



INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

**CODIFICACIÓN DE DATOS PARA
INSTRUMENTACIÓN VIRTUAL DISTRIBUIDA**

T E S I S

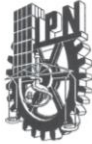
QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS EN
INGENIERÍA DE CÓMPUTO CON
OPCIÓN EN SISTEMAS DIGITALES.

P R E S E N T A:
ING. MARISOL SALINAS SALINAS.

DIRECTOR DE TESIS:
DR. LUIS PASTOR SÁNCHEZ
FERNÁNDEZ



MÉXICO D. F., OCTUBRE DE 2012.



INSTITUTO POLITÉCNICO NACIONAL SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 14:00 horas del día 23 del mes de abril de 2012 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Codificación de datos para instrumentación virtual distribuida"

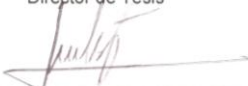
Presentada por la alumna:

SALINAS Apellido paterno	SALINAS Apellido materno	MARISOL Nombre(s)
		Con registro: B 0 1 1 3 7 6

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES**

Después de intercambiar opiniones los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA Director de Tesis



Dr. Luis Pastor Sánchez Fernández



Dr. Sergio Suárez Guerra



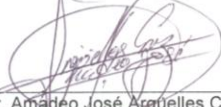
Dr. Oleksiy Pogrebnyak




M. en C. Osvaldo Espinosa Sosa



Dr. Rodrigo López Cárdenas



Dr. Amadeo José Argüelles Cruz


PRESIDENTE DEL COLEGIO DE PROFESORES


Dr. Luis Alfonso Villa Vargas
DIRECCIÓN



INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 30 del mes de octubre del año 2012, el (la) que suscribe Marisol Salinas Salinas alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Cómputo con opción en sistemas digitales con número de registro A011376, adscrito al Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección del Dr. Luis Pastor Sánchez Fernández y cede los derechos del trabajo intitulado Codificación de Datos para Instrumentación Virtual Distribuida, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráfico o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección msalinass@ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Marisol Salinas Salinas.

Nombre y Firma



RESUMEN

En la industria de las comunicaciones es cada vez más importante transmitir datos dentro de los sistemas, independientemente de su localización y su uso, por este motivo se necesita una herramienta que permita llevar la información a diferentes destinos y de forma segura. Esta herramienta puede ser la criptografía, en sus comienzos era considerado un arte, actualmente se desempeña como un conglomerado de técnicas matemáticas que permiten conectar computadoras enviando información de manera segura a través de un canal inseguro. Por lo que es posible cifrar y descifrar datos para enviarlos por un canal de datos en un ambiente distribuido utilizando instrumentación virtual con un desempeño satisfactorio.

Una de las primeras actividades desarrolladas en un sistema de intercambio de datos cifrados entre aplicaciones distribuidas es seleccionar el algoritmo, el cual es DES (Estándar de Cifrado de Datos). Después se programa el algoritmo seleccionado, para el cifrado y descifrado de datos, la siguiente etapa consiste en la implementación de un sistema distribuido para el intercambio de datos cifrados entre aplicaciones desarrolladas mediante herramientas de instrumentación virtual.

Uno de los aportes de este trabajo de tesis, con respecto a otras implementaciones consiste en haber generado un par de módulos para cifrar y descifrar datos en instrumentación virtual para que sea utilizado en sistemas distribuidos, pues actualmente no existen herramientas y funciones pre-programadas para el uso de criptosistemas dentro de las aplicaciones en los sistemas distribuidos que realizan funciones de monitoreo o supervisión y adquisición de datos, por lo tanto se obtiene una innovación tecnológica.

Además, se diseñó y programó en instrumentación virtual un sistema de adquisición de datos y control supervisorio (SCADA), que integra una aplicación cliente para intercambiar datos cifrados desde un servidor TCP, para después ser descifrados con el mismo criptosistema y la misma clave de cifrado.

Al terminar, se tienen los resultados del envío de información generada del SCADA por medio de un canal, que permiten cifrar mensajes simples en un emisor y enviarlos al receptor para que sea únicamente éste quien pueda descifrarlos con la única clave de este criptosistema. La comunicación TCP del cliente-servidor fue implementada utilizando instrumentación virtual.





ABSTRACT

In the communications industry the data transmission inside the systems, regardless of the location and applications, is becoming more important. For this reason it is necessary to develop a tool that allows transferring information to different locations in a safe way. This tool can be cryptography that in the beginning was considered almost as an “art”, nowadays this is a set of mathematical techniques that allow a safe connection between computers through a safe channel. Actually it is possible to encode and decode data in order to be sent through a safe channel in a distributed computer environment, using virtual instrumentation with a very good performance.

One of the first tasks developed in an encoded exchange data system is the selection of the algorithm, in this case DES (Data Encryption Standard). Then the selected algorithm is implemented to encode and decode the information. The next stage is the implementation of a distributed system that allows the encoded data exchange between applications developed through virtual instrumentation.

One of the main contributions in this thesis is the generation of a couple of software modules that encode and decode data in virtual instrumentation, that will be used in distributed systems, since actually there are no tools neither functions preprogrammed that could be applied to cryptosystems in distributed systems that need to deal with supervising and data acquisition, therefore a technological innovation is obtained.

In addition, it was developed a supervisory control and data acquisition system (SCADA), which integrates a client application whose main aim is the exchange of encoded data from a TCP server that will be decoded with the same cryptosystem and the same encoded key.

Finally, the information generated by the SCADA system is recovered through a channel, that allows to encode simple messages in a transmitter and then send them to a receiver, that will be the only able to decode the message with the encoded key generated by the cryptosystem; The TCP client-server communication was implemented using virtual instrumentation.





AGRADECIMIENTOS

A la memoria de mi mamá Goya (Gregoria Bardales Gómez†), a mi tía Aurelia Salinas Bardales† y a mi padre, Simón Salinas Jiménez†.

Con amor para Juan Francisco Novoa Colín, por ser un maravilloso esposo.

A mis hermanos, muy especialmente a Alfonso Salinas Salinas.

A mi asesor el Dr. Luis Pastor Sánchez Fernández por su gran apoyo.

A mis amigos: el Dr. Rodrigo López Cárdenas, al Ing. Carlos Rojas Hernández, a la Lic. Janet Guerrero Mendoza y a la Dra. Jessica Azucena Escobar.





ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT.....	V
AGRADECIMIENTOS.....	VI
ÍNDICE DE CONTENIDOS	VII
ÍNDICE DE FIGURAS	XI
ÍNDICE DE TABLAS.....	XIV
GLOSARIO	XV
1. INTRODUCCIÓN	1
1.1 Antecedentes y motivación	1
1.2 Problemas a resolver	4
1.3 Justificación	4
1.4 Hipótesis	5
1.5 Objetivos	5
1.5.1 General.....	5
1.5.2 Específicos	6
1.6 Alcance	6
1.7 Contribuciones del trabajo	6
1.8 Resumen de la metodología y desarrollo de la investigación	6
1.9 Estructuración de la tesis.....	7
2. ESTADO DEL ARTE	8
2.1 Introducción.....	8
2.2 Criptografía	9
2.2.1 Conceptos básicos.....	9





2.3 Criptosistemas	12
2.3.1 Elementos fundamentales y clasificación.	12
2.4 Criptosistemas de clave privada	15
2.4.1 Transposición:.....	16
2.4.2 Cifrados monoalfabéticos	17
2.4.3 Cifrados polialfabéticos.	18
2.4.4 Cifrados por sustitución homofónica	19
2.4.5 Redes de Feistel	19
2.4.6 S-Cajas.....	21
2.4.7 CAST	23
2.4.8 Blowfish y Twofish	23
2.4.9 El algoritmo IDEA.....	25
2.4.10 El algoritmo AES.....	26
2.4.11 Modos de operación del cifrado por bloques	27
2.4.11.1 ECB (Electronic Codebook).....	28
2.4.11.2 CBC (Cipher Book Chaining Mode).....	28
2.4.11.3 CFB (Cipher-Feedback Mode)	29
2.4.12 El Algoritmo DES.....	30
2.5 Algunas propiedades de DES	36
2.5.1 Variantes de DES.....	37
2.5.1.1 DES múltiple	37
2.5.1.2 DES con subclaves independientes	38
2.5.1.3 DES generalizado	38
2.5.1.4 DES con S-Cajas alternativas	38
2.5.1.5 Robustez del DES.....	38
2.5.2 Claves débiles, semi-débiles, y posiblemente débiles	39
2.6 Criptosistemas de clave pública	40
2.6.1 El Algoritmo RSA	40
2.6.2 Algoritmo de ElGamal.....	41
2.6.3 Algoritmo de Rabin	42
2.6.4 La nueva dirección en criptografía.....	43
2.7 Aplicación práctica de cifrado de datos en distintos lenguajes de programación	44
2.8 Patentes de los algoritmos criptográficos	44
2.9 Sistemas distribuidos hoy en día	45
2.9.1 Implementaciones en sistemas distribuidos con instrumentación virtual:	47
2.9.2 Sistema de monitoreo y control en el país	49
2.9.3 Sistema digital de monitoreo y control	49
2.10 Conclusiones del capítulo	51
3. MARCO TEÓRICO	52
3.1 Introducción	52
3.2 Selección del algoritmo	53





3.3 Sistemas SCADA	53
3.3.1 Arquitectura de un sistema SCADA	54
3.3.1.1 El hardware.....	56
3.3.1.2 Interfaz hombre-máquina (HMI, MMI).....	56
3.3.1.3 Unidad central (MTU, Master Terminal Unit)	56
3.3.1.4 Unidad remota (RTU, Remote Terminal Unit).....	57
3.3.1.4.1 RTU	58
3.3.1.4.2 PLC.....	58
3.3.1.4.3 IED	58
3.3.1.4.4 Sistemas remotos	59
3.3.1.5 Sistemas de comunicación.....	60
3.4 Sistema distribuido.....	61
3.4.1 El modelo cliente/servidor	61
3.4.2 Elementos básicos del modelo cliente/ servidor	63
3.4.3 Infraestructura.....	64
3.4.4 Servicios de seguridad	64
3.5 Programación gráfica.....	65
3.6 La arquitectura de protocolos TCP/IP	65
3.6.1 Las capas de TCP/IP	66
3.6.2 TCP y UDP.....	67
3.6.3 TCP/IP en LabVIEW	67
3.6.3.1 Puerto.....	68
3.6.3.2 Dirección IP.....	68
3.6.3.3 Servidor de datos	68
3.6.3.4 Cliente de datos	69
3.7 Lenguajes de programación visuales	70
3.8 Propuesta de la tesis	72
3.9 Conclusiones del capítulo.....	73
4. PROGRAMACIÓN DEL CRIPTOSISTEMA	74
4.1 Introducción.....	74
4.2 Elección del algoritmo de clave simétrica, DES	74
4.3 Panel del cifrado de datos	76
4.4 Panel frontal del descifrado de datos.....	77
4.5 Descripción de la programación del criptosistema	79
4.6 Conclusiones del capítulo.....	83





5. INSERCIÓN DEL CRIPTOSISTEMA EN EL MÓDULO DEL SISTEMA SCADA 84

5.1 Introducción..... 84

5.2 Implementación del sistema cliente/servidor 85

 5.2.1 Descripción de la implementación del cliente 85

 5.2.2 Implementación del modelo servidor..... 88

5.3 Panel de cambio de parámetros de la base de datos..... 93

 5.3.1 Panel de la base de datos del sistema..... 96

 5.3.2 Panel de gráfico de la variable Vce..... 97

 5.3.3 Panel de gráficos de las variables Vce del sistema 100

 5.3.4 Panel del gráfico de la variable histórico del Vce..... 101

 5.3.5 Panel de la recopilación de datos 104

 5.3.6 Panel de la lectura de datos binaria 106

5.4 Conclusiones del capítulo..... 107

6. PRUEBAS Y RESULTADOS 108

6.1 Funcionamiento del criptosistema 108

6.2 Valores de prueba en el criptosistema 117

 6.2.1 Valores de prueba en el criptosistema aplicando clave débil..... 117

 6.2.2 Valores de prueba en el criptosistema aplicando clave semi-débil 119

6.3 Pruebas de comunicación del sistema 122

6.4 Pruebas de tiempo de procesamiento del criptosistema..... 123

6.5 Conclusiones del capítulo..... 126

CONCLUSIONES 127

RECOMENDACIONES Y TRABAJO FUTURO..... 128

REFERENCIAS BIBLIOGRÁFICAS..... 129

APÉNDICES 134





ÍNDICE DE FIGURAS

Fig. 1. Esquema fundamental de un criptosistema.	13
Fig. 2. Esquema de un criptosistema simétrico.	15
Fig. 3. Estructura de las redes de Feistel.	20
Fig. 4. Ejemplo de redes de Feistel.	21
Fig. 5. Esquema del funcionamiento de S-Cajas.	22
Fig. 6. Modo ECB (Electronic Codebook).	28
Fig. 7. Modo CBC (Cipher Book Chaining Mode).	29
Fig. 8. Modo CFB (Cipher-Feedback Mode).....	29
Fig. 9. Esquema general del algoritmo DES.	30
Fig. 10. Cálculo de las subclaves k_i	32
Fig. 11. Ronda de la red de Feistel del algoritmo DES.	34
Fig. 12. Red Ethernet de computadoras y otros nodos de cálculo.	46
Fig. 13. Ejemplo de una red híbrida ethernet con E/S.	47
Fig. 14. Estructura básica de un sistema de supervisión y mando.	55
Fig. 15. Arquitectura general de una RTU.	60
Fig. 16. Esquema del modelo cliente/servidor.	62
Fig. 17. Componentes básicos del modelo cliente/servidor.	63
Fig. 18. Interfaz con las funciones de TCP/IP en LabVIEW.	68
Fig. 19. Diagrama de bloques del servidor de datos en LabVIEW.	69
Fig. 20. Diagrama de bloques del cliente de datos en LabVIEW.....	70
Fig. 21. Interfaz de automatización gráfica de usuario (GUI) para aplicaciones Windows y Web. (Cortesía de National Instruments)	71
Fig. 22. Panel frontal del cifrado de datos	77
Fig.23. Panel frontal del descifrado de datos.	78
Fig.24. Muestra el VI de la jerarquía del sistema de cifrado de datos	78
Fig.25. Diagrama de flujo del criptosistema DES	80
Fig.26. Diagrama de flujo de la obtención de las claves para el DES.....	81
Fig.27. Rondas de las redes de Feistel del DES.	82
Fig. 28. Panel frontal de la conexión del cliente.	85
Fig.29. Diagrama de flujo de la conexión del cliente.	86
Fig. 30. Panel frontal del SCADA como cliente con los módulos principales.	87
Fig. 31. Diagrama de flujo del SCADA como cliente.....	88
Fig. 32. Panel frontal de la conexión del servidor.	89
Fig. 33. Diagrama de flujo del SCADA como servidor.....	90
Fig.34. Panel frontal del SCADA como servidor con los módulos principales.....	91
Fig.35. Diagrama de flujo del servidor	92
Fig. 36. Jerarquía de los módulos principales	93
Fig. 37. Pantalla del panel de cambios de parámetros de la base de datos.	94
Fig. 38. Pantalla para decidir guardar los cambios de los parámetros del panel de la base de datos.	94
Fig. 39. Pantalla para teclear el nombre con el que se almacenará el archivo de la base de datos.	95
Fig.40. Diagrama de flujo del panel de cambio de parámetros de la base de datos	96
Fig. 41. Base de datos para las variables del sistema.	97
Fig. 42. Sub-VI frontal del gráfico de la variable Vce.	98
Fig. 43. Diagrama de flujo del gráfico de la variable Vce.	99
Fig. 44. Panel de gráficos de las variables Vce del sistema.	100
Fig. 45. Diagrama de flujo del panel de gráfico de la visualización de la variables Vce del sistema.	101
Fig. 46. Panel del gráfico histórico de la variable Vce.....	102
Fig. 47. Diagrama de flujo del panel de gráfico histórico de una variable Vce.	103





