

Una comunicación segura requiere un canal seguro para la distribución de claves, lo cual resulta difícil de garantizar cuando el número de usuarios es considerable. La solución para este problema es usar un sistema criptográfico basado en algoritmos asimétricos para cifrar la información.

**Criptografía de clave asimétrica**

Se lo conoce también como cifrado de clave pública, su nombre se debe a que cualquier usuario puede conocer esta clave pues se encuentra disponible para todo el mundo. La seguridad de este tipo de cifrado se basa en mantener secreta la clave privada.

Cada usuario tiene un par de claves para cifrar y descifrar el mensaje, como se observa en la Figura 1.16, las cuales se obtienen empleando matemática compleja, siendo prácticamente imposible conocer una clave a partir de la otra.

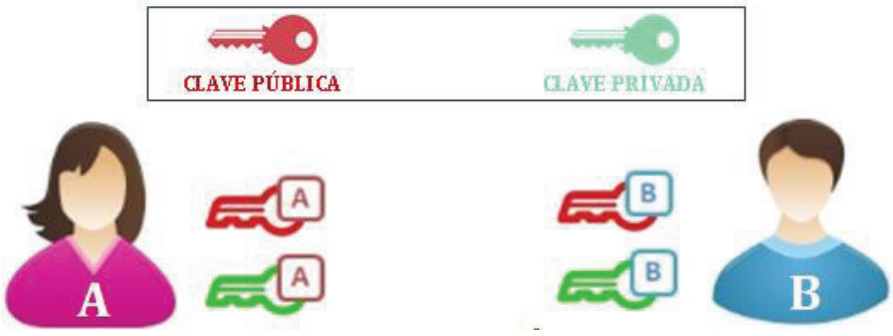


**Figura 1.16.** Cifrado Asimétrico

Cada usuario tiene dos claves y estas pueden emplearse de dos modos diferentes: a) modo de confidencialidad y b) modo de autenticación, los cuales se describen a continuación:

a) **Modo de confidencialidad:** Si el emisor cifra un mensaje con la clave pública del receptor, solamente éste será capaz de descifrarlo con su correspondiente clave privada, como se observa en la Figura 1.17.

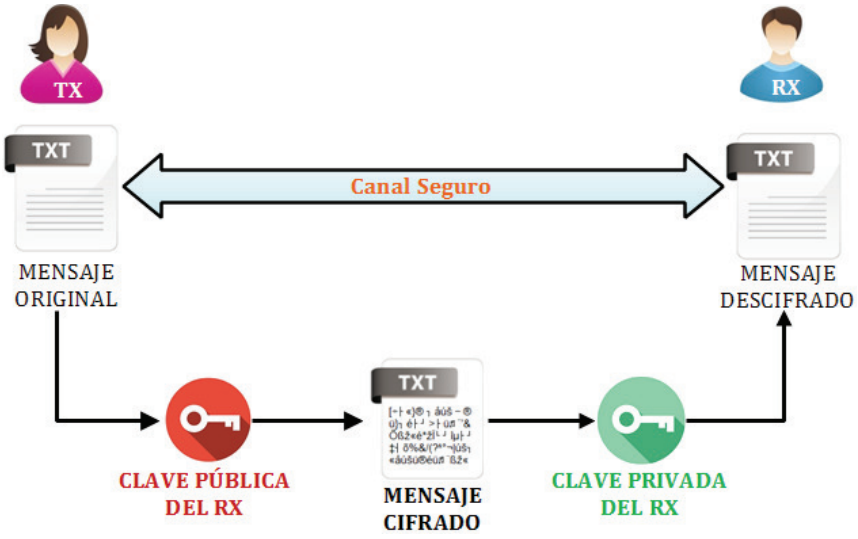


**Figura 1.17.** Cifrado Asimétrico: Modo de Confidencialidad

Si alguien intercepta el mensaje no podrá entender su contenido, solamente el receptor podrá descifrarlo ya que solamente él conoce la clave privada correspondiente.

b) **Modo de autenticación:** Como se observa en la Figura 1.18, si el emisor cifra un mensaje con su clave privada, cualquiera podrá descifrarla empleando su correspondiente clave pública, la cual se encuentra a disposición de todo el mundo.

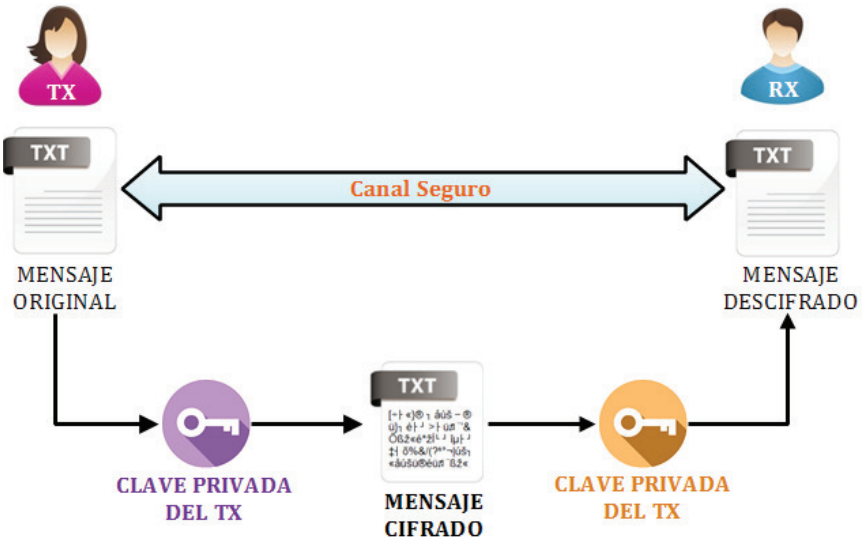


**Figura 1.18.** Cifrado Asimétrico: Modo de Autenticación

Si el emisor desea asegurar al receptor que él es la persona que envía el mensaje (autenticación), puede cifrar el mensaje con su clave privada. Y en el destino, el receptor debe descifrar con la clave pública del transmisor, si el texto se descifra correctamente se garantiza la identidad del emisor. Este tipo de cifrado además de autenticación proporciona no repudio, pues el emisor nunca podrá decir que él no envió el mensaje [15].

**Algoritmos de cifrado asimétrico**

Existen diferentes maneras de cifrar un mensaje con algoritmos asimétricos, a continuación, se describen los algoritmos más utilizados:

- **Diffie-Hellman**

  Este algoritmo fue desarrollado en 1976 por Whitfield Diffie y Martin Hellman, creadores del concepto de clave pública y privada [15]. El algoritmo Diffie-Hellman no es un algoritmo de cifrado, se usa para un intercambio seguro de claves solamente para el transporte de claves secretas.

  Permite a dos partes establecer una clave secreta compartida sobre un canal de comunicación inseguro. Los mensajes cifrados por una parte pueden ser descifrados solo por la otra parte que posee la clave secreta [16]. Los siguientes pasos están involucrados en el intercambio de clave [16]:

  - Las dos partes acuerdan dos números: un número primo grande y un número entero pequeño.
  - Ambas partes generan por separado otro número, equivalente a una clave privada, que se mantiene en secreto, y hacen cálculos que involucran a la clave privada y los números previamente acordados. El resultado del cálculo (la clave pública) se envía a la otra parte.
  - Las dos partes intercambian sus claves públicas, cada parte luego hace otro cálculo usando su clave privada y la clave pública para producir otra clave conocida como la clave de sesión S, la cual es calculada por cada parte de la comunicación. La clave de sesión se puede usar como clave secreta para mejorar el cifrado. Un tercero no puede descifrar el mensaje sin conocer la clave secreta.

  El procedimiento antes descrito se indica en el diagrama de bloques de la Figura 1.19.

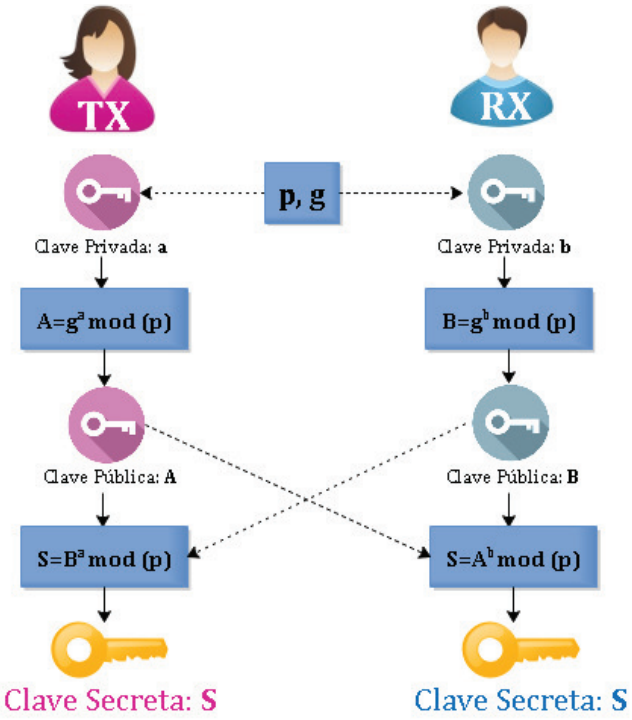


**Figura 1.19.** Diagrama de bloques del algoritmo asimétrico Diffie-Hellman

Si los números iniciales se eligen cuidadosamente, el intercambio de claves puede ser un algoritmo fuerte para proteger la clave secreta compartida [16].

IPSec[7] usa el algoritmo de intercambio de claves Diffie-Hellman junto con autenticación RSA para intercambiar claves de sesión. Este algoritmo se considera seguro contra el espionaje y ataques MITM[8] [16].

- **RSA**

RSA fue desarrollado en 1978 por tres profesores del MIT[9]: Ronald L. Rivest, Adi Shamir y Leonard M. Adleman (de ahí el nombre RSA). Los autores ofrecieron sus hallazgos a cualquiera que les enviara un sobre con su dirección. La NSA tomó una visión amarga de este enfoque y sugirió que los profesores cesaran y desistieran. Sin embargo, cuando se le preguntó acerca de la legalidad de su solicitud, la NSA no respondió y el algoritmo fue publicado [15].

[7] IPSec (*Internet Protocol Security)*: Conjunto de protocolos que autentica y cifra los paquetes de datos enviados por la red.
[8] MIMT (*Man-In-The-Middle)*: Ataque que intercepta la comunicación para capturar el tráfico que pasa a través de la red.
[9] MIT (*Massachusetts Institute of Technology)*: Universidad privada norteamericana reconocida a nivel mundial por su calidad de enseñanza en ingeniería, economía, ciencia y arte.

Se relaciona con Diffie-Hellman, pero es mucho más rápido. Es el primer algoritmo utilizado tanto para el cifrado como para la firma digital y todavía se usa ampliamente, especialmente en el comercio electrónico. Es importante comprender que RSA utiliza una función de una sola vía, una fórmula matemática para generar una clave que es fácil de calcular en una dirección, pero difícil o casi imposible de calcular en la dirección opuesta. Por ejemplo, multiplicar dos números primos grandes para determinar su producto es fácil, pero cuando se tiene únicamente el producto, es difícil determinar qué números se usaron en el cálculo [15].

Debido a que el algoritmo RSA se basa en operaciones exponenciales, es posible aplicarlo a cualquier tamaño de texto, aunque, en general suele ser utilizado solamente para cifrar el intercambio de una clave simétrica por su alto nivel de procesamiento [17]. El diagrama de bloques del algoritmo asimétrico RSA se muestra en la Figura 1.20.

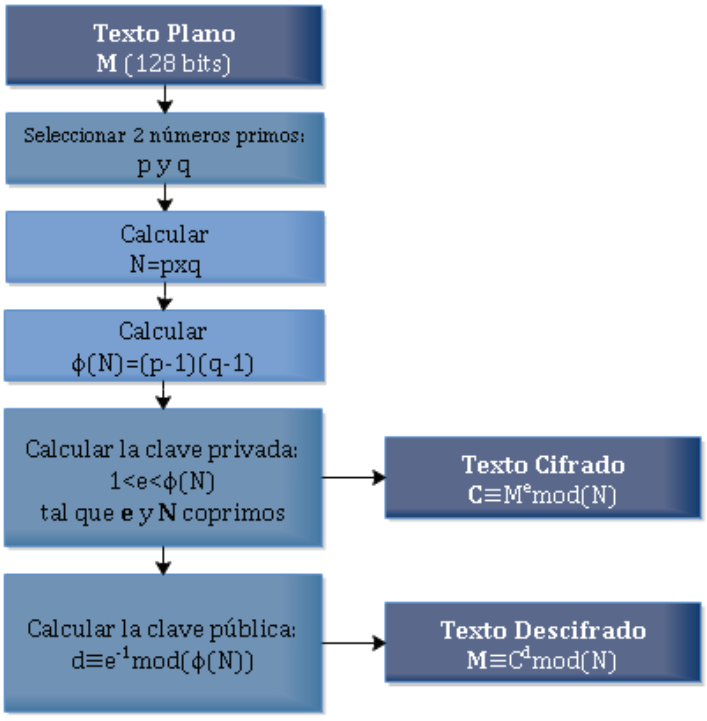


**Figura 1.20.** Diagrama de bloques del algoritmo asimétrico RSA

Muchos navegadores *web* que utilizan el protocolo SSL[10] utilizan el algoritmo RSA, que se basa en la dificultad de factorizar números grandes [15].

[10] SSL (*Secure Sockets Layer)*: Protocolo que protegen las comunicaciones empleando certificados digitales.

### Ataques al algoritmo RSA

Para comprobar cuán robusto es el algoritmo asimétrico RSA se crearon desafíos con el objetivo de conocer la longitud de las claves que son seguras y por cuánto tiempo lo son. El Laboratorio RSA incentivó con recursos económicos a los que encontraran la clave de cada desafío propuesto. El *RSA Challenge* se estableció en marzo de 1991; consiste en que, al conocer la clave pública N que es el producto de dos números grandes primos (N=pxq), encontrar p y q de tamaño aproximadamente igual.

La Tabla 1.5 muestra algunos resultados del Desafío RSA [18] [19].

**Tabla 1.5.** *RSA Challenge*

| Número | Dígitos Decimales | Dígitos Binarios | Fecha de factorización | Premio |
|---|---|---|---|---|
| RSA-100 | 100 | 330 | Abril de 1991 | $ 1 000 |
| RSA-110 | 110 | 364 | Abril de 1992 | $ 4 429 |
| RSA-120 | 120 | 397 | Junio de 1993 | $ 5 898 |
| RSA-129 | 129 | 426 | Abril de 1994 | $ 100 |
| RSA-130 | 130 | 430 | Abril de 1996 | $ 14 527 |
| RSA-140 | 140 | 463 | Febrero de 1999 | $ 17 226 |
| RSA-150 | 150 | 496 | Abril de 2004 | |
| RSA-155 | 155 | 512 | Agosto de 1999 | $ 9 383 |
| RSA-160 | 160 | 530 | Abril de 2003 | |
| RSA-170 | 170 | 563 | Diciembre de 2009 | |
| RSA-576 | 174 | 576 | Diciembre de 2003 | $ 10 000 |
| RSA-180 | 180 | 596 | Mayo de 2010 | |
| RSA-190 | 190 | 629 | Noviembre de 2010 | |
| RSA-640 | 193 | 640 | Noviembre de 2005 | $ 20 000 |
| RSA-200 | 200 | 663 | Mayo de 2005 | |
| RSA-210 | 210 | 696 | Septiembre de 2013 | |
| RSA-704 | 212 | 704 | Julio de 2012 | $ 30 000 |
| RSA-220 | 220 | 729 | Mayo de 2016 | |
| RSA-768 | 232 | 768 | Diciembre de 2009 | $ 50 000 |
| RSA-2048 | 617 | 2048 | | $ 200 000 |

RSA-129 se factorizó en 8 meses, con 1600 computadores distribuidos en la red, recibiendo un premio de $ 100, pero no fue considerado desafío. Seguidamente, RSA-155 fue roto en 5.2 meses con 300 equipos en red, mientras que RSA-140 se demoró 8,9 años en factorizar los números primos con 185 computadores en red.

El desafío RSA más recientemente resuelto, fue el RSA-220 en mayo de 2016, donde se consiguió factorizar un número de 220 dígitos. Además, existen varios desafíos más en la lista con premios de hasta $ 200,000, que aún no se han roto, pero en el año 2007 RSA canceló la competición de desafíos.

Existen cuatro tipos de ataques contra RSA, los cuales se muestran a continuación [14]:

1. **Ataques de protocolo:** Explotan las debilidades en la forma en que se usa RSA, muchos de ellos se pueden evitar mediante el uso de relleno.
2. **Ataques de fuerza bruta:** Intenta todas las posibles claves privadas.
3. **Ataques matemáticos:** El mejor método criptoanalítico matemático que se conoce es factorizar el módulo. Para prevenir este ataque, el módulo debe ser lo suficientemente grande, por lo cual hasta hace poco la longitud recomendada del número primo era 1024 bits, pero se cree que podría ser posible factorizarlo en un tiempo de 10 a 15 años aproximadamente. Por lo tanto, hoy en día se recomienda emplear un número primo en el rango de 2048 a 4096 bits, lo cual proporciona seguridad a largo plazo.
4. **Ataques de canal lateral:** Son ataques completamente diferentes, ya que explotan información sobre la clave privada que se filtra a través de canales físicos, tales como el consumo de energía o el comportamiento en el tiempo.

- **Criptografía de curva elíptica**

  La criptografía de curva elíptica (ECC), desarrollada en 1985, se utiliza para cifrado, así como para firmas digitales e intercambio de claves. Adicionalmente, es un algoritmo eficiente que requiere pocos recursos (memoria, espacio en disco, ancho de banda, etc.), por lo que es un candidato perfecto para dispositivos inalámbricos y teléfonos celulares [15].

  Este algoritmo se utiliza para cifrar una gran cantidad de datos, por su eficiencia en el cálculo, sin demandar un elevado tiempo de procesamiento.

Para cifrar información con el algoritmo ECC se debe acordar entre emisor y receptor un punto de partida para dibujar una curva elíptica. Una vez creadas las curvas, se dibujan líneas para crear el punto de intersección $P$, donde ambos usuarios eligen una clave secreta $K_a$ y $K_b$ y calculan su propia clave, la cual es enviada al otro usuario. A partir de esta clave, cada usuario calcula la clave secreta S.

En la Figura 1.21 se indica un diagrama de bloques del algoritmo asimétrico ECC descrito anteriormente.

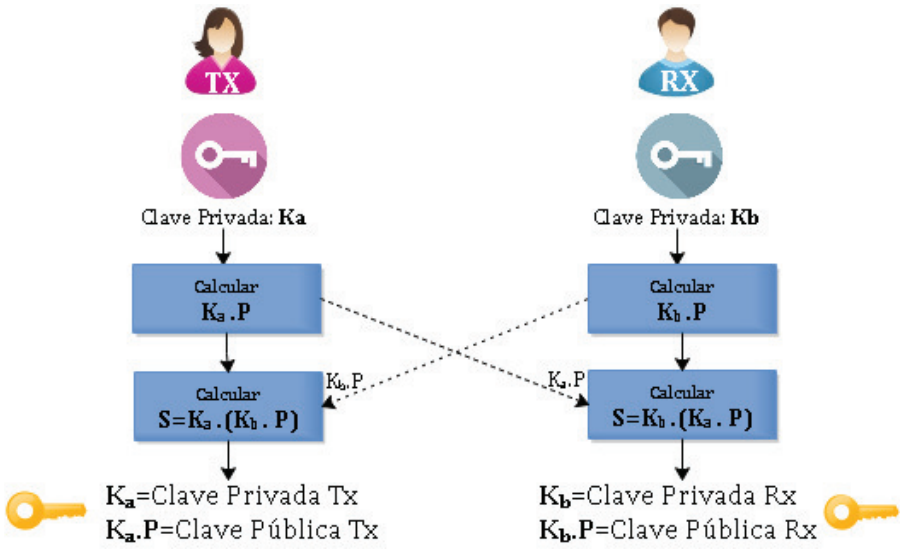


**Figura 1.21.** Diagrama de bloques del algoritmo asimétrico ECC

Según el NIST, el algoritmo ECC con clave de 256 bits proporciona una seguridad aproximadamente igual a los algoritmos AES y RSA empleando claves de 128 y 3072 bits respectivamente [20].

- **ElGamal**

El algoritmo ElGamal fue desarrollado por Taher Elgamal en 1985, es un algoritmo asimétrico utilizado para generar claves, firmas digitales y para cifrar información, prácticamente, es una versión mejorada del algoritmo de intercambio de claves Diffie-Hellman, y se considera tan seguro como RSA [16], utiliza logaritmos discretos que son complejos de resolver, el cual puede tomar muchos años y requerir operaciones intensivas de CPU [15].

ElGamal produce un texto cifrado de gran tamaño y solo se puede emplear en enlaces rápidos de WAN[11]. Su uso es frecuente en algunas versiones recientes de PGP[12] [16].

El diagrama de bloques del algoritmo se detalla en la Figura 1.22.

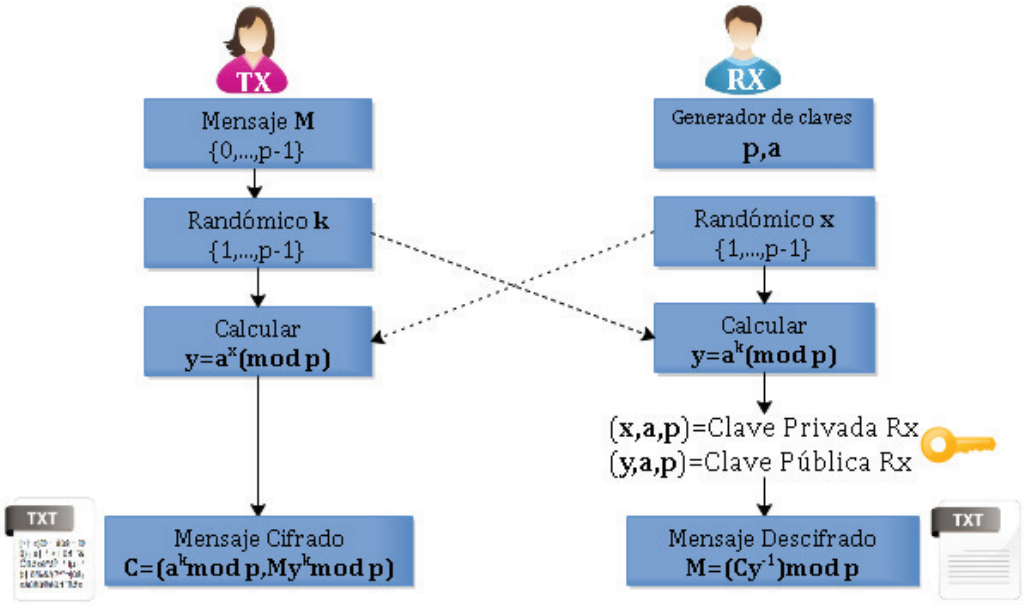


**Figura 1.22.** Diagrama de bloques del algoritmo asimétrico ElGamal

El Algoritmo de firma digital (DSA[13]) es una variante del esquema de firma digital ElGamal y se basa en el algoritmo de ElGamal [16].

### Ventajas y desventajas del cifrado asimétrico

El cifrado asimétrico presenta las siguientes ventajas y desventajas:

#### Ventajas

- La principal ventaja consiste en emplear dos claves diferentes para el cifrado, distribuyendo así la clave pública sin poner en riesgo la privacidad de los mensajes cifrados, por lo tanto es más seguro.

[11] WAN (*Wide Area Network)*: Red de computadoras que interconectan diferentes redes geográficamente separadas.
[12] PGP (*Pretty Good Privacy*): Un programa de cifrado que combina la criptografía simétrica con la asimétrica para garantizar seguridad en las comunicaciones.
[13] DSA (*Digital Signature Algorithm)*: Estándar estadounidense para firmas digitales.

- El problema de distribución de claves se resuelve al suprimir la necesidad de enviar la clave, por lo tanto es más segura que utilizar cifrado simétrico.

**Desventajas**

- Debido al procesamiento matemático, el cifrado asimétrico es mucho más lento que el cifrado simétrico para la misma longitud de mensaje.
- La medida de la seguridad del sistema es el tamaño de la clave, lo cual a pesar de ser de gran tamaño ha tenido ataques de vulnerabilidad.
- La longitud de llave que actualmente se considera segura es de 2048 o 5096 bits, por lo tanto, las claves son de mayor tamaño.
- El mensaje cifrado ocupa más espacio que el original.
- A medida que aumenta la potencia tecnológica, y a la vez se descubren algoritmos de factorización más eficientes, también aumenta la capacidad de factorizar números de mayor tamaño.

Para solventar el inconveniente de las desventajas de los algoritmos asimétricos, la solución consiste en emplear un algoritmo simétrico en conjunto con un algoritmo asimétrico.

**Criptografía asimétrica en conjunto con cifrado simétrico**

Los algoritmos simétricos consumen menor cantidad de recursos en relación a los algoritmos asimétricos, se los emplea mayormente para el cifrado de gran cantidad de datos por su velocidad de cifrado, sin embargo, el problema que presenta es la distribución de claves.

El cifrado asimétrico resuelve el problema de distribución de claves al suprimir la necesidad del envío de la misma, convirtiéndolo en un sistema más seguro, sin embargo, debido a su alto nivel de procesamiento requiere más potencia de cálculo y ralentiza el proceso de cifrado, siendo adecuado para pequeñas cantidades de datos. Por lo tanto, el cifrado asimétrico es perfecto para el cifrado de claves [12].

Es posible complementar ambos métodos de cifrado, aprovechando la rapidez del cifrado simétrico con la robustez del cifrado asimétrico, lo que se conoce como cifrado híbrido, que en la actualidad emplean los protocolos seguros en *internet* y en las redes utilizando el siguiente funcionamiento, el cual se describe gráficamente en la Figura 1.23.

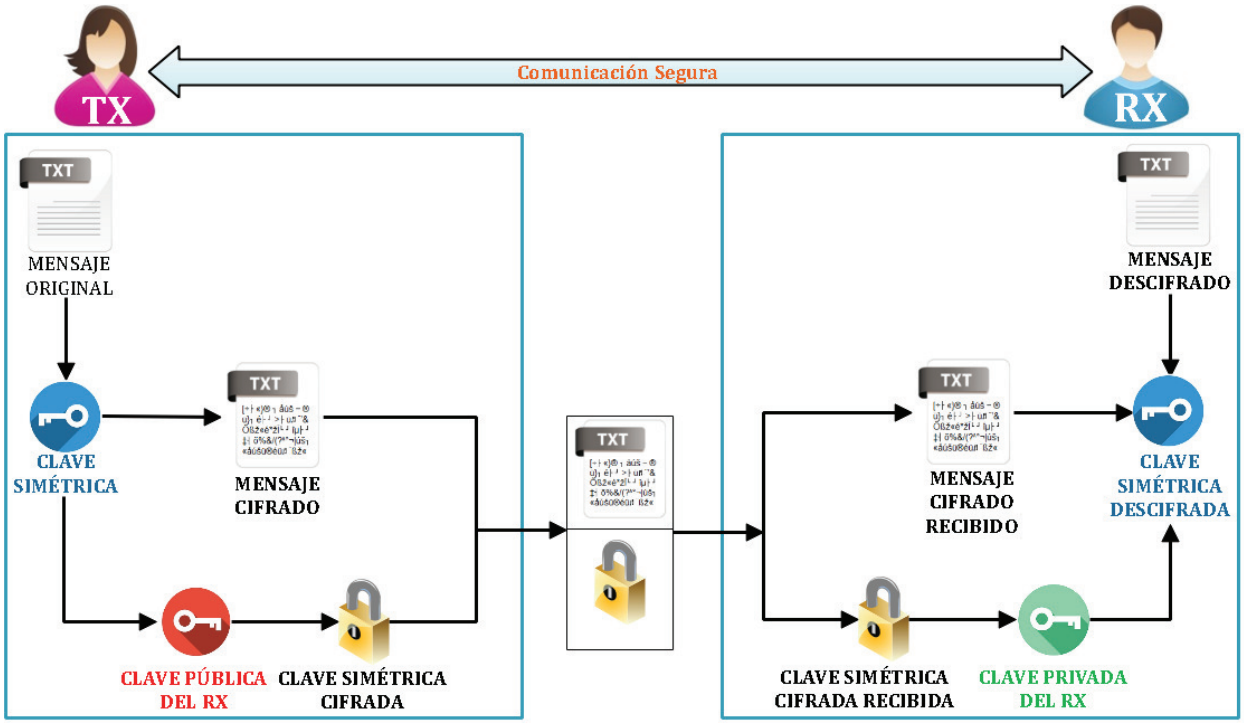


**Figura 1.23.** Criptografía asimétrica en conjunto con cifrado simétrico

a) El mensaje se cifra con un algoritmo simétrico empleando la clave simétrica que es generada por el transmisor.

b) El transmisor cifra solamente la clave simétrica empleando un algoritmo asimétrico.

c) Se envía al receptor el mensaje cifrado junto con la clave simétrica cifrada.

d) El receptor recibe ambas informaciones: la clave simétrica se descifra con el algoritmo asimétrico y una vez obtenida se emplea para descifrar el mensaje con un algoritmo simétrico.

Para complementar el estudio del servicio de cifrado se van a describir brevemente los algoritmos más usados para cada tipo de cifrado.

### 1.1.4. Servicio de Integridad

#### Funciones *HASH*

Un algoritmo *hash* es una función que toma una cadena o mensaje de longitud variable y produce un valor *hash* de longitud fija, también llamado resumen de mensaje, que se emplea para verificar la integridad de los datos o mensajes, se representa como una

cadena corta de letras aleatorias y números, es como una huella digital de un mensaje, un proceso unidireccional, pues no es posible crear el texto original utilizando cualquier función de *hash* inverso [16], si los datos originales cambian incluso por un caracter, la función *hash* producirá un valor *hash* diferente, por lo tanto, el receptor sabrá que la información original ha cambiado.

Los algoritmos *hash* se utilizan para proporcionar integridad y autenticación de los datos enviados a través de los medios de red de una computadora a otra. De hecho, es muy utilizado en seguridades por el uso de cero claves, pues es un algoritmo muy conocido y muy robusto.

Por otro lado, es común almacenar y transferir contraseñas cifradas como *hash* en redes seguras, cuando el usuario establece su contraseña, se genera una función *hash*, y solo el *hash* cifrado se almacena. Cuando el usuario inicia sesión en la red, su contraseña es resumida nuevamente y los dos valores de *hash* se comparan, como resultado, si se encuentra una coincidencia, el usuario se le concede acceso; de lo contrario, se le niega [16]. El algoritmo de *hash* más comúnmente utilizado es *Message Digest* 5 (MD5) y el más seguro *Secure Hash Algorithm* (SHA).

**Propiedades del *hash***

Los *hash* criptográficos deben poseer algunas propiedades específicas [17]:

- Debe ser prácticamente imposible reconstruir los datos originales del *hash*.
- Dos documentos similares deben producir *hash* muy diferentes.
- Debe ser prácticamente imposible construir dos datos que generen el mismo *hash*.
- La computación del *hash* es un proceso rápido

**Algoritmos de resumen de mensajes**

Las funciones principales de funciones *hash* se utilizan debido a que son ampliamente seguros y ofrecen velocidad de ejecución. A continuación, se mencionan algunos de estos algoritmos:

- **MD5 (*Messaje Digest 5*)**

  MD5 es la quinta generación del algoritmo de resumen, el cual es ampliamente utilizado.

En agosto de 1992, Ronald L. Rivest del MIT presentó un documento al IETF14 titulado "*The MD5 Message-Digest Algorithm*", el cual describe la teoría de este algoritmo.

Genera un valor *hash* de 128 bits, se representa típicamente como un número hexadecimal de 32 dígitos, especificado en el RFC 132115. El *hash* resultante se emplea principalmente en firmas digitales para verificar la integridad de los datos [16].

Adicionalmente, el algoritmo divide al texto de entrada en bloques fijos de 512 bits, realiza operaciones lógicas entre el primer bloque de mensaje y 4 vectores iniciales de 32 bits cada uno: ABCD fijos, generando una salida de 128 bits de 4 nuevos vectores iniciales A'B'C'D'. De igual manera, se realiza la misma función con todos los bloques de entrada, donde los 128 bits generados en el último bloque son los que corresponden al resumen del texto de entrada, como se ilustra en el diagrama de bloques de la Figura 1.24.

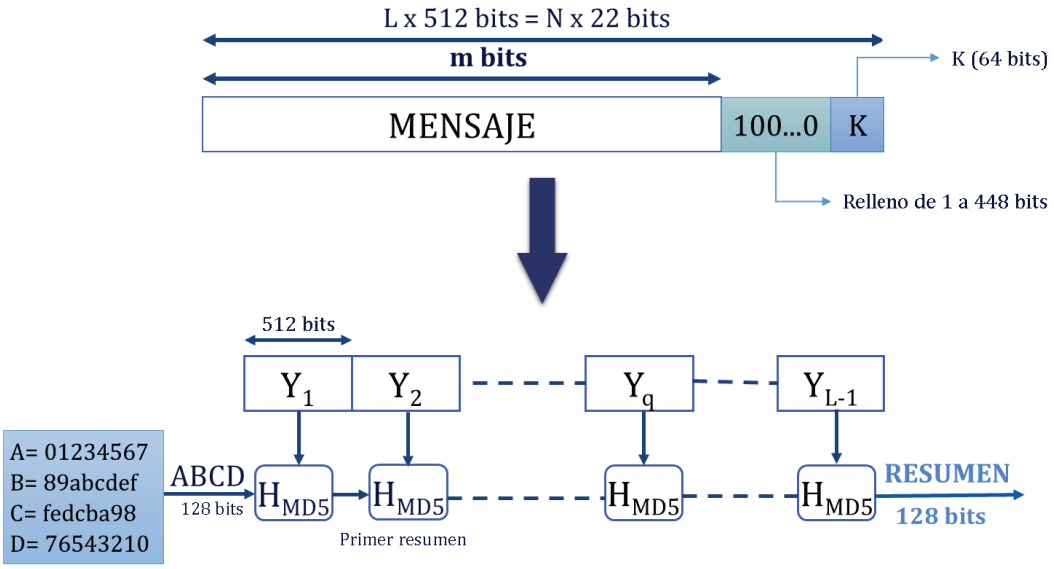


**Figura 1.24.** Diagrama de bloques del algoritmo de resumen MD5

Existe una nueva versión de MD no muy conocida: MD6, fue desarrollada en el año 2008, utiliza bloques de mensajes de entrada muy grandes (hasta 512 Bytes) y produce *hash* de longitud variable (hasta 512 bits). MD6 se presentó originalmente

[14] IETF (*Internet Engineering Task Force)*: Organización internacional de normalización que tiene como función principal el anális, desarrollo e investigación de nuevas tecnologías.
[15] Más información en la página *web*: tools.ietf.org/html/rfc1321

para su consideración como el nuevo estándar SHA-3, pero fue eliminado en julio de 2009.