Fig. 48. SubVI de la recopilación de los datos 104

Fig. 49. Diagrama de flujo del subVI recopilación de datos. 105

Fig. 50. SubVI que realiza la lectura binaria de los datos..... 106

Fig.51. Conversión del mensaje de entrada a un formato binario. 109

Fig.52. Muestra el subVI que se utilizó como prueba de las claves, iniciando con K en hexadecimal. . 110

Fig.53. La prueba de los corrimientos de las claves..... 113

Fig.54. Pruebas de las S-Cajas..... 114

Fig.55. Prueba de la permutación inicial IP 114

Fig.56. Pruebas de cifrado y descifrado con DES. 115

Fig.57. Prueba de cifrado y descifrado en bloques. 116

Fig. 58. Pruebas del cifrado de datos aplicando una clave..... 117

Fig. 59. Prueba del criptosistema con una clave débil. 119

Fig. 60. Pruebas del criptosistemas tras aplicar claves semi-débiles..... 120

Fig. 61.Pruebas del criptosistema tras aplicar claves semi-débiles 121

Fig.62. Pruebas de la comunicación del sistema en la conexión del servidor. 123

Fig.63. Pruebas de la comunicación del sistema en la conexión de cliente 123

Fig. 64. Prueba del tiempo de ejecución de procesamiento en el cifrado de datos..... 124

Fig. 65. Prueba del tiempo de ejecución de procesamiento del descifrado. 125

Fig.66. Diagrama encargado de obtener las variables a cifrar y convertirlas a la cadena mensaje simple.
..... 134

Fig.67. SubVI para dividir el mensaje simple en bloques de tamaño fijo de 64 bits..... 135

Fig.68. SubVI general del DES con sus entradas por un lado el mensaje simple y la clave y su salida se
obtienen los mensajes cifrados en bloques. 136

Fig.69. Permutación inicial IP a partir de la entrada de datos binarios. 136

Fig.70. SubVI para realizar las redes de Feistel..... 137

Fig.71. SubVI que muestra la sustitución de S-Cajas..... 138

Fig.72. Última permutación IP-1 dentro de las rondas de las redes de Feistel..... 138

Fig.73. SubVI que aplica la permutación PC-1 a la clave Ke. 139

Fig.74. Rotación de un bit o dos para las claves..... 140

Fig.75. Permutación PC-2 aplicada al final de cada subclave. 141

Fig.76. SubVI que muestra como se tienen los mensajes descifrados y descompactados en las variables
descifradas. 141

Fig.77. Generación de las claves en orden inverso para ocuparlas en el descifrado..... 142

Fig. 78. Inicialización de las variables del panel de la base de datos..... 143

Fig.79. Actualización y guardar las variables del panel de base de datos. 144

Fig. 80. Obtiene el máximo común divisor de los periodos de muestreo..... 145

Fig. 81. Almacena el archivo de la base de datos. 146

Fig. 82. Ejecuta la escritura del archivo para almacenar la base de datos 147

Fig. 83. Reinicializa la recolección de datos para la base de datos. 148

Fig. 84. Realiza la gráfica de una variable Vce. 149

Fig. 85. Realiza las graficas para visualización de alarmas. 150

Fig. 86. Genera el archivo para el gráfico histórico de una Vce. 151

Fig.87. Realiza la recopilación de datos 151

Fig. 88. Determina ¿Cuántas Vce hay que leer? y ¿Cuáles? 152

Fig. 89. Ejecuta SubVI de la opción Gen 153

Fig. 90. Lee datos de la tarjeta USB 154

Fig. 91. Realiza la conversión a unidades de ingeniería. 155

Fig. 92. Almacena el histórico en un arreglo. 156

Fig. 93. Realiza el proceso de la alarma. 157

Fig. 94. Inicializa la base de datos. 158

Fig. 95. Almacena los datos en un archivo binario. 159

Fig. 96. Lee los datos de la base de datos. 160





Fig. 97. Realiza la comunicación del servidor. 161
Fig. 98. Maneja los errores en la comunicación del servidor. 161
Fig. 99. Inicializa el arreglo de la variable MENSAJES CIFRADOS EN BLOQUES. 162
Fig. 100. Transmite los MENSAJES CIFRADOS EN BLOQUES. 162





ÍNDICE DE TABLAS

Tabla 1. Transposición de elementos.....	16
Tabla 2. Ejemplo de S-Cajas del DES.....	23
Tabla 3. Permutación Inicial (IP).....	31
Tabla 4. Permutación 1 (PC-1).....	31
Tabla 5. Permutación 2 (PC-2).....	31
Tabla 6. Desplazamientos de los bits en cada ronda.....	33
Tabla 7. Expansión (E).....	34
Tabla 8. Permutación (P)	34
Tabla 9. Permutación Final (IP-1)	34
Tabla 10. S-Cajas del algoritmo DES	35
Tabla 11. Tiempo medio para la búsqueda exhaustiva de claves.	39
Tabla 12. Claves débiles para el algoritmo DES (64 bits), expresadas en hexadecimal.	39
Tabla 13. Claves semi-débiles para el algoritmo DES (64 bits), expresadas en hexadecimal.	40
Tabla 14. Patentes registradas en EE.UU.....	45
Tabla 15. Pruebas de distintas claves débiles y el resultado obtenido en los mensajes cifrados.....	118
Tabla 16. Pruebas de distintas claves semi-debiles y el resultado obtenido en los mensajes cifrados...	121
Tabla 17. Tiempos de ejecución del cifrado y descifrado de datos.....	125





GLOSARIO

Términos

Acondicionamiento de señal: Proceso mediante el cual la señal medida se adecua para que los dispositivos o criterios de tratamiento la puedan manejar más fácilmente.

ActiveX: Tipo de componente COM que puede autoregistrarse, también conocido como control *ActiveX* que implementa algunas interfaces estándar para incrustación, interfaces de usuario, métodos, propiedades, eventos y persistencias.

Adquisición de datos: Proceso mediante el cual se toman muestras de una señal analógica para tener una representación discreta de la misma.

Algoritmo criptográfico: Algoritmo cuyo objetivo es el cifrado de datos, generalmente basado en una función matemática.

Algoritmo de optimización: Algoritmo que encuentra los parámetros más apropiados del modelo matemático para ajustarlo a la curva adquirida.

Algoritmo: Detalle de pasos específicos, ordenados y finitos para la solución de un problema.

Ataque criptográfico: Intento de compromiso de una parte o de la totalidad de la seguridad de un criptosistema.

Ataque por fuerza bruta: Ataque que requiere la prueba de todos (o una fracción importante) los valores posibles hasta que el correcto sea encontrado. También es llamado de búsqueda exhaustiva.

Bidireccional: Término que se utiliza para indicar que un solo puerto de comunicaciones puede transmitir y recibir datos.

Bus de campo: Sistema de transmisión de información (datos) que simplifica la instalación y operación de máquinas y equipamientos industriales.

Canal: Camino por el que circula o se transmite información en un sistema.

Clave de cifrado: Grupo de caracteres utilizada para el cifrado y descifrado de información.

Confusión: La forma más sencilla de obtener confusión es mediante sustituciones. En el caso de cifrado por bloques, se sustituye una secuencia larga de caracteres (un bloque), por otra a base de manipular sub-bloques del mensaje como unidades completas.

Control distribuido: Tipo de control de donde una serie de controladores locales, se hallan distribuidos en diferentes puntos de la planta, y a su vez estos se conectan a un cuarto de control central.

Criptanálisis: Es el estudio de las técnicas matemáticas para intentar romper esquemas criptográficos.

Criptografía: Un conjunto de técnicas matemáticas para crear esquemas de cifrado y descifrado de mensajes

Criptología: Del griego criptos = oculto y logos = tratado, ciencia que designa a dos disciplinas complementarias que son la criptografía y el criptoanálisis.

DataSocket: Grupo de primitivas de la capa 4 del modelo OSI, la capa de transporte.





Difusión: Buscando por el efecto de disimular las redundancias del texto claro al extenderlas por todo el texto cifrado, la manera más sencilla de obtener difusión es mediante permutaciones o transposiciones, se describen a continuación.

Dispositivo inteligente: En el área de control industrial, dispositivo digital de campo que además de proporcionar datos, hacen funciones de estado, funcionamiento, configuración, materiales de construcción, etc.

Estándar RS-232: Interface entre un equipo terminal de datos y un equipo de comunicación de datos empleando un intercambio en modo serie de datos binarios.

Frecuencia de muestreo de Nyquist: Por el teorema de muestreo de *Nyquist*, tasa de muestreo mínima para el evitar el *aliasing*. Debe ser por lo menos el doble de la frecuencia de la componente más alta en frecuencia de la señal a muestrear.

Frecuencia de muestreo: Frecuencia a la cual se toman las muestras de una señal.

Instrumento virtual: Programa en *LabVIEW*.

Interfaz: Zona de comunicación o acción de un sistema sobre otro.

Interpolación: Proceso mediante el cual se introducen puntos entre otros dos de una gráfica.

Intervalo de muestreo: Se conoce como el recíproco de la frecuencia de muestreo.

Panel de operador: Panel donde se visualizan todos los parámetros del proceso monitoreado.

Proceso industrial: Es la secuencia de operaciones desarrolladas progresivamente, en forma continua o discreta, caracterizada por una serie de cambios graduales que se suceden en forma fija o con cierta probabilidad, para llegar a un fin o resultado determinado dentro de un sistema.

Protocolo: También conocido como control de acceso al medio, es el factor que caracteriza el funcionamiento de una red. De él dependen parámetros básicos como rendimiento, fiabilidad, disponibilidad y la gestión de la red.

Ronda: Cada iteración o aplicación de las técnicas de *confusión* y *difusión* en un algoritmo de cifrado.

Sensor: Dispositivo empleado para transformar un tipo de energía en otro.

Sistema: Es un conjunto de objetos, conceptos o entidades que relacionadas de tal manera forman un todo y funcionan como una unidad. Un sistema puede ser parte de otro más grande o estar integrado por otros más pequeños.

Valor crítico: Valor que toma la variable de proceso, que indica que el mismo ha llegado a un rango de funcionamiento peligroso.

Variable: Parámetro no constante que requiere de control dentro de un proceso industrial.





Siglas

A/D: *Analog to Digital Converter*. Convertidor Analógico Digital.

AES: *Advanced Encryption Standard*. Estándar de Cifrado Avanzado.

ANSI: *American National Standards Institute*. Instituto Nacional de Estándares Americano.

ASI: *Actuator Sensor Interface*. Interfaz Sensor Actuador.

CAN: *Controller Area Network*.

CIM: *Computer Integrated Manufacturing*.

COM: *Component Object Model*. Modelo Objeto Componente.

DCS: *Distributed Control System*. Sistema de Control Distribuido.

DES: *Data Encryption Standard*. Estándar de Cifrado de Datos.

DSA: *Digital Signature Algorithm*. Algoritmo de Firma Digital.

DSS: *Digital Signature Standard*. Estándar de Firma Digital.

FIPS: *Federal Information Processing Standard*.

FPGA: *Field Programmable Gate Arrays*.

GPIB: *General Purpose Interface Bus*. Bus de Interfaz de Propósito General.

IDEA: *International Data Encryption Algorithm*. Algoritmo Internacional de Cifrado de Datos.

IEC: *International Electrotechnical Commission* o Comisión Electrotécnica Internacional.

IEEE: *Institute of Electrical and Electronics Engineers*. Instituto de Ingenieros Eléctricos y Electrónicos.

IETF: *Internet Engineering Task Force*.

IP: *Internet Protocol*. Protocolo de Internet.

IRQ: *Interrupt Request Line*. Línea de Petición de Interrupción.

ISO: *International Standard Organization*. Organización Internacional de Estándares.

LabVIEW: *Laboratory Virtual Instrument Engineering Workbench*.

LIA: Límite Inferior de Alarma.

LSA: Límite Superior de Alarma.

MATLAB: *Matrix Laboratory*. Laboratorio de Matrices.





MMI: *Man Machine Interface*. Interfaz Máquina Hombre.

MTU: *Master Terminal Unit*. Unidad Central.

NBS: National Bureau of Standards. Oficina Nacional de Estándares de los Estados Unidos.

NIST: National Institute of Standards and Technology. Instituto Nacional de Estandarización y Tecnología de Estados Unidos.

NSA: National Security Agency. Agencia de Seguridad Nacional de los EE.UU.

OSI: *Open System Interconnection*. Interconexión de Sistemas Abiertos.

OPC: *OLE for Process Control*. Control de Procesos para OLE.

PC: *Personal Computer*. Computadora Personal.

PDA: *Personal Digital Assistant*. Asistente Digital Personal.

PGP: Pretty Good Privacy

PLC: *Programable Logic Controller*. Controlador Lógico Programable.