Todos estos algoritmos de *Message Digest* son conocidos por tener fallas de resistencia a colisión que podrían resultar en la creación de dos documentos que generen el mismo *hash* [17].

**Colisiones del algoritmo MD5**

Los *hash* no son únicos, puesto que la cantidad de textos planos posibles es mucho mayor que la cantidad de *hash* posibles, por lo tanto, más de un documento podría generar el mismo *hash*, a esto se le llama colisión. Si bien las colisiones siempre son posibles (suponiendo que el texto plano sea más largo que el *hash*), deberían ser muy difíciles de encontrar. La búsqueda de una colisión para que coincida con un texto plano específico no debería ser posible en un tiempo razonable [21].

En 1993, B. den Boer y A. Bosselaers encontraron una especie de *pseudo*-colisión para MD5 que consiste en el mismo mensaje con dos conjuntos diferentes de valores iniciales. Este ataque revela que la debilidad se encuentra en el bit más significativo del mensaje [22]. En 1996, Dobbertin construyó colisiones MD5 con un valor inicial erróneo. Estuvo lo suficientemente cerca de las colisiones de MD5 para sugerir que MD5 ya no debería usarse en aplicaciones tales como esquemas de firma, donde se requiere una función de *hash* resistente a colisiones, sin embargo, todavía se usaba en muchas aplicaciones. En marzo de 2005, Xiaoyun W. y Hongbo Yu de la Universidad de Shandong en China publicaron un artículo[16] sobre un algoritmo que puede causar una colisión MD5, lo que significa que dos entradas arbitrarias pueden producir el mismo valor *hash*: *hash*(mensaje1) = *hash*(messaje2).

Estadísticamente, la posibilidad de producir el mismo hash de diferentes entradas es una en $2^L$ (donde $L$ es la longitud del resumen de salida que es de tamaño fijo (128)), lo cual resulta infinitamente pequeño para MD5.

El Departamento de Seguridad Nacional de los EEUU lo considera "criptográficamente roto", pues se pueden generar nuevas colisiones en un PC en cuestión de minutos [23].

[16] Artículo "*How to Break MD5 and Other Hash Functions*". Disponible en la página *web*: http://merlot.usc.edu/csac-f06/papers/Wang05a.pdf

El criptógrafo checo Vlastimil Klíma publicó su trabajo en el cual demostró que colisiones MD5 se pueden calcular en aproximadamente ocho horas en una PC doméstica estándar. En noviembre de 2007, los investigadores publicaron la capacidad de tener dos ejecutables Win32 completamente diferentes con diferentes funcionalidades, pero con el mismo *hash* MD5 [23].

El *hash* es una propiedad importante para una serie de aplicaciones, como en firmas digitales, en las que, a un mensaje se resume y luego se lo firma. Entonces, si dos archivos, uno legítimo y uno malicioso, pueden tener los mismos valores de *hash* y, por lo tanto, una firma en el documento legítimo también será válida en el malicioso. Arjen Lenstra y otros investigadores demostraron en el año 2005 que es posible crear dos certificados digitales de diferentes claves públicas que tengan el mismo *hash* MD5.

MD5 es probablemente el algoritmo de *hash* más utilizado en el mundo hoy en día y lo seguirá siendo durante varios años más. Esto se debe al hecho de que MD5 está codificado en sistemas operativos y aplicaciones populares. Solo cuando los sistemas operativos y las herramientas de *software* comunes pasen a SHA-1 u otro algoritmo, se disminuirá el uso de MD5.

- **SHA (*Secure Hash Algorithm*)**

  El algoritmo *Secure Hash* fue desarrollado por el NIST y se documentó por primera vez en la Publicación 180 de FIPS[17]. La versión actual es SHA-1 se especifica en FIPS 180-1 y RFC 2404[18]. El proceso de resumen de los algoritmos SHA es muy similares a MD5, separando el texto de entrada en bloques de 512 bits, con total de 80 rondas, pero el vector inicial tiene 32 bits más, lo que generará un resumen resultante de 160 bits.

  SHA-0 y SHA-1 producen cada uno un resumen de 160 bits desde cualquier mensaje de entrada hasta $2^{64}$-1 bits de longitud. SHA-1 fue desarrollado por la NSA, especificado en el RFC 3174[19], utiliza un valor de *hash* de 160 bits y se considera más seguro que MD5 y es comúnmente utilizado en IPSec [16].

  Para aplicaciones sensibles, NIST recomienda no usar SHA-1. Las agencias federales han recibido instrucciones de eliminar SHA-1 de futuras aplicaciones y

[17] FIPS (Federal Information Processing Standards): Estándares norteamericanos para el empleo en agencias no militares y contratistas públicos.
[18] Más información en la página *web*: tools.ietf.org/html/rfc2404
[19] Más información en la página *web*: tools.ietf.org/html/rfc3174

reemplazarlo con las versiones de resumen más largas de SHA, conocidas colectivamente como SHA-2 [15].

SHA-2 es la versión actual de SHA. SHA-2 es una colección de cuatro variaciones que incluyen SHA-256, SHA-384 y SHA-512 [16], son versiones que usan esencialmente el mismo algoritmo que SHA-160, pero los resúmenes más largos hacen que las colisiones sean más difíciles de encontrar [15]. En la Figura 1.25 se muestra el diagrama de bloques del algoritmo de resumen SHA-512.

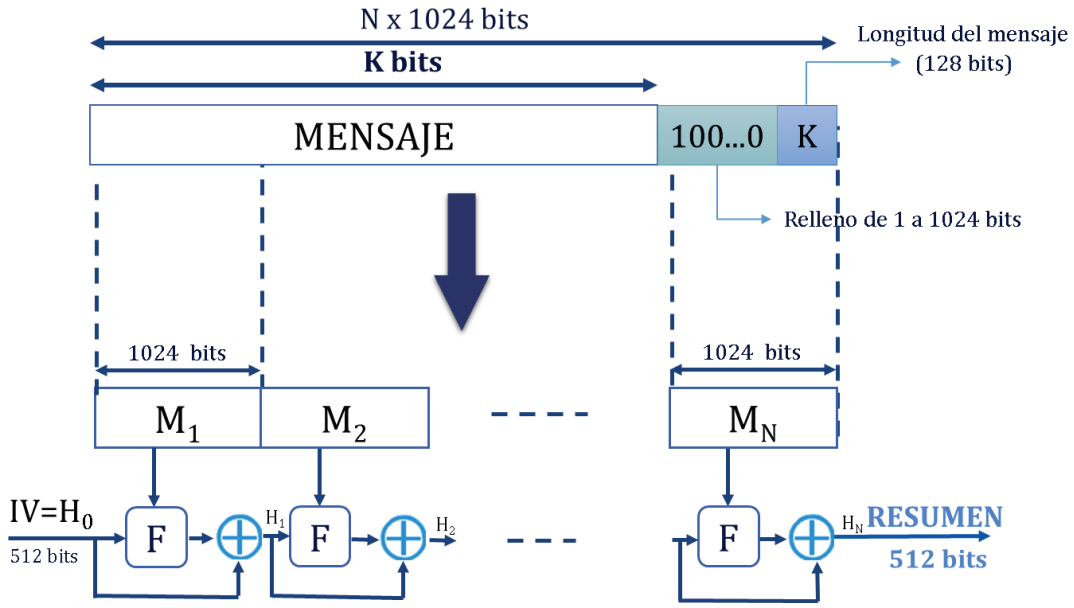


**Figura 1.25.** Diagrama de bloques del algoritmo de resumenSHA-512

En la Tabla 1.6 se comparan los diferentes tipos de resumen MD5, SHA-1 y SHA-2 [24]:

**Tabla 1.6.** Comparación de los algoritmos de resumen SHA-1 y SHA-2

| Algoritmo | Longitud máxima del mensaje (bits) | Tamaño del bloque de datos (bits) | Tamaño del resumen (bits) |
|---|---|---|---|
| **MD5** | Ilimitado | 512 | 128 |
| **SHA-1** | $2^{64}$ | 512 | 160 |
| **SHA-256** | $2^{64}$ | 512 | 256 |
| **SHA-384** | $2^{128}$ | 1024 | 384 |
| **SHA-512** | $2^{128}$ | 1024 | 512 |

El resumen del mensaje SHA-1 de 160 bits es más seguro que el resumen del mensaje MD5 de 128 bits, hay un precio a pagar en rendimiento por la seguridad adicional, pero si es necesario usar integridad del mensaje de la forma más segura, se debe seleccionar el algoritmo HMAC-SHA-1 [25] [15].

SHA-3 o también conocido como Keccak fue seleccionado como el estándar SHA-3 por el NIST en el 2015. SHA-3 puede tomar una entrada de cualquier tamaño y crear una salida de cualquier tamaño. SHA-3 no está destinado a reemplazar inmediatamente SHA-2 y varía mucho en diseño de sus predecesores de SHA [15].

- **HMAC (*Hash-Keyed Message Authentication Code*)**

  Es un tipo especial de *hash* con clave, fue diseñado por Hugo Krawczyk, Ran Canetti y Mihir Bellare. HMAC se desarrolló para mejorar la criptografía agregando una clave secreta en el cálculo del resumen del mensaje producido por algoritmos de *hash* estándar.

  En la Figura 1.26 se puede observar el diagrama de bloques del funcionamiento del algoritmo HMAC.

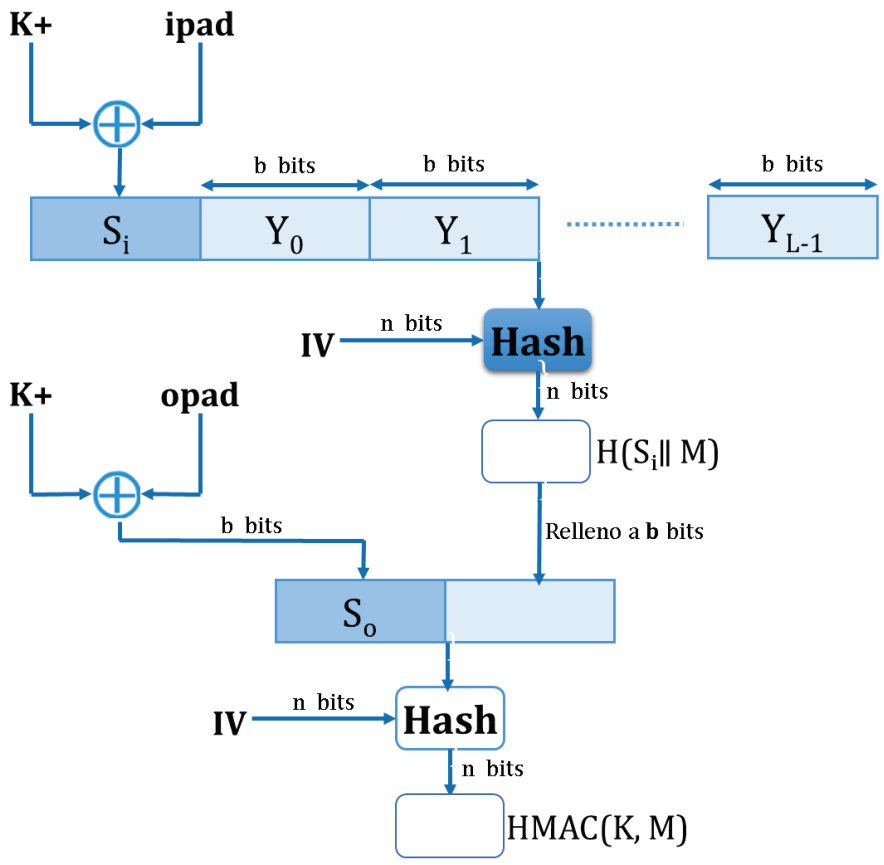


**Figura 1.26.** Diagrama de bloques del algoritmo de resumen HMAC

La especificación HMAC se encuentra disponible en el RFC2104[20] y se puede utilizar con cualquier función *hash* existente, por lo que SHA puede convertirse en HMAC-SHA y MD5 se convierte en HMAC-MD5. La clave secreta agregada a la fórmula es de la misma longitud del resumen del mensaje resultante dependiendo el algoritmo *hash* utilizado [25].

- **Variante HMAC – MD5**

  Cisco[21] utiliza la variante HMAC para autenticación denominada HMAC-MD5-96. Esta versión usa una clave secreta de 128 bits para producir un *hash* de 128 bits.

  MD5 sin HMAC tiene algunas debilidades conocidas que lo hacen una opción más débil para aplicaciones de alta seguridad, como las colisiones, pero esta variante no es susceptible a ataque, pues HMAC-MD5 aún no ha sido atacado con éxito [26].

### 1.1.5. Servicio de Autenticación y No Repudio

**Firma Digital**

Una firma digital es un medio electrónico para validar la autenticidad y la integridad de un mensaje, *software* o documento. Las firmas digitales se utilizan para proporcionar autenticación del emisor, integridad y no repudio de datos, esto asegura que los datos enviados no fueron modificados en su camino, desde el origen hasta el destino. [12]

Cuando se envía el mensaje, a éste se le aplica un algoritmo de resumen para generar un valor *hash*. El *hash* se cifra aún más utilizando la clave privada del remitente y se adjunta al mensaje. El receptor usa la clave pública del emisor para descifrar el *hash* recibido. El receptor también genera un *hash* del mensaje recibido y se comparan los dos valores *hash*: si el valor *hash* del receptor coincide con el del emisor, el receptor se asegura de que el mensaje no ha sido modificado en el camino y que el emisor es auténtico [12].

**Tipos de firma digital**

La mayoría de las firmas digitales utilizan la criptografía asimétrica para lograr la autenticidad, el proceso para la firma digital empleando algoritmos asimétricos se puede ver en la Figura 1.18. A continuación se indican los tipos de firma digital que existen:

[20] Más información en la página *web*: tools.ietf.org/html/rfc2104
[21] Cisco: Empresa global de telecomunicaciones que fabrica, comercializa y da mantenimiento de equipos de conectividad.

- **Firma simple**: Incluye el *hash* cifrado empleando la clave privada, es decir es el mensaje cifrado resumido.
- **Firma avanzada**: Empleo de técnicas PKI para garantizar la identidad del documento e identificar al emisor.
- **Firma reconocida**: Ejecución de un DSCF (Dispositivo Seguro de Creación de Firma) a la firma avanzada.

**Firma digital vs firma electrónica:** La firma electrónica es un término mucho más amplio que involucra el ámbito legal, organizacional, técnico, etc. Además, tiene validez jurídica.

Por lo tanto, la firma digital es uno de los elementos que conforman la firma electrónica.

**Estándar de firma digital**

En 1991, el NIST estableció el DSS (*Digital Signature Standard*) para asegurar que las firmas digitales pudieran ser verificadas. El gobierno federal especificó el uso de RSA y del *Digital Signature Algorithm* (DSA) para todas las firmas digitales y el uso de un algoritmo *hash* para garantizar la integridad del mensaje (es decir, verificar que no se haya manipulado) [12].

- **Proceso de firma digital DSA** [12]

  DSA es considerado como la Criptografía de Clave Pública (PKC), es decir, utiliza ambas claves privadas y públicas, diferente a la criptografía de clave secreta, como la utilizada en HMAC. DSA se utiliza sólo para las firmas digitales, no es utilizada para el cifrado.

  DSA utiliza dos claves para generar la firma digital. La primera clave proviene de la función *hash* SHA - 1. La actual versión de DSA es capaz de usar las nuevas versiones del algoritmo SHA como SHA - 2. La segunda clave es creada a partir de una clave privada. Estas dos claves se usan para generar la firma digital. Una vez DSA ha generado el resumen del mensaje, utiliza su clave privada para cifrar mismo resumen del mensaje. El resumen del mensaje cifrado es la firma digital.

  Debido a que utiliza un DSA dos tipos de distribución de clave, se considera que requiere un procesamiento más intensivo que otros algoritmos tales como RSA.

  Las principales características del proceso de firma digital DSA son los siguientes:

  - Utiliza varias claves para generar la firma digital.

- Utiliza SHA - 1 como el algoritmo de *hash*.
- Considerado más lento que el RSA.

- **Proceso de firma digital RSA** [12]

RSA puede descifrar un mensaje cifrado con una clave privada solo si el mensaje fue cifrado por el titular de la clave privada correspondiente [15]. El proceso de firma digital RSA es más ágil, más rápido y más flexible que el proceso de DSA. Sin embargo, se podría argumentar que a pesar de la velocidad y la flexibilidad, es menos seguro.

El valor de *hash* o resumen del mensaje se cifra con RSA mediante el uso de la clave privada. El resumen cifrado es la firma digital, a la cual se añaden los datos originales y se envían al destinatario. El destinatario genera su propio resumen del mensaje de los datos que recibió y además descifra la firma digital con la clave pública del remitente. Si el *hash* del mensaje generado y el *hash* del mensaje descifrado son iguales, la información es íntegra y el emisor está autenticado ya que la clave pública descifrará sólo los mensajes cifrados por la clave privada del remitente, tal como se muestra en la Figura 1.27.

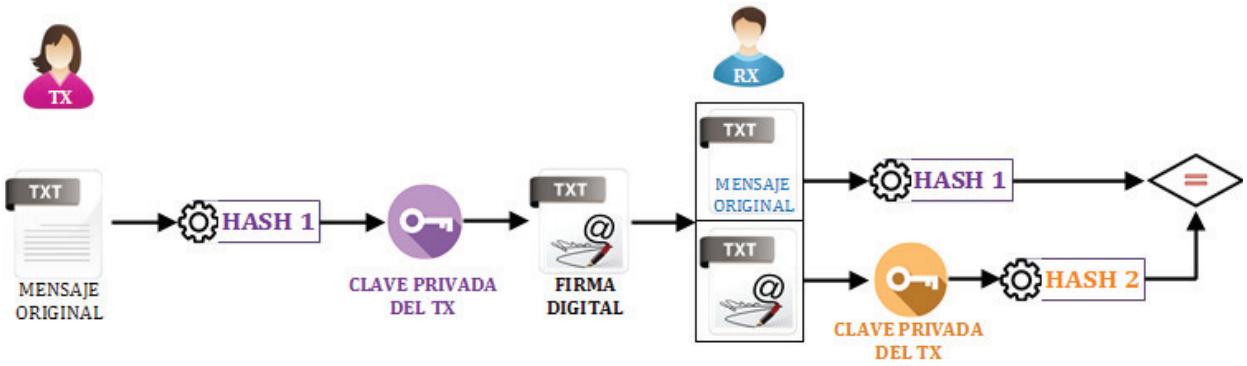


**Figura 1.27.** Firma Digital

Las principales características del proceso de firma digital RSA son los siguientes [12]:

- Considerado más rápido y más flexible que el DSA.
- Cifra sólo el resumen del mensaje.
- Utiliza la clave privada para cifrar y la clave pública para descifrar.

En la Tabla 1.7 se muestra un cuadro comparativo entre la firma digital DSA y RSA:

**Tabla 1.7.** Comparación de firma digital DSA y RSA

| | DSA | RSA |
|---|---|---|
| **Tipo de clave** | Clave pública | Clave pública |
| **Cifrado** | No | Si |
| **Velocidad de procesamiento** | Lento | Rápido |
| **Seguridad** | Fuerte | Fuerte |

**Infraestructura de Clave Pública (PKI)** [12]: PKI es un sistema de recursos: servidores, estaciones de trabajo, usuarios, redes y empresas, políticas y *software*, permite a los usuarios intercambiar de forma segura datos confidenciales utilizando claves públicas y privadas obtenidas por medio de una autoridad, es decir, facilita los medios para distribuir, administrar, verificar y revocar las identidades digitales. El estándar X.509 indica el funcionamiento de la infraestructura PKI [12] y puede ser encontrado en [27]:

1. Servidores *web* seguros para la autenticación de sitios *web* con *Secure Socket Layer* (SSL).
2. Navegadores *web* para servicios que implementan certificados de cliente SSL y Protocolos de Seguridad de capa Transporte (TLS).
3. Agentes de correo para admitir la protección del correo utilizando el protocolo *Secure Multipurpose Internet Mail Extension* (S/MIME)[22]
4. En IPSec VPNs[23], donde los certificados se usan como un mecanismo de distribución de clave pública para IKE[24] basado en autenticación.

El sistema PKI se utiliza principalmente en redes inseguras que utilizan criptografía de clave pública. Una manera de lograr generar un canal seguro para las comunicaciones y verificar la identidad de los participantes es usar PKI, sin embargo, existe otra manera llamada red de confianza, donde no hay una autoridad centralizada, esta técnica se utiliza con *Pretty Good Privacy* (PGP), la desventaja de este método es que no es escalable; mientras que PKI es utilizado para grandes entornos [12].

[22] *Secure Multipurpose Internet Mail Extension* (S/MIME): Método seguro para aplicaciones de mensajería electrónica que garantiza autenticación, integridad, no repudio y confidencialidad mediante cifrado.
[23] VPN (*Virtual Private Network*): Redes diferentes se interconectan en una red que emula ser local.
[24] IKE (*Internet Key Exchange*): Protocolo de intercambio secreto de claves en IPSec.

### Certificados digitales

Un certificado digital es una forma de crear identidades electrónicas. Estos certificados son parte de la información digital que proporciona una forma de verificar que cada entidad es quien dice ser.

El certificado es emitido por una Autoridad de Certificación (CA) [12], tiene periodo limitado de tiempo y no son válidos después de su fecha de caducidad; además, deja de ser válido por motivos como el cambio de empresa del propietario o la divulgación de la clave privada. La fecha de vencimiento del certificado debe caducar antes que la fecha de vencimiento de la clave pública [28].

**Clases de certificados** [12]: No existe un estándar para las clases de certificados, sin embargo, VeriSign fue el que inició el concepto y ha creado seis clases de certificados digitales, cada una tiene diferentes requisitos de verificación.

**a)** Clase 0 = Sin verificación.
**b)** Clase 1 = Certificado individual, destinado para la firma digital de correo electrónico.
**c)** Clase 2 = Certificado de Clase Business, demuestra el certificado pertenece a la empresa.
**d)** Clase 3 = Se utiliza para verificar que un programa de servidor o software es proporcionado por una fuente verificada.
**e)** Clase 4 = Certificados específicos para la comunicación entre dos empresas conocidas.
**f)** Clase 5 = Privada o de Gobierno, para su uso en entornos PKI privadas o gubernamentales.

El Certificado Digital debe cumplir con formatos estandarizados, y al menos debe contener la siguiente información:

1. ID de la Autoridad Certificadora emisora (CA).
2. ID del titular del titular del Certificado.
3. ID del Certificado.
4. Fechas de emisión y caducidad del Certificado.
5. Clave pública.
6. Otro tipo de información en función de la clase de Certificado.
7. Firma con la Clave Privada de la Autoridad Certificadora emisora (CA).

Al descargar *software* de *Internet* o al hacer compras en línea, se debe verificar la validez del certificado digital de la organización o del sitio *web*, seleccionando el signo de candado que aparece en la esquina inferior derecha del navegador *web*.

### Autoridades de Certificación (CA)

Las autoridades de certificación, también conocida como Tercera Parte Confiable (*Trusted Third Party* o TTP) proporcionan el poder y la autoridad en una PKI, identifican a una persona, la vinculan a su clave pública. Las CA brindan los siguientes servicios [28]:

- Emisión de certificados digitales
- Validación de certificados digitales
- Revocación de certificados digitales
- Distribución de claves pública

**Tipos de Certificados Digitales que emite una CA:**

- Certificados de Identidad
- Certificados de Autorización o potestad
- Certificados Transaccionales
- Certificados de Tiempo

CA puede ser pública o privada, VeriSign y Thawte son entidades emisoras públicas. Las CA privadas son las creadas en las organizaciones privadas para uso interno. Microsoft AD (*Active Directory*[25]) y OpenSSL son tipos de servidores CA privados [12].

**Procedimiento para obtener un certificado de una CA:**

1. Generar la clave privada.
2. Crear una Solicitud de Firma de Certificado (CSR), la cual debe contener información de la organización y del servidor *web*, código de país, localización completa, nombre de dominio que se desea proteger.
3. Enviar la petición de certificado, junto con los documentos que prueben su identidad a una CA.
4. Seguir las instrucciones que indique la CA para obtener un certificado.

[25] *Active Directory* (AD): Servicio distribuido para administrar información de los recursos de la red.

5. La CA realiza pruebas de autenticación y una vez confirmada la identidad, la CA enviará el certificado digital.
6. Instalar el certificado en un servidor seguro.
7. Navegar seguros a través de la red.

# 2. METODOLOGÍA

El presente trabajo de estudio técnico se basa en investigación de tipo descriptiva, debido a que su objeto de estudio se concentrará en el estudio del arte de los servicios de seguridad y adaptar los diferentes algoritmos criptográficos (DES, RSA, MD5) en el *software* Matlab, y así implementar los diferentes servicios de seguridad informática, con el objetivo de solventar la necesidad de herramientas didácticas, que ayudarán al profesor que imparte asignaturas relacionadas con la Seguridad Informática a exponer de forma práctica los conceptos impartidos a los estudiantes, los mismos que reforzarán los conocimientos adquiridos durante su formación.

Para la realización de este proyecto de investigación se utilizará la entrevista como técnica de recopilación de los requerimientos para la implementación del Sistema de Seguridad Informático, la cual consiste en un diálogo entre dos personas: el entrevistador y el entrevistado (profesor), con el objetivo de obtener información de una persona entendida en Seguridad Informática.

## 2.1. Entrevista

La ficha tipo de la entrevista se encuentra en el ANEXO I.

Luego de realizar la entrevista a dos profesores que tienen conocimiento en el tema que se investiga y que dictan asignaturas relacionadas con el mismo durante la formación del estudiante, se determinan los requerimientos para desarrollar la aplicación de uso didáctico.

### 2.1.1. Análisis de resultados de la entrevista

Una vez recopilada la información respecto a las respuestas de la entrevista realizada a os profesores, se realizó un análisis, cuyos resultados se detallan a continuación:

- Los profesores opinan que las herramientas didácticas son de gran ayuda para comprender conceptos que involucran aplicaciones prácticas, por ejemplo, los algoritmos criptográficos.
- Los profesores señalan que a pesar de que la seguridad informática es un tema que se debería desarrollar en la práctica, el contenido que se imparte es en mayor parte teórico.

- Los profesores no han empleado alguna aplicación de uso didáctico para que los estudiantes refuercen el conocimiento respecto a seguridad de la información.
- Los profesores sí utilizan programas criptográficos como complemento para impartir sus clases, por ejemplo: CrypTool,
- Sería de gran utilidad disponer de una aplicación de uso didáctico para reforzar los conocimientos de los estudiantes en seguridad de la información.
- Los requerimientos propuestos para la implementación de la aplicación son los siguientes: la aplicación debería ser de fácil uso, para que cualquier persona pueda comprender el significado de seguridad de la información, ya sea que tengan conocimiento en el tema o estén en proceso de aprenderlo; además debería ser de fácil ejecución y debería incluir un entorno gráfico para facilitar el aprendizaje. Cada servicio de seguridad informática debe presentarse por separado para poder entender paso a paso cada algoritmo criptográfico y luego deberían conjugarse todos los algoritmos para implementar un sistema completo de seguridad informática. Los botones de la aplicación deberían ser interactivos e intuitivos. Para el envío de datos de origen a destino se debería realizar entre usuarios conectados en la misma red, como en un laboratorio.

### 2.1.2. Requerimientos para desarrollar la aplicación

Los requerimientos describen los servicios que deben cumplir el sistema y las limitaciones asociadas a su funcionamiento. En base al análisis de las entrevistas descritas previamente, se propone desarrollar la aplicación de uso didáctico con los siguientes requerimientos:

- Una aplicación de uso didáctico para cada servicio de seguridad, que sea auto descriptiva.
- Una pantalla principal, en la que sea posible seleccionar el tipo de servicio de seguridad que se desee ejecutar.
- Una aplicación completa que integre los servicios de seguridad implementados.
- Mensajes informativos en caso de un mal uso de la aplicación.
- Fácil manejo para todo tipo de usuarios.
- Fácil ejecución.

- Didáctica, con botones interactivos e intuitivos.
- El envío de información se debe realizar entre entidades emisora y receptora que se encuentren en la misma red local, conectados con el medio cableado, empleando direccionamiento IPv4.

En la siguiente sección se detallará el desarrollo matemático de cada algoritmo y su implementación en Matlab. Para la implementación del servicio de confidencialidad se emplea el algoritmo simétrico (DES) en conjunto con el algoritmo asimétrico (RSA). El servicio de integridad se implementa con el algoritmo de resumen MD5, el servicio de autenticación de emisor y no repudio se garantizan a través de la implementación de la firma digital, aplicando el algoritmo RSA al resumen MD5. Los servicios acoplados en conjunto generan un sistema de seguridad informática para el envío seguro de mensajes de texto a través de la red.

## 2.2. Servicio de confidencialidad

### 2.2.1. Algoritmo de cifrado DES

El algoritmo DES se basa en el método de cifrado en bloque denominado Feistel.

**Esquema Feistel**

Denominado también algoritmo simétrico por rondas, pues realiza reiteradamente las mismas operaciones; la ventaja de este algoritmo es que es reversible, de esta manera la operación de cifrado es idéntica a la operación de descifrado, invirtiendo el orden de utilización de las subclaves.

En el esquema Feistel los bloques $L_n$ y $R_n$ están sujetos a un conjunto de transformaciones iterativas, formando las rondas de DES, tal como se muestra en la Figura 2.1. En total son 16 rondas, en cada una se emplea una subclave generada previamente para realizar el proceso de cifrado de un bloque en específico. La Ecuación 2.1 describe las operaciones matemáticas de la función de Feistel.

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

**Ecuación 2.1.** Función de Feistel

Donde, $K_i$ es el índice de la subclave, mientras que $L_i$ y $R_i$ son índices del bloque de mensaje.

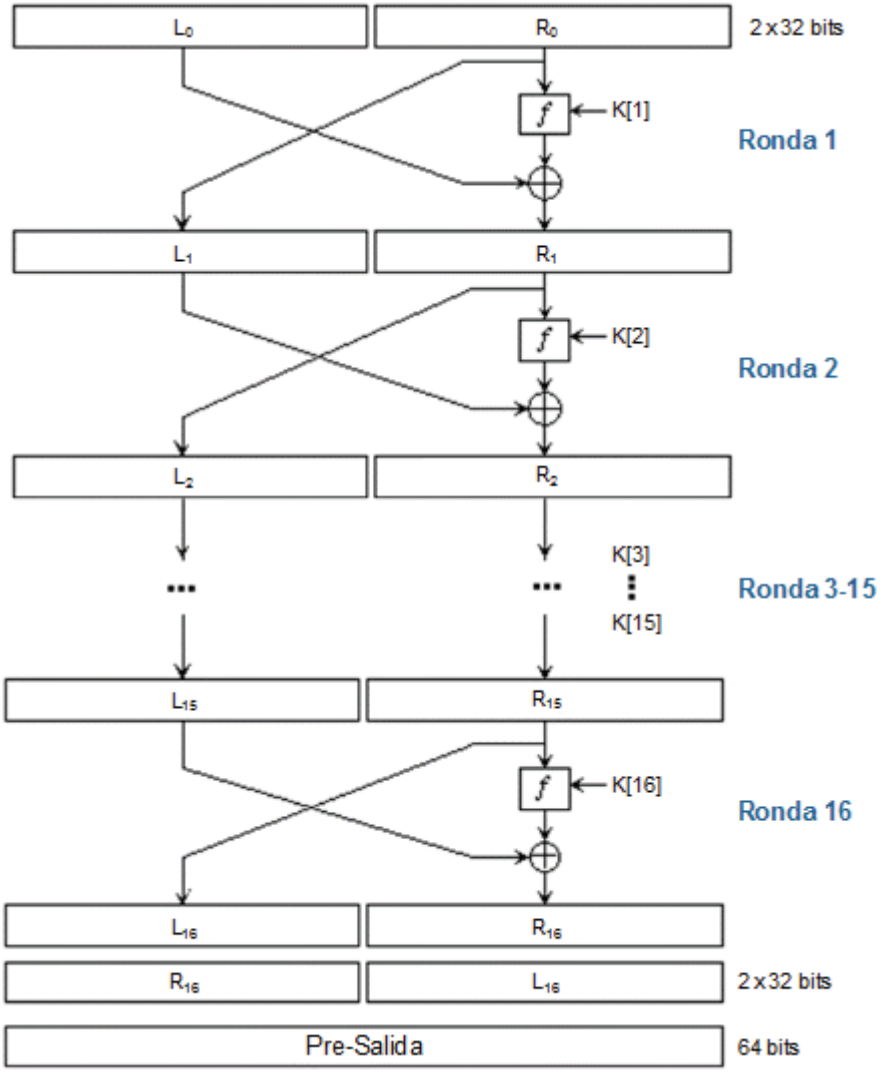


**Figura 2.1.** Función de Feistel

### Etapas de cifrado

El cifrado se divide en dos etapas y cada una de ellas en subetapas: Generación de subclaves y Cifrado del mensaje.

1. **Generación de subclaves**

Esta etapa consiste en generar una llave inicial K, y a partir de esa 16 subclaves, a las que se realizan procesos de permutación y desplazamiento. A partir de la clave simétrica de 64 bits se generarán 16 subclaves de 48 bits cada una, tal como se observa en el diagrama de flujo de la Figura 2.2.

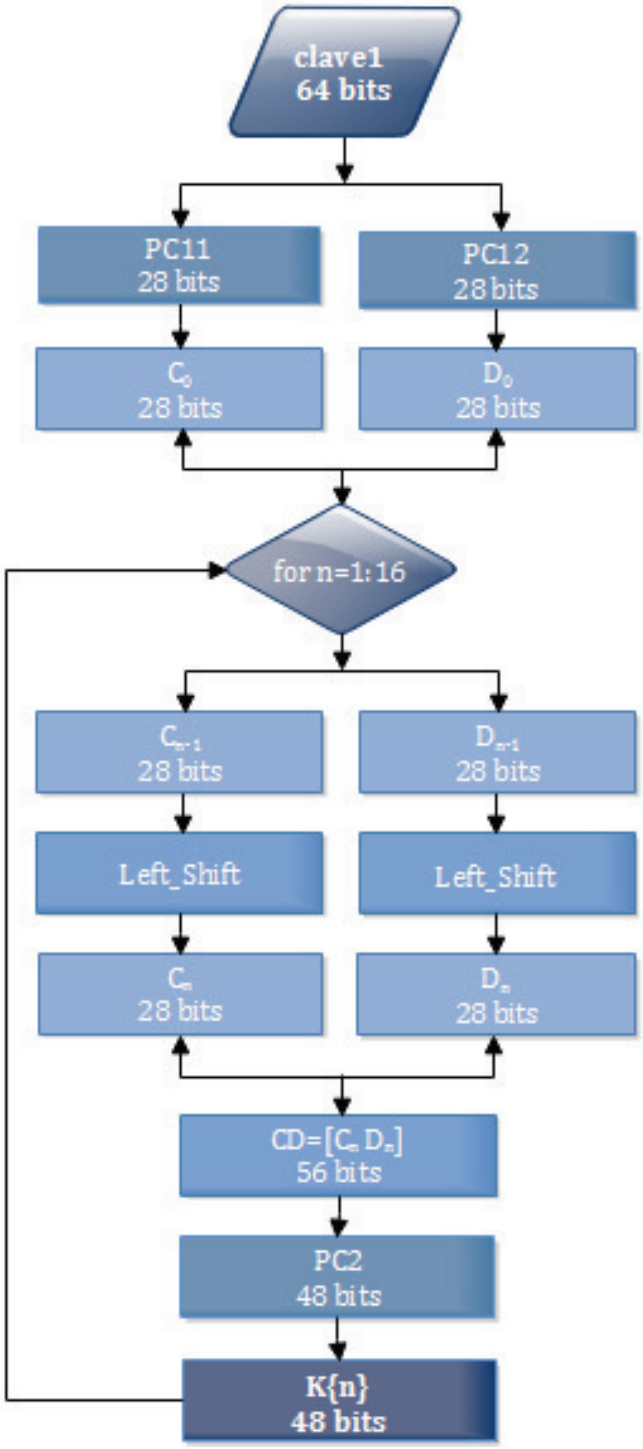


**Figura 2.2.** Diagrama de flujo para la Generación de subclaves

**Ejemplo:** Se va a cifrar el mensaje M con la clave K= 133457799BBCDFF1, la cual se encuentra en formato hexadecimal, el cifrado DES se realiza en bits, por lo cual se debe transformar a binario la clave K.

K= 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

### 1. Clave efectiva

Antes de realizar el proceso de permutación de la clave, el último bit por cada byte de la clave, es decir cada octavo bit (bit menos significativo), que son los bits que se encuentran representados de color rojo, se utilizan únicamente para verificar la paridad, tal como se observa en la Figura 2.3, estos bits no afectan en el proceso de cifrado, por lo tanto, la longitud efectiva de la clave para el cifrado es de 56 bits.

K= 0001001**1** 0011010**0** 0101011**1** 0111100**1** 1001101**1** 1011110**0** 1101111**1** 1111000**1**

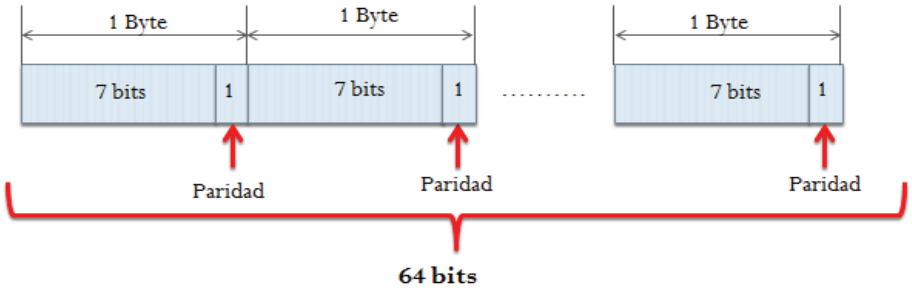


**Figura 2.3** Bits de paridad

### 2. Permutación Inicial 1

El siguiente paso es hacer una permutación llamada *Permuted Choice* 1 (PC1) a la clave K de 64 bits. Los bits de la clave K son reordenados según la Figura 2.4, de tal manera que el primer elemento de la matriz va a ser el bit 57, el segundo elemento será el bit 49, etc, tal como se muestra en la Figura 2.5.

$$\begin{bmatrix} 57 & 49 & 41 & 33 & 25 & 17 & 9 \\ 1 & 58 & 50 & 42 & 34 & 26 & 18 \\ 10 & 2 & 59 & 51 & 43 & 35 & 27 \\ 19 & 11 & 3 & 60 & 52 & 44 & 36 \\ 63 & 55 & 47 & 39 & 31 & 23 & 15 \\ 7 & 62 & 54 & 46 & 38 & 30 & 22 \\ 14 & 6 & 61 & 53 & 45 & 37 & 29 \\ 21 & 13 & 5 & 28 & 20 & 12 & 4 \end{bmatrix}$$

**Figura 2.4.** Permutación PC1

Los bits 8, 16, 24, 32, 40, 48, 56 y 64 no están en la matriz de permutación debido a que son los bits de paridad. Por lo tanto, la clave permutada es de 56 bits:

K[1] = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

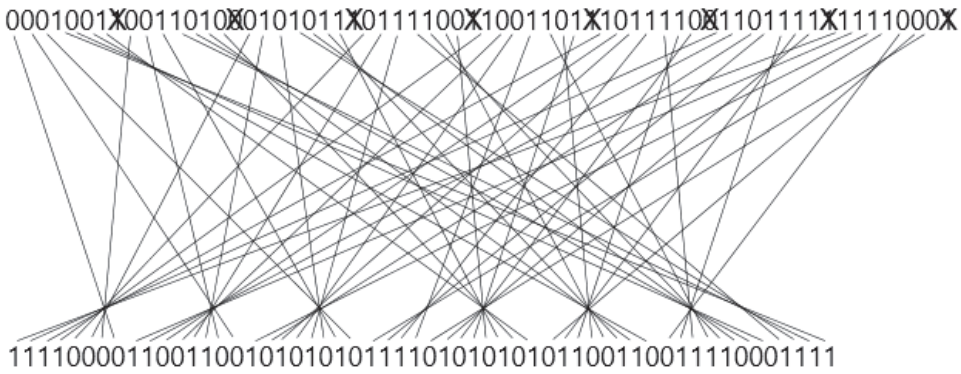


**Figura 2.5.** PC1 a la clave K

### 3. Desplazamiento izquierdo

Después de la permutación PC1, el vector resultante se divide en dos mitades $C_0$ y $D_0$, obteniéndose dos matrices de 28 bits cada una (7 filas x 4 columnas).

Para $C_n$ y $D_n$, se debe realizar un desplazamiento de 1 o 2 bits a la izquierda, es decir, en la primera ronda todos los bits se desplazarán una posición a la izquierda, excepto el primero que se desplazará al final del bloque. Para cada ronda se debe desplazar la cantidad indicada en la Figura 2.6, considerando que cada par de bloques $C_n$ y $D_n$ se genera a partir de los anteriores $C_{n-1}$ y $D_{n-1}$.

| RONDA | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bits a desplazarse | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

**Figura 2.6.** Bits de desplazamiento

En la Figura 2.7 se muestra el resultado de la ronda 1 del desplazamiento a la izquierda de una posición, tanto del vector $C_0$ y $D_0$.

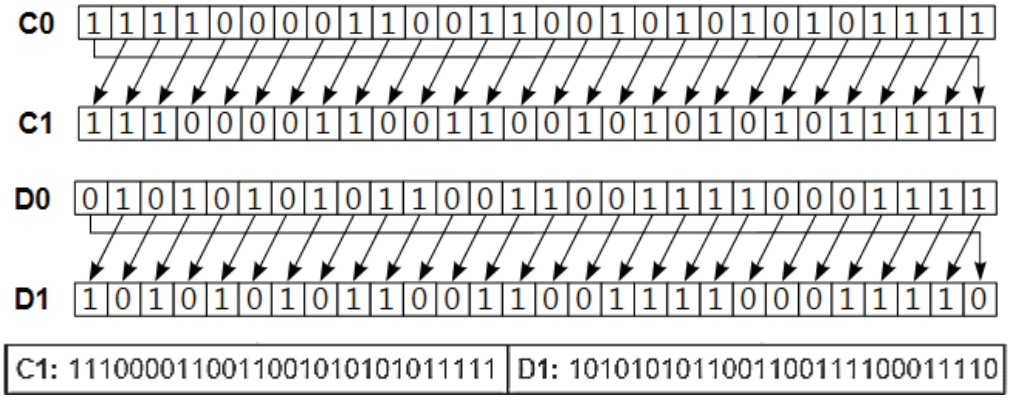


**Figura 2.7.** Resultado del desplazamiento

El resultado de las 16 iteraciones es el que se muestra en la Figura 2.8.

$C_1$ = 11100001100110010101011111
$D_1$ = 10101010110011001111100011110

$C_2$ = 11000011001100101010101111111
$D_2$ = 01010101100110011110001111101

$C_3$ = 00001100110010101010111111111
$D_3$ = 01010110011001111100011110101

$C_4$ = 00110011001010101011111111100
$D_4$ = 01011001100111110001111010101

$C_5$ = 11001100101010101111111110000
$D_5$ = 01100110011111000111101010101

$C_6$ = 00110010101010111111111000011
$D_6$ = 10011001111000111101010101101

$C_7$ = 11001010101011111111100001100
$D_7$ = 01100111100011110101010110

$C_8$ = 00101010101111111110000110011
$D_8$ = 10011110001111010101011001

$C_9$ = 01010101011111111100001100110
$D_9$ = 00111100011110101010110011

$C_{10}$ = 01010101111111110000110011001
$D_{10}$ = 11110001111010101011001100

$C_{11}$ = 01010111111110000110011001 01
$D_{11}$ = 11000111101010101100110011

$C_{12}$ = 01011111111000011001100101 01
$D_{12}$ = 00011110101010101100110011 11

$C_{13}$ = 01111111100001100110010101 01
$D_{13}$ = 01111010101010110011001111 00

$C_{14}$ = 11111110000110011001010101 01
$D_{14}$ = 11101010101011001100111100 01

$C_{15}$ = 11111000011001100101010101 11
$D_{15}$ = 10101010101100110011110000 111

$C_{16}$ = 11110000110011001010101011 11
$D_{16}$ = 01010101011001100111100011 11

**Figura 2.8.** Resultado de 16 rondas de $C_n$ y $D_n$

### 4. Permutación Inicial 2

Una vez realizado el desplazamiento, se concatenan $C_1$ y $D_1$ en una matriz denominada CD. A la nueva matriz CD de 56 bits se le realizará una nueva permutación denominada *Permuted Choice 2* (PC2), indicada en la Figura 2.9.

$$\begin{bmatrix} 14 & 17 & 11 & 24 & 1 & 5 \\ 3 & 28 & 15 & 6 & 21 & 10 \\ 23 & 19 & 12 & 4 & 26 & 8 \\ 16 & 7 & 27 & 20 & 13 & 2 \\ 41 & 52 & 31 & 37 & 47 & 55 \\ 30 & 40 & 51 & 45 & 33 & 48 \\ 44 & 49 & 39 & 56 & 34 & 53 \\ 46 & 42 & 50 & 36 & 29 & 32 \end{bmatrix}$$

**Figura 2.9.** Permutación PC2

El resultado es la subclave de la primera ronda K[1], la cual tiene 8 bits menos que CD, es decir tiene 48 bits, Figura 2.10.

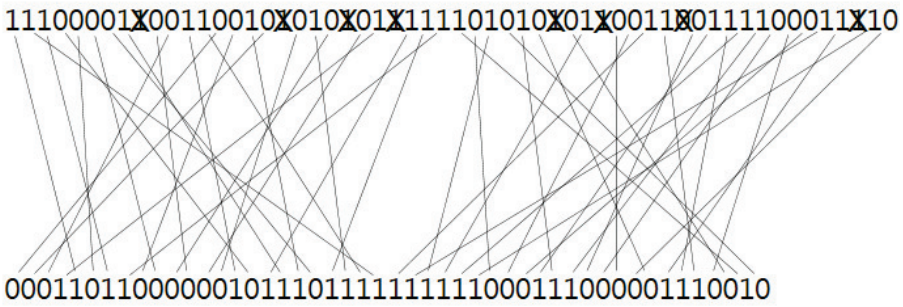


**Figura 2.10.** PC2 a CD

Son en total 16 rondas que se generan de la misma manera a partir de la clave inicial: K[1]… K[16], como se muestra en la Figura 2.11.

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010
$K_2$ = 011110 011010 111011 011001 110110 111100 100111 100101
$K_3$ = 010101 011111 110010 001010 010000 101100 111110 011001
$K_4$ = 011100 101010 110111 010110 110110 110011 010100 011101
$K_5$ = 011111 001110 110000 000111 111010 110101 001110 101000
$K_6$ = 011000 111010 010100 111110 010100 000111 101100 101111
$K_7$ = 111011 001000 010010 110111 111101 100001 100010 111100
$K_8$ = 111101 111000 101000 111010 110000 010011 101111 111011
$K_9$ = 111000 001101 101111 101011 111011 011110 011110 000001
$K_{10}$ = 101100 011111 001101 000111 101110 100100 011001 001111
$K_{11}$ = 001000 010101 111111 010011 110111 101101 001110 000110
$K_{12}$ = 011101 010111 000111 110101 100101 000110 011111 101001
$K_{13}$ = 100101 111100 010111 010001 111110 101011 101001 000001
$K_{14}$ = 010111 110100 001110 110111 111100 101110 011100 111010
$K_{15}$ = 101111 111001 000110 001101 001111 010011 111100 001010
$K_{16}$ = 110010 110011 110110 001011 000011 100001 011111 110101

**Figura 2.11.** Subclaves K

### 2. Cifrado del mensaje

El proceso de cifrado se lo realiza empleando el Esquema Feistel empleando las subclaves generadas anteriormente.

Una vez finalizados los prerrequisitos para el desarrollo del algoritmo, se procede a implementar la función principal DES, cuyo procedimiento se detalla en el diagrama de flujo de la Figura 2.12.

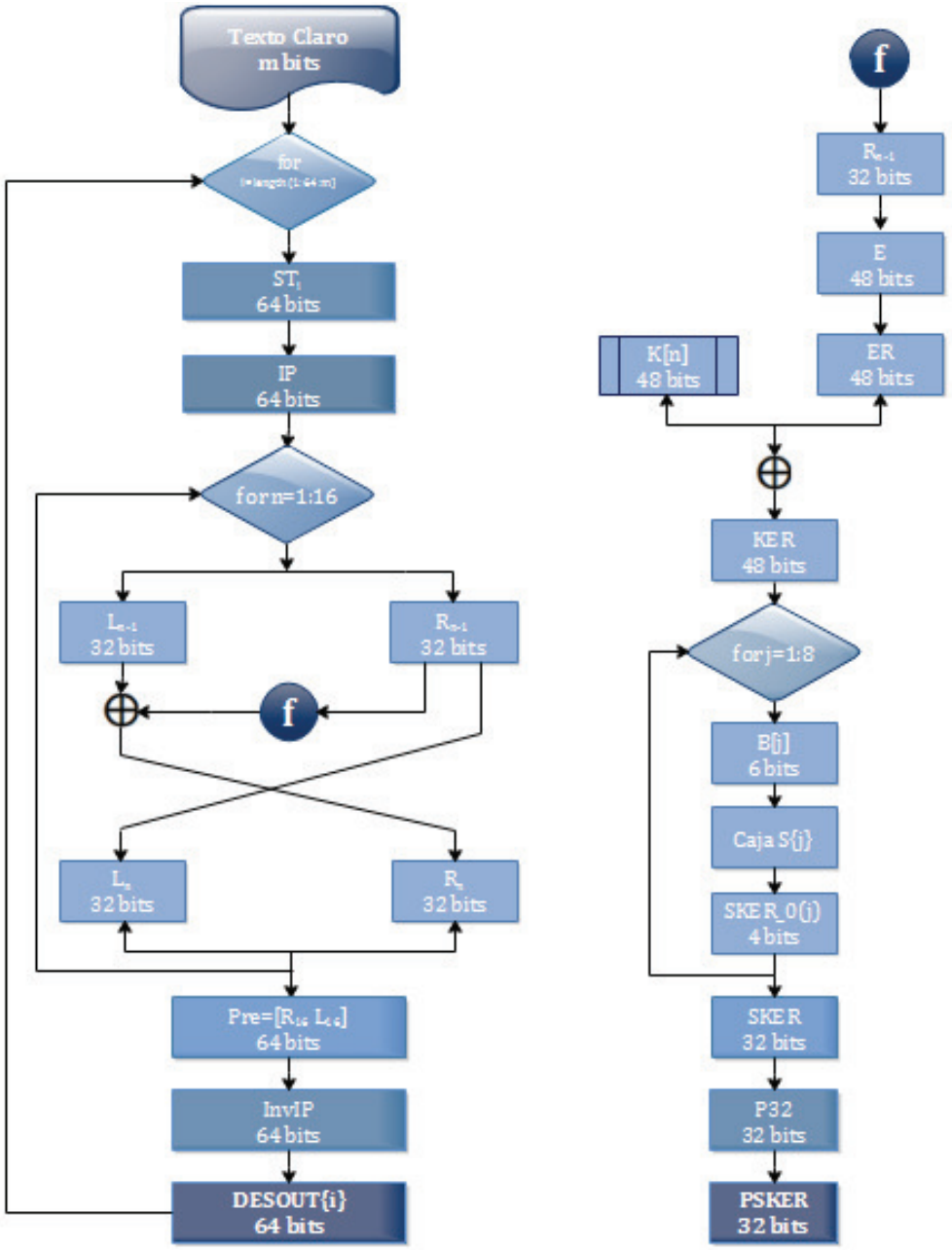


**Figura 2.12.** Diagrama de flujo de cifrado DES

### a. Bloque de Entrada X

A medida de ejemplo se va a cifrar el mensaje M = 0123456789ABCDEF, el cual se encuentra en formato hexadecimal, transformándolo a binario sería:

M = 00000001 00100011 01000101 01100111 10001001 10101011 11001101 11101111

En primer lugar, el texto se divide en bloques de 64 bits, como ese muestra en la Figura 2.13.

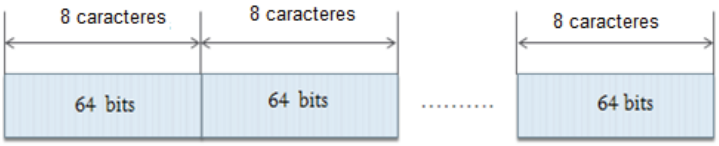


**Figura 2.13.** División del texto plano en bloques de 64 bits (8 caracteres)

### b. Permutación Inicial IP

Cada bit de un bloque está sujeto a una permutación inicial, que puede representarse mediante la matriz de Permutación Inicial (denominada IP), Figura 2.14.

$$\begin{bmatrix} 58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 \\ 60 & 52 & 44 & 36 & 28 & 20 & 12 & 4 \\ 62 & 54 & 46 & 38 & 30 & 22 & 14 & 6 \\ 64 & 56 & 48 & 40 & 32 & 24 & 16 & 8 \\ 57 & 49 & 41 & 33 & 25 & 17 & 9 & 1 \\ 59 & 51 & 43 & 35 & 27 & 19 & 11 & 3 \\ 61 & 53 & 45 & 37 & 29 & 21 & 13 & 5 \\ 63 & 55 & 47 & 39 & 31 & 23 & 15 & 7 \end{bmatrix}$$

**Figura 2.14.** Matriz de permutación inicial IP

Es interesante observar que los primeros 32 bits resultantes de la permutación contienen todos los bits que se encuentran en posición par en el mensaje inicial, mientas que la otra mitad contiene los bits en posición impar, como se puede observar en la Figura 2.15.

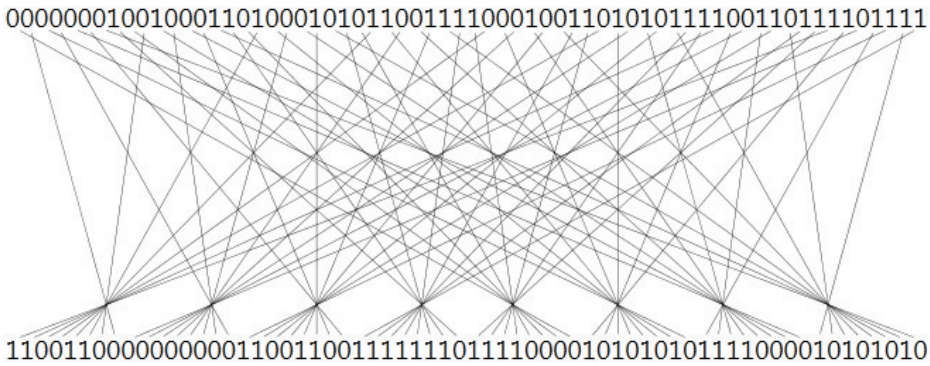


**Figura 2.15.** Permutación IP del mensaje

### c. División en bloques de 32 bits

Una vez que la permutación inicial se completó, el bloque de 64 bits se divide en dos bloques de 32 bits denominados $L_0$ y $R_0$ respectivamente (para izquierda y derecha, siendo la anotación en anglosajón L y R por *Left* y *Right*), Figura 2.16.

| $L_0$: 11001100000000000110011001111111 | $R_0$: 11110000101010101111000010101010 |
|---|---|

**Figura 2.16.** División de $L_0$ y $R_0$

### d. Función de expansión de $R_0$

En la función de expansión los 32 bits del bloque $R_0$ se expanden a 48 bits debido a la matriz de expansión que se denota como E, Figura 2.17, en la que los 48 bits se mezclan y 16 de ellos se duplican:

$$\begin{bmatrix} 32 & 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & 11 & 12 & 13 \\ 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 \\ 28 & 29 & 30 & 31 & 32 & 1 \end{bmatrix}$$

**Figura 2.17.** Matriz de expansión

En esta función el último bit de $R_0$, se convierte en el primero, el primero en el segundo, etc. Además, los bits 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28 y 29 de $R_0$ son duplicados y dispersos en la matriz, como se muestra en la Figura 2.18. El vector resultante de 48 bits se denomina ER.

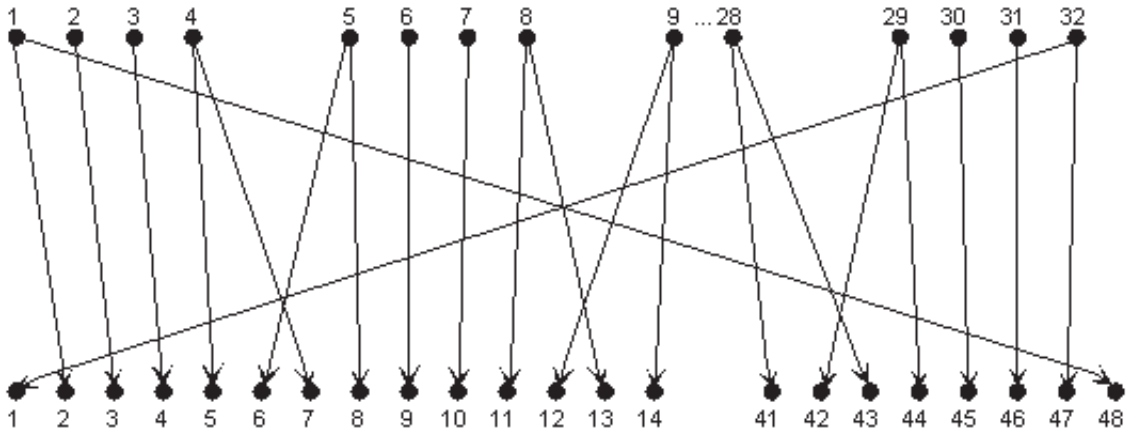


**Figura 2.18.** Función de expansión

### e. OR exclusiva con la clave

El algoritmo DES aplica la operación matemática binaria OR exclusiva entre la primera clave K[1] y ER, Figura 2.19.

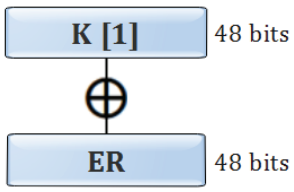


**Figura 2.19.** OR exclusiva con la clave

La operación XOR produce una salida verdadera si una y solo una de las entradas es verdadera, si se cumple que ambas entradas son verdaderas o falsas, la salida es falsa. Es decir, si ambas entradas son iguales la salida es falsa, y viceversa, Figura 2.20.


| A | B | A ⊕ B |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Figura 2.20.** Operación XOR

El resultado de este OR exclusivo es un vector de 48 bits, Figura 2.21.

```
K[1]: 000110110000001011101111111111000111000001110010
                              ⊕
ER:   011110100001010101010101011110100001010101010101
      ------------------------------------------------
KER:  011000010001011110111010100001100110010100100111
```

**Figura 2.21.** Operación XOR entre K1 y Exp

### f. División en 8 bloques

Luego la matriz resultante KER se debe dividir en 8 bloques de 6 bits cada uno (B1,B2,…,B8), como se indica en la Figura 2.22.

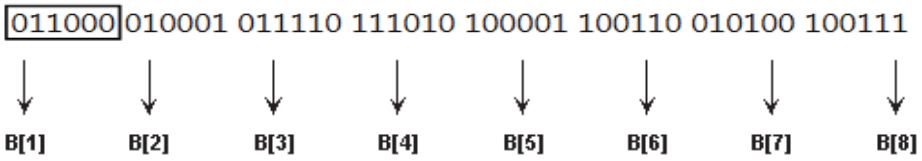


**Figura 2.22.** División en 8 bloques

### g. Cajas-S

Luego de la división se realiza una sustitución de cada bloque B, los cuales se sustituyen por valores presentes en las cajas S (cajas de sustitución) que se establecen con valores constantes de 8 matrices de dimensión 4x16, Figura 2.23.

| Fila | Columna | | | | | | | | | | | | | | | | S-Caja |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 | $S_1$ |
| 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 | |
| 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 | |
| 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 | |
| 0 | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 | $S_2$ |
| 1 | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 | |
| 2 | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 | |
| 3 | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 | |
| 0 | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 | $S_3$ |
| 1 | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 | |
| 2 | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 | |
| 3 | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 | |
| 0 | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 | $S_4$ |
| 1 | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 | |
| 2 | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 | |
| 3 | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 | |
| 0 | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 | $S_5$ |
| 1 | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 | |
| 2 | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 | |
| 3 | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 | |
| 0 | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 | $S_6$ |
| 1 | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 | |
| 2 | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 | |
| 3 | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 | |
| 0 | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 | $S_7$ |
| 1 | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 | |
| 2 | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 | |
| 3 | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 | |
| 0 | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 | $S_8$ |
| 1 | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 | |
| 2 | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 | |
| 3 | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 | |

**Figura 2.23.** Cajas S

Para cada S[n] se usan los primeros y últimos bits de B[n] como índice de filas, y los cuatro bits medios como el índice de la columna. Luego el valor binario es transformado a decimal para sustituir el valor de B[n] por el de S[n]. Para la primera Caja S de la Figura 2.24, se muestra un ejemplo para B[1] y S[1]:

B[1]: 011000

Fila (m): 0____0 → 0

Columna (n): _1100_ → 12

Salida S[1]: 5 → 0101

| Fila | Columna | | | | | | | | | | | | | | | S-Caja |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 | $S_1$ |
| 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 | |
| 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 | |
| 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 | |

**Figura 2.24.** Caja S[1]

Ahora en la Figura 2.25 se sustituye B[1] por el valor adquirido de la caja S.

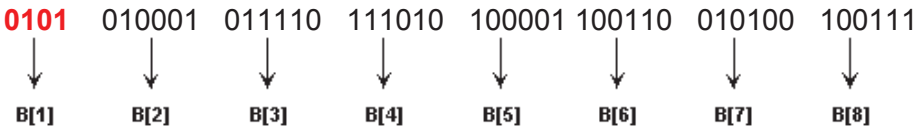


**Figura 2.25.** Sustitución de B[1]

Se realiza el mismo procedimiento de B[2] a B[8], y se concatenan en una matriz denominada SKER: 01011100100000101011010110010111.

Finalmente, SKER pasa a través de la permutación P de la Figura 2.26.

$$\begin{bmatrix} 16 & 7 & 20 & 21 \\ 29 & 12 & 28 & 17 \\ 1 & 15 & 23 & 26 \\ 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 \\ 32 & 27 & 3 & 9 \\ 19 & 13 & 30 & 6 \\ 22 & 11 & 4 & 25 \end{bmatrix}$$

**Figura 2.26.** Matriz de Permutación P

Tal como se indica en la Figura 2.27, que al igual que las Cajas S, esta matriz es constante, P es el resultado de la función $f(R_0, K_1)$.

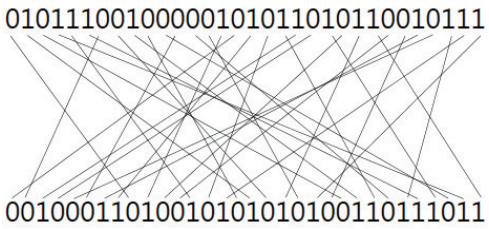


**Figura 2.27.** Permutación P de SKER

**OR exclusiva con la parte izquierda del mensaje**

El siguiente paso es aplicar la función de Feistel de la Ecuación 2.1, se realiza nuevamente la operación XOR entre $L_0$ y la función $f(R_0, K_1)$, el resultado de la operación se muestra en la Figura 2.28.

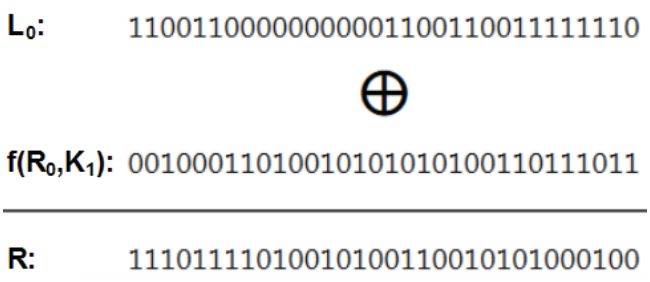


**Figura 2.28.** Operación XOR con la parte izquierda del mensaje

**h. Iteración**

El esquema Feistel se repite 16 veces, tal como se indicó en la Figura 2.1. En cada ronda se usa una subclave para realizar el cifrado de un bloque en específico.

Finalmente se intercambian L y R después de la ronda 16. El resultado se conoce como Pre-Salida.

Después de las 16 rondas, el primer bloque se cifra como se muestra en la Figura 2.29.

| Ronda | L | R |
|---|---|---|
| 0 | 11001100000000001100110011111110 | 11110000101010101111000010101010 |
| 1 | 11110000101010101111000010101010 | 11101111010010100110010101000100 |
| 2 | 11101111010010100110010101000100 | 11001100000000001011101110000100 0 |
| 3 | 11001100000000001011101110000100 0 | 10100010010111000000101111110100 |
| 4 | 10100010010111000000101111110100 | 01110111001000100000000001000100 |
| 5 | 01110111001000100000000001000100 | 10001010010011111010011000110110 |
| 6 | 10001010010011111010011000110110 | 11101001011001111100110101101000 |
| 7 | 11101001011001111100110101101000 | 00000110010010101011101000010000 |
| 8 | 00000110010010101011101000010000 | 11010101011010010100101110010000 |
| 9 | 11010101011010010100101110010000 | 00100100011111001100011001111010 |
| 10 | 00100100011111001100011001111010 | 10110111110101011101011110110010 |
| 11 | 10110111110101011101011110110010 | 11000101011110000011110001111000 |
| 12 | 11000101011110000011110001111000 | 01110101101111010001100001011000 |
| 13 | 01110101101111010001100001011000 | 00011000110000110001010101011010 |
| 14 | 00011000110000110001010101011010 | 11000010100011001001011000001100 |
| 15 | 11000010100011001001011000001100 | 01000011010000100011001000110100 |
| 16 | 01000011010000100011001000110100 | 00001010010011001101100110010100 |

**Figura 2.29.** Función de Feistel en las 16 rondas

### i. Permutación Inicial Inversa

Al final de las iteraciones, los dos bloques $\mathbf{L}_{16}$ y $\mathbf{R}_{16}$ se vuelven a concatenar y se someten a una permutación inicial inversa $IP^{-1}$, Figura 2.30.

$$\begin{bmatrix} 40 & 8 & 48 & 16 & 56 & 24 & 64 & 32 \\ 39 & 7 & 47 & 15 & 55 & 23 & 63 & 31 \\ 38 & 6 & 46 & 14 & 54 & 22 & 62 & 30 \\ 37 & 5 & 45 & 13 & 53 & 21 & 61 & 29 \\ 36 & 4 & 44 & 12 & 52 & 20 & 60 & 28 \\ 35 & 3 & 43 & 11 & 51 & 19 & 59 & 27 \\ 34 & 2 & 42 & 10 & 50 & 18 & 58 & 26 \\ 33 & 1 & 41 & 9 & 49 & 17 & 57 & 25 \end{bmatrix}$$

**Figura 2.30.** Matriz de Permutación Inicial Inversa

El resultado de la permutación se muestra en la Figura 2.31.