RTU: *Remote Terminal Units*. Unidades Remotas

SCADA: *Supervisory Control and Data Acquisition*. Adquisición de Datos y Control Supervisorio.

SCMA: *Carrier Sense Multiple Access*.

SDS: *Smart Distributed System*. Sistema Distribuido Inteligente.

SHA: *Secure Hash Algorithm*. Algoritmo de Hash Seguro.

TCP: *Transmission Control Protocol*. Protocolo de Control de Transmisión.

UDP: *User Datagram Protocol*. Protocolo de Datagramas de Usuario.

URL: *Uniform Resource Locator*. Localizador Uniforme de Recursos.

VPN: *Virtual Protocol Network*. Redes Privadas Virtuales.

.





1. INTRODUCCIÓN

1.1 Antecedentes y motivación

El avance de la ingeniería y la ciencia son de extrema importancia para utilizar sistemas de vehículos espaciales, militares, guía de misiles, así como controlar automáticamente las variables de procesos industriales, como presión, temperatura, humedad, viscosidad y flujo, entre otras.

Una vez que se tienen las señales de la variable a controlar, es necesario que se acondicionen antes de que lleguen al convertidor analógico-digital (A/D). El acondicionamiento óptimo de las señales permite que el proceso posterior que se le aplique a dicha señal sea correcto.

Un sistema de adquisición de datos en un ambiente distribuido estándar es en esencia una red de computadoras. El sistema debe ser confiable, seguro y usar estándares en la industria en lo relativo a protocolos en interfaces de comunicación.

Referente a la parte de seguridad de un sistema de adquisición de datos en un ambiente distribuido estándar veremos útil el aplicar técnicas matemáticas en el envío de información, así transformar la información y que ésta pueda viajar a través de un canal de manera segura aún si ésta llegará a manos no deseadas.

Una de las transformaciones matemáticas que se le puede dar a la información consiste en cifrarla para protegerla de intrusos.

En vez de ocultar la existencia del mensaje, se busca proteger su contenido o hacerlo ininteligible para receptores no autorizados. Al estudio de los sistemas de cifrado y su descifrado se le llama Criptografía.

La Criptografía tiene una historia larga y fascinante. El libro enciclopédico escrito por D. Kahn llamado *The Codebreakers* contiene los datos de criptología desde su uso inicial con los Egipcios hace unos 4000 años hasta el siglo XX en el que ha desempeñado un papel





crucial en el resultado de las dos guerras mundiales, donde fue utilizada la Criptografía como una herramienta para proteger los secretos y estrategias nacionales [34].

En la época de los romanos se utilizaron sistemas de sustitución, siendo el método más conocido el de sustitución Monoalfabética de Julio César, llamado así porque es el que empleaba Julio César para enviar mensajes secretos, es uno de los algoritmos criptográficos más simples. Se conocen también los Cifrados Polialfabéticos, un ejemplo típico debe su nombre a Blaise de Vigenère su creador, y que data del siglo XVI. Existe además, una familia de algoritmos que trata de ocultar las propiedades estadísticas del mensaje simple conocida como Cifrado por Sustitución Homofónica. Al cambiar el orden dentro del mensaje se tiene un Cifrado de Transposición. Considerando los Cifrados mencionados como parte de la Criptografía Clásica[17][31][51].

Fundamentalmente la teoría criptográfica se apoya en el trabajo de Claude Shannon en sus artículos [9][11], sienta las bases de la Teoría de la Información y de la criptografía moderna. Según la Teoría de Shannon, las dos técnicas básicas para ocultar la redundancia en un mensaje simple son la *confusión* y la *difusión*[10].

La proliferación de computadoras y sistemas de comunicación en los años 60's demandaron protección para la información en forma digital en el sector privado y también servicios seguros. Entonces, surgió la necesidad de comunicaciones confidenciales entre equipos de cómputo diversos. Los fabricantes de estos equipos proveían de software (programas) para tal fin, pero no había garantía alguna del nivel de seguridad proporcionado por dichos programas y además no existía compatibilidad entre programas hechos por diferentes fabricantes. Muchos de los mecanismos de cifrado de datos eran hechos a la medida del cliente, poco generales en su rango de aplicación y basados en mecanismos oscuros[21].

Ante esta situación la oficina nacional de estándares de los Estados Unidos (National Bureau of Standards o NBS, actualmente al National Institute of Standards and Technology o NIST) estableció un programa de protección de datos. Como parte de este programa se pretende establecer un algoritmo criptográfico estándar para ser utilizado por las dependencias gubernamentales. Así el 15 de mayo de 1973 el NBS emitió una convocatoria pública para recibir propuestas para un algoritmo criptográfico. Dicho algoritmo debía satisfacer los siguientes requisitos:

1. Proveer un alto nivel de seguridad.
2. Estar completamente especificado y ser fácil de entender.
3. La seguridad del algoritmo debe residir enteramente en la clave y no en el supuesto de mantener secreto el algoritmo mismo.
4. El algoritmo debe poder ser puesto a disposición del público sin restricciones (i.e. no debe haber licencias de uso sobre él).





Nadie se presentó a esta convocatoria, por lo que se emitió una nueva el 27 de agosto de 1974. En atención a ésta última, se presentó un algoritmo diseñado por un grupo de trabajo de los laboratorios de investigación de IBM; el nombre del algoritmo era Lucifer[15].

La idea central de Lucifer fue de Horst Feistel, un inmigrante alemán que arribó a los Estados Unidos en 1932 según algunas fuentes[28], o en 1934 según otras [53]. Feistel siempre quiso trabajar en criptografía, pero debido a su origen durante la segunda guerra mundial estuvo en arresto domiciliario. Luego de la guerra, trabajó un tiempo como investigador en el área de criptografía para la Fuerza Aérea y más tarde trabajo en la misma área para una compañía privada (Maitre Corporation). Al parecer [53] los proyectos de Feistel en ambos empleos fueron cancelados por intervención de la NSA (National Security Agency), dependencia gubernamental que tiene a su cargo todo lo relacionado con la seguridad de comunicaciones, la criptografía, y el criptoanálisis en los Estados Unidos. Finalmente Feistel se incorporó en 1967 al IBM Thomas J. Watson Reserch Laboratory, de donde surgió Lucifer, a principios de los setenta.

La oficina de estándares había solicitado a la NSA la evaluación de las propuestas que recibiera, así que la NSA se dio a la tarea de evaluar a Lucifer y hacerle algunas modificaciones.

En el esquema original de Lucifer el mensaje era un bloque de 128 bits de longitud al igual que la clave usada para cifrarlo. De los 128 bits de clave realmente sólo se usaban 112, porque el octavo bit de cada grupo de ocho bits consecutivos (que en adelante llamaremos bytes) se consideraba un bit de verificación de paridad de los otros siete. Luego de las modificaciones de NSA el mensaje era de 64 bits de longitud al igual que la clave, de la que sólo se usaban realmente 56 bits, siguiendo el mismo esquema de verificación de paridad descrito. Otras de las modificaciones a Lucifer fueron en las llamadas cajas S (S- box), el corazón criptográfico de Lucifer y añadir una permutación inicial al texto de entrada, misma que se deshace (aplicando la permutación inversa) al final; esta permutación no tiene propósito criptográfico alguno, es sólo para que al implementar en hardware el algoritmo fuera eficiente la carga del texto a cifrar.

Por supuesto muchos expertos del área (como Whitfield Diffie) sospecharon de las modificaciones de NSA al algoritmo original. Las modificaciones se habían hecho sin justificación técnica alguna, se redujó el tamaño de la clave, lo que aparenta debilitar el algoritmo. Algunos sospechaban que NSA había puesto “puertas traseras” en el algoritmo que le permitieran a sus expertos y sus máquinas descifrar fácilmente las comunicaciones cifradas con el algoritmo modificado. De hecho en 1978 un comité especial del Senado de los Estados Unidos investigó la controversia y la parte no clasificada del resumen se exonera a NSA de haber hecho modificaciones maliciosas. Más adelante serán claras las razones por las que NSA modificó Lucifer y también será claro que la NSA sabía, evidentemente, más de lo que todos sospechaban (que, paradójicamente, era lo que algunos sospechaban).





Finalmente, el 23 de noviembre de 1976 el algoritmo propuesto, con las modificaciones de la NSA, se adoptó como estándar para comunicaciones gubernamentales no clasificadas, el famoso estándar de cifrado de datos o DES (Data Encryption Standard). En 1981 se adoptó como estándar para el sector privado bajo el nombre de DEA (Data Encryption Algorithm). La especificación del estándar hacia obligatoria la revisión del mismo periódicamente: en 1983 DES fue certificado nuevamente; igual que en 1987, a pesar de las reticencias de NSA que sospechaba que podría romperse pronto. En 1993, a pesar de que ya había tenido problemas la re-certificación en 1987, se volvió a confirmar, en buena medida, porque no había otras alternativas. En enero de 1997 se anunció el proyecto del Advanced Encryption Standard o AES y en septiembre de ese año se abrió una nueva convocatoria para recibir propuestas que reemplazaran a DES. DES fue certificado por última vez en enero de 1999[38] ya que el 2 de octubre de 2000 se anunció oficialmente al algoritmo ganador de la convocatoria de 1997: Rijndael, un algoritmo propuesto por los belgas Joan Daemen y Vincent Rijmen. En noviembre de 2001 se publicó el nuevo estándar AES[37] y entró en vigor el 26 de mayo de 2002.

1.2 Problemas a resolver

La necesidad de mantener confidenciales las comunicaciones dentro de un sistema distribuido que utilice una red local Ethernet e Internet.

Carencia de módulos pre-programados para el cifrado de datos en aplicaciones de instrumentación virtual.

1.3 Justificación

Existe una demanda cada vez mayor en aumentar el grado de automatización en todas las plantas industriales y en diversas tareas de mediciones y monitoreo de tal manera que por un lado aumente la productividad, por otro se disminuya el gasto de energía y, finalmente, que se reduzca el nivel de contaminación del medio ambiente. Ello hace que la aplicación de computadoras sea una necesidad ineludible.

Además de que hoy en día, tener un sistema que cumpla con los estándares de gestión de la seguridad de información sinónimo de calidad de servicio. Principalmente por el uso masivo de Internet, el tema de la protección de la información se ha transformado en una necesidad y con ello se popularizan las técnicas asociadas a la criptología como son: cifrado, descifrado, criptoanálisis, firma digital, etc.

Ya no sólo se comentan estos temas en las universidades. Ahora, cualquier usuario desea saber, por ejemplo, qué significa firmar un e-mail o qué significa que en una comunicación





con su banco aparezca un candado en la barra de tareas de su navegador o al adquirir un software actual que contenga seguridad añadida.

Aunque no será la única, una de las herramientas de protección de datos más efectiva es el uso de técnicas criptográficas.

Aplicar un algoritmo directo a la información que se enviará por un canal inseguro que permita transmitir dicha información con la certeza de que si es capturado por un intruso, este desconocerá del significado de dicha información.

Un sistema de adquisición de datos en un ambiente distribuido envía y recibe datos desde lugares remotos y por ello debe ser seguro ya que con esta información se toman decisiones dentro del sistema.

En los ambientes de programación gráficos basados en el concepto de programación de flujo de datos, existe uno que es *LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench)*. Este tipo de programación ha sido ampliamente adoptado por la industria, ambientes universitarios y laboratorios de investigación. En este tipo de programación no existen módulos de cifrados de datos.

En varios proyectos es posible intercambiar datos en una red inalámbrica como el denominado *sistema de monitoreo ambiental de ruido en el centro histórico*, pero en sistemas como éste, puede resultar necesario elevar el nivel de seguridad en el envío de la información.

1.4 Hipótesis

Es posible cifrar y descifrar información para enviarlos por un canal de datos en un ambiente distribuido, utilizando instrumentación virtual, con un desempeño satisfactorio mediante el algoritmo Estándar de Cifrado de Datos (DES).

1.5 Objetivos

1.5.1 General

Desarrollar un sistema de intercambio de datos cifrados entre aplicaciones distribuidas mediante herramientas de instrumentación virtual, con el algoritmo Estándar de Cifrado de Datos (DES).





1.5.2 Específicos

1. Seleccionar un algoritmo de cifrado de datos, para intercambio de información entre aplicaciones distribuidas de instrumentación virtual.
2. Programar el algoritmo seleccionado.
3. Implementar un sistema de información para el intercambio de datos cifrados entre aplicaciones distribuidas desarrolladas mediante herramientas de instrumentación virtual.
4. Desarrollar una aplicación distribuida con instrumentación virtual para el intercambio de datos cifrados y descifrados.

1.6 Alcance

El trabajo está limitado a la implementación del cifrado y descifrado de datos para que estos datos viajen a través de un canal de datos de un sistema distribuido con instrumentación virtual.

1.7 Contribuciones del trabajo

Los resultados obtenidos de este proyecto se enumeran como sigue:

- Se diseñó y programó un sistema de adquisición de datos y control Supervisorio (SCADA), que integra una aplicación cliente para adquirir datos cifrados desde un servidor TCP en *LabVIEW*.
- Se programó el algoritmo de cifrado y descifrado de datos, para intercambio de información entre aplicaciones distribuidas de instrumentación virtual.
- Se desarrollo una aplicación distribuida de instrumentación virtual que intercambie datos cifrados y que descifre los mismos.

1.8 Resumen de la metodología y desarrollo de la investigación

La metodología de la tesis comienza en la implementación del cifrado y descifrado de datos de uno de los algoritmos más trascendentes dentro de la criptografía, DES. Además del desarrollo del sistema de comunicaciones con un Servidor TCP en *LabVIEW* para verificar el envío y recepción de la información. Al concluir, se realiza la inserción de lo anterior en un sistema de adquisición de datos y monitoreo.