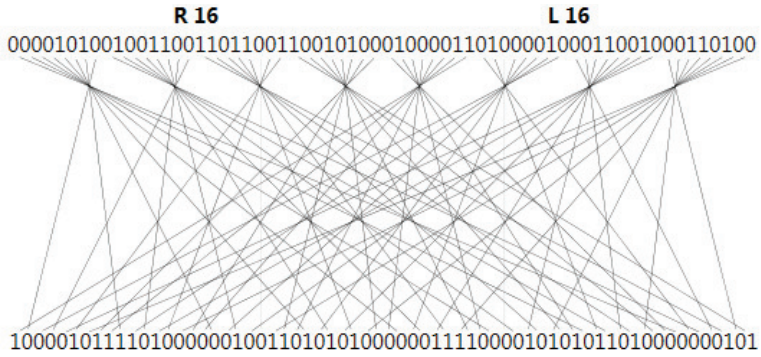


**Figura 2.31.** Permutación Inicial Inversa final

El resultado es el texto cifrado de 64 bits, el cual transformado en hexadecimal es el mensaje que se muestra en la Figura 2.32.

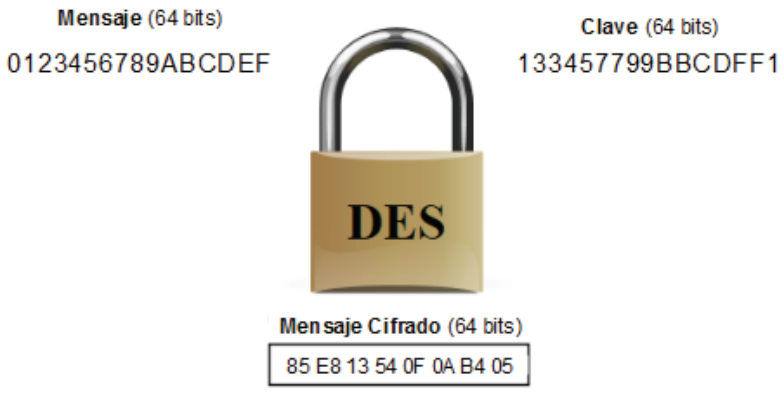


**Figura 2.32.** Texto Cifrado

La codificación en Matlab para el proceso de generación de subclaves se describe en el Segmento de código 2.1, en la cual se crean 16 iteraciones para generar la respectiva subclave.

```
%Permutación de la clave simétrica de 64 bits.
C_0=clave1(PC11); %Permutación a la primera mitad de la clave
D_0=clave1(PC12); %Permutación a la segunda mitad de la clave

K=cell(1,16); Se crea un arreglo de matrices de 1x16

%Generación de las 16 subclaves
for n=1:16
     %Desplazamiento izquierdo de la mitad C_0
     CT=circshift(C_0',-sum(Left_Shift(1:n)));
     C=CT'; %Transpuesta de la matriz
     DT=circshift(D_0',-sum(Left_Shift(1:n)));
     %Desplazamiento izquierdo de la mitad D_0
     D=DT'; %Transpuesta de la matriz
     CD=[C D]; %Concatenación de las dos mitades
     K{n}=CD(PC2); %Generación de las subllaves
end
```

**Segmento de código 2.1.** Generación de subclaves

La codificación en Matlab para el proceso de segmentación del mensaje en bloques de 64 bits se muestra en el Segmento de código 2.2, si el último bloque tiene menos de 64 bits se rellena con 0L al final.

```
%% SEGMENTACIÓN DE DATOS DE ENTRADA PARA CIFRADO
PT_0=dec2bin(a,8)'; %Transformación a binario del mensaje
PT_1=reshape(PT_0,1,numel(PT_0)); %Reubica los elementos de PT_0
PT_2=logical(PT_1=='1'); %Convierte a valores lógicos PT_0
PT_3=zeros(1,numel(PT_2)+(64-mod(numel(PT_2),64))); %Matriz 0L
PT_3(1:numel(PT_2))=PT_2; %Se hace un relleno de 0L
ST=reshape(PT_3,64,numel(PT_3)/64)'; %Segmenta el mensaje 64 x n
```

**Segmento de código 2.2.** Segmentación de datos binarios para el cifrado

La codificación en Matlab para cifrar el mensaje con el algoritmo simétrico DES se detalla paso a paso en el Segmento de código 2.3, en el cual se puede apreciar que la primera iteración está relacionada a la cantidad total de bloques de 64 bits, mientras que la segunda a las 16 rondas de DES.

```
%% CIFRADO EN BLOQUES DE 64-bits
DESOUT=cell(1,size(ST,1)); %% Asignación de espacio de 1 x n
for i=1:size(ST,1); %Cifrado DES de cada bloque del texto
    IPST_0=ST(i,:); %Selección de cada bloque para el cifrado
    IPST=IPST_0(IP); %Permutación IP del respectivo bloque
    L_0=IPST( 1:32); %Separación del bloque a la izquierda
    R_0=IPST(33:64); %Separación del bloque a la derecha
    R=cell(1,16); %Se crea un arreglo de matrices vacías 1 x 16
    L=cell(1,16); %Se crea un arreglo de matrices vacías 1 x 16
    L{1}=R_0; %Se almacena el vector R_0 de 32 elementos

%% 16 RONDAS de DES
    for n=1:16 %Inicio del lazo for para las 16 rondas
        if n == 1 %En la 1ra ronda función de expansión a R_0
            ER=R_0(E); %Función de expansión del lado derecho
        else %Función de expansión a R_0 de la ronda anterior
            ER=R{n-1}(E); %Función de expansión del lado derecho
        end %Fin lazo for
        KER=xor(K{n},ER); %Función XOR entre K{n} y ER
        SKER_0=[]; %Creación de una matriz vacía

        for j=1:8 %Iteración para cada caja-S
            posR=bin2dec(num2str([KER(6*j-5) KER(6*j)]));
            posC=bin2dec(num2str([KER(6*j-4) KER(6*j-3) KER(6*j-
2) KER(6*j-1)])); %Posición n de la Caja-S
            SKER_0(j)=S{j}(posR*16+posC+1);
        end
        SKER_N=logical(reshape(dec2bin(SKER_0,4)',1,32)=='1');
        PSKER=SKER_N(P32); %Permutación P

        if n == 1 %Iteración para el esqueme Feistel
            R{1}=xor(L_0,PSKER); %Operación XOR entre L0 y PSKER
        else %Para las demás rondas (2-16)
            R{n}=xor(L{n-1},PSKER); % XOR entre L y PSKER
            L{n}=R{n-1}; % L{n}=R{n} de la ronda anterior
        end
    end
     Pre=[R{16} L{16}]; DESbinary=Pre(InvIP);
     DESOUT{i}=binaryVectorToHex(DESbinary); %Salida
end
DES=horzcat(DESOUT{:}); %Concatenación de las salidas del bloque
```

**Segmento de código 2.3.** Cifrado de mensaje

### Algoritmo de Descifrado DES

El proceso de descifrado de DES es metódicamente el mismo que el de cifrado, tiene dos etapas: la generación de las subclaves y el cifrado. A partir de la clave simétrica se generan las 16 subclaves que serán empleadas para cifrar, siendo el mismo procedimiento que el descrito en el diagrama de flujo de la Figura 2.2, la diferencia en la segunda etapa de

descifrado es que el texto de entrada es el texto cifrado, y las claves se las emplea para la operación XOR en orden inverso, esto es, en la primera ronda se debe usar la subclave K[16] en forma descendente hasta llegar a la última ronda, en la cual se debe usar la subclave K[1], tal como se muestra en el diagrama de flujo de la Figura 2.33.

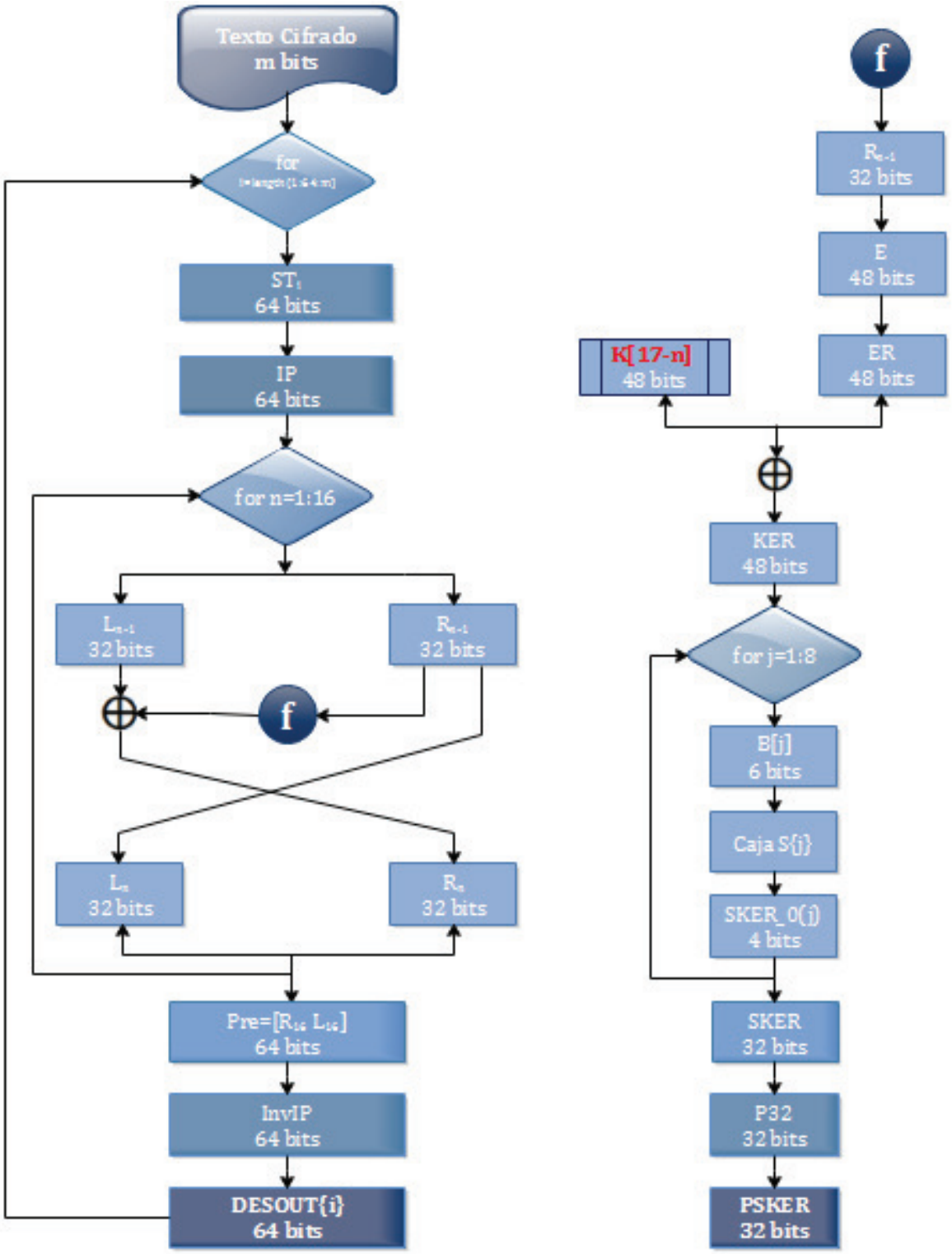


**Figura 2.33.** Diagrama de flujo del algoritmo de descifrado DES

### 2.2.2. Algoritmo de cifrado RSA

#### Etapas de cifrado

Este algoritmo consta de dos etapas para el cifrado: Generación del par de claves y cifrado del mensaje. En la Figura 2.34 se observa el diagrama de flujo del algoritmo para el proceso de cifrado del mensaje.

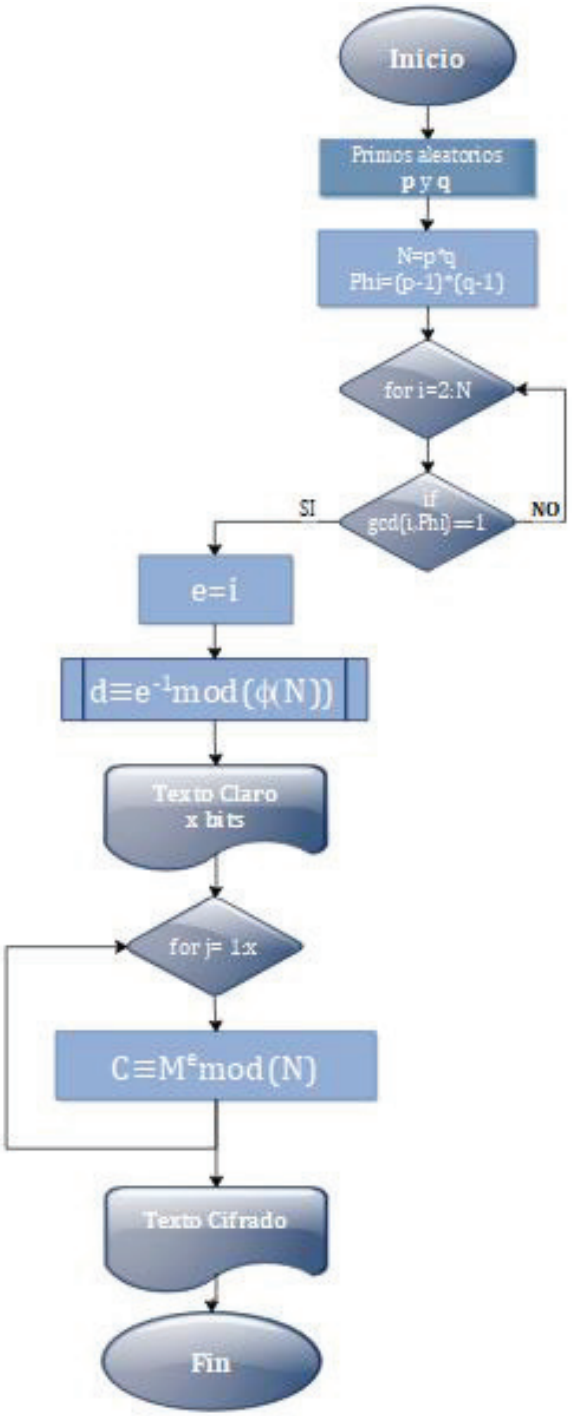


**Figura 2.34.** Diagrama de flujo del cifrado del Algoritmo RSA

A diferencia del sistema de clave simétrica, cada usuario $i$ que participa en la comunicación posee dos claves $(e_i, d_i)$, donde: $e_i$ es la clave pública y $d_i$ es la clave privada.

1. Para generar las claves, se debe elegir aleatoriamente dos números primos distintos de gran tamaño, denominados $p$ y $q$, los cuales se recomienda que sean del mismo tamaño para una máxima seguridad. Los valores mínimos de estos dos números que se recomienda para los certificados digitales X.509 es de 1024 bits cada uno, cuyo producto es un número de 2048 bits, esto es unos 620 dígitos, lo cual resulta un problema de difícil solución, pero asegura un margen de seguridad razonable, y por el mismo motivo se lo utiliza en aplicaciones y certificados digitales.

2. Se calcula el producto de dichos números: $N = pq$. Este número es utilizado tanto por las claves públicas como privadas y proporciona el enlace entre ellas.

3. Cada usuario calcula la Función de Euler $\phi(N) = (p-1)(q-1)$, cuyo valor debe ser secreto y únicamente debe ser conocido por su propietario. Los números p y q ya no serán útiles, por lo tanto, deben descartarse, pero nunca revelarse.

   Esta función indica la cantidad de números primos relativos con $N$ y menores que $N$.

   En el Segmento de código 2.4 se describe la implementación de los pasos 1 al 3.

```
%Selección arbitraria de dos números primos p y q
num=randsample(primes(2^30),2);
p=num(1); %Número primo p
q=num(2); %Número primo q
N=p*q; %Hallar el producto N=p×q
Phi=(p-1)*(q-1); %Calcular la función indicador de Euler
```

**Segmento de código 2.4.** Selección arbitraria de números primos

4. Seleccionar un número entero randómico $e$, coprimo de $\phi(N)$, de modo que el Máximo Común Divisor entre $\phi(N)$ y $e$ sea 1: $mcd(e, \phi(N)) = 1$, si $1 < e < \phi(N)$. Este valor será la clave pública, cuya codificación se muestra en el Segmento de código 2.5.

```
%CLAVE PÚBLICA
%Calcular un entero e primo menor que ϕ(N) / mcd(e, ϕ(N))=1
for i=2:N
        if gcd(i,Phi)==1 %Comprobación si el módulo es 1
            break;
        end
end
e=i; %Clave pública es el valor donde se cumple la igualdad.
```

**Segmento de código 2.5.** Generación clave pública

5. Calcular el inverso multiplicativo de $e$ usando el Algoritmo Extendido de Euclides para computarizar el entero $d$, que verifique que: $ed \equiv 1(\mathrm{mod}\,\phi(N))$, tal que $1 < d < \phi(N)$. Este valor será la clave privada. El Segmento de código 2.6 muestra la línea de comandos en la cual llama a la función del Algoritmo Extendido de Euclides y posteriormente se muestra la función que implementa dicho Algoritmo, cuya teoría se detalla en el ANEXO III.

```
%CLAVE PRIVADA
%Calcular el inverso de e, tal que e×d=1(mod ϕ(N))
[d]=Euclidesf(e,Phi);

%Algoritmo Extendido de Euclides
function [h ] = Euclides1(a,b)
%Inicialización de variables
x=[1 0]; %Valor constante de x
y=[0 1]; %Valor constante de y
q=0; r=0; m=0; n=0; %Inicialización de variables

while a~=0 %Lazo hasta encontrar el inverso
    q=fix(b/a); %Redondear la división
    r=mod(b,a); %Módulo de los números b y a
    m=x(2)-q*x(1); %Ecuaciones de Euclides
    n=y(2)-q*y(1); %Ecuaciones de Euclides
    x(2)=x(1); %Almacenar temporalmente el valor de x(1)
    y(2)=y(1); %Almacenar temporalmente el valor de y(1)
    x(1)=m; y(1)=n; %Almacenar temporalmente el valor de m y n
    b=a; a=r; %Los nuevos b y a serán a y r respectivamente
end
    if x(2)>0 %Si el valor obtenido es mayor a 0
        h=x(2) %El inverso es x(2)
    else %Si el valor obtenido es menor a 0
        h=x(1)+x(2) %El inverso es x(1)+x(2)
    end
end
```

**Segmento de código 2.6.** Generación clave privada - Algoritmo Extendido de Euclides

6. La clave pública está constituida por ( $N$ , $e$ ), la cual debe ser publicada en el directorio a todo el mundo que desee enviar mensajes a dicho destinatario.

7. Guardar $d$ como la clave privada, que solamente el destinatario de la comunicación debe conocerla para poder descifrar el mensaje.

- **Cifrado del mensaje**

  El cifrado del mensaje se lo realiza por exponenciación modular de exponente y módulo fijo, de la siguiente manera:

8. Transformar a código ASCII cada caracter del texto original.

9. Utilizar la función de la Ecuación 2.2:

$$C \equiv M^e mod(N)$$

**Ecuación 2.2.** Cifrado RSA

siendo $M^e$ el mensaje original y $C$ el mensaje cifrado, empleando la clave pública ( $N$ , $e$ ).

El Segmento de código 2.7 muestra la codificación en Matlab para el cifrado de RSA.

```
%FUNCIÓN DE CIFRADO ASIMÉTRICO RSA
function[cifra]=rsa1(a)
global N e; %Valores globales de N y e
x=length(a); %Conocer la longitud del mensaje

c=double(a); %Convertir a decimal los valores de la clave
disp(['El código ASCII de la Clave es: ' num2str(c)]);

%Proceso de cifrado
for j= 1:x
     cifra(j)=mod(c(j)^e,N); %C=M^e mod (N)

end
end
```

**Segmento de código 2.7.** Cifrado RSA

Los fundamentos matemáticos para el proceso de cifrado y descifrado de RSA se encuentran disponibles en el ANEXO III.

**Ejemplo con números pequeños**

1. Se escoge dos número primos, por ejemplo, $p = 11$ y $q = 19$.
2. Producto $N = 11(19) = 209$.
3. Función de Euler: $\phi(209) = (11-1)(19-1) = 10(18) = 180$, lo cual significa que en el ejemplo hay 180 números enteros primos relativos con $N = 209$.
4. Para encontrar $e$ se selecciona el valor de 7, ya que $mcd(7,180) = 1$.
5. Empleando el Algoritmo Extendido de Euclides se encuentra el valor de $d$ para $e = 7$, el valor $d \equiv e^{-1} \bmod(\phi(N))$ es igual a 103.
6. La clave pública está constituida por $(N, e) = (34,3)$, que puede publicar en un directorio público.
7. La clave privada es $d = 103$, debe mantenerse en secreto.
8. Cifrar el mensaje empleando la función $C \equiv M^e mod(N)$, como se muestra en la Tabla 2.1.

**Tabla 2.1.** Texto cifrado RSA

| Texto Plano | E | P | N |
|---|---|---|---|
| Texto en decimal | 69 | 80 | 78 |
| Mensaje cifrado $C \equiv M^e mod(N)$ | 31 | 196 | 166 |

**Algoritmo de Descifrado RSA**

En la Figura 2.35 se puede observar el diagrama de flujo del proceso de descifrado del algoritmo RSA. El cálculo principal en el descifrado de RSA es la exponenciación modular dada por la Ecuación 2.3:

$$\boldsymbol{M \equiv C^d (mod\ N)}$$

**Ecuación 2.3.** Descifrado RSA

Donde $C$ es el mensaje de texto cifrado, $d$ es la clave privada que es un secreto, y $N$ es el producto de dos primos grandes $p$ y $q$ . Existen algunos métodos diferentes para calcular una exponenciación modular, como el Algoritmo Binario de Exponenciación de Cuadrados

y Múltiplos, el Teorema Remanente Chino (CRT) y el Algoritmo de Exponenciación de Ventanas Deslizantes (SWE).

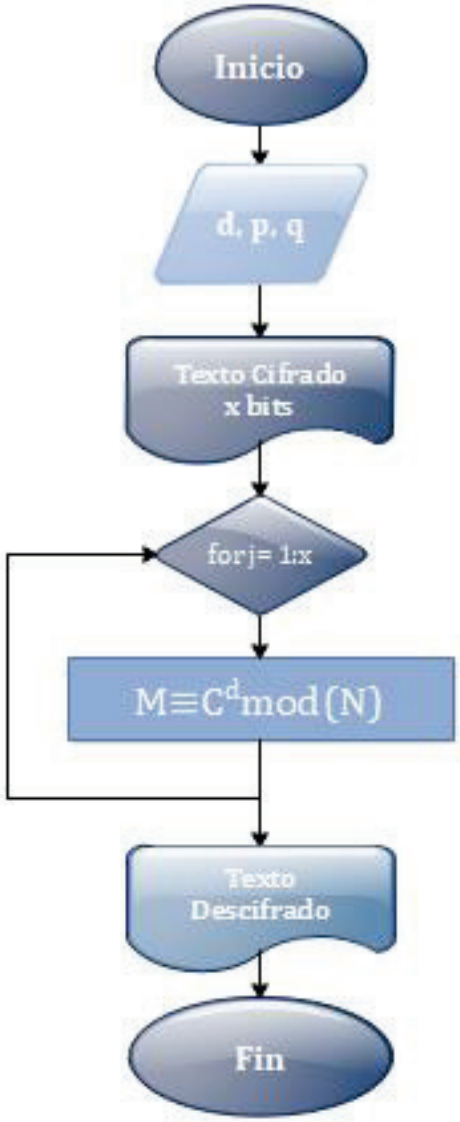


**Figura 2.35.** Diagrama de flujo del descifrado del Algoritmo RSA

## 2.3. Servicio de Integridad

### 2.3.1. Algoritmo de *hash* MD5

MD5 produce un valor de longitud fija de *hash* de 128 bits, lo cual equivale a 32 caracteres hexadecimales, independientemente del tamaño de la entrada de mensaje. MD5 es muy resistente a colisiones y fue diseñado para generar valores únicos de *hash* para cada entrada única de datos. El algoritmo de resumen MD5 se muestra en el diagrama de flujo de la Figura 2.36.

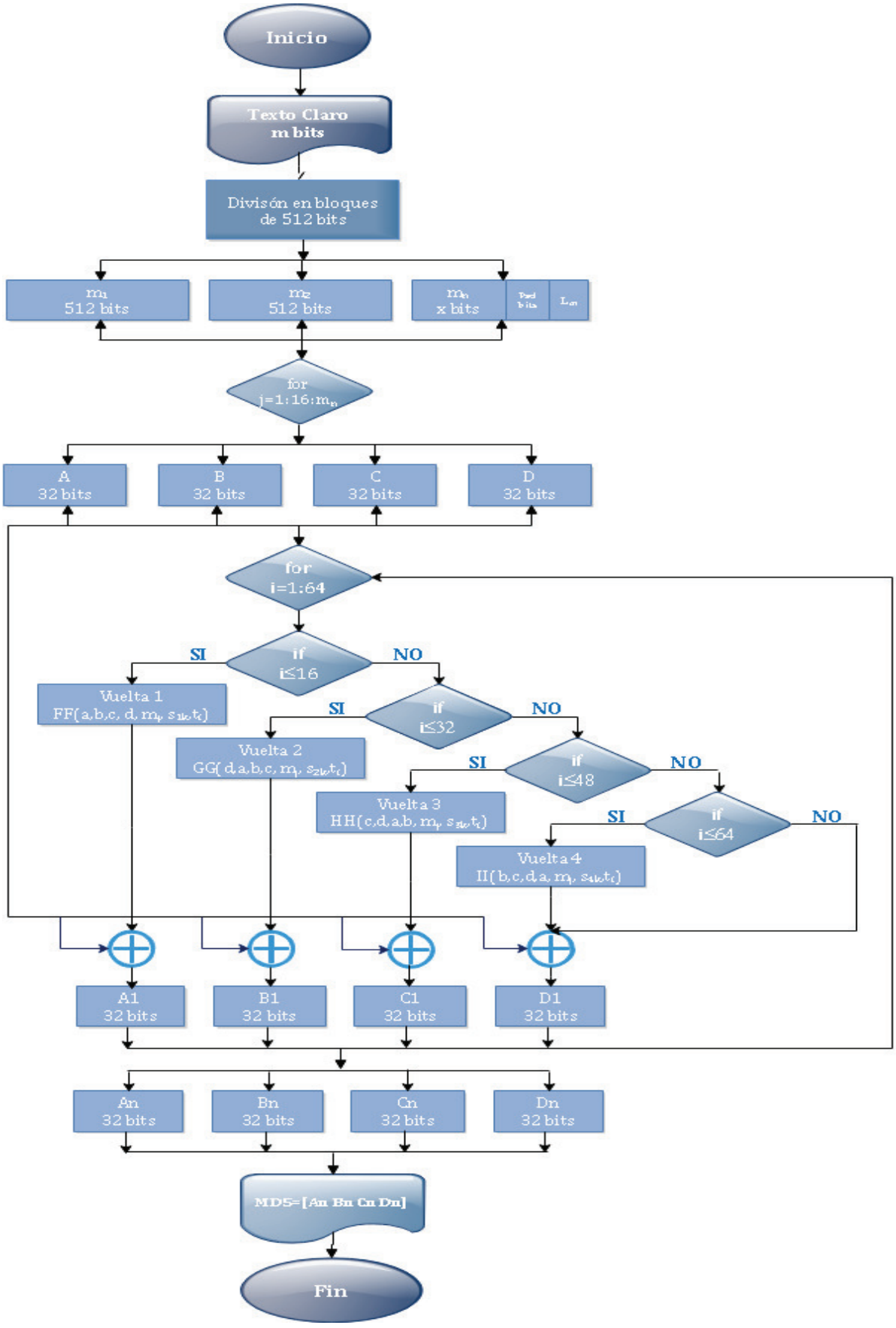


**Figura 2.36.** Diagrama de Flujo del Algoritmo MD5

**Descripción del Algoritmo MD5**

MD5 procesa la información en representación *Little-endian*, es decir los bits menos significativos se leen primero y después los bits más significativos.

El cálculo del valor de resumen de MD5 se realiza en etapas separadas que procesan cada bloque de datos de 512 bits junto con el valor calculado en la etapa anterior [29].

El proceso que realiza el algoritmo para generar el resumen MD5 se describe a continuación:

- La entrada de datos se divide en bloques de 512 bits, divididos en 16 sub-bloques de 32 bits cada uno, añadiendo un relleno en el último bloque si es necesario. Después del último bit de datos se añade un 1L seguido de tantos 0L como sea necesario hasta completar el bloque de 448 bits, los últimos 64 bits se emplean para indicar el tamaño del mensaje.

- Antes de procesar el primer bloque de datos, se inicializan 4 palabras iniciales públicas de 32 bits cada uno, los cuales se conoce como A, B, C, D:

$$A = 01\ 23\ 45\ 67$$
$$B = 89\ ab\ cd\ ef$$
$$C = fe\ dc\ ba\ 98$$
$$D = 76\ 54\ 32\ 10$$

  Posteriormente empieza el lazo principal del algoritmo, el cual se repetirá para cada bloque de 512 bits del mensaje.

- Para empezar la operación MD5 se definen 4 funciones auxiliares, cada una de ellas toma como entrada 3 palabras iniciales y devuelve una salida de 32 bits :

$$F(X,Y,Z) = (X \wedge Y) \vee ((\sim X) \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee ((Y \wedge (\sim Z))$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee (\sim Z))$$

  Los operadores $\oplus$, $\wedge$, $\vee$, $\sim$ son las funciones XOR, AND, OR y NOT respectivamente.

- Luego se definirán las siguientes 4 funciones que se aplicarán a cada bit del sub-bloque:

$$FF(a, b, c, d, m_j, s, t_i) \Rightarrow a = b + ((a + F(b, c, d) + m_j + t_i) <<< s)$$

$$GG(a, b, c, d, m_j, s, t_i) \Rightarrow a = b + ((a + G(b, c, d) + m_j + t_i) <<< s)$$

$$HH(a, b, c, d, m_j, s, t_i) \Rightarrow a = b + ((a + H(b, c, d) + m_j + t_i) <<< s)$$

$$II(a, b, c, d, m_j, s, t_i) \Rightarrow a = b + ((a + I(b, c, d) + m_j + t_i) <<< s)$$

Donde: - $+$ es la suma en módulo $2^{32}$.

- $m_j$ es el j-ésimo sub-bloque de 32 bits del mensaje $m$.

- $f <<< s$ Representa desplazar circularmente a la izquierda $s$ bits a la entrada $f$.

- $t_i$ $t_{[1...64]}$ es el número entero resultante de multiplicar $2^{32}$ (4294967296) veces la función sinusoidal: $abs(sin(i))$, donde $i$ es el número de vuelta expresado en radianes.

El valor en decimal se transforma a hexadecimal:

$$t = \begin{bmatrix} d76aa478 & e8c7b756 & 242070db & c1bdceee \ldots \\ f57c0faf & 4787c62a & a8304613 & fd469501 \ldots \\ 698098d8 & 8b44f7af & ffff5bb1 & 895cd7be \ldots \\ 6b901122 & fd987193 & a679438e & 49b40821 \ldots \\ f61e2562 & c040b340 & 265e5a51 & e9b6c7aa \ldots \\ d62f105d & 2441453 & d8a1e681 & e7d3fbc8 \ldots \\ 21e1cde6 & c33707d6 & f4d50d87 & 455a14ed \ldots \\ a9e3e905 & fcefa3f8 & 676f02d9 & 8d2a4c8a \ldots \\ fffa3942 & 8771f681 & 6d9d6122 & fde5380c \ldots \\ a4beea44 & 4bdecfa9 & f6bb4b60 & bebfbc70 \ldots \\ 289b7ec6 & eaa127fa & d4ef3085 & 4881d05 \ldots \\ d9d4d039 & e6db99e5 & 1fa27cf8 & c4ac5665 \ldots \\ f4292244 & 432aff97 & ab9423a7 & fc93a039 \ldots \\ 655b59c3 & 8f0ccc92 & ffeff47d & 85845dd1 \ldots \\ 6fa87e4f & fe2ce6e0 & a3014314 & 4e0811a1 \ldots \\ f7537e82 & bd3af235 & 2ad7d2bb & eb86d31 \end{bmatrix}$$

- $s$ es una matriz de valores constantes.

Por lo tanto, para cada bloque de 512 bits se realizan un total de 64 operaciones MD5, en la Figura 2.37 se muestra el diagrama de bloque de una operación MD5.

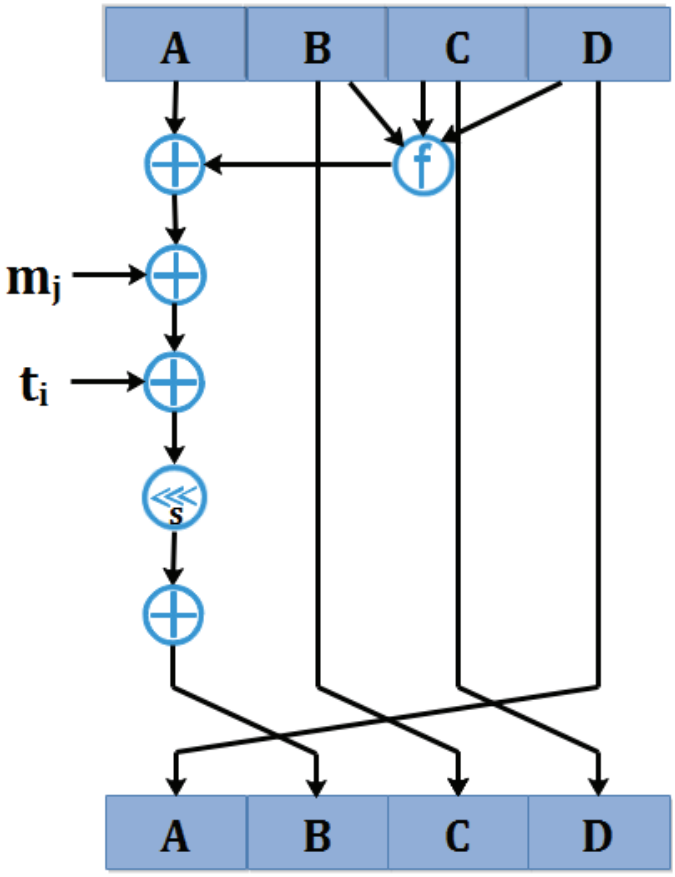


**Figura 2.37.** Función de una operación MD5

Para cada bloque de mensaje de 64 bits se realizan cuatro vueltas de operaciones MD5. El valor de $m_j$ que se empleará será está definido en función de la vuelta, tal como se indica en la Tabla 2.2.

**Tabla 2.2.** Selección del valor de $\boldsymbol{m_j}$ en cada vuelta

| Vuelta | $\boldsymbol{m_j}$ $(\boldsymbol{i = 0:63})$ |
|---|---|
| 1 | $i + 1$ |
| 2 | $(5i + 1) mod 16 + 1$ |
| 3 | $(3i + 5) mod 16 + 1$ |
| 4 | $m_j = (7i) mod 16 + 1$ |

En la Tabla 2.3 se muestra la función resultante para cada vuelta de las operaciones MD5.

**Tabla 2.3.** Operaciones MD5 en función de cada vuelta

| Función | | | | $t_i$ $(i = 1{:}64)$ | | | | $s_{vk}$ $(k = 1{:}4)$ | | | | Función Resultante en función de la vuelta |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vuelta | | | | Vuelta | | | | Vuelta | | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | |
| $FF$ | $GG$ | $HH$ | $II$ | $t_1$ $t_5$ $t_9$ $t_{13}$ | $t_{17}$ $t_{21}$ $t_{25}$ $t_{29}$ | $t_{33}$ $t_{37}$ $t_{41}$ $t_{45}$ | $t_{49}$ $t_{53}$ $t_{57}$ $t_{61}$ | $s_{11}$ | $s_{21}$ | $s_{31}$ | $s_{41}$ | $(a, b, c, d, m_j, s_{vk}, t_i,)$ |
| | | | | $t_2$ $t_6$ $t_{10}$ $t_{14}$ | $t_{18}$ $t_{22}$ $t_{26}$ $t_{30}$ | $t_{34}$ $t_{38}$ $t_{42}$ $t_{46}$ | $t_{50}$ $t_{54}$ $t_{58}$ $t_{62}$ | $s_{12}$ | $s_{22}$ | $s_{32}$ | $s_{42}$ | $(d, a, b, c, m_j, s_{vk}, t_i,)$ |
| | | | | $t_3$ $t_7$ $t_{11}$ $t_{15}$ | $t_{19}$ $t_{23}$ $t_{27}$ $t_{31}$ | $t_{35}$ $t_{39}$ $t_{43}$ $t_{47}$ | $t_{51}$ $t_{55}$ $t_{59}$ $t_{63}$ | $s_{13}$ | $s_{23}$ | $s_{33}$ | $s_{43}$ | $(c, d, a, b, m_j, s_{vk}, t_i,)$ |
| | | | | $t_4$ $t_9$ $t_{12}$ $t_{16}$ | $t_{20}$ $t_{24}$ $t_{28}$ $t_{32}$ | $t_{36}$ $t_{40}$ $t_{44}$ $t_{48}$ | $t_{52}$ $t_{56}$ $t_{60}$ $t_{64}$ | $s_{14}$ | $s_{24}$ | $s_{34}$ | $s_{44}$ | $(b, c, d, a, m_j, s_{vk}, t_i,)$ |

Con los valores de $i$ comprendidos entre 0 y 63 es posible generar el valor de $j$ correspondiente para cada vuelta de la operación MD5, la cual se muestra en la Tabla 2.4.

A la función resultante de la suma binaria se le debe realizar el proceso de desplazamiento a la izquierda, dependiendo el valor de la vuelta y de la matriz $s$:

$$s = \begin{bmatrix} 7 & 12 & 17 & 22 \\ 5 & 9 & 14 & 20 \\ 4 & 11 & 16 & 23 \\ 6 & 10 & 15 & 21 \end{bmatrix}$$

Los valores de las palabras obtenidas en la vuelta anterior se convertirán en las palabras de la vuelta en curso, realizando el siguiente intercambio:

- La palabra $A_1$ es el valor anterior de D.
- La palabra $C_1$ es el valor anterior de B.
- La palabra $D_1$ es el valor anterior de B.
- La palabra $B_1$ resulta de la suma binaria del valor anterior de B con el resultado de la función calculada.

**Tabla 2.4.** Valores de $m_j$ en función de la vuelta

| Vuelta $v$ | | | |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| $m_1$ | $m_2$ | $m_6$ | $m_1$ |
| $m_2$ | $m_7$ | $m_9$ | $m_8$ |
| $m_3$ | $m_{12}$ | $m_{12}$ | $m_{15}$ |
| $m_4$ | $m_1$ | $m_{15}$ | $m_6$ |
| $m_5$ | $m_6$ | $m_2$ | $m_{13}$ |
| $m_6$ | $m_{11}$ | $m_5$ | $m_4$ |
| $m_7$ | $m_{16}$ | $m_8$ | $m_{11}$ |
| $m_8$ | $m_5$ | $m_{11}$ | $m_2$ |
| $m_9$ | $m_{10}$ | $m_{14}$ | $m_9$ |
| $m_{10}$ | $m_{15}$ | $m_1$ | $m_{16}$ |
| $m_{11}$ | $m_4$ | $m_4$ | $m_7$ |
| $m_{12}$ | $m_9$ | $m_7$ | $m_{14}$ |
| $m_{13}$ | $m_{14}$ | $m_{10}$ | $m_5$ |
| $m_{14}$ | $m_3$ | $m_{13}$ | $m_{12}$ |
| $m_{15}$ | $m_8$ | $m_{16}$ | $m_3$ |
| $m_{16}$ | $m_{13}$ | $m_3$ | $m_{10}$ |

Este procedimiento de sustitución se realiza para cada palabra, por lo tanto son 64 vueltas, una vez finalizadas, al valor final de cada palabra se realiza una suma binaria con el valor anterior de la palabra correspondiente:

$$A_1 = A_1 + A$$
$$B_1 = B_1 + B$$
$$C_1 = C_1 + C$$
$$D_1 = D_1 + D$$

Los valores resultantes serán las nuevas 4 palabras que se concatenarán con el segundo bloque de 512 bits de entrada de texto. Se realiza el mismo procedimiento en los siguientes bloques, hasta el último bloque de 512 bits. Finalmente, la concatenación de los valores ABCD resultantes de la última vuelta de 32 bits cada uno es el valor del resumen de 128 bits.

# 3. RESULTADOS Y DISCUSIÓN

La interfaz gráfica se desarrolla en Matlab, empleando su entorno de programación visual GUIDE. En el

ANEXO II se detalla el funcionamiento básico, la instalación y configuración del *software*.

La aplicación del Sistema de Seguridad Informática consta de una ventana Principal y seis ventanas secundarias. A continuación, se expondrá la vista principal de cada una de las ventanas, detallando las pruebas para probar el funcionamiento de cada algoritmo implementado.

En la Figura 3.1 se muestra la vista de la pantalla principal, la cual contiene un grupo de selección que permite ejecutar cada uno de los algoritmos.


**Figura 3.1.** Pantalla Principal

### Condiciones de realización de pruebas

Para todas las pruebas las condiciones de operación fueron las siguientes:

- *Hardware*: Computador persona, con las siguientes características:
  - Procesador Intel Core i7 @ 4 GHz
  - 8 GB de memoria RAM
  - Disco duro de 1 TB

- *Software*: Entorno de Matlab 8.5.0 (R2017a) ejecutándose bajo Windows 10 Pro de 64 bits

A continuación, se describirán las pruebas de funcionamiento de cada módulo desarrollado en la aplicación principal.

## 3.1. Servicio de Confidencialidad

### 3.1.1. Algoritmo DES

En la Figura 3.2 se muestra la vista de la aplicación de uso didáctica implementada del algoritmo DES.

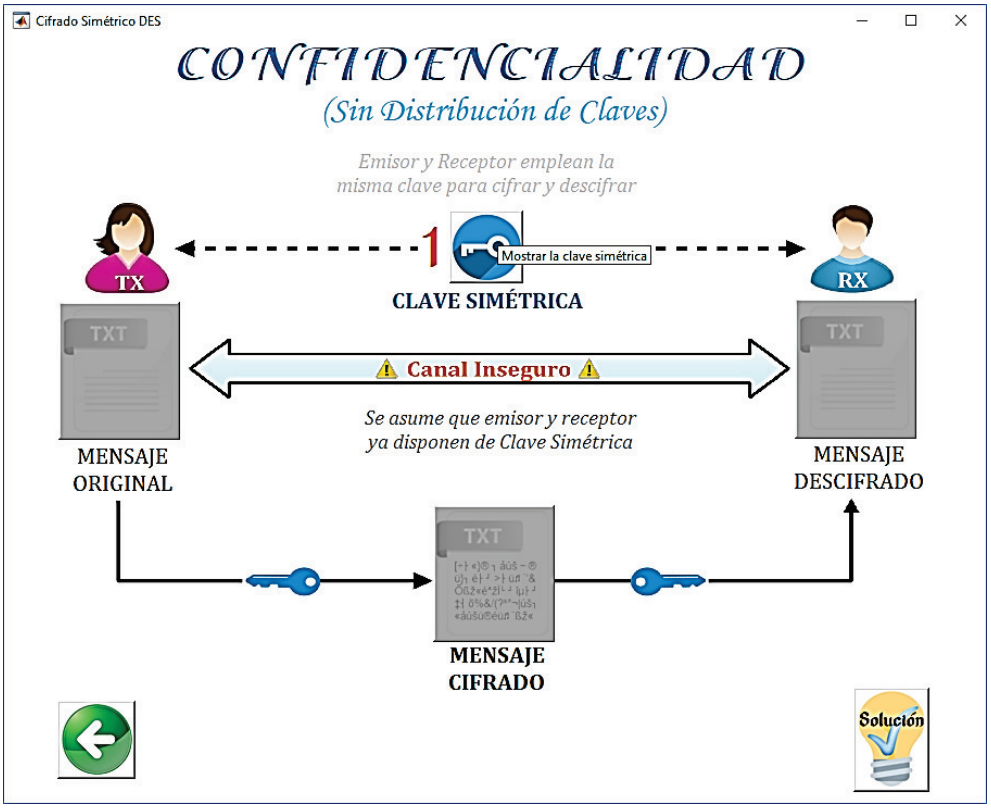


**Figura 3.2.** Aplicación del algoritmo de cifrado simétrico DES

La interfaz gráfica diseñada es de fácil uso, didáctica y con botones interactivos, de modo que sea de fácil entendimiento para el usuario.

El código fuente que corresponde a la implementación de la interfaz gráfica de usuario del algoritmo simétrico DES desarrollado se muestra en el ANEXO V.

### Cifrado con el algoritmo simétrico DES

Antes de cifrar el archivo, en el lado del usuario emisor se autogenera la clave simétrica, la cual debe ser conocida por el receptor para descifrar el mensaje. En la Figura 3.3 se indica la clave generada por el emisor para cifrar el mensaje.


**Figura 3.3.** Clave simétrica auto generada por la aplicación

A continuación, el emisor podrá seleccionar un archivo en texto claro del computador y al seleccionar el botón de MENSAJE CIFRADO, hace que se cifre el mensaje original. Por ejemplo, se va a cifrar un mensaje que tiene el alfabeto en mayúsculas, minúsculas, números, y caracteres especiales, el cual se muestra en la Figura 3.4.

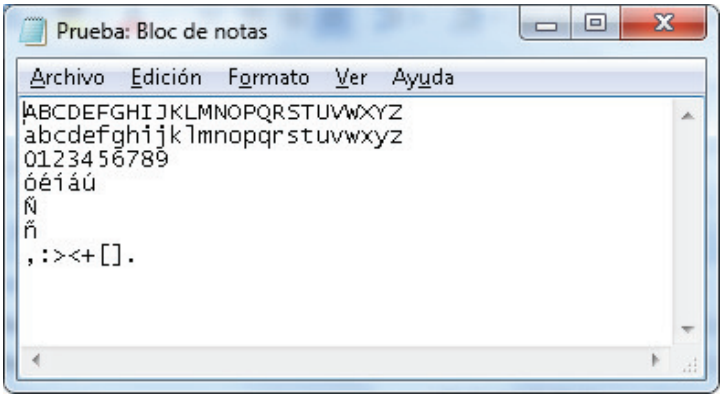


**Figura 3.4.** Archivo de prueba para cifrar con el algoritmo DES

El mensaje cifrado con el algoritmo simétrico que se muestra en la Figura 3.5 se guarda automáticamente en un archivo de texto y se abre para visualización del usuario.

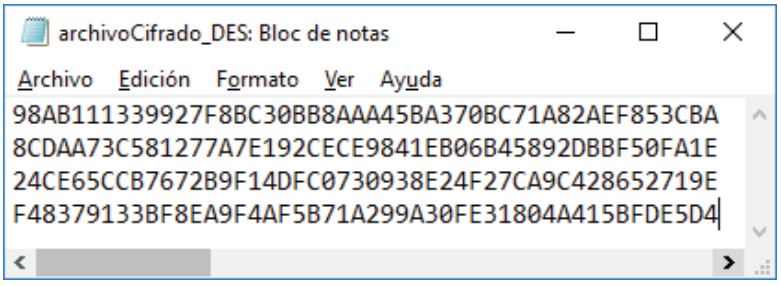


**Figura 3.5.** Archivo cifrado con el algoritmo simétrico DES

### *Software* CrypTool

Para comprobar la correcta implementación de cada algoritmo se utilizará el programa CrypTool versión 1.4.40, el cual es un *software* libre y *open source* que permite aplicar y analizar mecanismos criptográficos en el sistema operativo Windows. Este programa tiene implementado casi la mayoría de las funciones criptográficas y permite aprender acerca de la criptografía clásica y moderna.

### Comprobación de cifrado del algoritmo simétrico DES con el *software* CrypTool

Con la finalidad de comprobar que el algoritmo simétrico DES se ha implementado correctamente, se compara el mensaje cifrado con el *software* CrypTool.

A continuación, en la Figura 3.6 se puede observar el mensaje a ser cifrado, el cual es el mismo de la Figura 3.4 y su respectivo mensaje cifrado empleando la misma clave simétrica de la Figura 3.3.

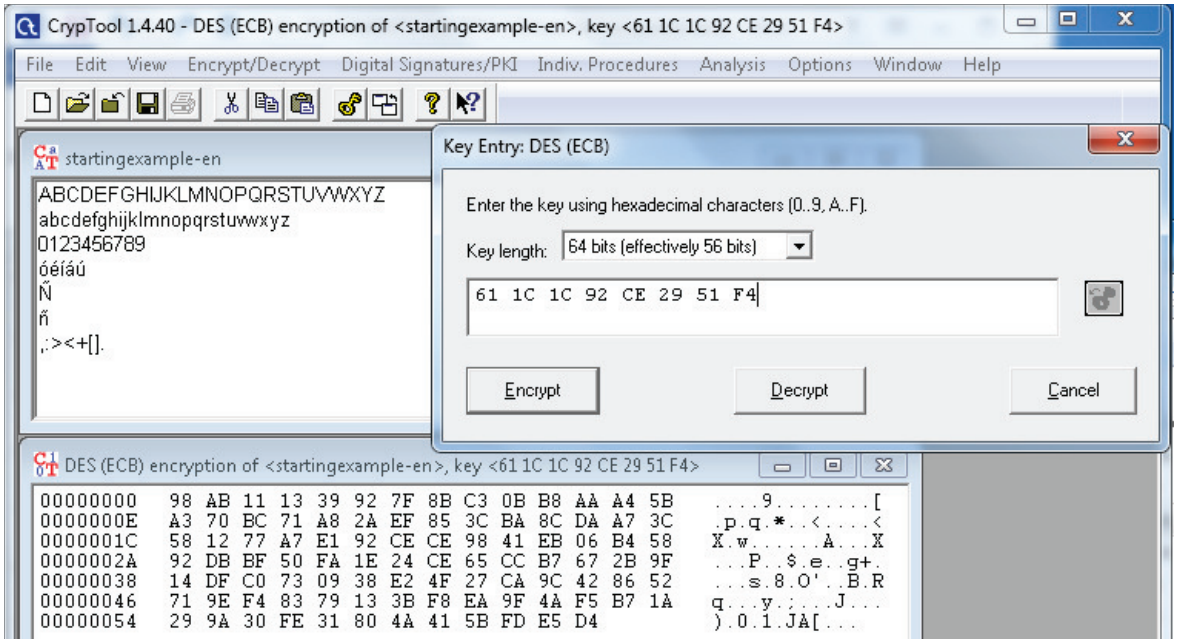


**Figura 3.6.** Cifrado DES del mensaje con el *software* CrypTool

Se puede comprobar que el mensaje cifrado empleando el algoritmo simétrico DES implementado en Matlab es exactamente el mismo que el mensaje cifrado con el *software* CrypTool. De esta manera se garantiza que el algoritmo simétrico DES ha sido desarrollado correctamente.

### Descifrado con el algoritmo simétrico DES

Luego de verificar la correcta implementación del algoritmo se procede a descifrar el mensaje enviado.

En el lado del receptor, se emplea la clave simétrica enviada por el emisor para poder descifrar el mensaje, el cual, de igual manera, una vez descifrado se almacena y se muestra al receptor para comprobar que el mensaje descifrado es el mismo que el original. En la Figura 3.7 se muestra el archivo descifrado, y se comprueba que tiene el mismo contenido que el mensaje original enviado, de esta manera se garantiza que la información es confidencial.

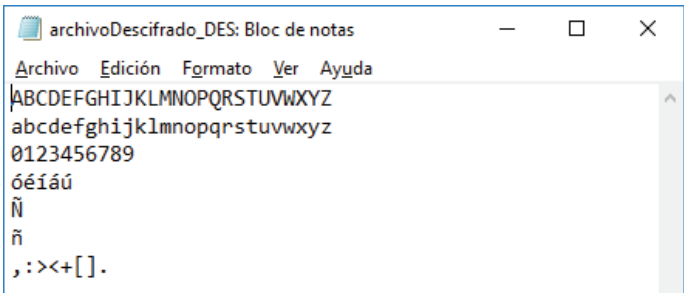


**Figura 3.7.** Mensaje descifrado con el algoritmo simétrico DES

## 3.1.2. Algoritmo RSA

En la Figura 3.8 se muestra la vista de la aplicación de uso didáctico del algoritmo RSA implementada.

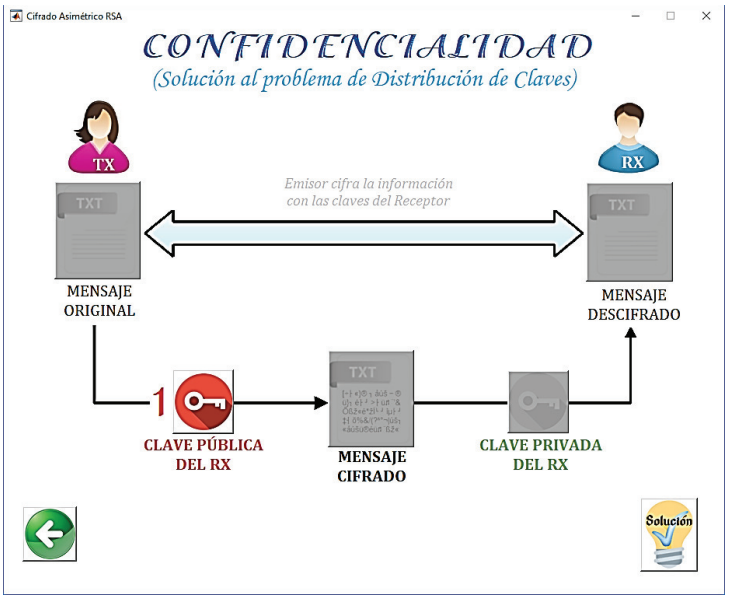


**Figura 3.8.** Aplicación del algoritmo de cifrado asimétrico RSA

El código fuente que corresponde a la implementación de la interfaz gráfica de usuario del algoritmo asimétrico RSA desarrollado se muestra en el ANEXO V.

### Cifrado con el algoritmo asimétrico RSA

Antes de que el emisor pueda enviar un mensaje es necesario que el receptor genere su clave pública y el emisor pueda emplear esta clave para cifrar el mensaje. Ambas claves se generan automáticamente en la aplicación y se muestran en la Figura 3.9

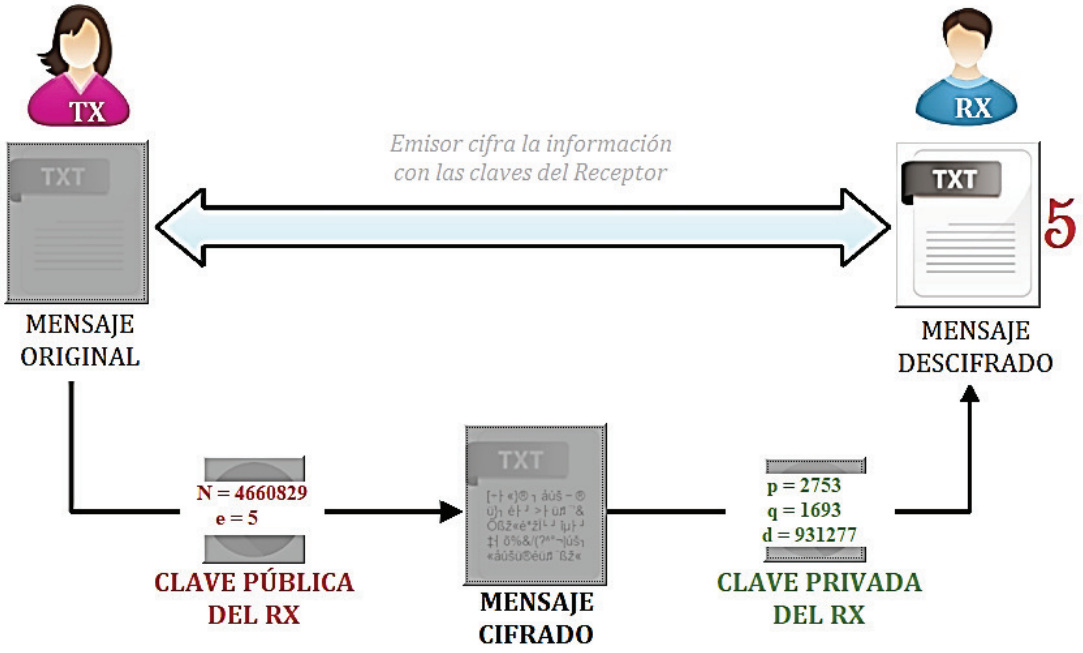


**Figura 3.9.** Claves pública y privada del Receptor

Con la clave pública, el emisor podrá seleccionar un archivo en texto claro del computador y al presionar el botón de cifrado se efectuará dicho proceso, el cual dependiendo del tamaño de la clave empleada para cifrar y del tamaño del texto, variará el tiempo en el cual se cifre. En este caso, se ha cifrado el mismo documento que se empleó para cifrar con el algoritmo DES, Figura 3.4.

El mensaje cifrado con el algoritmo asimétrico que se muestra en la Figura 3.10 se guarda automáticamente en un archivo de texto.

### Comprobación de las claves de cifrado del algoritmo asimétrico RSA con el *Software* CrypTool

El *software* CrypTool no permite cifrar texto de gran longitud, sin embargo, si es posible al menos generar las claves pública y privada.

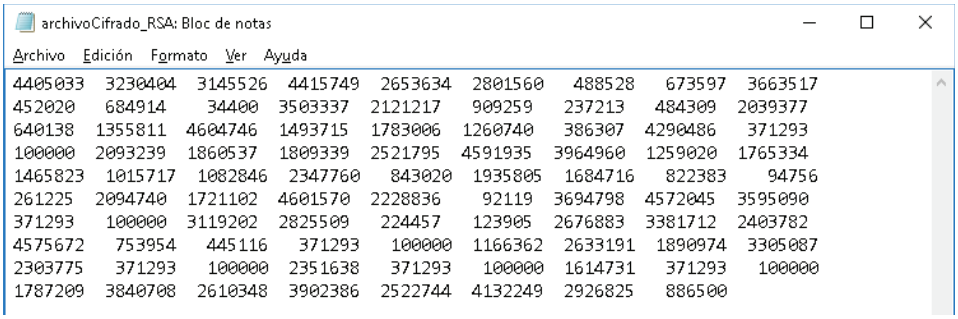


**Figura 3.10.** Archivo cifrado con el algoritmo asimétrico RSA

Con la finalidad de comprobar que las claves del algoritmo asimétrico RSA se han generado correctamente, se comparan con el *software* CrypTool. En la Figura 3.11 se pueden observar las claves pública y privada generadas, las cuales son las mismas de la Figura 3.9.

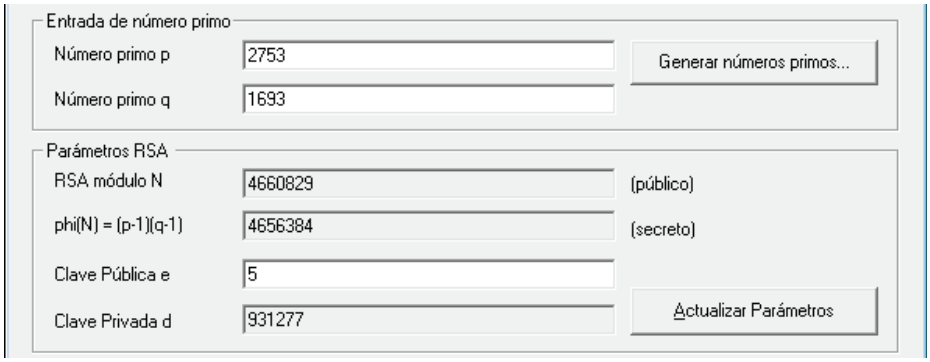


**Figura 3.11.** Generación de claves pública y privada con el *software* CrypTool

Por lo tanto, se puede comprobar que las claves pública y privada generadas empleando las funciones de generación de claves implementadas en Matlab son exactamente las mismas que las generadas con el *software* CrypTool. De esta manera se garantiza que las claves del algoritmo asimétrico RSA han sido desarrolladas correctamente en nuestra implementación.

### Descifrado con el algoritmo asimétrico RSA

Luego de verificar la correcta implementación de las claves de cifrado se procede a descifrar el mensaje enviado. En el lado del receptor, se emplea la clave privada del receptor para poder descifrar el mensaje, el cual, de igual manera, una vez descifrado se almacena y se muestra al receptor para comprobar que el mensaje descifrado es el mismo que el original.

En la Figura 3.12 se muestra el archivo descifrado, y se comprueba que tiene el mismo contenido que el mensaje original enviado, de esta manera se garantiza que la información es confidencial.

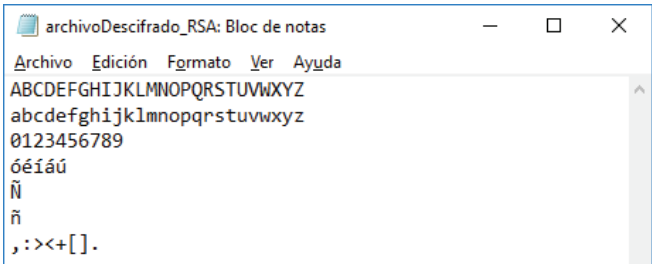


**Figura 3.12.** Mensaje descifrado con el algoritmo asimétrico RSA

### 3.1.3. Pruebas de tiempos de ejecución de DES versus RSA

Para comparar el tiempo de ejecución que toma cifrar un archivo de la misma longitud de mensaje empleando el algoritmo simétrico DES y también con el asimétrico RSA, se utiliza el comando `tic – toc` de Matlab, el cual muestra el tiempo que se demora en ejecutarse una porción de código. La comparación de ambos algoritmos se muestra en la Tabla 3.1.

El algoritmo simétrico DES en promedio es más rápido que el algoritmo asimétrico RSA, por lo cual se concluye que es no es útil para cifrar mensajes de gran longitud, y se lo emplea para cifrar mensajes de corta longitud, como claves, por ejemplo.

Por este motivo es importante analizar la criptografía híbrida, que se detalla a continuación.

**Tabla 3.1.** Comparación de los tiempos de ejecución del algoritmo DES con el RSA

| Longitud del mensaje | Tiempo de ejecución [seg] | | Razón |
|---|---|---|---|
| | DES | RSA | $t_{RSA}/t_{DES}$ |
| 1 palabra | 0.13 | 1.65 | 12.69 |
| 2 palabras | 0.99 | 2.1 | 2.12 |
| 4 palabras | 1.13 | 3.36 | 2.97 |
| 8 palabras | 1.39 | 5.95 | 4.28 |
| 16 palabras | 1.94 | 11.22 | 5.78 |
| 32 palabras | 3.02 | 21.42 | 7.09 |
| 64 palabras | 5.2 | 42.21 | 8.12 |
| 128 palabras | 9.66 | 89.12 | 9.23 |
| 512 palabras | 35.99 | 377.57 | 10.49 |
| 1024 palabras | 70.29 | 691.2 | 9.83 |
| 2048 palabras | 138.27 | 1385.7 | 10.02 |

### 3.1.4. Criptografía híbrida

En base a las desventajas del algoritmo asimétrico implementado anteriormente, y con el aprovechamiento de este tipo de cifrado para la criptografía híbrida se diseñó una nueva interfaz gráfica en la que únicamente va a ser posible cifrar una clave simétrica del usuario almacenada en texto claro en el computador. En la Figura 3.13 se muestra la vista de la aplicación de uso didáctica implementada de la criptografía híbrida.

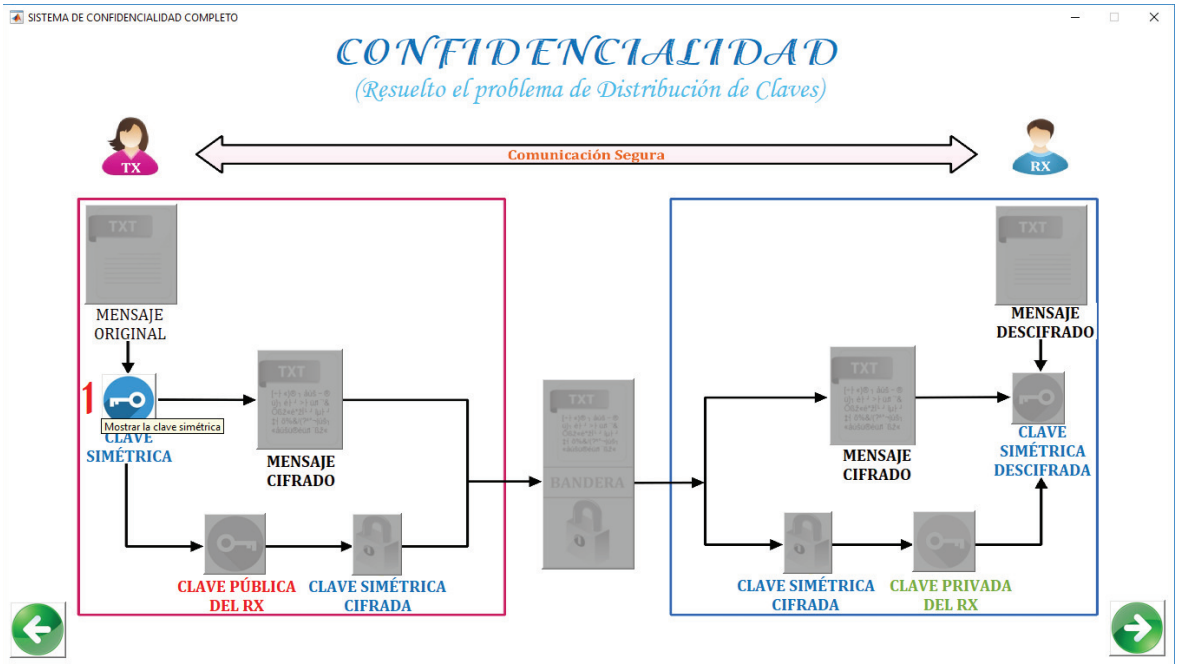


**Figura 3.13.** Aplicación de la criptografía híbrida

El código fuente que corresponde a la implementación de la interfaz gráfica de usuario de la criptografía híbrida desarrollada se muestra en el ANEXO V.

#### Cifrado del texto con el algoritmo simétrico DES

Para el cifrado del texto se sigue el siguiente procedimiento:

a) Seleccionar un archivo de texto claro donde se encuentre almacenada la clave secreta simétrica, la cual será la misma que la aplicada en el cifrado simétrico DES, Figura 3.3.
b) Seleccionar un mensaje de texto claro de la computadora que se desea enviar confidencialmente por la red, el cual será el mismo mensaje que se cifró en el cifrado simétrico DES, Figura 3.4.

c) Cifrar el mensaje utilizando la clave secreta simétrica, aplicando el algoritmo de cifrado simétrico DES. El mensaje cifrado con el algoritmo simétrico DES va a ser el mismo que se indicó en la Figura 3.5, pues es el mismo mensaje aplicada la misma clave simétrica, se guarda automáticamente en un archivo de texto y se abre el documento para visualización del usuario.