1.9 Estructuración de la tesis

- **1.-Introducción.**
- **2.-Estado del arte.** Se dan algunos conceptos básicos sobre criptografía, criptosistemas, sus elementos fundamentales y su clasificación. Se describen algunos algoritmos de cifrado de datos de clave privada y clave pública. Este capítulo se centra en el cifrado simétrico, resaltando la técnica de cifrado que más se usa, el DES (Data Encryption Standard). Además se incluye una descripción breve de las aplicaciones prácticas, de algunos cifrados de datos en distintos lenguajes de programación y las patentes de los algoritmos criptográficos. Se comentan algunas implementaciones importantes, sobre los sistemas distribuidos realizados con instrumentación virtual, así como algunas características sobre estos.
- **3. Marco Teórico.** Se selecciona el algoritmo a implementar. Además de los elementos de un sistema distribuido y un sistema SCADA. También se pone la descripción de la programación gráfica, la arquitectura de los protocolos TCP/IP de comunicación, y la programación visual. Por último, en este capítulo se da la propuesta de tesis.
- **4. Programación del criptosistema.** Aquí se expone la implementación del sistema de cifrado de datos y descifrado de datos, programado en el ambiente de desarrollo gráfico de instrumentación virtual.
- **5. Inserción del criptosistema en el módulo del sistema SCADA.** Se describe la inserción del criptosistema en el módulo del sistema de adquisición y monitoreo de datos. Es descrito el sistema SCADA.
- **6. Pruebas y Resultados.** Se muestran los resultados finales obtenidos del proyecto. Al criptosistema se le aplican pruebas con las claves débiles y semidébiles para obtener distintos cifrados de datos y se obtiene una tabla de mensajes cifrados con sus respectivos mensajes simples. Para finalizar con las pruebas de tiempo de ejecución del criptosistema.





2. ESTADO DEL ARTE

2.1 Introducción

No cabe duda de que la información se está convirtiendo en la mayor fuente de poder que ha conocido la humanidad, y que la Criptografía es una herramienta esencial para su control. Es necesario, que se conozcan sus ventajas e inconvenientes, sus peligros y sus mitos o leyendas.

La criptografía es una disciplina con multitud de aplicaciones, muchas de las cuales están en uso hoy en día. Entre las más importantes están la seguridad de las comunicaciones que refiere principalmente a la aplicación de la criptografía a las redes de computadoras, que debido a la potencia de cálculo que se tiene hoy en día y empleando cifrado simétrico (utiliza sólo una clave) se consigue privacidad sin perder velocidad en la transferencia.

Además se observó la falta de módulos de cifrado de datos en ambientes de programación gráficos, entonces se comienza este trabajo con una sencilla implementación de uno de algoritmos más difundidos, DES y así observar que tan útil puede ser dentro de los sistemas distribuidos.

Como ejemplos del uso de las técnicas criptográficas, se encuentran en la historia comenzando con una de las más conocidas, Julio César empleaba una sencilla técnica para evitar que sus comunicaciones militares fueran interceptadas. Leonardo Da Vinci escribía las anotaciones sobre sus trabajos de derecha a izquierda y con la mano zurda. Otros personajes, como Sir Francis Bacon o Edgar Allan Poe eran conocidos por su afición a los códigos criptográficos, que en muchas ocasiones constituían un apasionante divertimento y un reto para el ingenio.

En este capítulo haremos un breve repaso de los mecanismos criptográficos considerados clásicos. Podemos llamar así a todos los sistemas de cifrado anteriores al nacimiento de las computadoras. Estas técnicas tienen en común que pueden ser empleadas usando simplemente lápiz y papel, y que pueden ser criptoanalizadas casi de la misma forma. De hecho, con la ayuda de las computadoras, los mensajes cifrados mediante el uso de estos códigos son fácilmente descifrables, por lo que cayeron rápidamente en desuso.





2.2 Criptografía

En sus inicios, la criptografía era entendida como un arte, porque requería de habilidades especiales que no todas las personas poseían, por lo que aquellas personas que pudiesen hacer uso de esta, eran consideradas personas elegidas. Por supuesto que la criptografía desde hace tiempo dejó de ser un arte, para convertirse en una técnica o conglomerado de técnicas matemáticas para crear esquemas de cifrado y descifrado de mensajes.

2.2.1 Conceptos básicos.

La Criptología (del griego criptos = oculto y logos = tratado, ciencia) designa a dos disciplinas complementarias que son la criptografía y el criptoanálisis, siendo la primera un conjunto de técnicas matemáticas para crear esquemas de cifrado y descifrado de mensajes, mientras que el criptoanálisis es el estudio de las técnicas matemáticas para intentar romper esquemas criptográficos.

El desarrollo de las comunicaciones electrónicas, unido al uso masivo y generalizado de datos en las computadoras, hace posible la transmisión y almacenamiento de grandes cantidades de información confidencial que es necesario proteger; entonces la criptografía es requerida.

La parte fundamental de la criptografía son los criptosistemas. Un sistema criptográfico consta esencialmente de dos partes:

- ✓ Cifrado de mensajes.
- ✓ Descifrado de mensajes.

Los mensajes originales (por ejemplo, $m = \text{HOLA MUNDO}$), a los que llamaremos mensajes simples, se construyen con caracteres del alfabeto usual $A = \{A, B, C, \dots, Z\}$ y M será el conjunto de todos los mensajes simples.

$$M = \{ m \mid m \text{ es un mensaje simple} \}$$

Al conjunto de los mensajes cifrados (por ejemplo $c = \text{ODNUM ALOH}$) lo llamaremos C :

$$C = \{ c \mid c \text{ es un mensaje cifrado} \}$$

Además de los conjuntos A , M y C , se requiere de una transformación para cifrar (E) y de una transformación para descifrar (D).

A continuación se ilustrarán los conceptos y definiciones anteriores de la criptografía por medio de ejemplos sencillos:





Ejemplo 1. Iniciemos con un mensaje simple $m = \text{HOLA MUNDO}$. A este mensaje le aplicaremos una transformación de cifrado (E) muy simple, que consiste en poner la última letra al principio, la penúltima en segundo lugar, y así hasta llegar a que la primera letra de nuestro mensaje simple ocupe la última posición en el mensaje cifrado:

$$E(\text{HOLA}) = \text{ALOH}$$

Donde E es una función con dominio M y codominio C; $E: M \rightarrow C$

Para descifrar el mensaje cifrado c, sólo es necesario aplicar la transformación de descifrado (D), la cual es una función con dominio C y codominio M; $D: C \rightarrow M$. En este caso, la transformación de descifrado D es similar a la transformación de cifrado E (colocar las letras en orden inverso):

$$D(\text{ALOH}) = \text{HOLA}$$

Entonces se tiene que la transformación de descifrado D es la inversa de la transformación de cifrado E, es decir: $D(E(m)) = m$

■

Ejemplo 2. Tenemos un mensaje simple $m = \text{HOLA MUNDO PELIGROSO}$ que necesitamos cifrar y enviar a un receptor. Utilizaremos la clave siguiente: A=1, E=2, I=3, O=4 y U=5; ahora se aplica la transformación de cifrado que consiste en sustituir las vocales por números de acuerdo con la clave, de forma tal que la “O” es sustituida por un “4”, así consecutivamente con cada una de las vocales y al final tenemos en mensaje cifrado:

$$c = \text{H4L1 M5ND4 P2L3GR4S4}$$

Para descifrar este mensaje debemos realizar la operación inversa, lo que significa sustituir los números por vocales según la clave.

De acuerdo con lo anterior, sabemos que el alfabeto para M (A_M) es diferente del alfabeto para C (A_C), por lo que:

A_M es el alfabeto usual

$$A_C = \{ x \mid x \text{ es una consonante} \} \cup \{ 1, 2, 3, 4, 5 \}$$

Se necesita una clave para cifrar y otra para descifrar (en este ejemplo es la misma clave)

Si llamamos k a la clave, E y D se convierten en E_k y D_k , respectivamente (su efecto depende también de k).

■





Ejemplo 3. La clave usada en el Ejemplo 2 podemos escribirla de la siguiente manera: 12345. Realicemos el mismo Ejemplo 2, pero ahora con una nueva clave $k = 23514$, de forma tal que la A se sustituye por 2, la O es sustituida por 1, y así sucesivamente. El mensaje simple es el mismo

$$m = \text{HOLA MUNDO PELIGROSO}$$

La transformación de cifrado consiste en sustituir las vocales por números de acuerdo con la nueva clave k .

$$c = \text{H1L2 M4ND1 P3L5GR1S1}$$

Para descifrar lo hacemos por medio de la función de transformación que consiste en sustituir los números por vocales de acuerdo con la nueva clave k .

Hemos visto que en el ejemplo.1 se ilustra el uso de permutaciones (transposiciones), y en los ejemplos.2 y 3, el uso de sustituciones.

Mientras que en el ejemplo 1 el mismo alfabeto sirve para M y para C, en los ejemplos 2 y 3 se requieren alfabetos diferentes (A_M y A_C), y se requiere además el uso de clave; estos ejemplos también ilustran el concepto de criptografía simétrica, porque se usa la misma clave en E_k y en D_k (el uso de claves diferentes se ilustrará en el ejemplo. 4).

Ejemplo 4. Ahora utilizaremos dos claves: $k_e = 12345$ para cifrar y $k_d = 5464968494$ la clave para descifrar.

El mensaje simple es:

$$m = \text{HOLA MUNDO PELIGROSO}$$

La transformación de cifrado consiste en sustituir las vocales por números de acuerdo con la clave k_e , como se realizó en el ejemplo 2, obteniendo el siguiente cifrado:

$$c = E_{k_e}(m) = \text{H4L1 M5ND4 P2L3GR4S4}$$

El emisor envía el valor de c y el de la clave k_d .

El receptor realiza la siguiente operación para obtener la clave K_e :

$$\begin{aligned} 5-4 &= \underline{1} \\ 6-4 &= \underline{2} \\ 9-6 &= \underline{3} \\ 8-4 &= \underline{4} \\ 9-4 &= \underline{5} \end{aligned}$$





Así, el receptor calcula la clave $k_e=12345$ y con ella puede realizar la transformación de descifrado, obteniendo el mensaje simple.

$$Dk_d(c) = \text{HOLA MUNDO PELIGROSO}$$

Este ejemplo ilustra la criptografía asimétrica.

2.3 Criptosistemas.

Los criptosistemas son la columna vertebral de la criptografía.

2.3.1 Elementos fundamentales y clasificación.

El esquema fundamental de un criptosistema (Fig. 1.) simbólicamente se presenta por medio de un sistema de comunicación.

- ❖ El emisor pretende enviar información al receptor.
- ❖ La fuente de información contiene el mensaje simple.
- ❖ El mensaje simple es cifrado.
- ❖ El transmisor transforma el mensaje en una señal que es enviada por el canal de comunicación al receptor.
- ❖ Posiblemente hay un intruso que penetra al canal de comunicaciones e intenta romper el mensaje cifrado para obtener el mensaje simple, información que el emisor y el receptor pretenden mantener secreta.
- ❖ El receptor hace las veces de un transmisor invertido que cambia la señal transmitida en un mensaje y pasa este mensaje a su destinatario.
- ❖ El receptor descifra el mensaje cifrado para obtener el mensaje simple.

El emisor envía $m \in M$ siendo: $M = \{ m \mid m \text{ es un mensaje simple } \}$.

En el proceso de cifrado del mensaje simple es utilizada la clave k la cual es un elemento del conjunto K , definido como sigue:

$$K = \{ k \mid k \text{ es una clave } \}$$

La clave para cifrar que se representa por $k_e \in K$ y E_{k_e} es la función para cifrar; esta función se aplica a $m \in M$ y se obtiene $c \in C$. Es decir, $E_{k_e}(m) = c$ donde c pertenece al conjunto:

$$C = \{ c \mid c \text{ es un mensaje cifrado } \}$$





En este momento c viaja a través del canal, el intruso intenta descifrar c (romper el criptosistema), c llega al receptor, el cual utilizará la clave $k_d \in K$ para descifrar, aplica D_{k_d} que es la función para descifrar $c \in C$ y se obtiene $m \in M$.

Finalmente $D_{k_d}(c) = m$.

Con este procedimiento el receptor recibe el mensaje m .

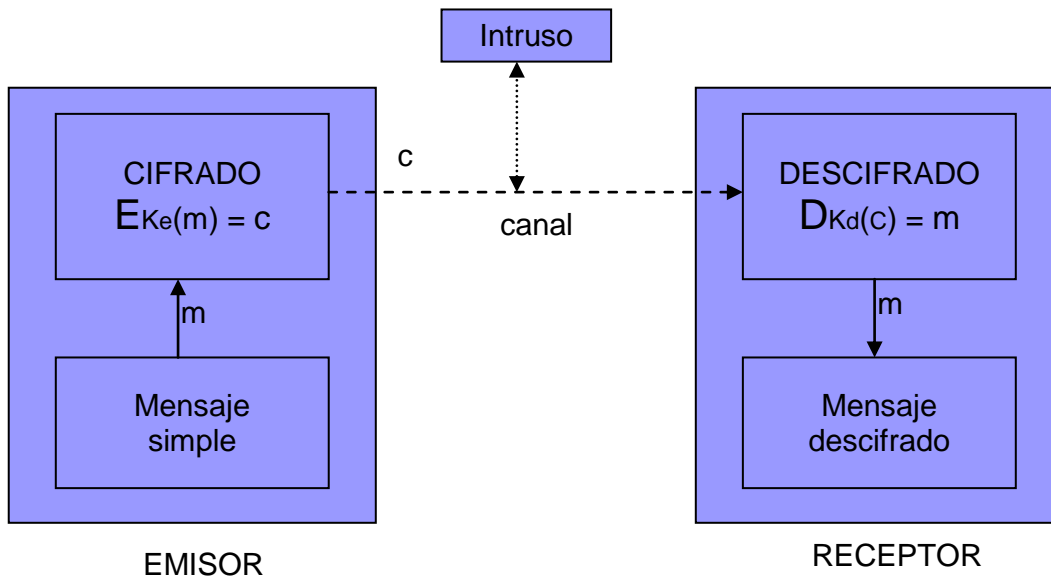


Fig. 1. Esquema fundamental de un criptosistema.

En criptografía simétrica las claves tienen el mismo valor, tanto para cifrar como para descifrar:

$$k_e = k_d$$

Dado A_M , los $m \in M$ son cadenas que se forman con los elementos de A_M .

Dado A_C , no cualquier cadena formada con sus elementos es un elemento de C ; para que una de estas cadenas sea un $c \in C$ válido, es preciso que se obtenga cifrando un $m \in M$ con una función E_{k_e} .

La manera como se construye C , permite asegurar que la función $E_{k_e} : M \rightarrow C$ es suprayectiva.





Por otro lado, la función $E_{k_e} : M \rightarrow C$ debe ser inyectiva, para asegurar que dos mensajes cifrados diferentes correspondan a dos mensajes simples diferentes.

Esto es, la función $E_{k_e} : M \rightarrow C$ es biyectiva, y eso significa que tiene inversa.

La inversa es precisamente $D_{k_d} : C \rightarrow M$, es decir

$$D_{k_d} = (E_{k_e})^{-1}$$

Para construir un criptosistema se requiere seleccionar un espacio de mensajes simples M , un espacio de mensajes cifrados C , un espacio de claves K , un conjunto de funciones de cifrado $\{E_{k_e} \mid k_e \in K\}$ y un conjunto correspondiente de funciones de descifrado $\{D_{k_d} \mid k_d \in K\}$; este último par de conjuntos debe tener la propiedad de que para cada clave de cifrado $k_e \in K$ existe una única clave de descifrado $k_d \in K$, tales que

$$D_{k_d} = (E_{k_e})^{-1}$$

Esto significa que $D_{k_d}(E_{k_e}(m)) = m, \forall m \in M$.

Las claves k_e y k_d forman un par de claves (k_e, k_d) .

Una premisa fundamental en criptología, es el hecho de que, dado un criptosistema, los conjuntos $M, C, K, \{E_{k_e} \mid k_e \in K\}$ y $\{D_{k_d} \mid k_d \in K\}$ son del dominio público. Criptógrafos y criptoanalistas sólo deben ocuparse del par de claves (k_e, k_d) , único elemento de un criptosistema que se conserva en secreto.

Los tipos fundamentales de criptosistema son dos:

1. Simétricos: criptosistemas donde se cumple que $k_e = k_d$
2. Asimétricos: criptosistemas que emplean un par de claves (k_e, k_d) , tales que $k_e \neq k_d$.

Además existen en criptografía dos técnicas básicas del diseño de los criptosistemas, son confusión y difusión. Estas se deben a Claude Elmwood Shannon.

La confusión busca esconder la relación entre el mensaje original y el mensaje cifrado. La forma más sencilla de obtener confusión es mediante sustituciones. En el caso de cifrado por bloques, se sustituye una secuencia larga de caracteres (un bloque), por otra a base de manipular sub-bloques del mensaje como unidades completas.

La difusión busca disipar la información, buscando por el efecto de disimular las redundancias del texto claro al extenderlas por todo el texto cifrado, la manera más sencilla





de obtener difusión es mediante permutaciones o transposiciones, se describen a continuación.

2.4 Criptosistemas de clave privada

Los métodos simétricos son propios de la criptografía clásica o criptografía de clave privada, mientras que los métodos asimétricos corresponden a la criptografía de clave pública, introducida en la Universidad de Stanford por Diffie y Hellman, en 1975.

Un esquema fundamental de un criptosistema simétrico (Fig.2.) simbólicamente se presenta por medio de un sistema de comunicación.

El emisor tiene $m \in M$ y una clave $k_e \in K$, a las cuales les aplica una función de transformación E_{k_e} para obtener $c \in C$. El emisor envía c a través del canal, mientras que el intruso intenta romper el criptosistema para obtener el mensaje simple m . Finalmente, el receptor aplica la función de descifrado $D_{k_e}(c)$, con la misma clave k_e para obtener el mensaje simple m .

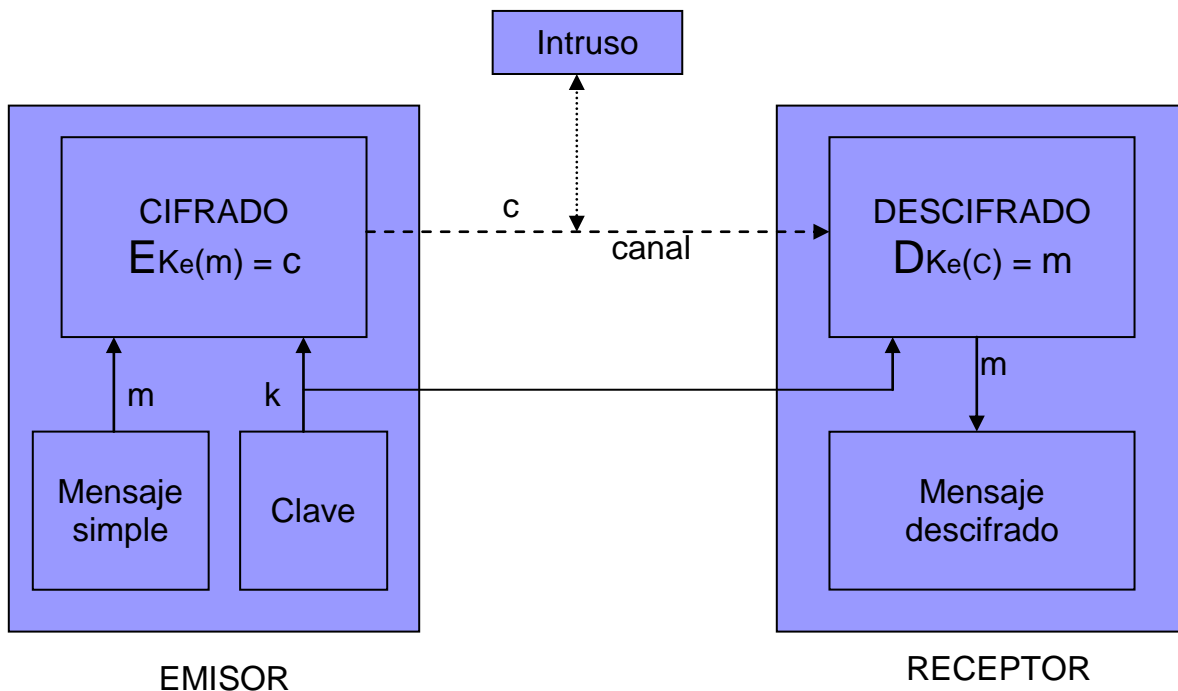


Fig. 2. Esquema de un criptosistema simétrico.





Por otra parte, los sistemas criptográficos simétricos pueden tener dos formas de funcionamiento:

- Cifrado en flujo, bit a bit o byte a byte (stream cipher): el algoritmo de cifrado se aplica a un elemento de información (carácter, bit) mediante un flujo que constituye la clave y que en teoría es aleatorio y de un tamaño superior al del mensaje. Para generar el flujo que constituye la clave, se emplea un generador de secuencias pseudoaleatorias. En este tipo de algoritmos sólo se realizan sustituciones, mediante una operación XOR entre cada bit de información y cada bit de la secuencia que forma la clave[24].
- Cifrado en bloque o poligráfico (block cipher): el mismo algoritmo de cifrado se aplica a un bloque de información (grupo de caracteres o número de bytes) repetidas veces, usando la misma clave. De este modo, es posible combinar varias sustituciones y transposiciones.

En un *criptosistema simétrico por bloques* m se parte en cadenas de longitud t , las cuales se llaman bloques. Se cifra un *bloque* a la vez.

Hay dos clases principales de criptosistemas simétricos por bloques:

- ✓ Transposición (usan permutaciones).
- ✓ Sustitución.

2.4.1 Transposición:

Un mecanismo simple de transposición podría consistir en colocar el mensaje simple conociendo que t es igual al tamaño del largo del bloque en una tabla de n columnas, y dar como mensaje cifrado los símbolos de una columna ordenados de arriba hacia abajo con los de otra.

La clave $k_e = \{1, 2, \dots, t\}$ para cada $k_e \in K$ se compone del número n junto con el orden en el que se deben leer las columnas.

Ejemplo 5. Queremos cifrar el siguiente mensaje simple:

$m =$ LO QUE TE DIGA TRES VECES ES CIERTO

Colocamos el texto en una Tabla 1, ponemos un orden en el envío con el número de columnas $n=5$ y la permutación de las columnas 42513, para obtener:

Tabla 1. Transposición de elementos.

1	2	3	4	5
L	O		Q	U





E		T	E	
D	I	G	A	
T	R	E	S	
V	E	C	E	S
	E	S		C
I	E	R	T	O

Realizando la concatenación de las columnas y respetando los espacios en blanco tenemos el siguiente cifrado:

$$c = \text{QEASE TO IREEEU SCOLEDTV ITGECSR}$$

La transformación de cifrado es:

$$E_{k_e}(m) = (m_{k_e(1)} m_{k_e(2)} \dots m_{k_e(t)})$$

Donde : $m = (m_1 m_2 \dots m_t) \in M$

La transformación de descifrado es:

$$D_{k_d}(c) = (c_{k_d(1)} c_{k_d(2)} \dots c_{k_d(t)})$$

Donde: $c = (c_1 c_2 \dots c_t) \in C$.

El cifrado por transposición se puede criptoanalizar efectuando un estudio estadístico sobre la frecuencia de aparición de pares y tripletas de símbolos, en el lenguaje en que esté escrito el mensaje simple.

2.4.2 Cifrados monoalfabéticos

En este tipo de cifrado cada símbolo del mensaje simple se transforma en otro símbolo, de modo que a un carácter dado siempre le corresponde el mismo carácter, de acuerdo con la clave de cifrado.

Ejemplo 6. (Método de Julio César). Con un alfabeto de 26 letras, se suma 3 al número de orden de cada letra, para obtener el orden de la letra que debe sustituirla.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Entonces al tener un mensaje:

$$m = \text{HOLA MUNDO MAYA}$$





El mensaje cifrado con el método de Julio César queda así:

$$c = KROD PXQGR PDBD$$

■

Si A es el alfabeto y M el conjunto de mensajes simples, la función $E: A \rightarrow A$ debe ser inyectiva, suprayectiva y diferente a la identidad. Al mensaje m formado por la concatenación de t elementos de A se le aplica la transformación E de la siguiente manera:

$$Ek_e(m) = (Ek_e(m_1) Ek_e(m_2) \dots Ek_e(m_t)) = (c_1 c_2 \dots c_t) = c$$

Donde: $m = (m_1 m_2 \dots m_t) \in M$

La transformación de descifrado es:

$$Dk_d(c) = (k_d(c_1) k_d(c_2) \dots k_d(c_t)) = (m_1 m_2 \dots m_t) = m$$

Donde: $c = (c_1 c_2 \dots c_t) \in C$

2.4.3 Cifrados polialfabéticos.

Es la aplicación cíclica de n cifrados monoalfabéticos. Entonces es la sustitución aplicada a cada carácter que varía en función de la posición que ocupe esté dentro del mensaje simple. Un ejemplo típico de cifrado polialfabético es el Cifrado de Vigènere. Lo inventó Blaise de Vigenère hacia 1550, y fue inquebrantable durante 300 años.

Un simple Cifrado de Vigènere de periodo t , sobre un alfabeto de s - caracteres, involucra un número t de permutaciones a utilizarse con desplazamientos: $k_0, k_1 \dots k_{t-1}$.

El mapeo del mensaje simple $m = m_0 m_1 m_2 \dots$ al mensaje cifrado $c = c_0 c_1 c_2$ es definido en caracteres individuales por $c_i = (m_i + k_{i \bmod t}) \bmod s$

Ejemplo 7. Tenemos la clave $k = (p_1, p_2, p_3) = (3, 7, 10)$ en este caso el valor de $t=3$. Si el mensaje simple es

$$m = \text{HOLA MUNDO}$$

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
 H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
 K L M N O P Q R S T U V W X Y Z A B C D E F G H I J





La sustitución para obtener el mensaje cifrado es:

$$c = KVV \text{ DTE QKY}$$



2.4.4 Cifrados por sustitución homofónica

Frente a ataques basados en el estudio de las frecuencias de aparición de los símbolos, existe una familia de algoritmos que trata de ocultar las propiedades estadísticas del mensaje simple empleando un alfabeto de salida (A_C) con más símbolos que el alfabeto de entrada (A_M).

Supongamos que nuestro alfabeto de entrada posee cuatro letras, (a, b, c, s). Supongamos además que en nuestros mensajes la letra *a* aparece con una probabilidad 0.4, y el resto con probabilidad 0.2. Podríamos emplear el siguiente alfabeto de salida $\{\alpha, \beta, \gamma, \delta, \varepsilon\}$ efectuando la siguiente asociación:

$$\begin{aligned} A_M &= \{a, b, c, s\} \\ A_C &= \{\alpha, \beta, \gamma, \delta, \varepsilon\} \end{aligned}$$

La probabilidad de cada elemento es: $P(a)=0.4$ y $P(b,c,s)=0.2$

La transformación de cifrado es:

$$\begin{aligned} E(a) &= \begin{cases} \alpha \text{ con probabilidad } 1/2 \\ \beta \text{ con probabilidad } 1/2 \end{cases} \\ E(b) &= \gamma \\ E(c) &= \delta \\ E(s) &= \varepsilon \end{aligned}$$

Ejemplo 8. Si $m = \text{casa}$, el mensaje cifrado con el método anterior es:

$$c = \delta\alpha\varepsilon\beta$$



2.4.5 Redes de Feistel

Los cifrados de producto tienen en común que dividen un bloque de longitud n en dos mitades, L y R . Se define entonces un cifrado de producto iterativo en el que la salida de cada ronda se usa como entrada para la siguiente (ver Fig. 3):



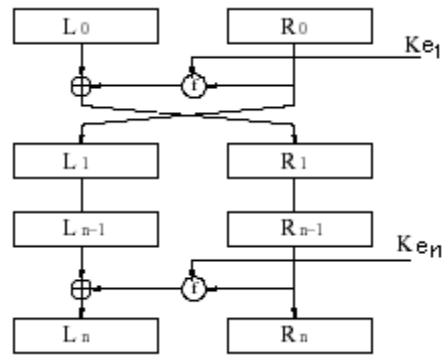


Fig. 3. Estructura de las redes de Feistel.

La estructura de las redes de Feistel es empleada en multitud de algoritmos, como CAST, DES, Lucifer, Blowfish, entre otros.

La transformación de cifrado es la siguiente

$$\left. \begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_{e_i}) \end{aligned} \right\} \text{ si } i < n$$

$$\begin{aligned} L_n &= L_{n-1} \oplus f(R_{n-1}, K_{e_n}) \\ R_n &= R_{n-1} \end{aligned}$$

En donde (L_n, R_n) son los dos bloques a los que se les aplica la función de cifrado. Para una red de Feistel de n rondas $E_{k_e}(L_n, R_n)$ y $D_{k_d}(L_n, R_n)$. El símbolo \oplus representa a la operación XOR

La función de descifrado es análoga: se aplica el algoritmo en orden inverso.

Ejemplo 9.