**Cifrado de la clave simétrica empleando el algoritmo asimétrico RSA**

Asimismo, para el cifrado de la clave simétrica se sigue el siguiente procedimiento:

a) Se obtiene la clave pública del receptor, la cual se indica en la Figura 3.14.

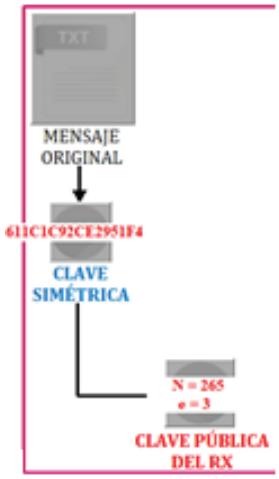


**Figura 3.14.** Clave pública del receptor para cifrado de la clave simétrica

b) Se cifra la clave secreta simétrica con la clave pública del receptor aplicando el algoritmo de cifrado asimétrico RSA. La clave simétrica cifrada se muestra en la Figura 3.15.

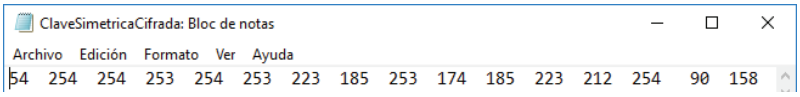


**Figura 3.15.** Clave simétrica cifrada con el algoritmo asimétrico RSA

c) El emisor transmite tanto la clave simétrica cifrada junto con el mensaje cifrado al receptor. Para poder distinguir el mensaje cifrado de la clave simétrica cifrada se ha empleado una bandera formada de una combinación binaria de cinco 0L y cinco 1L, esta va a estar localizada después del texto cifrado y antes de la clave simétrica cifrada, tal como se observa en la Figura 3.16.

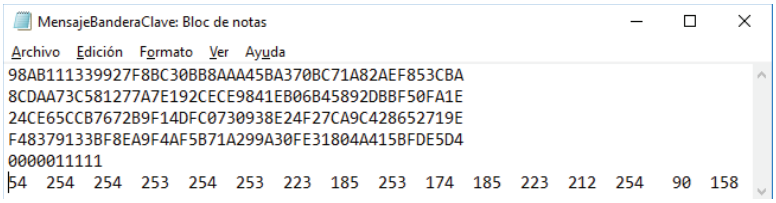


**Figura 3.16.** Mensaje cifrado con DES y clave simétrica cifrada con RSA

### Comprobación del cifrado de la clave simétrica empleando el algoritmo asimétrico RSA implementado con el *Software* CrypTool

El *software* CrypTool permite cifrar texto de corta longitud, en este caso se va cifrar la clave simétrica, la cual contiene únicamente 16 dígitos hexadecimales.

Con la finalidad de comprobar que la clave simétrica cifrada empleando el algoritmo asimétrico RSA implementado se ha generado correctamente, se compara con el *software* CrypTool. A continuación, en la Figura 3.17 se puede observar las claves pública y privada generadas y la clave simétrica cifrada.

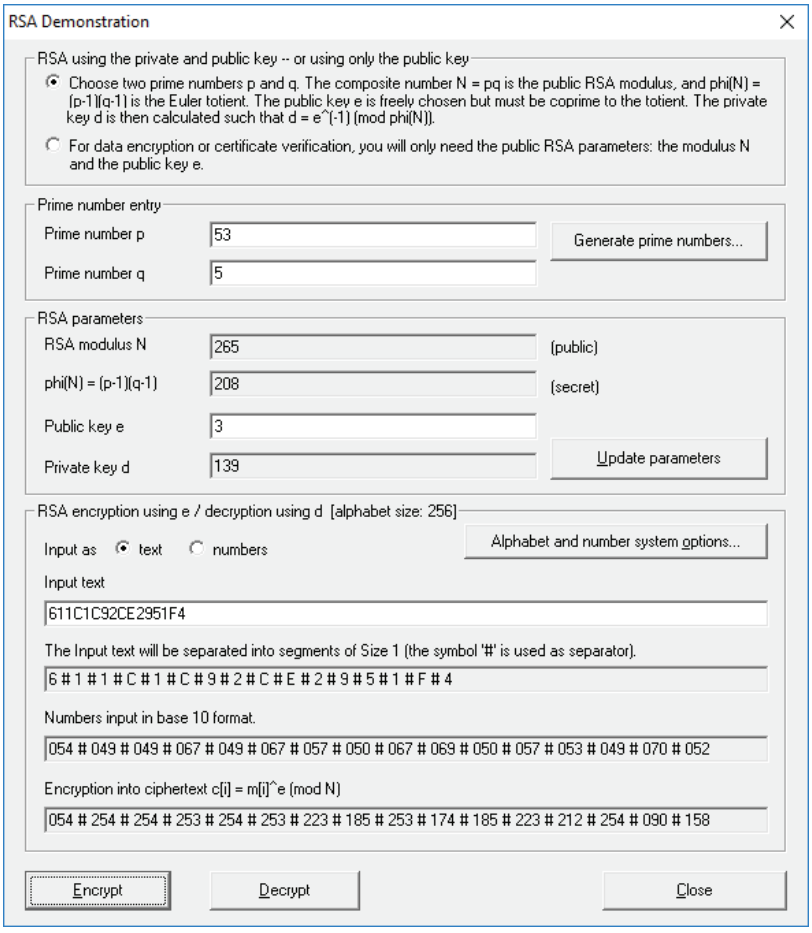


**Figura 3.17.** Cifrado de la clave simétrica con el *software* CrypTool

Por lo tanto, se puede comprobar que las claves pública y privada generadas empleando el algoritmo asimétrico RSA implementado en Matlab son exactamente las mismas que las generadas con el *software* CrypTool. La comprobación del cifrado de mensaje con el algoritmo simètrico DES se realizó previamente en la sección Algoritmo DES. Así se garantiza que tanto el cifrado del mensaje como el cifrado de la clave simétrica han sido desarrollados correctamente.

**Descifrado de la información**

Luego de verificar la correcta implementación del algoritmo RSA se procede a descifrar el mensaje enviado.

Al igual que en el proceso de cifrado, para descifrar el mensaje y la clave simétrica se sigue el siguiente procedimiento:

a) Se emplea la clave privada del receptor para descifrar la clave secreta simétrica, la cual se muestra en la Figura 3.18.


**Figura 3.18.** Clave privada del receptor para descifrado de la clave simétrica

b) Se emplea la clave secreta simétrica para descifrar la información, la cual, de igual manera, una vez descifrada se almacena y se muestra al receptor para comprobar que el mensaje descifrado es el mismo que el original. En la Figura 3.19 se muestra el archivo descifrado, y se comprueba que tiene el mismo contenido que el mensaje original enviado.

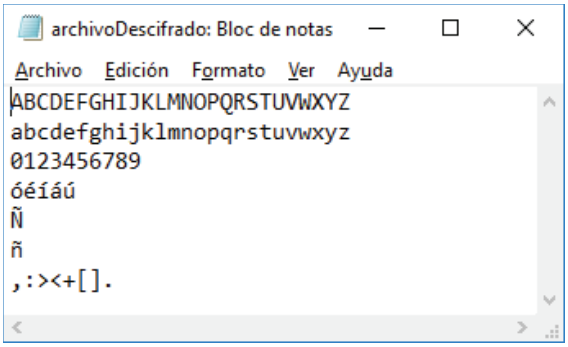


**Figura 3.19.** Mensaje descifrado empleando la criptografía híbrida

De esta forma se logra conseguir la confidencialidad de la información.

### 3.1.5. Prueba de *sniffer* para comprobar la confidencialidad

El método más apropiado para comprobar la confidencialidad de la información es realizar un *sniff* a la red para capturar los paquetes que se transmiten a través de la misma.

A modo de ejemplo, se va a enviar un mensaje cifrado con el algoritmo simétrico DES junto con la clave simétrica cifrada con el algoritmo asimétrico RSA, Figura 3.20.

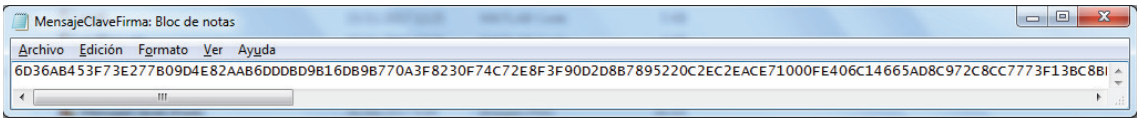


**Figura 3.20.** Mensaje cifrado enviado para *sniff*

Así es como se garantiza que la información es confidencial, y cualquier persona que la intercepte no puede comprenderla, únicamente las entidades que forman parte de la comunicación podrán descifrarla empleando las claves respectivas.

La transmisión de información se realiza desde la computadora de origen cuya dirección IP es 172.31.38.215 a la computadora destino cuya dirección IP es 172.31.38.216.

Para realizar la captura de paquetes se utilizó el *software* Wireshark. En la Figura 3.21 se puede observar el mensaje cifrado transmitido, seguido de la bandera y de la clave cifrada.

En el paquete enviado se puede ver la trama Ethernet, el paquete IP, el datagrama TCP y la información, la cual sería el mensaje cifrado y la clave simétrica cifrada. Por lo tanto, se demuestra claramente que el mensaje es confidencial, pues así una persona esté capturando la información enviada a través de la red, solo quien posea las claves podrá descifrar dicha información.

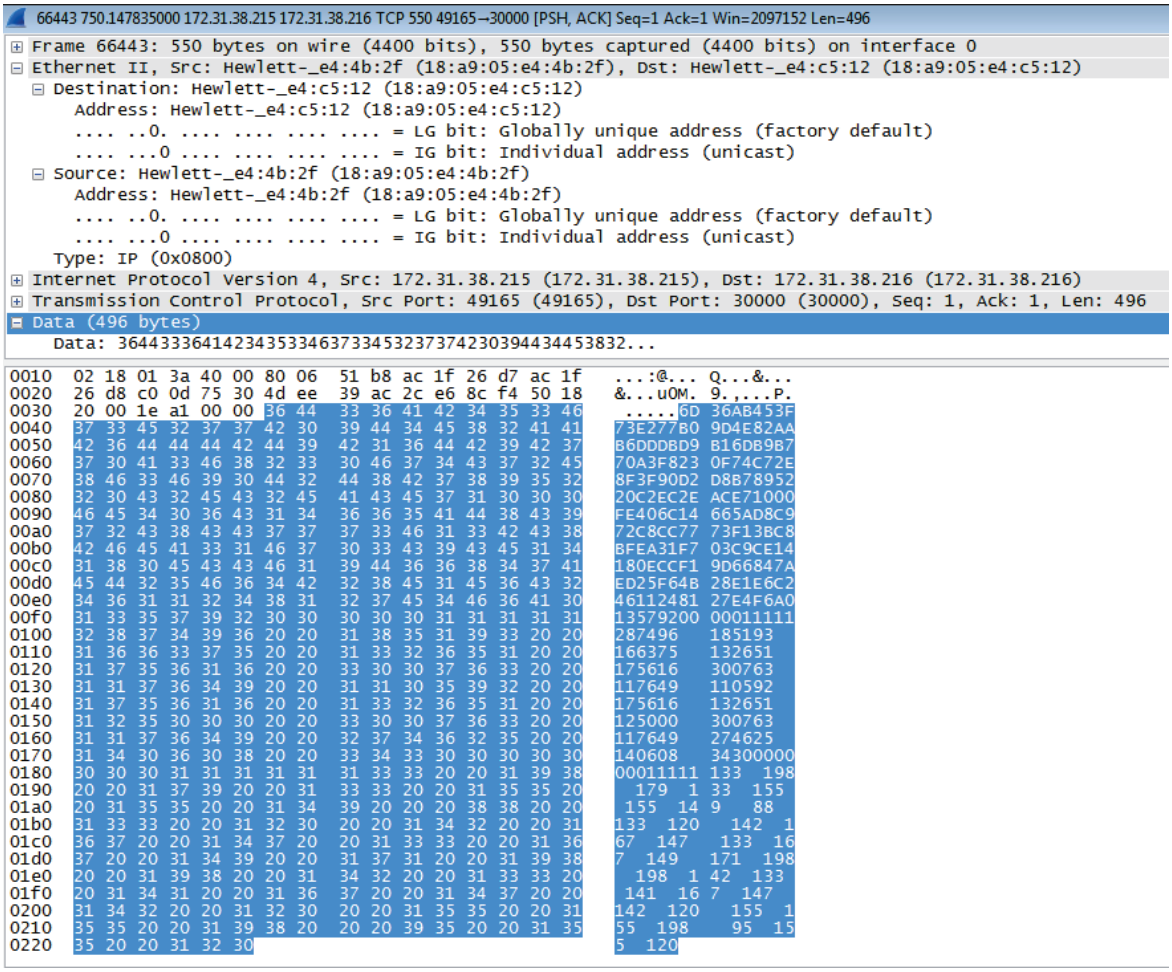


**Figura 3.21.** Captura del paquete de la clave simétrica enviada de origen a destino

## 3.2. Servicio de Integridad

### 3.2.1. Algoritmo MD5

En la Figura 3.22 se muestra la vista de la aplicación de uso didáctica del algoritmo MD5 implementada para proporcionar el servicio de integridad de la información. Esta aplicación genera un resumen de cualquier documento de texto plano que seleccione el emisor, y en recepción, realiza una comprobación del resumen enviado con el generado por el receptor, de esta manera se garantiza la integridad.

El código fuente que corresponde al algoritmo de resumen MD5 desarrollado se muestra en el ANEXO V.

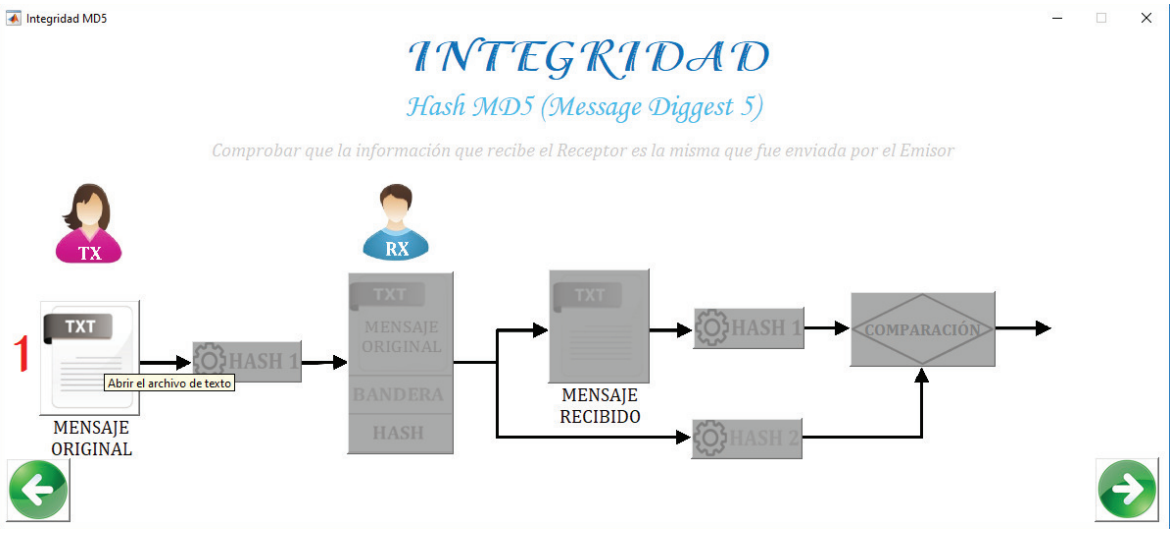


**Figura 3.22.** Aplicación del algoritmo de resumen MD5

### Generación del resumen MD5 en el lado del emisor

El emisor debe seleccionar un mensaje para generar el *hash* del mismo. El mensaje será el mismo de la Figura 3.4.

Al seleccionar el botón HASH 1 se muestra el resumen del mensaje, tal como se indica en la Figura 3.23.

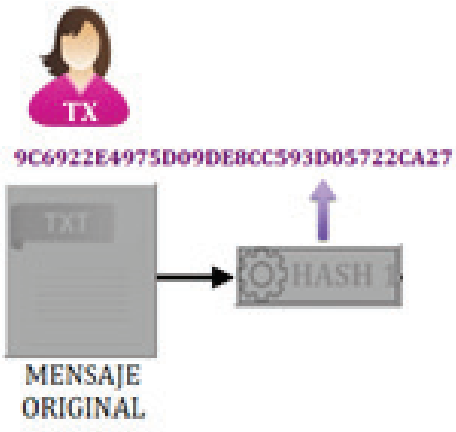


**Figura 3.23.** *Hash* del mensaje de texto generado

Consecutivamente, el documento se envía junto con su *hash* al receptor, incluyendo la bandera que se especificó en la sección 3.1.4, la cual servirá para poder separar y diferenciar el texto enviado del resumen del mismo. El documento que se envía al receptor es el que se indica en la Figura 3.24.

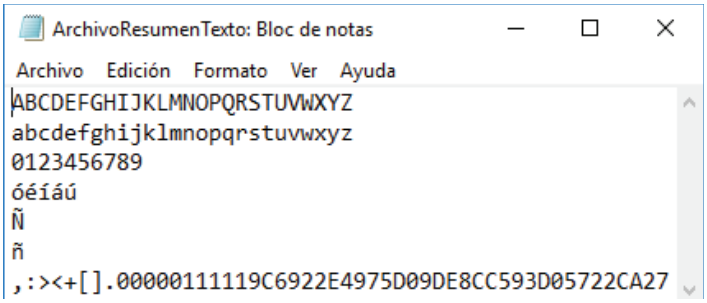


**Figura 3.24.** Archivo enviado al receptor: mensaje original junto con su *hash*

### Comprobación del resumen empleando el algoritmo MD5 implementado con el *Software* CrypTool

Con la finalidad de comprobar que el resumen del mensaje generado empleando el algoritmo MD5 implementado se ha generado correctamente, se compara con el *software* CrypTool, el cual también permite generar el resumen de un mensaje, en este caso se va a obtener el resumen del texto empleado en la Figura 3.4.

A continuación, en la Figura 3.25 se puede observar el resumen del mensaje original empleando dicho *software*.

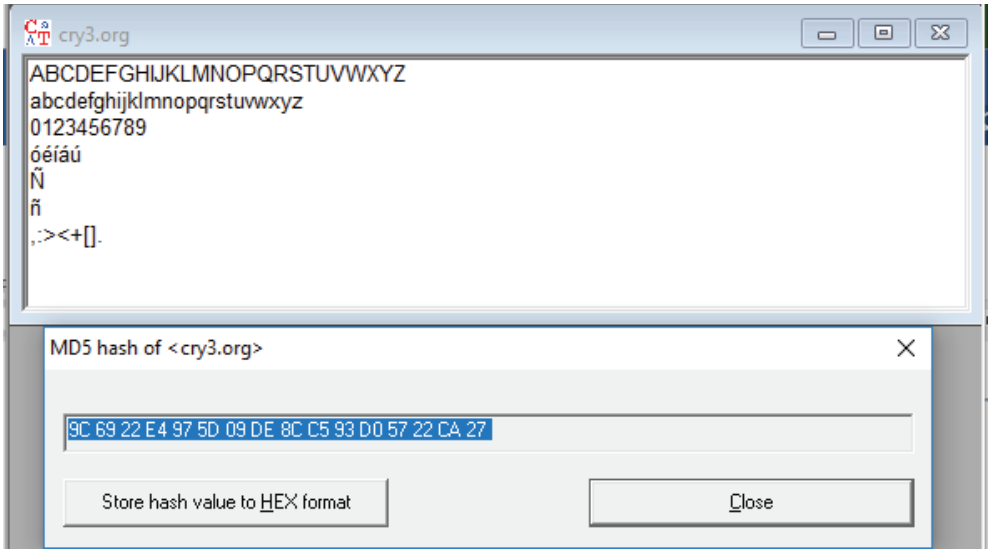


**Figura 3.25.** Resumen MD5 del mensaje con el *software* CrypTool

Por lo tanto, se puede comprobar el resumen generado empleando el algoritmo de resumen MD5 implementado en Matlab es exactamente el mismo que el generado con el *software* CrypTool. Así se garantiza el algoritmo MD5 ha sido desarrollado correctamente.

**Verificación de la integridad de la información en el lado del receptor**

El receptor debe separar las dos partes del archivo recibido: el documento en texto claro enviado y el *hash*. A partir del mensaje en texto claro recibido generará nuevamente un *hash* para verificar la integridad del mensaje, si tanto el *hash* recibido como el *hash* generado son iguales, el documento es íntegro, como se indica en la Figura 3.26; caso contrario el documento está corrupto.

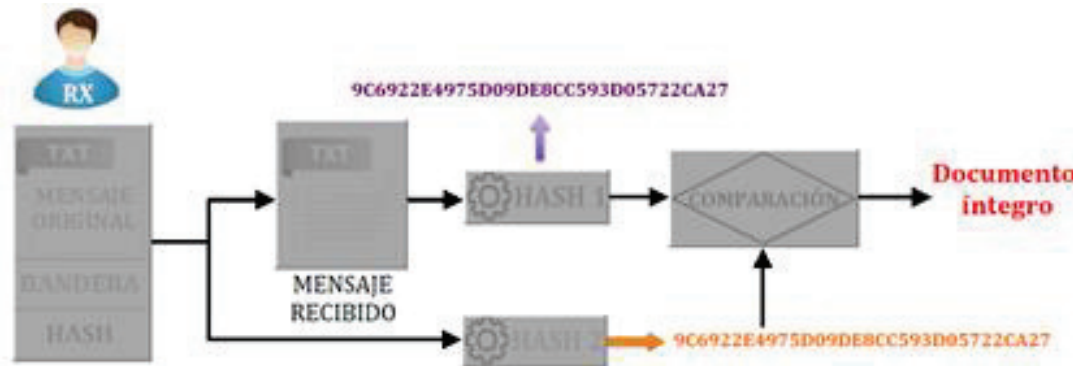


**Figura 3.26.** Comprobación de integridad del mensaje en el lado del receptor

## 3.3. Servicio de Autenticación de emisor y no repudio

### 3.3.1. Firma Digital

En base a la utilidad de la firma digital que proporciona conjuntamente tres servicios: integridad de la información, autenticación de emisor y no repudio, se diseñó una nueva interfaz gráfica en la cual es posible generar la firma digital de un documento. En la Figura 3.27 se muestra la vista de la aplicación implementada de la firma digital.

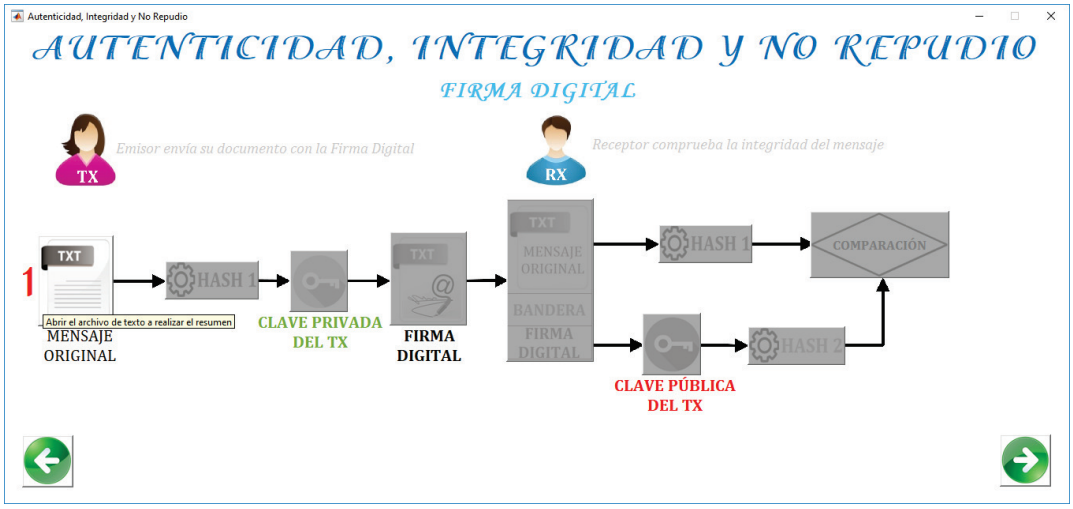


**Figura 3.27.** Aplicación de la firma digital

### Generación y cifrado del hash empleando el algoritmo asimétrico RSA

Para crear la firma digital se emplea el RSA como mecanismo para cifrar el *hash* MD5 generado. Como se vio en la sección Criptografía de clave asimétrica, para el modo de autenticación el proceso de cifrado es diferente: el emisor cifra el mensaje que se ha venido usando y se muestra en la Figura 3.4, empleando la clave privada del remitente. El mensaje resumido cifrado es la firma digital, a la cual se añade la información original y se envían en conjunto al destinatario.

En la Figura 3.28 se muestra el *hash* del texto claro y la clave privada del emisor.

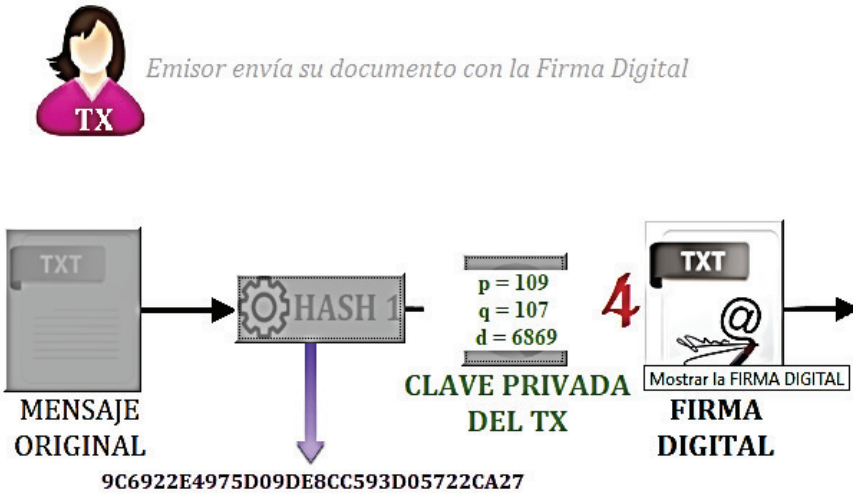


**Figura 3.28.** *Hash* del mensaje original y clave privada el emisor

Se cifra el resumen del mensaje original con la clave privada del emisor aplicando el algoritmo de cifrado asimétrico RSA. El resumen cifrado se muestra en la Figura 3.29.

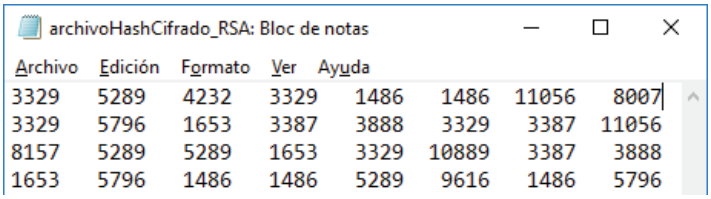


**Figura 3.29.** Cifrado del resumen MD5 del mensaje

### Comprobación del cifrado de la clave simétrica empleando el algoritmo asimétrico RSA implementado con el *Software* CrypTool

Para comprobar que el resumen MD5 cifrado empleando el algoritmo asimétrico RSA implementado se ha generado correctamente, se compara con el *software* CrypTool.

A continuación, en la Figura 3.30 se puede observar las claves pública y privada generadas y la clave simétrica cifrada.

La comprobación del *hash* MD5 del mensaje se realizó previamente en la sección Comprobación del resumen empleando el algoritmo MD5 implementado con el *Software* CrypTool.

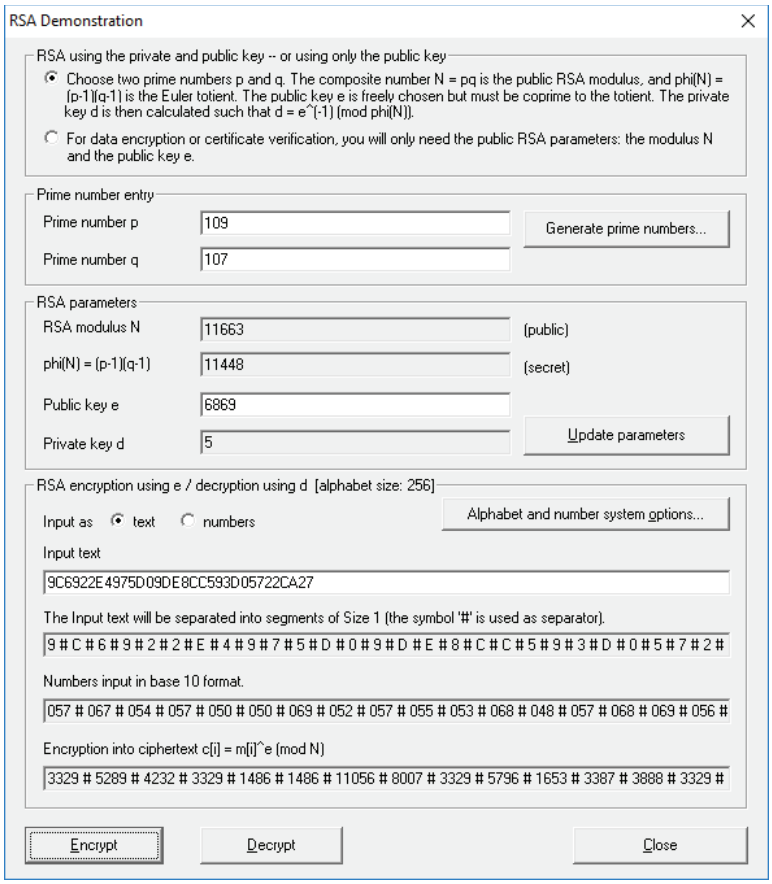


**Figura 3.30.** Cifrado del resumen MD5 con el *software* CrypTool

En vista de que en el *software* no se puede mostrar todo el resumen cifrado, se lo ha copiado en un nuevo documento de texto para mejor visualización, el cual se muestra en la Figura 3.31.

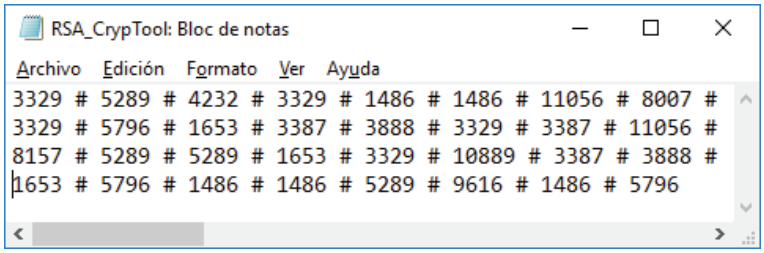


**Figura 3.31.** Cifrado completo del resumen MD5 con el *software* CrypTool

Por lo tanto, se puede comprobar que el cifrado del resumen MD5 empleando el algoritmo asimétrico RSA codificado en Matlab es exactamente el mismo que el generado con el *software* CrypTool. más atrásAsí se garantiza que tanto el *hash* MD5 como el cifrado del mismo han sido desarrollados correctamente.

### Comprobación de la autenticación del emisor y la integridad del mensaje en el lado del receptor

El receptor debe separar las dos partes del archivo recibido: el documento en texto claro enviado y el *hash* cifrado, el cual es la firma digital. A partir del mensaje en texto claro recibido genera nuevamente un *hash* y descifra la firma digital con la clave pública del remitente.

Si el *hash* descifrado como el *hash* generado son iguales, el documento es íntegro, como se indica en la Figura 3.32; caso contrario el documento está corrupto.

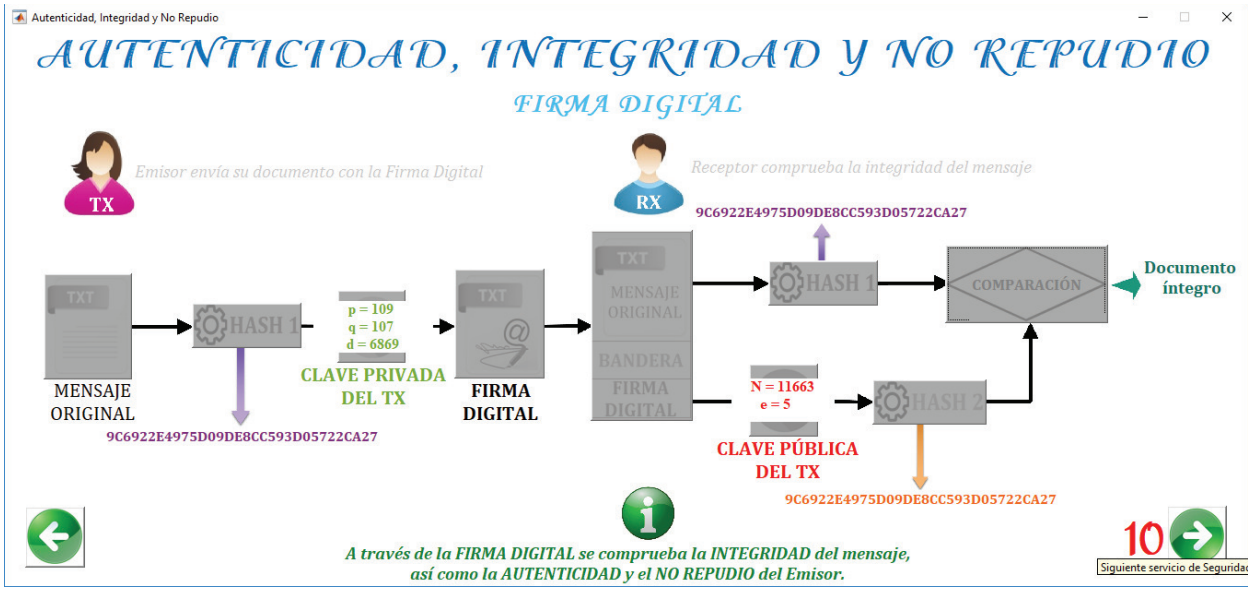


**Figura 3.32.** Comprobación de integridad del mensaje y autenticación de emisor

El emisor transmite la información cifrada empleando su clave privada, por lo tanto, cualquier persona que desee conocer la identidad del emisor emplea la clave pública del transmisor, la cual se encuentra disponible a todo el mundo puede descifrar los mensajes creados por el remitente, de este modo se garantiza que únicamente el mensaje pudo haber sido generado por el emisor.