En la Fig. 4. se tienen los siguientes valores como muestra, para observar el funcionamiento del algoritmo de redes de Feistel:

Clave = 1101011010110101

Mensaje simple = HOLA

Mensaje simple en hexadecimal = 73 7A 77 6C

Mensaje simple en binario = 01110011011110100 **111011101101100**

$R_0 = 0111011101101100$

$K_1 = 1101011010110101$





$$R_0 \text{ XOR } K_1 = 1010000111011001$$

$$L_0 = 0111001101111010$$

$$R_1 = 1101001010100011$$

$$K_2 = 1010110101101011 \quad \text{XOR}$$

$$0111111111001000$$

$$L_1 = 0111011101101100 \quad \text{XOR}$$

$$R_2 = 0000100010100100$$

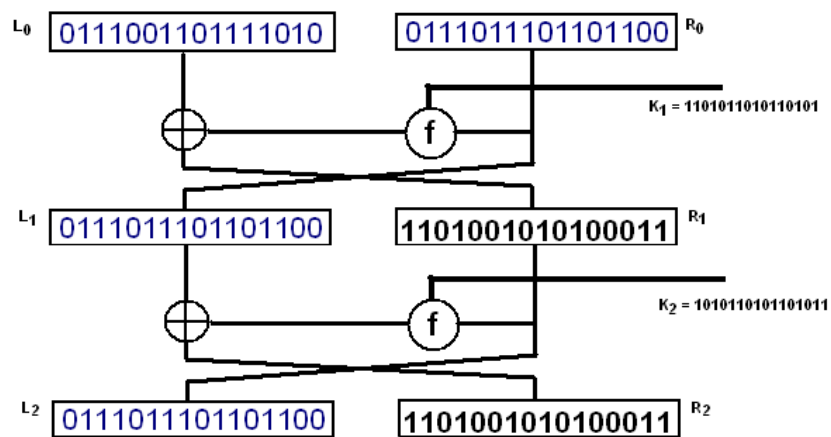


Fig. 4. Ejemplo de redes de Feistel.

2.4.6 S-Cajas

Las tablas pequeñas de sustitución se denominan de forma S-Cajas, consisten en tener de entrada cadenas de m bits y da como salida cadenas de n bits. El valor de m y n pueden ser iguales (Fig. 5).

Este tipo de sustitución que es muy sencilla de utilizar: se divide el bloque original en trozos de m bits y cada uno de ellos se sustituye por otro de n bits, haciendo uso de la S-Caja correspondiente. Existen algoritmos criptográficos como DES que utiliza ocho S-Cajas de 6×4 bits y otro es CAST que divide bloques de 64 bits para emplear S-Cajas de 8×32 bits[17].



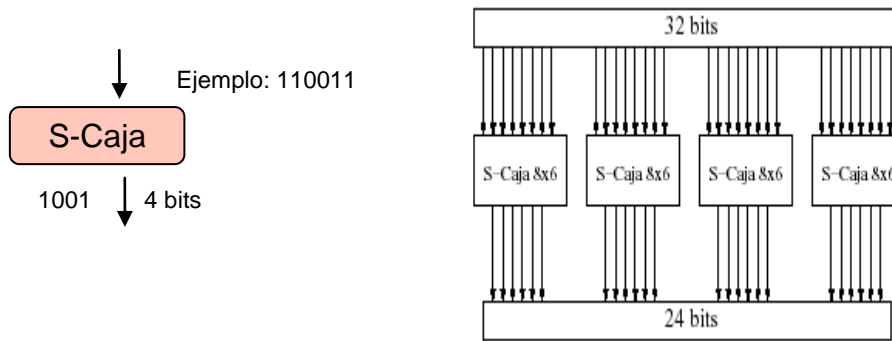


Fig. 5. Esquema del funcionamiento de S-Cajas.

Ejemplo 10.

A continuación se muestra la Tabla 2 que corresponde a un pequeño segmento de la tabla original con la finalidad de dar el ejemplo de S-Cajas del DES.

1. Se toman los bits 1° y 6° de $B(j)$ y se forma un número de dos bits que llamaremos m . Este valor nos indicará la fila en tabla de sustitución correspondiente $S(j)$. Obsérvese que $m=0$ representa la 1ª fila y $m=3$ la última.
2. Con los bits 2° al 5° de $B(j)$ formar otro número, n , de cuatro bits que indicará la columna de $S(j)$ en la que se buscar el valor de sustitución. En esta ocasión $n=0$ representa la 1ª columna y $n=15$ la última columna.
3. Reemplazar $B(j)$ con $S(j)(m,n)$, m fila y n columna.

$B=48$ bits, $B(j) = 6$ bits $1 \leq j \leq 8$

Sea $B(3)=42$, en binario $B(3)=101010$.

Se buscará el nuevo valor de $B(3)$ en $S(3)$. Fila m y columna n , según lo expuesto anteriormente.

$m=10$, $n=0101$, y en decimal $m=2$ y $n=5$.

Por tanto, $B(3)$ será $S(3)(2,5)=15$. [17]





Tabla 2. Ejemplo de S-Cajas del DES.

↓

Fila	Columna																S-Caja
	0	1	2	3	4	<u>5</u>	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	
<u>2</u>	13	6	4	9	8	<u>15</u>	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	

→

2.4.7 CAST

CAST cifra bloques de 64 bits empleando claves de 64 bits, consta de ocho rondas y deposita prácticamente toda su fuerza en las S-Cajas. De hecho, existen muchas variedades de CAST, cada una con sus S-Cajas correspondientes —algunas de ellas secretas—. Este algoritmo se ha demostrado resistente a las técnicas habituales de criptoanálisis, y sólo se conoce la fuerza bruta como mecanismo para atacarlo [31].

2.4.8 Blowfish y Twofish

El algoritmo Blowfish ha sido desarrollado o diseñado por Bruce Schneier en el año 1993. Fue ideado con la intención de reemplazar al algoritmo DES, proponiéndolo como una alternativa sin problemas de patentamiento y libre de uso o de dominio libre. Se trata también, por supuesto, de un algoritmo criptográfico de clave simétrica y cifrado por bloques.

El algoritmo consta de dos partes: la expansión de la clave y el cifrado de la información de entrada (que se procesará en bloques de 64 bits de longitud). La primera de estas partes se refiere a la conversión de la clave (de hasta 448 bits de longitud) en varios arreglos de sub-





claves. La segunda parte, respecto del cifrado, consiste en una función que se aplica 16 veces. Cada una de estas 16 iteraciones, o rondas, consistirá de una permutación dependiente de la clave y de una sustitución dependiente, tanto de la clave como de la información de entrada. Casi todas las operaciones son adiciones y operaciones XOR en variables de 32 bits de longitud.

Veamos en símbolos, de manera resumida, cómo trabajaría el algoritmo para el cifrado de datos, teniendo en cuenta que se trata de una red Feistel consistente en 16 rondas. La entrada x es un bloque de 64 bits de longitud. El arreglo P contiene a las 18 sub-claves de 32 bits. El símbolo \oplus representa a la operación XOR.

Se divide x en dos mitades de 32-bits: X_L, X_R

Iterando, con $i=1$ hasta 16:

$$X_L = X_L \oplus P_i$$

$$X_R = F(X_L) \oplus X_R$$

Se intercambian X_L y X_R

Se intercambian X_L y X_R

$$X_R = X_R \oplus P_{17}$$

$$X_L = X_L \oplus P_{18}$$

Se combinan X_L y X_R

La función $F()$ se especifica del siguiente modo, dividiendo en primera instancia a X_L en cuatro cuartos de 8 bits cada uno: a, b, c , y d . Se define que:

$$F(X_L) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \oplus S_{3,c}) + S_{4,d} \bmod 2^{32}$$

Teniendo en cuenta que S representa a las S -cajas o Cajas de Sustitución, de las cuales se vale el algoritmo; en este caso, cuatro de 32 bits teniendo 256 items cada una:

$$S_{1,0} \ S_{1,1}, \dots, \ S_{1,255}$$

$$S_{2,0} \ S_{2,1}, \dots, \ S_{2,255}$$

$$S_{3,0} \ S_{3,1}, \dots, \ S_{3,255}$$

$$S_{4,0} \ S_{4,1}, \dots, \ S_{4,255}$$

Comentaremos brevemente acerca del algoritmo Twofish, que fue creado por el mismo autor de Blowfish y, si bien está relacionado con este último, es significativamente más complejo. El algoritmo vio la luz en el año 1998 y se trata también de un cifrado por bloques (ahora de 128 bits de longitud) que acepta una clave de longitud variable de hasta 256 bits.

El algoritmo fue finalista entre los algoritmos propuestos para el AES en el *Advanced Encryption Standard Contest*, el concurso en que se estableció ganador el algoritmo Rijndael.





Las características distintivas del algoritmo podrían resumirse en que las S-cajas utilizadas serán dependientes de la clave y un manejo de esta última relativamente complejo (una parte afectará al algoritmo de cifrado).

Notaremos de manera breve que, técnicamente, el algoritmo consta de una red Feistel, realiza 16 rondas, aplicando una función biyectiva basada en cuatro S-Cajas o cajas de sustitución dependientes de la clave[32].

2.4.9 El algoritmo IDEA

El algoritmo IDEA (International Data Encryption Algorithm) data de 1992. Trabaja con bloques de 64 bits de longitud y emplea una clave de 128 bits. Como en el caso de DES, se usa el mismo algoritmo tanto para cifrar como para descifrar.

El algoritmo IDEA consta de ocho rondas. Dividiremos el bloque M a codificar, de 64 bits, en cuatro partes m_1 , m_2 , m_3 y m_4 de 16 bits. Llamaremos Y_i a cada una de las 52 subclaves de 16 bits que vamos a necesitar.

Las operaciones que realizaremos en cada ronda son:

1. Multiplicar m_1 por Y_1 .
2. Sumar m_2 con Y_2 .
3. Sumar m_3 con Y_3 .
4. Multiplicar m_4 por Y_4 .
5. Realizar un XOR entre el resultado de 1 y 3.
6. Realizar un XOR entre el resultado del 2 y 4.
7. Multiplicar el resultado del paso 5 por Y_5 .
8. Sumar los resultados de los pasos 6 y 7.
9. Multiplicar el resultado del paso 8 por Y_6 .
10. Sumar los resultados de los pasos 7 y 9.
11. Realizar un XOR entre los resultados de los pasos 1 y 9.
12. Realizar un XOR entre los resultados de los pasos 3 y 9.
13. Realizar un XOR entre los resultados de los pasos 2 y 10.
14. Realizar un XOR entre los resultados de los pasos 4 y 10.

Al final tendremos de salida de cada ronda los cuatro sub-bloques generados en los pasos 11, 12, 13 y 14, que serán la entrada del siguiente ciclo, en el que utilizaremos las siguientes seis subclaves, hasta llegar a 48. Para finalizar cada ronda intercambiaremos los dos bloques centrales.

Al empezar la octava ronda, se realiza la siguiente operación:

1. Multiplicar m_1 por Y_{49} .





2. Sumar m_2 con Y_{50} .
3. Sumar m_3 con Y_{51} .
4. Multiplicar m_4 por Y_{52} .

Las ocho subclaves se calculan dividiendo la clave de entrada en bloques de 16 bits. Las ocho siguientes se obtienen rotando la clave de entrada 25 bits a la izquierda y volviendo a dividirla, y así consecutivamente.

IDEA es un algoritmo bastante seguro, no presenta claves débiles y su longitud de clave hace imposible un ataque por la fuerza bruta.[17]

2.4.10 El algoritmo AES

El algoritmo AES, siglas de *Advanced Encryption Standard* o Estándar para el Cifrado Avanzado, es, como su predecesor DES, un algoritmo de criptografía simétrica por bloques. Es ya un estándar del gobierno de los E.E.U.U., siendo su objetivo remplazar al algoritmo DES.

Fue anunciado por el Instituto Nacional de Estándares y Tecnología, o *National Institute of Standards and Technology* (NIST), en el año 2001. Es, desde 2002, un estándar del gobierno federal de los E.E.U.U., bajo un *Federal Information Processing Standard* (FIPS): Concretamente, como FIPS PUB 197.

Es también conocido como Rijndael, sobre la base de los apellidos de sus creadores, Joan Daemen y Vincent Rijmen, quienes presentaron el algoritmo al *Advanced Encryption Standard Contest* y fue seleccionado entre otros quince algoritmos finalistas (entre ellos Twofish, de Bruce Schneier).

Nótese que, ya que el estándar AES ha sido algo más restrictivo, Rijndael permite un mayor rango de tamaños posibles de bloques y de longitudes de claves. AES especifica un tamaño de bloque fijo de 128 bits de longitud; para la clave tamaños de 128, 192 ó 256 bits. En Rijndael, en tanto sean múltiplos de 32 bits, las longitudes de clave y bloque podrán variar entre 128 y 256 bits.

Con respecto a la especificación, o funcionamiento en sí del algoritmo, se destacará que Rijndael (o AES) utiliza lo que se conoce como campo de Galois (apellido de un matemático francés, precursor de esta teoría del álgebra abstracta) para llevar a cabo gran parte de sus operaciones matemáticas. Este campo es una construcción matemática especial, en donde las operaciones de adición, sustracción, multiplicación y división son redefinidas y donde se dispondría de un número limitado de enteros.





Más precisamente, en Rijndael, el campo Galois utilizado sólo permite un número de 8 bits (un número del 0 al 255) dentro de él. Todas las operaciones matemáticas definidas en este ámbito resultarán en un número de 8 bits.

Por ejemplo Rijndael, ambas, la suma y la resta representan operaciones XOR, no hay diferencia entre la adición y la sustracción. Para el caso de la multiplicación, las cosas ya no son tan simples. Veámoslo rápidamente: Teniendo dos números de 8 bits a y b , y un producto p , también de 8 bits, los pasos que se han de seguir implicarían:

$$p=0$$

Iterando, con $i=1$ hasta 8:

Si el bit menos significativo de b es 1 entonces:

$$p=p\oplus a$$

Se realiza una copia de resguardo de a , luego

$$a=a\ll 1$$

Si la copia de resguardo de a , en su bit más significativo, poseía un 1:

$$a=a\oplus 0x1b \text{ (constante expresada en hexadecimal)}$$

$$b=b\gg 1$$

Terminando el ciclo de ocho iteraciones, p contendrá el valor resultado del producto entre a y b .

El símbolo \oplus representa a la operación XOR, y \ll al desplazamiento a la izquierda, \gg a derecha, de bits, aplicado en el operando de la izquierda, en la cantidad que especifique el operando de la derecha.

Se comentó acerca de este proceso de multiplicación porque su inversa se utiliza en la generación de la S-Cajas o caja de sustitución del algoritmo. A grandes rasgos, el múltiplo inverso de un número de entrada se almacenará en dos variables temporales, sobre las cuales se realizarán operaciones de rotación, desplazamientos y XOR, para la obtención, finalmente, del valor transformado.[32]

2.4.11 Modos de operación del cifrado por bloques

Discutiremos algunos métodos para aplicar cifrados por bloques a mensajes de gran longitud

$$m = m_1 \parallel m_2 \parallel \dots \parallel m_N$$





- ✓ ECB (Electronic Codebook)
- ✓ CBC (Cipher Book Chaining Mode)
- ✓ CFB (Cipher Feedback Mode)

2.4.11.1 ECB (Electronic Codebook)

El modo más sencillo de aplicar un algoritmo de cifrado por bloques es el ECB (Electronic Codebook). Se subdivide la cadena que se quiere codificar en bloques del tamaño adecuado y se cifran todos ellos empleando la misma clave (ver Fig.6).

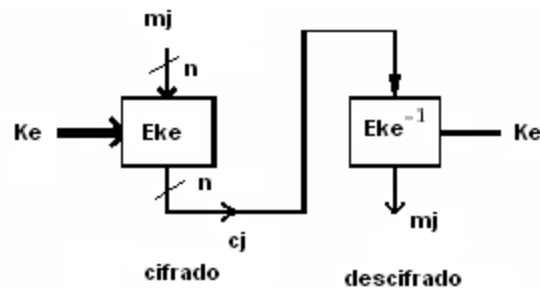


Fig. 6. Modo ECB (Electronic Codebook).

Es posible dentro de este método codificar en el orden que se desee. En caso de que uno de los bloques sufriera una alteración, el resto quedaría intacto, por lo que es resistente a errores. Ahora, si el mensaje presenta patrones repetitivos, el texto cifrado también los presentará, y eso no es conveniente, sobre todo cuando se codifican datos redundantes.

Algún intruso puede cambiar un bloque sin mayores problemas, y alterar los mensajes incluso desconociendo la clave y el algoritmo del cual procede.

2.4.11.2 CBC (Cipher Book Chaining Mode)

El modo CBC presenta un mecanismo de retroalimentación en el cifrado por bloques. El cifrado de bloques anteriores condiciona el cifrado del bloque actual, por lo que será imposible sustituir un bloque individual en el mensaje cifrado.

Ilustraremos este modo en la Fig.7 donde tenemos como entrada del criptosistema:

- ✓ k-bits en la clave k
- ✓ n-bit IV = vector de inicialización(bloque aleatorio)
- ✓ n-bit en bloque del mensaje simple m_1, \dots, m_t



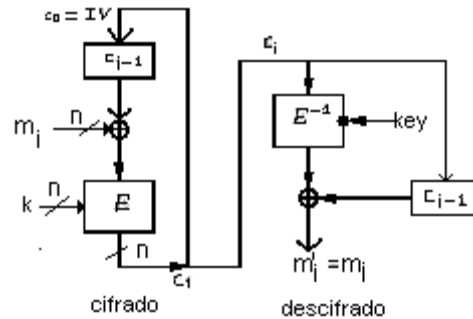


Fig. 7. Modo CBC (Cipher Block Chaining Mode).

Cifrado: $c_0 \leftarrow IV$. Para $1 \leq j \leq t$, $c_j \leftarrow Ek(c_{j-1} \oplus x_j)$.

Descifrado: $c_0 \leftarrow IV$. Para $1 \leq j \leq t$, $m_j \leftarrow c_{j-1} \oplus Ek^{-1}(c_j)$.

Esto se consigue efectuando una operación XOR entre el bloque del mensaje que queremos cifrar y el último criptosistema obtenido.

2.4.11.3 CFB (Cipher-Feedback Mode)

Al tener dos mensajes simples idénticos se codifican de la misma forma, pero para evitar que sean cifrados iguales, se utiliza un vector de inicialización (IV) que puede ser un bloque aleatorio; con esto se garantiza que siempre haya diferentes mensajes cifrados.

El modo de operación CFB (Cipher-Feedback Mode) permitirá codificar la información en unidades inferiores al tamaño del bloque, lo cual permite aprovechar totalmente la capacidad de transmisión del canal de comunicaciones, manteniendo además un nivel de seguridad adecuado.

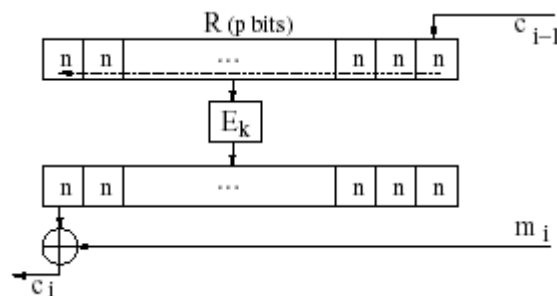


Fig. 8. Modo CFB (Cipher-Feedback Mode)





En la Fig. 8, p es el tamaño de bloque del algoritmo simétrico, y sea n el tamaño de los bloques que queremos transmitir (n ha de ser divisor de p). Sea m_i el i -ésimo bloque del mensaje simple, de tamaño n .

Empleamos entonces un registro de desplazamiento R de longitud p y lo cargamos con un vector de inicialización. Codificamos el registro R con el algoritmo simétrico y obtenemos en r sus n bits más a la izquierda. El bloque que deberemos enviar es $c_i = r \oplus m_i$. Desplazamos R n bits a la izquierda e introducimos c_i por la derecha.

Para descifrar basta con cargar el vector de inicialización en R y codificarlo, calculando r . Entonces $m_i = r \oplus c_i$. Desplazamos luego R e introducimos c_i por la derecha como hacemos en el algoritmo de cifrado.

2.4.12 El Algoritmo DES

DES cifra bloques de 64 bits, mediante permutación, sustitución y usando una clave de 64 bits, de los que 8 son de paridad generando 64 bits cifrados.

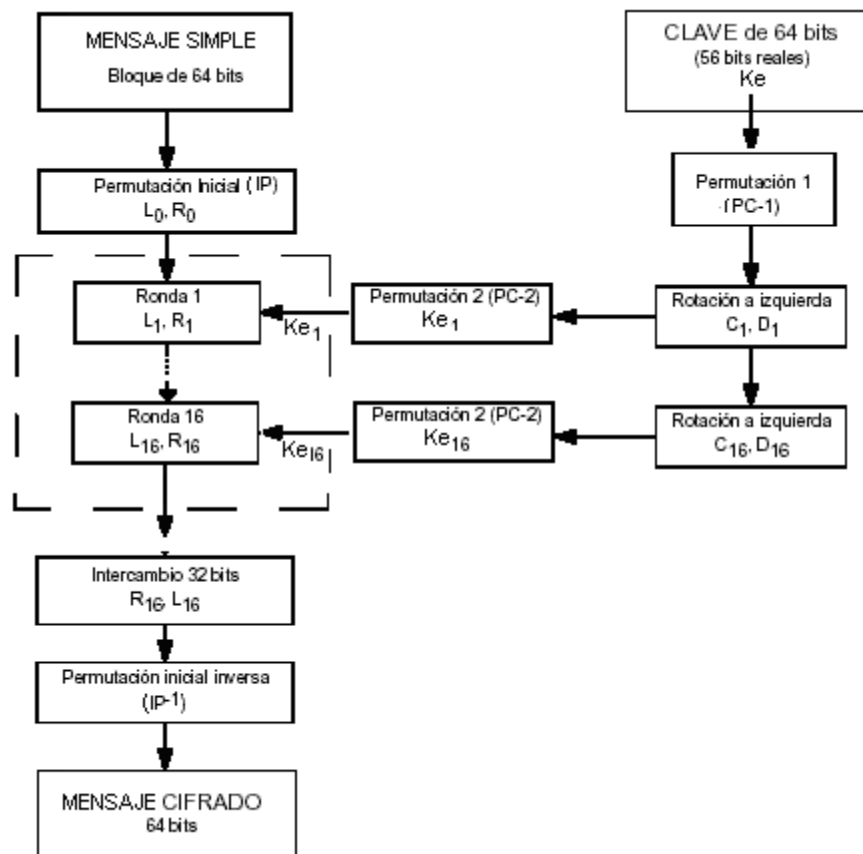


Fig. 9. Esquema general del algoritmo DES.





DES tiene 19 etapas diferentes. La primera etapa es una transposición, una permutación inicial (IP) mostrada en la Tabla 3 del mensaje simple de 64 bits, independientemente de la clave. La penúltima etapa consiste en realizar un intercambio de 32 bits y la última etapa es otra transposición (IP^{-1}) mostrada en la Tabla 9, exactamente la inversa de la primera (Fig.9).

Tabla 3. Permutación Inicial (IP)

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

Tabla 4. Permutación 1 (PC-1)

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

Tabla 5. Permutación 2 (PC-2)

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

En la penúltima etapa se intercambian los 32 bits de la izquierda y los 32 bits de la derecha. Las 16 etapas restantes se realizan con una Red de Feistel de 16 rondas (Fig. 3).



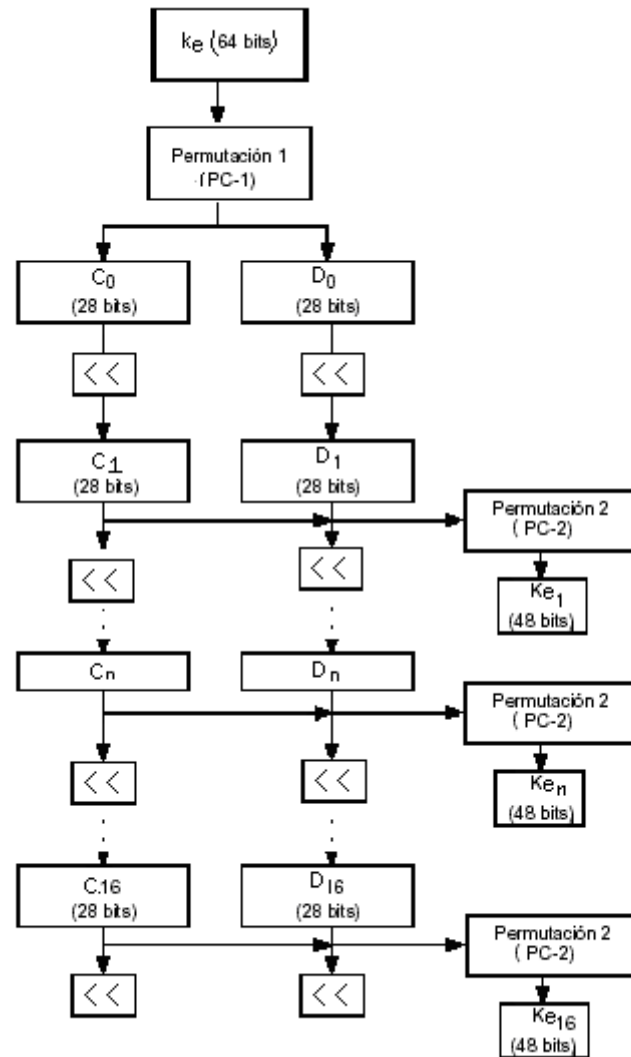


Fig. 10. Cálculo de las subclaves k_i .

En cada una de las 16 iteraciones se emplea un valor, K_i , obtenido a partir de la clave de 56 bits y distinto en cada iteración (Fig. 10).

Se realiza una permutación inicial (PC-1) en la Tabla 4 sobre la clave, y luego la clave obtenida se divide en dos mitades de 28 bits, cada una de las cuales se rota a la izquierda un número de bits determinado que no siempre es el mismo. K_i se deriva de la elección permutada (PC-2) en la Tabla 5 de 48 de los 56 bits de estas dos mitades rotadas.





Los desplazamientos de los bits son de acuerdo a la ronda que se esté realizando como se muestra en la Tabla 6:

Tabla 6. Desplazamientos de los bits en cada ronda.

Ronda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits despl	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

La función de la red de Feistel (Fig.11) se compone de una permutación de expansión (E) en la Tabla 7, que convierte el bloque correspondiente de 32 bits en uno de 48. Después realiza una OR-Exclusiva (El símbolo \oplus representa a la operación XOR) con el valor K_i , también de 48 bits, aplica ocho *S-Cajas* de 6×4 bits (Tabla 10). Para conocer mejor como funciona las *S-Cajas* de DES se explica con un ejemplo:

Se dividieron los 48 bits, se tienen bloques de 6 bits que son llamados $B(j)$ y el valor de j está en función del número del bloque comenzando con 1 hasta 8 y después:

1. Se toman los bits 1. y 6. de $B(j)$ y se forma un número de dos bits que es m . Este valor nos indicará la fila en la tabla de sustitución correspondiente $S(j)$. Obsérvese que $m=0$ representa la 1ª fila y $m=3$ la última de acuerdo a la Tabla 10.
2. Con los bits 2º al 5º de $B(j)$ formar otro número, n , de cuatro bits que indicará la columna de $S(j)$ en la que se busca el valor de sustitución. En esta ocasión $n=0$ representa la 1ª columna y $n=15$ la última columna.
3. Reemplazar $B(j)$ con $S(j)(m,n)$, m es la fila y n es la columna.

Entonces si se tiene $B(j)=110011$

$B=48$ bits, $B(j) = 6$ bits $1 \leq j \leq 8$

Sea $B(3)=42$, en binario $B(3)=101010$.

Se obtiene el nuevo valor de $B(3)$ en $S(3)$. Fila m y columna n , según lo expuesto anteriormente.

$m=10$, $n=0101$, y en decimal $m=2$ y $n=5$.

Por tanto, $B(3)$ será $S(3)(2,5)=15$

Al final se efectúa una nueva permutación (P) en la Tabla 8.