## 3.4. Sistema de seguridad informática

### 3.4.1. Sistema completo

En vista de que todos los servicios se encuentran ya desarrollados y se ha comprobado su correcta implementación, es posible crear un nuevo sistema que integre todos los anteriores servicios en un único sistema de seguridad informática. De este modo se puede garantizar el servicio de confidencialidad de la información por medio de la criptografía híbrida, también se puede garantizar el servicio de integridad de la información, la autenticación y el no repudio del emisor a través de la firma digital. En la Figura 3.33 se muestra la vista de la aplicación de uso didáctica implementada del sistema de seguridad informática.

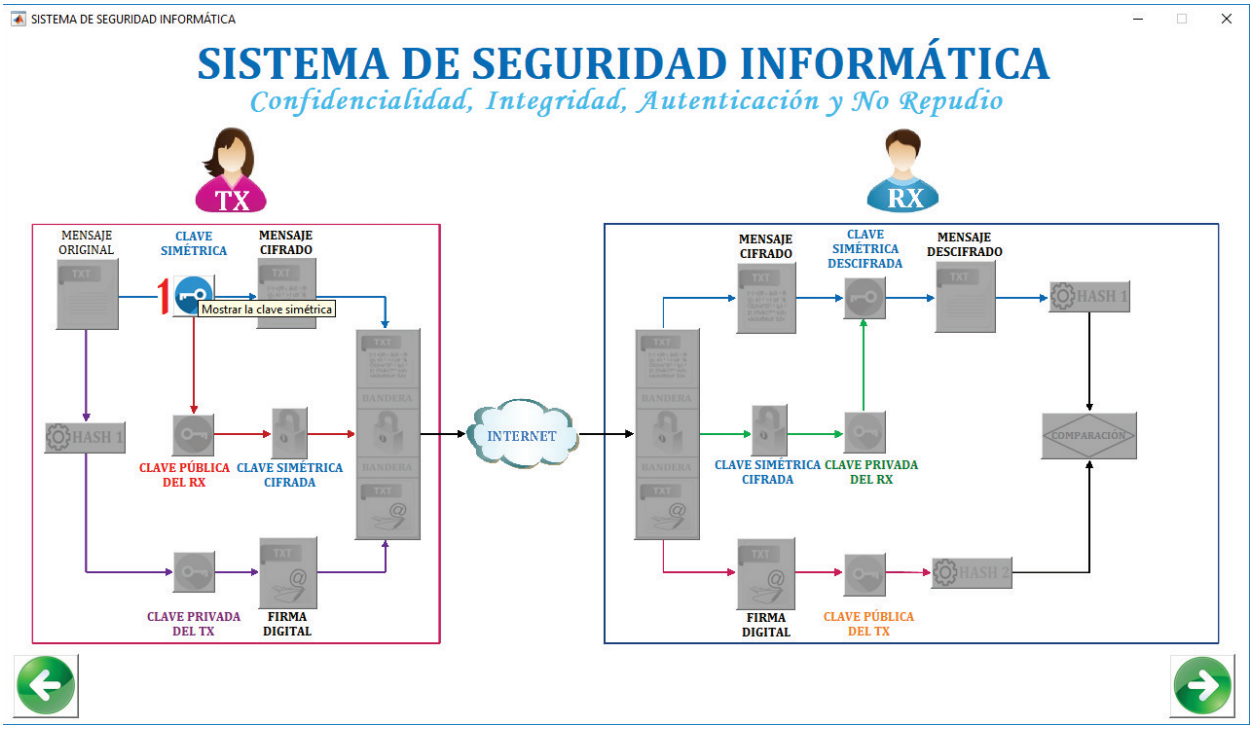


**Figura 3.33.** Aplicación del sistema de seguridad informática

El código fuente que corresponde a la implementación de la interfaz gráfica de usuario de la criptografía híbrida desarrollada se muestra en el ANEXO V.

## 3.5. Comunicación TCP/IP en Matlab

En la sección del Capítulo 1 se habla acerca de la Comunicación TCP/IP, en la cual es necesario establecer la comunicación entre el cliente y el servidor.

El Segmento de código 3.1 describe la codificación en Matlab de la comunicación TCP/IP de lado del servidor.

```
% COMUNICACIÓN TCP/IP SERVIDOR
t = tcpip('0.0.0.0', 30000, 'NetworkRole', 'server'); %Iniciar
la comunicación TCP/IP de tipo servidor que escuche a todos los
clientes, a través del puerto 30000.
disp(['El servidor está escuchando a través del puerto: ',
num2str(t.RemotePort)]);
set(t,'InputBufferSize',10000); %Modificar el buffer de entrada
set(t,'Timeout',30); %Modificar el timeout de la conexión
fopen(t); %Abrir la conexión con el cliente
disp('Se ha establecido una conexión...');

%Recibir bit a bit la información enviada por el cliente
datos = fread(t); %Recibir los datos enviados por el cliente
archID=fopen('ArchivoRecibido.txt', 'w'); %Abrir un archivo para
sobreescribirlo
fwrite(archID,datos); %Escribir el archivo con la inf recibida
fclose(archID); %Cerrar el archivo de texto
winopen('ArchivoRecibido.txt');
fclose(t); %Cerrar la conexión con el cliente
disp('Se ha cerrado la conexión...');
```

**Segmento de código 3.1.** Comunicación TCP/IP del servidor

En el Segmento de código 3.2 se describe la codificación en Matlab de la comunicación TP/IP de lado del cliente.

```
% COMUNICACIÓN TCP/IP CLIENTE
t = tcpip('172.31.38.144', 30000, 'NetworkRole', 'client');
%Iniciar la comunicación TCP/IP de tipo cliente
%al servidor 172.31.38.173, a través del puerto 30000.
disp(['El cliente está escuchando a través del puerto: ',
num2str(t.RemotePort)]);
set(t,'OutputBufferSize',10000); %Modificar el buffer de salida
set(t,'Timeout',30); %Modificar el timeout de la conexión
fopen(t); %Abrir la conexión con el cliente
disp(['Se ha establecido una conexión con el servidor: ',
num2str(t.RemoteHost)]);

%Enviar bit a bit la información al servidor
archivo=fileread('F:\Texto Claro.txt'); %Leer archivo de texto
fwrite(t,archivo); %Enviar el archivo a través de la red
fclose(t); %Cerrar la conexión con el cliente
disp('Se ha cerrado la conexión...');
```

**Segmento de código 3.2.** Comunicación TCP/IP del cliente

**Pruebas de envío de información a través del protocolo TCP/IP en Matlab**

En la Figura 3.34 se muestra la captura de pantalla de la conexión de la comunicación TCP/IP exitosa entre cliente y servidor.

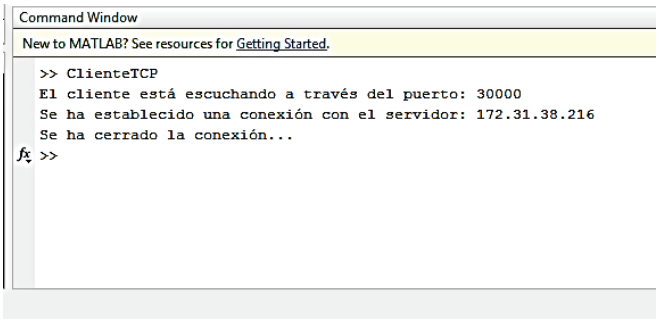


**Figura 3.34.** Conexión del cliente establecida exitosamente con el servidor

## 3.6. Encuesta

Ésta encuesta tiene como objetivo medir la utilidad, funcionalidad y apariencia para así conocer el grado de satisfacción de los usuarios una vez que se les presentó la aplicación como un método didáctico para la seguridad de la información.

**Tamaño de la muestra**

El tamaño de la muestra se definió en base a un promedio de estudiantes que utilizarían la aplicación dentro de un ambiente de clase didáctica de seguridad. Por esto, se consideró viable realizar la encuesta a 11 estudiantes de la carrera de Ingeniería en Electrónica y Redes.

### 3.6.1. Resultados de las encuestas realizadas

- **PREGUNTA 1:** ¿Considera Ud. la aplicación como uso didáctico y de fácil entendimiento?

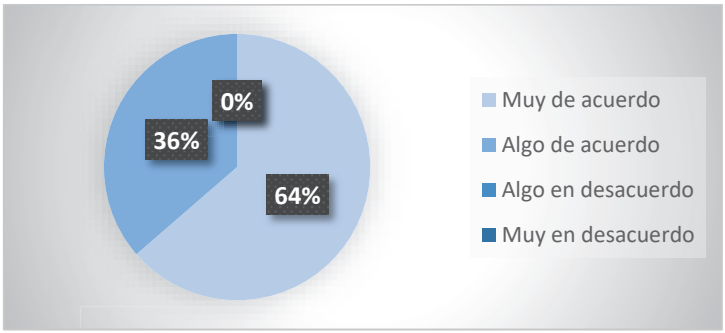


**Figura 3.35.** Resultados Pregunta 1

- **PREGUNTA 2:** ¿Usaría Ud. la aplicación de uso didáctico para reforzar conocimientos de seguridad informática?

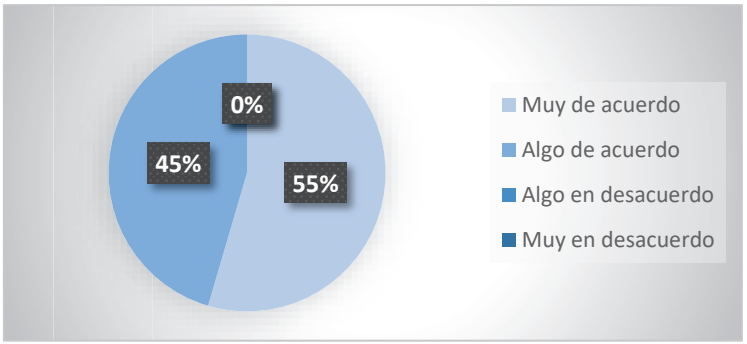


**Figura 3.36.** Resultados Pregunta 2

- **PREGUNTA 3:** ¿Pudo observar y entender claramente la implementación y funcionamiento de los algoritmos de seguridad informática implementados?

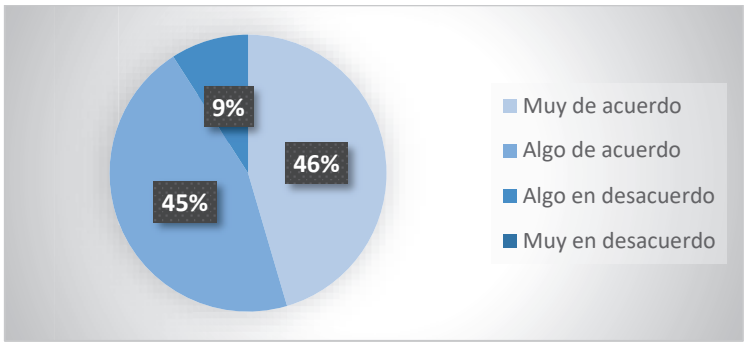


**Figura 3.37.** Resultados Pregunta 3

- **PREGUNTA 4:** ¿A través del uso de la aplicación pudo entender claramente el envío seguro de información a través de la red?

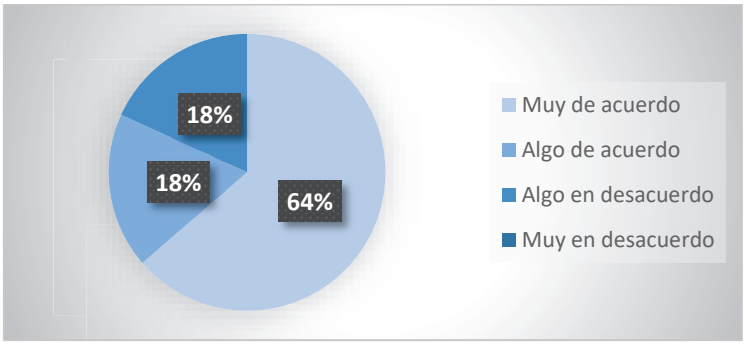


**Figura 3.38.** Resultados Pregunta 4

- **PREGUNTA 5:** ¿Cómo calificaría el nivel de aprendizaje que ha proporcionado la aplicación de uso didáctica?

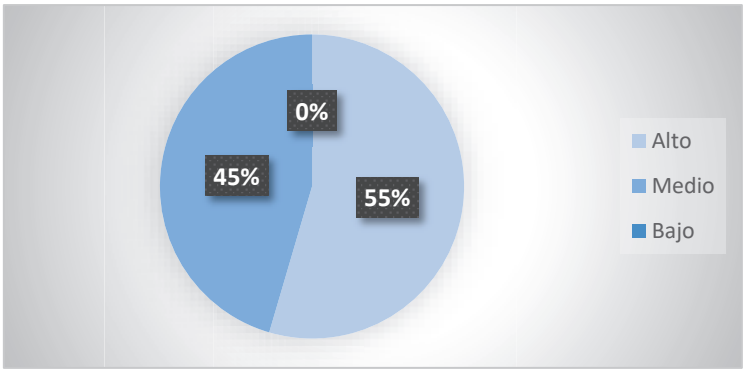


**Figura 3.39.** Resultados Pregunta 5

- **PREGUNTA 6:** En general, ¿qué opina de la aplicación de uso didáctica?

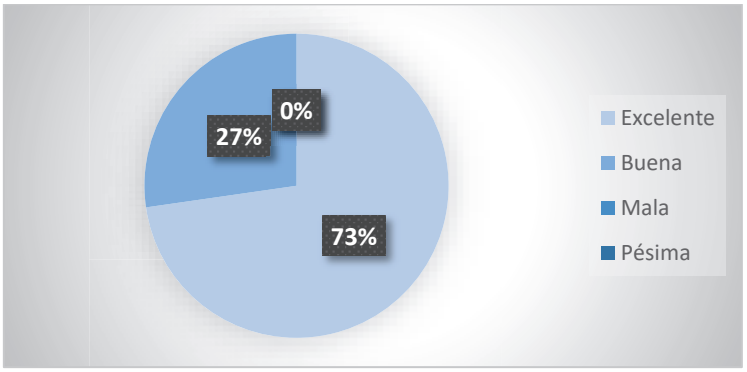


**Figura 3.40.** Resultados Pregunta 6

- **PREGUNTA 7:** ¿Qué opinión tiene en relación al ambiente gráfico de la aplicación?

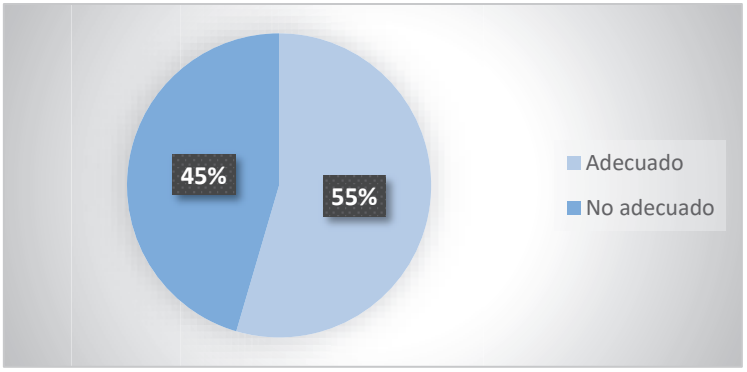


**Figura 3.41.** Resultados Pregunta 7

- **PREGUNTA 8:** ¿Recomendaría la aplicación de uso didáctica?

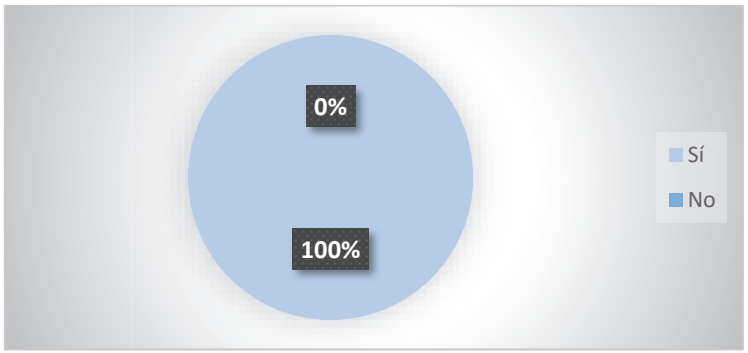


**Figura 3.42.** Resultados Pregunta 8

- **PREGUNTA 9:** ¿Cree Ud. que hacen falta más herramientas de uso didáctico en la materia de seguridad para mejorar el nivel de aprendizaje?

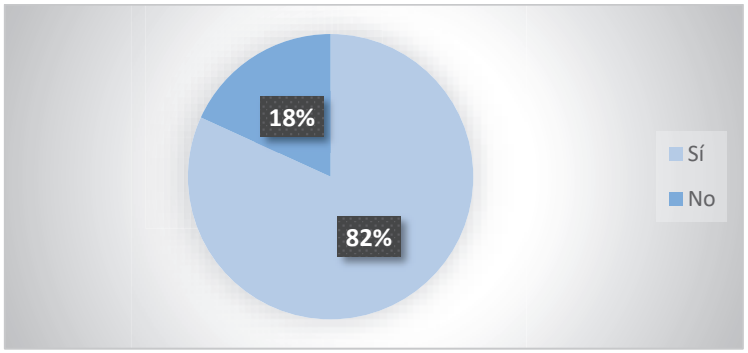


**Figura 3.43.** Resultados Pregunta 9

- **PREGUNTA 10:** Explique las dificultades de uso que haya encontrado en la aplicación.

  **RESPUESTA GENERALIZADA:** Se encontró como dificultad el poder regresar a un proceso anterior.

- **PREGUNTA 11:** Indique las recomendaciones que se debería hacer para mejorar la aplicación.

  **RESPUESTA GENERALIZADA:** La aplicación debería mostrar mensajes informativos de cada botón. Además, programar botones para regresar a procesos anteriores y deshabilitar los botones que ya se han utilizado previamente.

# 4. CONCLUSIONES

- Durante la búsqueda de investigación, no se ha encontrado un *software* que implemente todos los servicios de seguridad acoplados en uno solo sistema. Si bien el *software* libre CrypTool es el único programa que implementa cada algoritmo criptográfico por separado, no permite la ejecución conjunta de los cuatro servicios de seguridad: confidencialidad, integridad, autenticación y no repudio. Por lo tanto, el mérito del presente Trabajo de Titulación es el desarrollo de un sistema completo que proporciona un alto nivel de seguridad de la información.

- El Sistema de Seguridad Informática desarrollado cifra todo tipo de caracteres del mensaje en texto plano, inclusive caracteres especiales, por tanto, la seguridad se garantiza a todo el contenido del mensaje.

- La organización de contenido de cualquier fuente bibliográfica respecto a la seguridad de la información está basada en los algoritmos criptográficos, sin embargo, en el presente Trabajo de Titulación se realiza en base a los servicios de seguridad informática para un mejor entendimiento del lector.

- La interfaz gráfica de los módulos del Sistema de Seguridad cumple con su objetivo didáctico, según el índice de satisfacción de las encuestas realizadas a los estudiantes, proporcionando una herramienta de aprendizaje práctico para la materia Seguridad de Redes.

- Antes de seleccionar un mecanismo de cifrado e integridad de la información y de autenticación del emisor, se debe evaluar el nivel de seguridad requerido y los recursos físicos disponibles para su implementación, no es lo mismo cifrar un mensaje con una clave de 64 bits que empleando una clave de 5096 bits, la cual se recomienda criptográficamente segura actualmente para el cifrado empleando RSA. Por lo tanto, si se desea cifrar información empleando una clave de gran longitud se recomienda ejecutar el cifrado en servidores con recursos computacionales (procesamiento y memoria RAM) potencialmente óptimos.

- El presente trabajo garantiza la seguridad de la información que se transmite digitalmente, utilizando algoritmos criptográficos, lo cual permite aumentar la confiabilidad de la comunicación entre emisor y receptor, y podría ser aplicada en diferentes áreas: comercio electrónico, transacciones, compras en línea, correo electrónico, comunicados oficiales, entre otras.

- Durante la búsqueda de información para el desarrollo del presente Trabajo de Titulación, no se encuentra información respecto al proceso a detalle de cada algoritmo que se ha efectuado, por tal motivo, para una futura implementación, el presente Trabajo presenta de forma ordenada la matemática de cada algoritmo criptográfico.

- El diseño de este Sistema podría ser la base para el desarrollo de futuros sistemas de cifrado de diferentes tipos de información, como transmisión de voz, imágenes, video, con transmisión sobre un sistema real (ejemplo, canal telefónico) ya que los elementos que conforman el sistema con los diferentes algoritmos desarrollados podrán ser adaptados a un canal de comunicación real, con objetivo de transmitir información cifrada en tiempo real.

**Confidencialidad**

- En cuanto al servicio de confidencialidad, los algoritmos simétricos son rápidos para el cifrado de la información, sin embargo, el principal inconveniente es la distribución de claves. Por otro lado, los algoritmos asimétricos resuelven este problema, pero en la Tabla 3.1 se evidencia que son lentos para el cifrado de gran cantidad de información. Por lo anterior, el servicio de confidencialidad se consigue a través de un mecanismo de criptografía híbrida: el mensaje se cifra empleando cifrado simétrico y la distribución de claves se resuelve a través del cifrado asimétrico; esta técnica la emplean algunos protocolos para brindar seguridad respecto a la confidencialidad de la información.

- Para cada establecimiento de la comunicación entre emisor y receptor se emplean claves aleatorias generadas por la aplicación, tanto para el cifrado simétrico como asimétrico, lo cual determina un elevado nivel de seguridad, adicional al que proporciona el algoritmo criptográfico.

- Para el proceso de descifrado en el Sistema de Seguridad diseñado, lo único que se modificó fue la subrutina de los algoritmos de cifrado, cumpliéndose con una de las ventajas respecto a la matemática de los algoritmos criptográficos.

- El algoritmo simétrico DES al ser el primer algoritmo criptográfico moderno es la base para comprender la matemática de la criptografía, por lo cual, a pesar de la matemática compleja está sujeto a una serie de amenazas que requieren una gran cantidad de recursos de tiempo, procesamiento y computación distribuida, por lo tanto, no es apto para aplicaciones bancarias o que involucren información críticamente confidencial.

- Los actuales estudios criptográficos robustecen la seguridad de la información aumentando la longitud de las claves con el objetivo de dificultar los ataques, sin

embargo, es muy evidente que a medida que mejora la tecnología, la potencia computacional a futuro puede aumentar la vulnerabilidad de dichos algoritmos, incluso de algunos que se consideran irrompibles hasta la fecha.

**Integridad**

- En la actualidad existen algoritmos de resumen más seguros que MD5, como por ejemplo el algoritmo SHA-2, sin embargo, en el presente proyecto se ha empleado MD5 debido a que es ampliamente utilizado tanto para generar resúmenes de documentos o programas, garantizando la integridad de los mismos.
- Los algoritmos de resumen además de emplearse para verificar la integridad de la información, garantizan la autenticación en las comunicaciones. Algunos protocolos emplean este mecanismo criptográfico para resumir las contraseñas, comprobando la identidad de las personas con la comparación de los resúmenes.

**Autenticación y No Repudio**

- El proceso para obtener el servicio de autenticación a través de la firma digital es utilizando certificados digitales, en el presente Trabajo de Titulación se ha desarrollado cada servicio de seguridad con propósitos de uso didáctico, y por lo tanto, sería de gran aporte para proyectos futuros que se desarrolle un sistema que permita generar certificados digitales que garanticen la propiedad de las claves.
- Una Autoridad de Certificación (AC) puede ser implementada en Microsoft Server, por lo cual se recomienda que, aprovechando el recurso de licenciamiento que tiene disponible la Escuela Politécnica Nacional para toda la comunidad politécnica, se pueda implementar dicha AC, que sirva como metodología de aprendizaje didáctica para los estudiantes.
- La firma digital tiene el mismo propósito que una firma manuscrita, sin embargo, la segunda es vulnerable a falsificación, mientras que la digital es imposible si la clave privada permanece oculta y nadie logra descubrirla.
- Debido a que la firma digital tiene las mismas repercusiones legales que una firma manuscrita se recomienda comprender las implicaciones tanto jurídicas como administrativas del uso de la misma durante el intercambio de información, razón por lo cual es importante avalar la firma digital para que ninguna entidad externa mal intencionada pueda conocer la clave privada y pueda hacer un mal uso de la misma.

# 5. REFERENCIAS BIBLIOGRÁFICAS


[1] S. Srivastav y N. Verma, «Improving Data Security in Cloud Computing Using RSA Algorithm and MD5 Algorithm,» *International Journal of Innovative Research in Science, Engineering and Technology,* vol. 4, nº 7, pp. 5450-5457, July 2015.

[2] J. Costas Santos, Seguridad Informática, RA-MA EDITORIAL, 2010, p. 308.

[3] P. Beltrán Canessa, «Pilares Básicos,» de *SEGURIDAD INFORMÁTICA*, Trujillo, Perú, p. 14.

[4] M. Soriano, Seguridad en redes y seguridad de la información, Primera Edición ed., Praga: Innovative Methodology for Promising VET Areas, 2013, p. 80.

[5] Surety, «Provable Electronic Record Integrity, Authenticity, and Ownership,» Surety, [En línea]. Available: http://www.surety.com/digital-copyright-protection. [Último acceso: 08 10 2017].

[6] Delaware, «Notario Público de Delaware,» [En línea]. Available: https://notary.delaware.gov/electronic-notarization/. [Último acceso: 09 10 2017].

[7] Banco Central del Ecuador, Certificación Electrónica, [En línea]. Available: https://www.eci.bce.ec/preguntas-frecuentes#31. [Último acceso: 98 10 2017].

[8] MathWorks, «Create a TCP/IP Connection,» MathWorks, 2017. [En línea]. Available: https://www.mathworks.com/help/matlab/import_export/create-a-tcpip-connection.html. [Último acceso: 10 31 2017].

[9] MathWorks, «TCP/IP and UDP Interface,» MathWorks, 2017. [En línea]. Available: https://www.mathworks.com/help/instrument/tcp-ip-and-udp-interface.html. [Último acceso: 31 10 2017].

[10] S. Talens-Oliag, «Introducción a la Criptología,» 04 2003. [En línea]. Available: https://www.uv.es/sto/articulos/BEI-2003-04/criptologia.pdf. [Último acceso: 09 08 2017].

[11] Swiscoin, «WHAT IS CRYPTOGRAPHY?,» SwisCoin Crypto Inc., 2015. [En línea]. Available: https://www.swiscoin.com/Cryptography. [Último acceso: 02 08 2017].

[12] T. Boyles, CCNA Security, Indianapolis, Indiana: Wiley Publishing, Inc., 2010, p. 282.

[13] W. Stallings, NETWORK SECURITY ESSENTIALS: APPLICATIONS AND STANDARDS, Fourth Edition ed., New Jersey: Prentice Hall, 2011.

[14] C. Paar y J. Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Berlin: Springer Science & Business Media, 2009.

[15] M. T. Simpson, K. Backman y J. E. Corley, «Cryptography,» de *Hands-On Ethical Hacking and Network Defense*, Boston, Cengage Learning, 2012, p. 464 páginas.

[16] P. K. Bhardwaj, de *A+, Network+, Security+ Exams in a Nutshell: A Desktop Quick Reference*, First Edition ed., Sebastopol, O'Reilly Media, Inc., 2007, p. 672.

[17] C. Cobb, «Crypto Basics & What You Really Need to Know,» de *Cryptography For Dummies*, Hoboken, John Wiley & Sons, 2004, p. 324 páginas.

[18] A. G. Konheim, Computer Security and Criptography, Hoboken: Wiley, 2007.

[19] RSASecurity, «NEWS REPORT ON RSA DATA SECURITY FACTORING CHALLENGE,» RSASecurity, 2002.

[20] A. Q. Sierra, «CRIPTO 101 - La criptografía de curva elíptica,» 1 01 2007. [En línea]. Available: http://www.ugr.es/~aquiran/cripto/enigma/boletin_enigma_54.htm. [Último acceso: 25 10 2017].

[21] S. Misenar, J. Feldman y E. Conrad, CISSP Study Guide, Waltham: Syngress, 2012.

[22] X. Wang y H. Yu, «How to Break MD5 and Other Hash Functions,» Shandong University, Jinan, 2005.

[23] W. A. Conklin, R. Davis, C. Cothren, G. White y D. Williams, CompTIA Security+ All-in-One Exam Guide, 3rd ed., New York: McGraw-Hill, 2011.

[24] B. Furht, Encyclopedia of Multimedia, 2nd Edition ed., Boca Ratón, Florida: Springer Science & Business Media, 2008, p. 1001 páginas.

[25] J. F. Roland y M. J. Newcomb, «Overview of VPN and IPSec Tecnologies,» de *CCSP Cisco Secure VPN*, Indianapolis, Cisco Press, 2003, pp. 46-52.

[26] N. Doraswamy y D. Harkins, «Cryptographic History and Techniques,» de *IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks*, Upper Saddle River, Prentice Hall Professional, 2003, pp. 1-20.

[27] S. Wilkins y F. H. S. III, CCNP Security Secure, Official Cert Guide ed., Indianapolis: Cisco Press, 2011, pp. 414-420.

[28] IBM, «Infraestructura de claves públicas (PKI),» IBM, 13 12 2016. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009900_.htm. [Último acceso: 30 08 2017].

[29] Rivest, «El Algoritmo de Resumen de Mensajes MD5,» rfc, 1992. [En línea]. Available: https://www.rfc-es.org/rfc/rfc1321-es.txt. [Último acceso: 24 11 2017].

[30] A. Gilat, Matlab: Una introducción con ejemplos prácticos, Barcelona: Reverté S.A., 2006, pp. 5-9.

[31] MathWorks, «Ways to Build Apps,» MathWorks, 2017. [En línea]. Available: https://www.mathworks.com/help/matlab/creating_guis/ways-to-build-matlab-guis.html. [Último acceso: 02 09 2017].

[32] J. G. Joachim von zur Gathen, Modern Computer Algebra, Second Edition ed., New York: Cambridge University Press, 2003, pp. 45-60.

[33] B. Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C, 20th Anniversary Edition, John Wiley & Sons, 2015, pp. 233-261.

[34] D. J. R. Aguirre, «Introducción a la seguridad informática y criptografía clásica,» Criptored, 15 02 2016. [En línea]. Available: http://www.criptored.upm.es/crypt4you/temas/criptografiaclasica/leccion1.html. [Último acceso: 10 10 2017].

[35] S. C. C. o. t. C. S. o. t. IEEE, *IEEE Standard Glossary of Software Engineering Terminology,* New York: The Institute of Electrical and Electronics Engineers, 1990, p. 67.

[36] A. Hossain, K. Islam, S. Kumar Das y A. Nashiry, «Cryptanalyzing of Message Digest Algorithms MD4 and MD5,» *International Journal on Cryptography and Information Security (IJCIS),* vol. 2, nº 1, pp. 1-13, March 2012.

[37] Fotoseimagenes.net, «Imágenes de Criptografía,» [En línea]. Available: http://www.fotoseimagenes.net/algoritmo-de-cifrado. [Último acceso: 10 10 2017].

[38] B. Holyfield, «RSAConference,» 2012. [En línea]. Available: https://www.rsaconference.com/writable/presentations/file_upload/asec-403.pdf. [Último acceso: 20 08 2017].

[39] T. Stockinger, «GSM network and its privacy - the A5 stream cipher,» 11 2005. [En línea]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.8718&rep=rep1&type=pdf. [Último acceso: 25 08 2017].

[40] Y. Bhaiji, «CCIE Security Certification,» 2009. [En línea]. Available: https://www.cisco.com/c/dam/global/en_ae/assets/exposaudi2009/assets/docs/ccie-security.pdf. [Último acceso: 31 10 2017].

[41] P. A. López, Seguridad informática, Editex, 2011.

[42] M. Gardner, TIME TRAVEL AND OTHER MATHEMATICAL BEWILDERMENTS, New York: W. H. Freeman and Company, 1988.

# 6. ANEXOS

En esta sección se incluyen los respectivos anexos como información complementaria al presente proyecto de titulación.

La distribución de esta sección se distribuye de la siguiente manera:

ANEXO I. Entrevista

ANEXO II. MATLAB

ANEXO III. Fundamentos matemáticos del algoritmo RSA

ANEXO IV. Encuesta final a los estudiantes

ANEXO V. Código MATLAB

# ANEXO I

| FICHA DE ENTREVISTA<br>Este cuestionario es estrictamente confidencial y sólo será utilizado con fines estadísticos. Sus respuestas serán muy útiles para el Desarrollo de una Aplicación de Uso Didáctico para envío seguro de información a través de la red. | |
|---|---|
| **ELABORADO POR:** | Jessica Jiménez |
| **LUGAR:** | Escuela Politécnica Nacional |
| **FECHA:** | |
| **PERSONA ENTREVISTADA:** | |

| Nº | PREGUNTAS |
|---|---|
| 1 | ¿Considera de utilidad el uso de aplicaciones didácticas para reforzar el aprendizaje? |
| 2 | ¿Se desarrollaron en este último año nuevas actualizaciones al contenido de seguridad de la información impartido dentro del sílabo de la materia de Seguridades? |
| 3 | ¿El contenido que se imparte respecto a seguridad informática es netamente teórica? |
| 4 | ¿Alguna vez ha empleado alguna aplicación de uso didáctico para que los estudiantes refuercen los conocimientos adquiridos? |
| 5 | ¿Utiliza algún programa criptográfico como complemento para impartir sus clases? |
| 6 | ¿Considera usted que una aplicación de uso didáctico sería de utilidad para el aprendizaje de seguridad informática? |
| 7 | ¿Cuáles son los recursos y actividades que le gustaría que contenga la aplicación para que sea considerada de uso didáctico? |

# ANEXO II

## MATLAB

MATLAB, abreviatura de MATrix LABoratory (Laboratorio de Matriz) es un software computacional desarrollado a mediados de los años 80 específicamente para cálculos científicos en el área técnica y de ingeniería.

MATLAB integra el análisis numérico, la computación matricial, el procesamiento de señal y los gráficos en un entorno fácil de aplicar, empleando funciones y cajas de herramientas (*toolbox*) diseñadas para aplicarlas en una variedad de áreas: Álgebra Lineal, Procesamiento de señales y comunicaciones, Procesamiento de imagen y video, Sistemas de control, Finanzas computacionales, Biología Computacional, Redes Neuronales, Seguridad Informática, Estadística, Matemática Avanzada, Cálculo Numérico.

La ventana principal de trabajo en MATLAB se muestra en la Figura II.1:

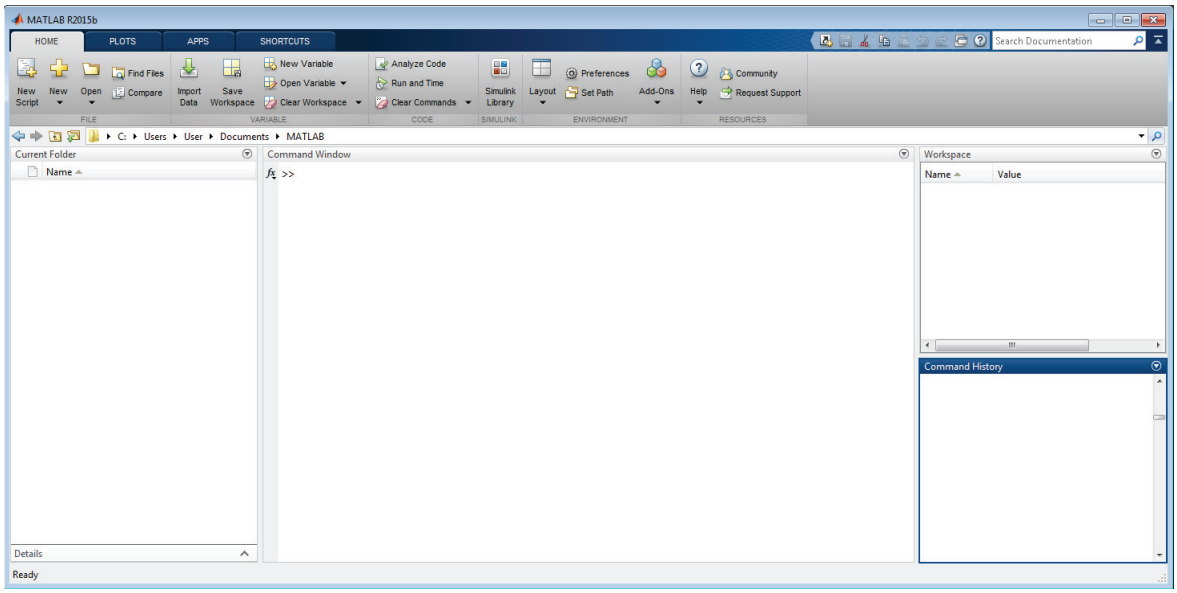


**Figura II.1** Ventana principal de MATLAB

La Tabla II.1 muestra un listado detallado de las ventanas y su funcionalidad [30]:

**Tabla II.1** Ventanas de Matlab

| Nombre | Significado | Funcionalidad |
|---|---|---|
| *Command Window* | Ventana de Comandos | Es la ventana principal, se utiliza para introducir variables y comandos; y ejecutar programas. |
| *Figure Window* | Ventana de Gráficos | Se utiliza para visualizar gráficos. |
| *Editor Window* | Ventana del Editor | Se usa para crear y depurar scripts y funciones. |
| *Help Window* | Ventana de Ayuda | Proporciona ayuda e información de las funciones de Matlab. |
| *Launch Pad Window* | Ventana de Plataforma | Facilita el acceso a herramientas, demos y documentación de Matlab. |
| *Command History Window* | Ventana del Historial de Comandos | Almacena y visualiza los comandos que se introducen en la ventana de comandos. |
| *Workspace Window* | Ventana del Espacio de Trabajo | Proporciona información sobre las variables almacenadas. |
| *Current Directory Window* | Ventana del Directorio de Trabajo Actual | Muestra los archivos que se encuentran en el directorio de trabajo actual. |

**Comandos básicos**

Los comandos e instrucciones de Matlab se escriben en la ventana de comandos.

Los comandos básicos más usados se encuentran descritos en Tabla II.2:

**Tabla II.2** Comandos básicos de Matlab

| Comando | Acción |
|---|---|
| `help` comando | Muestra ayuda e información de la función. |
| `ver` | Imprime la versión de Matlab. |
| `demo` | Muestra una breve introducción a Matlab |
| `clc` | Limpiar la ventana de comandos. |
| `save` variable | Almacenar la variable. |
| `load` | Cargar variables. |
| `exit` | Salir del programa. |
| `ans` | Variable creada automáticamente cuando no se especifica una variable de salida, muestra el resultado del último comando ejecutado. |

### *Scripts* y funciones en Matlab

Los archivos de comando en Matlab se los conoce como *scripts*, que son un conjunto de instrucciones creadas en la Ventana del Editor y almacenadas un fichero con la extensión .m para poder ejecutarse y así automatizar tareas.

Las funciones son programas que tienen argumentos de entrada y de salida, se diferencian de los scripts porque brindan flexibilidad para tareas de propósito general. Deben almacenarse en un fichero .m con el mismo nombre de la función. Se puede definir funciones anidadas, es decir, subfunciones dentro del código de la función principal.

### GUI (*Graphic User Interface*)

La interfaz gráfica es el conjunto de comandos o controles que tienen una función específica, por medio de los cuales el usuario se comunica con el programa, facilitando su uso para cualquier tipo de usuarios que lo utilicen.

Para diseñar una GUI se emplean dos ambientes de Matlab:

- El ambiente de desarrollo de la interfaz gráfica (GUIDE): Permite diseñar la interfaz gráfica que permite al usuario interactuar con el programa. Una vez la vista de la interfaz se encuentre lista se debe editar las funciones de llamada (*Callback*), programando el funcionamiento de cada control para su ejecución.
- El ambiente de trabajo (*Working Environment*): Es el espacio donde se desarrollan los *scripts* o funciones y se administran variables.

### Aplicaciones en Matlab

Existen diferentes maneras de construir aplicaciones [31]:

- ***App Designer*:** Introducido en la versión R2016a, las vistas de diseño y código están estrechamente vinculadas, de modo que los cambios que realice en una vista afectan inmediatamente a la otra, compatible con diagramas 2-D y 3-D.
- **Utilizar las funciones de MATLAB para crear aplicaciones mediante programación:** Codifica el diseño y el comportamiento de la aplicación utilizando por completo las funciones de MATLAB. Crea una figura básica y coloca componentes interactivos en esa figura mediante programación. Estas aplicaciones admiten los mismos tipos de gráficos y componentes interactivos compatibles con GUIDE, así como paneles con pestañas.

- **GUIDE:** Entorno de programación gráfica para diseñar interfaces de usuario que codifica el comportamiento de los controles su aplicación en el editor de MATLAB. Se utiliza para crear aplicaciones simples que puedan mostrar cualquier tipo de gráfico. En la Figura II.2 se puede observar la ventana de inicio de GUIDE en la cual es posible seleccionar crear una GUI en blanco, abrir una GUI ejemplo con controles básicos, abrir una GUI con Ejes y Menú y un Cuadro de Diálogo.

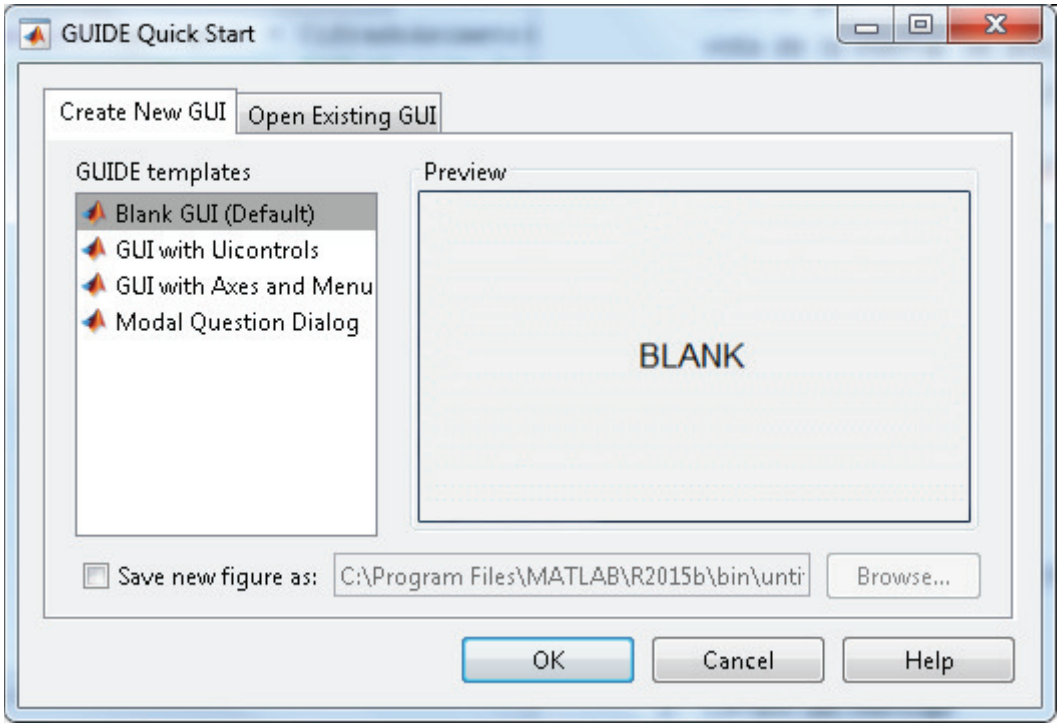


**Figura II.2** Entorno de programación gráfica en Matlab GUIDE

Matlab emplea objetos gráficos para poder visualizar los datos, se encuentran organizados jerárquicamente como se muestra en el siguiente organigrama, siendo la raíz el objeto más general, la cual puede contener varias ventanas y cada una de ellas varios controles: botones, slider, *radio button*, etc; menús normales, menús desplegables y uno o más ejes de coordenadas que a su vez podrán representar objetos gráficos. Para lo cual es importante indicar la herencia del objeto cuando están asociados entre sí.

**Instalación de Matlab**

Este es un software licenciado que se lo puede obtener a través de la licencia para el estudiante gracias al convenio que existe entre la universidad y MathWorks. Esto permite instalar a cada estudiante la versión de uso estudiantil y cuenta con todas las herramientas necesarias para el desarrollo de este proyecto.

Para la instalación se requiere generar las credenciales de cada estudiante a través de MathWorks y del área de la DGIP de la Escuela Politécnica Nacional.

Una vez que se tienen las credenciales para acceder a la licencia, se debe ejecutar el archivo *setup.exe* de la carpeta del instalador de MATLAB R2017a, de la Figura II.3.

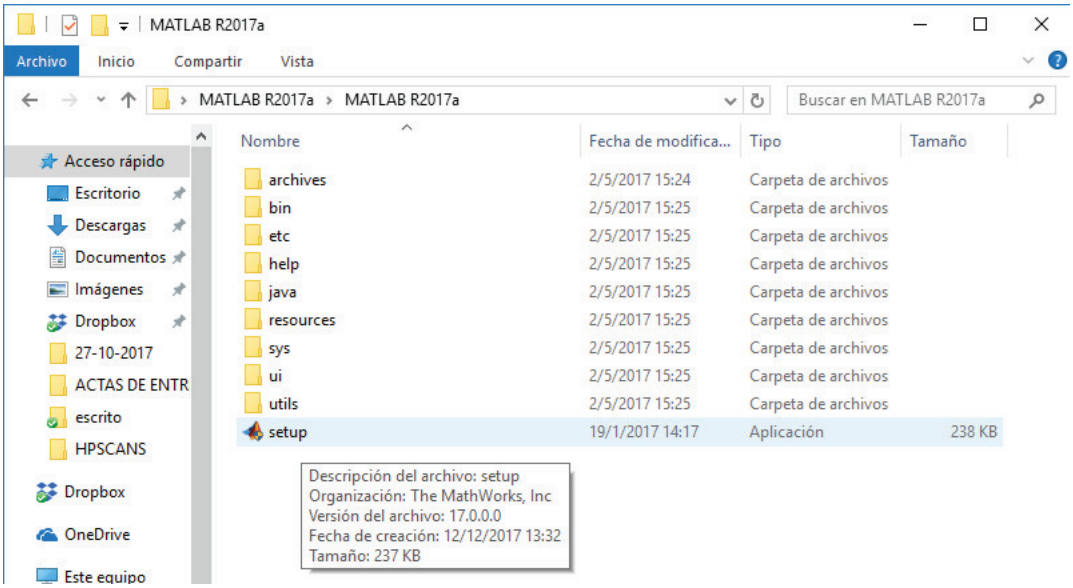


**Figura II.3** Archivos de instalación de Matlab

A continuación, se abre la ventana emergente para dar la autorización de ejecución de la aplicación *setup.exe* de la instalación de Matlab.

Después se debe seleccionar el método de instalación, escogiendo una de las dos opciones que se muestran en la Figura II.4.


**Figura II.4** Instalación de Matlab – *Select installation method*

En la ventana de la Figura II.5 se abre el acuerdo de la licencia, a la cual se debe aceptar los términos y continuar.

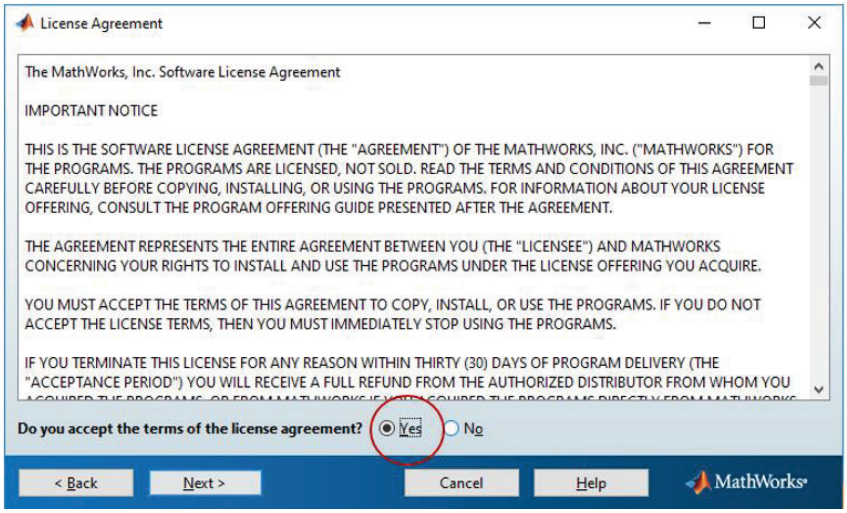


**Figura II.5** Instalación de Matlab - *License Agreement*

Una vez que se aceptó, en la ventana de la Figura II.6 se debe ingresar la cuenta de *Mathworks Account* con la que se creó la credencial y la contraseña respectiva.

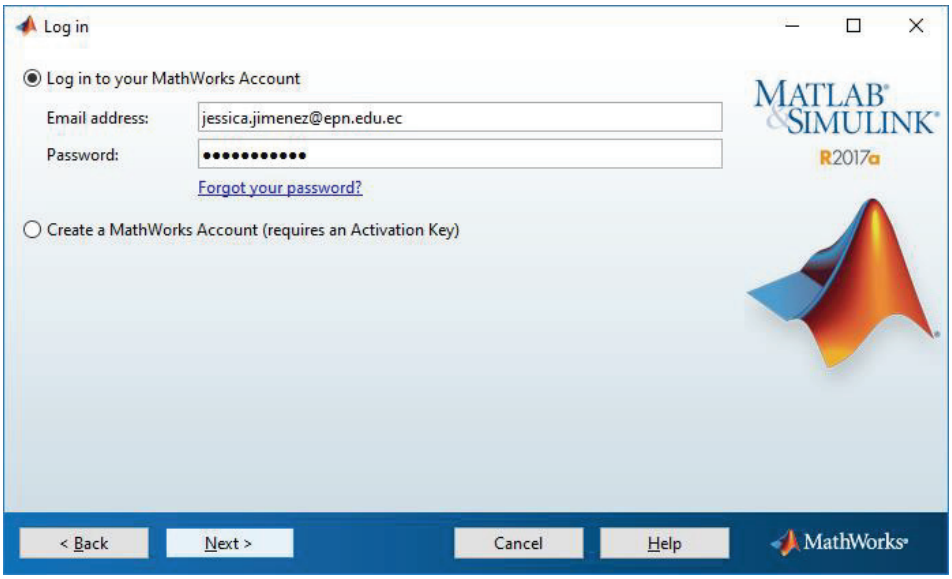


**Figura II.6** Instalación de Matlab – *Log in*

Una vez ingresados los datos para obtener la licencia del estudiante que brinda MathWorks en las ventanas siguientes se inicia ya la instalación de los paquetes del *software* y para finalizar la instalación como último requerimiento se debe realizar la activación de la cuenta, para lo cual se debe tomar en cuenta que se tenga conexión a *internet*.

Después de todos los pasos seguidos ya se tiene disponible esta herramienta de desarrollo para poder utilizar y aprovechar todas las utilidades que se muestran en la Figura II.7.

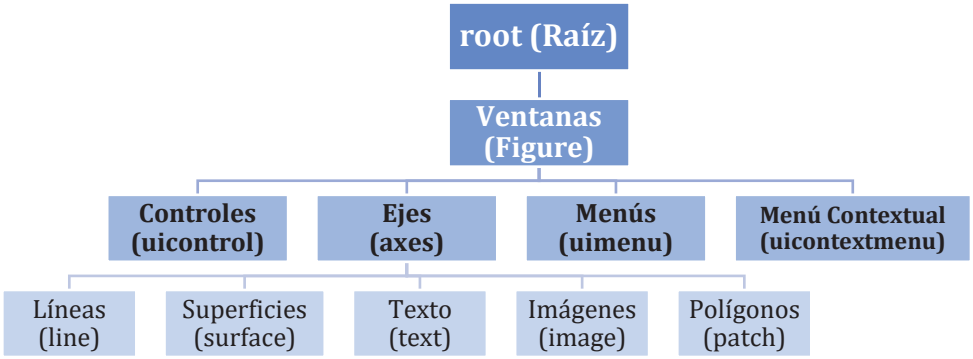


**Figura II.7** Jerarquía de objetos gráficos en Matlab

### Componentes de GUIDE

Para implementar una GUI se pueden emplear varios componentes que se encuentran disponibles en la GUIDE, en la Tabla II.3 se pueden observar una lista de los controles más conocidos y más empleados describiendo su función específica.

**Tabla II.3** Componentes de GUIDE

| Control | Ícono | Descripción |
|---|---|---|
| *Push Button* | | Invoca un evento inmediatamente. |
| *Slider* | | Barra de desplazamiento de un rango de valores. |
| *Radio Button* | | Seleccionar una opción. |
| *Check Box* | | Seleccionar el estado de una opción o atributo. |
| *Edit Text* | | Permite introducir o modificar texto. |
| *Static Text* | | Texto estático para dar instrucciones y etiquetas. |
| *Pop-Up Menu* | | Mostrar una lista de opciones. |
| *List Box* | | Seleccionar un elemento de una lista deslizable. |
| *Toggle Button* | | Indicar si una acción está activa (*on*) o inactiva (*off*). |
| *Table* | | Genera una tabla. |
| *Axes* | | Mostrar gráficas e imágenes. |
| *Panel* | | Organizan los componentes de la GUI en grupos. |
| *Button Group* | | Permiten selección exclusiva en los *Radio Button*. |
| *ActiveX Component* | | Mostrar controles ActiveX[26] (Microsoft) en la GUI. |

[26] Controles ActiveX: Bloques de Mircrosft Windows que se usan para crear programas.

# ANEXO III

## FUNDAMENTOS MATEMÁTICOS DEL ALGORITMO RSA

Antes de empezar a describir el funcionamiento del algoritmo, se va a resumir brevemente algunos conceptos matemáticos que serán empleados en esta etapa, las cuales deben tener una operación inversa para poder descifrar y recuperar el mensaje.

- **Número primo:** Es un entero positivo mayor que 1, que es divisible únicamente para sí mismo y la unidad; hay infinitos números primos en el universo de los números.
- **Números coprimos o primos relativos:** Son dos números primos entre sí cuyo máximo divisor común es 1, lo que significa que la fracción de ambos números no se la puede simplificar. Por ejemplo, 13 y 53 son coprimos porque tanto 13 como 53 son primos y 53 no es múltiplo de 13.
- **Función de Euler** $\phi(N)$: Devuelve el valor de enteros positivos menores o iguales que $N$ que son primos respecto a $N$. Estas son algunas propiedades de la función:
  a. $\phi(1) = 1$.
  b. $\phi(p) = p - 1$, si p es primo.
  c. $\phi(p^k) = (p-1)p^{k-1}$, si $p$ es primo y $k$ un número natural.
  d. $\phi(p * q) = \phi(p)\phi(q)$, si m y n son primos.

  Por lo tanto, aplicando la segunda y la cuarta propiedad se puede deducir que: $\phi(p * q) = (p-1)(q-1)$.
- **Cálculo de inversos:** El inverso de un número es otro número que multiplicado por otro da como resultado 1.

  El inverso modular de un entero $n$ mòdulo $p$ es un entero $m$ tal que: $n^{-1} \equiv m(\mathrm{mod}\ p)$, lo cual es equivalente a $mn \equiv 1(\mathrm{mod}\ p)$, se cumple solo si $n$ y $p$ son coprimos, es decir, si $mcd(n, p) = 1$.

  Existen algunas maneras de calcular inversos: Algoritmo Extendido de Euclides, Teorema de Euler, Teorema Chino del Resto, Ecuación Diofántica[27], entre otras.

[27] Ecuación diofántica: Ecuación de una o varias incógnitas, cuya solución son números enteros.

- **Inversos multiplicativos:** Esta operación matemática es muy común en criptografía de clave pública: un número $a$, elemento de $n$, tiene un inverso multiplicativo en $n$, si existe otro número $b$, tal que $abmod(n) = 1$, y para que esta condición se cumpla que $a$ y $n$ deben ser coprimos, es decir $mcd(a,n) = 1$. Si $a$ y $n$ no son coprimos, entonces $a^{-1} \equiv b\,(mod\,n)$ no tiene solución. Si $n$ es un número primo, entonces cada número desde 1 hasta $n-1$ es coprimo a $n$ y tiene exactamente un módulo inverso $n$ en ese rango.

  El método más empleado para el cálculo de inversos es el Algoritmo Extendido de Euclides.

- **Algoritmo de Euclides:** Permite conocer el máximo común divisor entre dos números.

- **Algoritmo Extendido de Euclides:** Este algoritmo permite generar las llaves pública y privada en el cifrado asimétrico RSA.

  Permite encontrar el máximo común divisor entre dos números $a$ y $b$ números enteros diferentes de 0, y expresarlo como la mínima combinación lineal, esto es encontrar dos números $u$ y $v$ tales que $mcd(a,b) = au + bv$, donde $mcd(a,b)$ es el divisor común más grande entre $a$ y $b$:

$$r = a\,u + b\,v$$

  El algoritmo permite encontrar eficientemente el cómputo de $u$ y $v$. Para lo cual se emplea el siguiente teorema [32]:

  **Teorema 1:** Dados los números enteros $a, b, r_0, \dots, r_{\lambda+1}$ y $q_1, \dots, q_\lambda$:

$u_0 \coloneqq 1,$ $\quad v_0 \coloneqq 0,$

$u_1 \coloneqq 0,$ $\quad v_1 \coloneqq 1,$

$u_{i+1} \coloneqq u_{i-1} - u_i q_i,$ $\quad v_{i+1} \coloneqq v_{i-1} - v_i q_i$

**Ejemplo:**

Encontrar el $mcd(393,267)$.

Para el ejemplo $a = 393$ y $b = 267$, primero se debe ir buscando el cociente y el residuo de dividir $a$ y $b$ hasta que $b$ sea $0$. Esto es, aplicar la división Euclideana de $a$ y $b$, de tal forma que $a = b\,c + r$, con $0 \leq r < b$.

$$393 = 267\,(1) + 126.$$

El cociente de esta operación es 1 y el residuo 126. La operación se repite hasta que $r$ sea 0:

$$267 = 126(2) + 15$$
$$126 = 15(8) + 6$$
$$15 = 6(2) + 3$$
$$6 = 3(2) + 0$$

Como se puede ver en las iteraciones, en la cada división el valor de $a$ es el anterior valor de $b$ y el valor de $b$ es el residuo de la anterior división.

Una vez hecho esto se puede crear una tabla para facilitar el cálculo de $u$ y $v$ de cada operación como se muestra a continuación en la Tabla III.1:

**Tabla III.1** Iteraciones del Algoritmo de Euclides Extendido

| $i$ | $r$ | $c$ | $u$ | $v$ |
|---|---|---|---|---|
| -1 | 393 | | $u_{-1}$ | $v_{-1}$ |
| 0 | 267 | | $u_0$ | $v_0$ |
| 1 | 126 | 1 | $u_1$ | $v_1$ |
| 2 | 15 | 2 | $u_2$ | $v_2$ |
| 3 | 6 | 8 | $u_3$ | $v_3$ |
| 4 | 3 | 2 | $u_4$ | $v_4$ |

Como se puede ver en la primera tabla se ha colocado los restos y cocientes calculados previamente. En rojo se tiene los coeficientes que cumplen la operación:

$$r = a\,u + b\,v$$

siendo $a = 393$ y $b = 267$.

Para realizar el cálculo de $u_1$ y $v_1$ en adelante se realiza el siguiente procedimiento:

$x_i = x_{i-2} - c\,x_{i-1}$ , esta regla se utiliza tanto para $u$ como para $v$.

$u_i = u_{i-2} - c\,u_{i-1}$ ,

si $i = 1$, entonces $u_1 = u_{-1} - c\,u_0 = 1 - 1(0) = 1$

$v_i = v_{i-2} - c\,v_{i-1}$,

si $i = 1$, entonces $v_1 = v_{-1} - c\,v_0 == 1 - 2(0) = 1$

Tras aplicar este proceso hasta el final el resultado se muestra en la Tabla III.2.

**Tabla III.2** Cálculo de los coeficientes empleando el Algoritmo Extendido de Euclides

| $i$ | $r$ | $c$ | $u$ | $v$ |
|---|---|---|---|---|
| -1 | 393 | | 1 | 0 |
| 0 | 267 | | 0 | -1 |
| 1 | 126 | 1 | 1 | -1 |
| 2 | 15 | 2 | -2 | 3 |
| 3 | 6 | 8 | 17 | -25 |
| 4 | 3 | 2 | -36 | 53 |

El resultado del algoritmo es el residuo anterior a la operación en que $b$ es $0$.

Resultado: $3 = 393(-36) + 267(53)$

Por lo tanto, $mcd(393,267) = 3$ y $mod(393,267) = 53$

**Fundamentos matemáticos para el proceso de descifrado del algoritmo RSA**

Existen algunos métodos diferentes para calcular una exponenciación modular, como el Algoritmo Binario de Exponenciación de Cuadrados y Múltiplos, el Teorema Remanente Chino (CRT) y el Algoritmo de Exponenciación de Ventanas Deslizantes (SWE).

**Exponenciación Modular**

Para calcular la potencia de un número grande por el módulo de otro número, $a^x mod\ n$ se realizan una serie de multiplicaciones y divisiones, empleando varios métodos que tienen como objetivo minimizar el número de multiplicaciones modulares. Sin embargo, debido a que las operaciones son distributivas, es más rápido hacer la exponenciación como un flujo de multiplicaciones sucesivas, a partir del módulo. No es muy útil para números pequeños, pero sí influenciará cuando se trabaje con números grandes.

Por ejemplo:

$$123^5 (mod\ 511) = 28153056843\ mod(511) = 359$$

Este proceso requiere $10^{31}$ gigabytes de memoria solamente para guardarle en la memoria.

**Exponenciación Binaria**

Por ejemplo, si se desea calcular $a^8 mod\ n$ , no es óptimo usar el método de las multiplicaciones sucesivas y una gran reducción modular:

$$(\ a * a * a * a * a * a * a * a\ )\ mod\ n$$

En lugar de eso, es mucho más eficiente realizar tres multiplicaciones más pequeñas y tres reducciones modulares más pequeñas:

$$(\,(\,a^2 mod\ n\,)^2\ mod\ n\,)^2\ mod\ n$$

A este método se le llama Exponenciación Binaria, la cual utiliza una cadena de adición simple y obvia basada en la representación binaria. Se basa en la propiedad distributiva de la multiplicación $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$ .

Por lo tanto: $a^{b+c} = a^b \cdot a^c$ y $a^{2b} = a^b \cdot a^b = (a^b)^2$.

En el ejemplo anterior, aplicando multiplicación modular, el proceso se podría reducir a:

$$117_{10} \equiv 1110101_2$$
$$117 = 2^0 + 2^2 + 2^4 + 2^5 + 2^6$$
$$117 = 1 + 4 + 16 + 32 + 64$$
$$5^{117}(mod\ 19) = 5^{(1+4+16+32+64)}(mod\ 19)$$
$$= (5 \cdot 5^4 \cdot 5^{16} \cdot 5^{32} \cdot 5^{64}) mod\ 19$$

Si: $$x = 5$$

$$x^2 = x \cdot x\ (mod\ 19) = 25\ (mod\ 19) = 6,$$
$$x^4 = x^2 \cdot x^2\ (mod\ 19) = 36\ (mod\ 19) = 17,$$
$$x^8 = x^4 \cdot x^4\ (mod\ 19) = 289\ (mod\ 19) = 4,$$
$$x^{16} = x^8 \cdot x^8\ (mod\ 19) = 16\ (mod\ 19) = 16,$$
$$x^{32} = x^{16} \cdot x^{16}\ (mod\ 19) = 256\ (mod\ 19) = 9,$$
$$x^{64} = x^{32} \cdot x^{32}\ (mod\ 19) = 81\ (mod\ 19) = 5,$$

Ahora se debe usar propiedades de multiplicación modular para combinar los valores calculados:

$$5^{117}(mod\ 19) = (5 mod\ 19 \cdot 5^4 mod\ 19 \cdot 5^{16} mod\ 19 \cdot 5^{32} mod\ 19 \cdot 5^{64} mod\ 19) mod\ 19$$
$$= (5 \cdot 17 \cdot 16 \cdot 9 \cdot 5) mod\ 19$$
$$= 61200 \cdot mod\ 19$$
$$= 1$$

Este método solamente requirió seis multiplicaciones modulares en lugar de hacer un gran cálculo matemático. Por lo tanto, es el método más eficiente y más utilizado para descifrar un mensaje en el algoritmo RSA con un gran exponente.

Otra forma eficiente de hacer reducciones modulares es empleando el Método de Montgomery, otro método se llama Algoritmo de Barrett [33].

# ANEXO IV

## ENCUESTA A LOS ESTUDIANTES

Esta encuesta busca medir la utilidad del Sistema de Seguridad Informática utilizado. Por favor llénela con total libertad y sinceridad.

1. **¿Considera Ud. la aplicación como uso didáctico y de fácil entendimiento?**

   ○ Muy de acuerdo ○ Algo de acuerdo ○ Algo en desacuerdo ○ Muy en desacuerdo

2. **¿Usaría Ud. la aplicación de uso didáctico para reforzar conocimientos de seguridad informática?**

   ○ Muy de acuerdo ○ Algo de acuerdo ○ Algo en desacuerdo ○ Muy en desacuerdo

3. **¿Pudo observar y entender claramente la implementación y funcionamiento de los algoritmos de seguridad informática implementados?**

   ○ Muy de acuerdo ○ Algo de acuerdo ○ Algo en desacuerdo ○ Muy en desacuerdo

4. **¿A través del uso de la aplicación pudo entender claramente el envío seguro de información a través de la red?**

   ○ Muy de acuerdo ○ Algo de acuerdo ○ Algo en desacuerdo ○ Muy en desacuerdo

5. **¿Cómo calificaría el nivel de aprendizaje que ha proporcionado la aplicación de uso didáctica?**

   ○ Alto ○ Medio ○ Bajo

6. **En general, ¿qué opina de la aplicación de uso didáctica?**

   ○ Excelente ○ Buena ○ Mala ○ Pésima

7. **¿Qué opinión tiene en relación al ambiente gráfico de la aplicación?**

   ○ Adecuado ○ No adecuado

8. **¿Recomendaría la aplicación de uso didáctica?**

   ○ Sí ○ No

9. **¿Cree Ud. que hacen falta más herramientas de uso didáctico en la materia de seguridad para mejorar el nivel de aprendizaje?**

   ○ Sí ○ No

**10. Explique las dificultades de uso que haya encontrado en la aplicación.**

**11. Indique las recomendaciones que se debería hacer para mejorar la aplicación.**

¡Gracias por su colaboración!

En el CD adjunto se encuentran las encuestas digitales realizadas a los estudiantes.

# ANEXO V

El código de cada módulo desarrollado se encuentra en el CD adjunto como información digital.

# ORDEN DE EMPASTADO