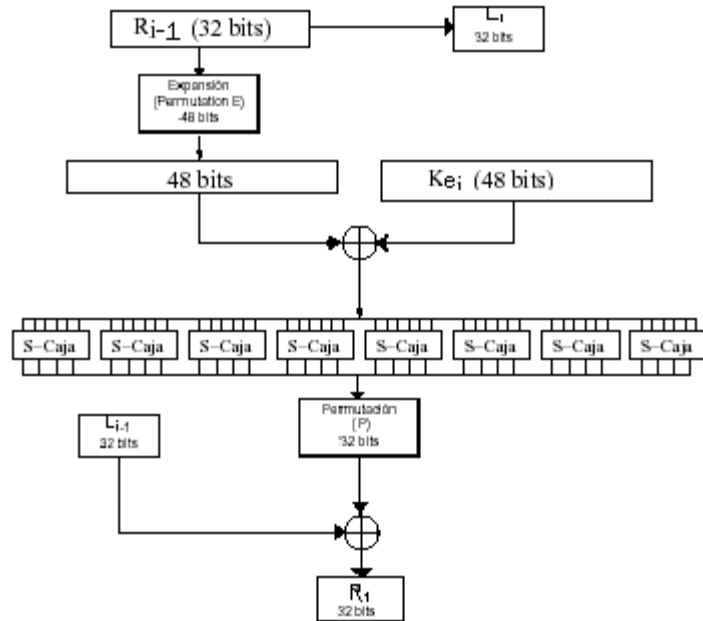


Fig. 11. Ronda de la red de Feistel del algoritmo DES.

Tabla 7. Expansión (E).

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

Tabla 8. Permutación (P)

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

Tabla 9. Permutación Final (IP-1)

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





Tabla 10. S-Cajas del algoritmo DES

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	





Para descifrar, basta con usar el mismo algoritmo empleando las K_i en orden inverso. Es decir, en la primera etapa se usa K_{16} , K_{15} en la segunda, y así hasta K_1 en la 16ª y última.

2.5 Algunas propiedades de DES

Los criptosistemas clásicos que en la actualidad han perdido gran parte de su eficacia, debido a que son fácilmente criptoanalizables empleando una computadora doméstica, bien mediante análisis estadístico o directamente por la fuerza bruta, pero que fueron empleados con éxito hasta principios del siglo XX. Algunos se remontan incluso, como el algoritmo de César, a la Roma Imperial. Sin embargo aún conservan el interés teórico, ya que algunas de sus propiedades resultan muy útiles para entender mejor los algoritmos modernos.

Refiriéndose a la criptografía moderna se puede decir que recién se desarrolla a partir de los trabajos de Claude Shannon en las áreas de la teoría de la información y de las comunicaciones. Es considerado por muchos autores el padre de la criptografía matemática de la comunicación y la teoría de los sistemas secretos, han establecido las bases para las implementaciones de los algoritmos criptográficos actuales.

Muchas son las voces que claman por la disponibilidad pública de la Criptografía. La experiencia ha demostrado que la única manera de tener buenos algoritmos es que éstos sean accesibles, para que puedan ser sometidos al escrutinio de toda la comunidad científica. Existe una máxima en Criptografía que afirma que cualquier persona —o equipo— es capaz de desarrollar un algoritmo criptográfico que él mismo no sea capaz de romper. Si la seguridad de nuestro sistema se basa en que nadie conozca su funcionamiento tiene varias implicaciones perversas: por un lado, aquellos que quieran conocer su verdadera resistencia tendrán que confiar en nuestra palabra, y por otro, provoca una falsa sensación de seguridad, ya que si algún enemigo encuentra un agujero, es bastante probable que no lo publique. En consecuencia, el único secreto que debe tener un sistema criptográfico es la clave.[31]

En 1995, por ejemplo, dos estudiantes de posgrado de la Universidad de California en Berkeley, David Wagner e Ian Goldberg, hicieron noticia cuando demostraron como romper fácilmente el sistema criptográfico que utilizaba el navegador Netscape para proporcionar seguridad. Parte de la seguridad del sistema residía en que Netscape no había publicado exactamente ¿cómo funcionaba el algoritmo?; a esto se le llama seguridad por obscuridad. En contraste, el estándar DES ha sido desde su origen un algoritmo públicamente conocido hasta en los más mínimos detalles y aún sobrevive.

El Estándar de Cifrado de Datos (DES), es el mecanismo criptográfico de clave simétrica más estudiado y conocido en la historia. Es el estándar más ocupado para la seguridad del comercio electrónico de muchas instituciones financieras alrededor del mundo [25][26][34].





Un buen propósito de explicar este algoritmo es que una buena cantidad de otros algoritmos de cifrado utilizan los mismos principios que el DES. Al entender las transformaciones que ocurren en el DES, será más fácil entender los algoritmos más recientes.

Muchos de los cifrados de producto como el denominado Red de Feistel, es empleado en multitud de algoritmos como DES, Lucifer, FEAL, CAST, Blowfish, etcétera.[31]

Además tenemos unas tablas pequeñas de sustitución se denominan de forma genérica S-Cajas. Existe un algoritmo criptográfico llamado CAST, que emplea seis S-Cajas. DES utiliza también S-Cajas [17].

2.5.1 Variantes de DES

A mediados de julio de 1998, una empresa sin ánimo de lucro, llamada EFF (Electronic Frontier Foundation), logró fabricar una máquina capaz de descifrar un mensaje DES en menos de tres días. Curiosamente, pocas semanas antes, un alto cargo de la NSA había declarado que dicho algoritmo seguía siendo seguro, y que descifrar un mensaje resultaba aun excesivamente costoso, incluso para organizaciones gubernamentales. DES-Cracker costó menos de 250.000 euros.

A pesar de su caída, DES sigue siendo ampliamente utilizado en multitud de aplicaciones, como por ejemplo las transacciones de los cajeros automáticos. De todas formas, el problema real de DES no radica en su diseño, sino en que emplea una clave demasiado corta (56 bits), lo cual hace que con el avance actual de las computadoras los ataques por la fuerza bruta comiencen a ser opciones realistas. Mucha gente se resiste a abandonar este algoritmo, precisamente porque ha sido capaz de sobrevivir durante veinte años sin mostrar ninguna debilidad en su diseño, y prefieren proponer variantes que, de un lado evitarían el riesgo de tener que confiar en algoritmos nuevos, y de otro permitirían aprovechar gran parte de las implementaciones por hardware existentes de DES[31]. Naturalmente, los precios del hardware continuarán bajando mientras la velocidad irá aumentando, haciendo al DES prácticamente inútil[55].

2.5.1.1 DES múltiple

Consiste en aplicar varias veces el algoritmo DES con diferentes claves al mensaje original. El más común de todos ellos es el triple-DES, que corresponde a la siguiente estructura:

$$C = E_{k_1}(E^{-1}_{k_2}(E_{k_1}(M)))$$

Es decir, codificamos con la clave K_{e_1} , decodificamos con K_{e_2} y volvemos a codificar con K_{e_1} . La clave resultante es la concatenación de k_1 y k_2 , con una longitud de 112 bits.





2.5.1.2 DES con subclaves independientes

Consiste en emplear subclaves diferentes para cada una de las 16 rondas de DES. Puesto que estas subclaves son de 48 bits, la clave resultante tendría 768 bits en total. No es nuestro objetivo entrar en detalles, pero empleando criptoanálisis diferencial, esta variante podría ser rotada con 2^{61} mensajes simples escogidos, por lo que en la práctica no presenta un avance sustancial sobre DES estándar.

2.5.1.3 DES generalizado

Esta variante emplea n trozos de 32 bits en cada ronda en lugar de dos, por lo que aumentamos tanto la longitud de la clave como el tamaño de mensaje que se puede codificar, manteniendo sin embargo el orden de complejidad del algoritmo.

2.5.1.4 DES con S-Cajas alternativas

Consiste en utilizar S-Cajas diferentes a las de la versión original de DES. En la práctica no se han encontrado S-Cajas mejores que las propias de DES. De hecho, algunos estudios han revelado que las S-Cajas originales presentan propiedades que las hacen resistentes a técnicas de criptoanálisis que no fueron conocidas fuera de la NSA hasta muchos años después de la aparición del algoritmo[31].

2.5.1.5 Robustez del DES

Los aspectos de robustez del DES se engloban en dos categorías: aspectos sobre el algoritmo mismo y aspectos sobre el uso de una clave de 56 bits. Los primeros se refieren a la posibilidad de que el criptoanálisis se realice explotando las características del algoritmo DES. A lo largo de los años, se han intentado encontrar debilidades que explotar en el algoritmo, lo que ha hecho del DES el algoritmo de cifrado existente más estudiado. A pesar de los numerosos enfoques, nadie ha conseguido descubrir ninguna debilidad grave en el DES.

Un aspecto de mayor importancia es la longitud de la clave. Con una clave de 56 bits, hay 2^{56} claves posibles, que es aproximadamente $7,2 \times 10^{16}$ claves. Por este motivo, no parece práctico un ataque de fuerza bruta. El enfoque por fuerza bruta implica intentar cada clave posible hasta que se obtenga una traducción legible del mensaje cifrado al mensaje simple. Suponiendo que, en promedio, se tiene que intentar la mitad del espacio de claves, una única máquina que realice un cifrado DES por microsegundo tardaría más de mil años en romper el cifrado (véase la Tabla 11).





La Tabla 11 muestra el tiempo necesario para distintos tamaños de clave. El tamaño de clave de 56 bits, se usa con el algoritmo DES (Data Encryption Standard). Para cada tamaño de clave se muestran los resultados suponiendo que cada operación de descifrado simple necesita un μs , lo cual es un valor razonable para las máquinas actuales.[55]

Tabla 11. Tiempo medio para la búsqueda exhaustiva de claves.

Tamaño de clave (bits)	Número de claves alternativas	Tiempo necesario a 1 cifrado/ μs	Tiempo necesario a 10^6 cifrados/ μs
32	$2^{32}=4.3 \times 10^9$	$2^{31} \mu\text{s} = 35.8$ minutos	2.15 milisegundos
56	$2^{56}=7.2 \times 10^{16}$	$2^{55} \mu\text{s} = 1.142$ años	10.01 horas
128	$2^{128}=3.4 \times 10^{38}$	$2^{127} \mu\text{s} = 5.4 \times 10^{24}$ años	5.4×10^{18} años
168	$2^{168}=3.7 \times 10^{50}$	$2^{167} \mu\text{s} = 5.9 \times 10^{36}$ años	5.9×10^{30} años
26 caracteres (permutación)	$26!=4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s} = 6.4 \times 10^{12}$ años	6.4×10^6 años

2.5.2 Claves débiles, semi-débiles, y posiblemente débiles

El algoritmo DES presenta algunas claves débiles. En general, todos aquellos valores de la clave que conducen a una secuencia inadecuada de K_i serán poco recomendables. Distinguiremos entre claves débiles (Tabla 12), que son aquellas que generan un conjunto de dieciséis valores iguales de K_i —y que cumplen $E_k(E_k(M)) = M$ —, y claves semidébiles (Tabla 13), que generan dos valores diferentes de K_i , cada uno de los cuales aparece ocho veces. En cualquier caso, el número de claves de este tipo es tan pequeño en comparación con el número total de posibles claves, que no debe suponer un motivo de preocupación.[31]

Tabla 12. Claves débiles para el algoritmo DES (64 bits), expresadas en hexadecimal.

Clave	Clave tras aplicar EP1
0101010101010101	0000000 0000000
1F1F1F1F0E0E0E0E	0000000 FFFFFFFF
E0E0E0E0F1F1F1F1	FFFFFFFF 0000000
FEFEFEFEFEFEFEFE	FFFFFFFF FFFFFFFF





Tabla 13. Claves semi-débiles para el algoritmo DES (64 bits), expresadas en hexadecimal.

Clave	Clave tras aplicar EP1
01FE01FE01FE01FE	AAAAAAAA AAAAAAAAA
FE01FE01FE01FE01	5555555 5555555
1FE01FE00EF10EF1	AAAAAAAA 5555555
E01FE01FF10EF10E	5555555 AAAAAAAAA
01E001E001F101F1	AAAAAAAA 0000000
E001E001F101F101	5555555 0000000
1FFE1FFE0EFE0EFE	AAAAAAAA FFFFFFFF
FE1FFE1FFE0EFE0E	5555555 FFFFFFFF
011F011F010E010E	0000000 AAAAAAAAA
1F011F010E010E01	0000000 5555555
E0FEE0FEF1FEF1FE	FFFFFFF AAAAAAAAA
FEE0FEE0FEF1FEF1	FFFFFFF 5555555

2.6 Criptosistemas de clave pública

A mediados de los años 70, la novedad en aplicaciones criptográficas giró en torno al concepto de algoritmo de clave pública, el cual fue ideado por Whitfield Diffie y Martin Hellman. La principal diferencia de este tipo de criptosistemas con respecto a los de criptografía simétrica, es que las claves de cifrado y descifrado no son iguales, sino que forman pares (k_e, k_d) .

Actualmente existen muchos algoritmos asimétricos o de clave pública que están basados en problemas matemáticos; de éstos, son pocos los que realmente llegan a ser de utilidad pues se basan en obtener números de forma compleja, lo que hace que computacionalmente sean poco eficientes.

Dentro de los algoritmos de clave pública que existen se pueden mencionar: el RSA, que es de los más sencillos, siguiendo ElGamal y Rabin, entre otros.

2.6.1 El Algoritmo RSA

Realizado en 1977, debe su nombre a sus tres inventores: Ronald Rivest, Adi Shamir y Leonard Adleman. Se considera como uno de los algoritmos asimétricos más seguros y sencillos de implementar.

Mientras que en los algoritmos simétricos o de clave privada la clave es igual tanto para cifrar como para descifrar, en los de clave pública la situación es diferente, pues se tienen





dos claves y no es del dominio público la relación que existe entre ellas; se necesita un procedimiento para calcular la clave pública a partir de la clave privada.

RSA se basa en la complicada operación de factorizar grandes números, pues las claves pública y privada se calculan a partir de un número que se obtiene como producto de dos primos grandes.

Se deben generar un par de claves (k_e ; k_d):

1. En primer lugar se eligen aleatoriamente dos números primos grandes, p y q (secretos),
2. Se calcula el producto $n = pq$.
3. Se obtiene un $\mu = (p-1)(q-1)$.
4. Seleccionamos un entero aleatorio e , $1 < e < \mu$, tal que $\text{mcd}(e, \mu) = 1$
5. De forma que e debe tener inversa módulo μ .
6. Por lo que existirá un número δ tal que $\delta e \equiv 1 \pmod{\mu}$, $1 < \delta < \mu$.
7. La clave pública es (n, e) .
8. La clave privada es δ .

Para cifrar se realizan las siguientes operaciones:

1. Obtener el mensaje simple en un sistema decimal o en binario.
2. Dividir el mensaje simple en partes m_i de tamaño fijo.
3. Se cifra cada m_i aplicando la función de transformación para obtener cada c_i con la siguiente ecuación:

$$c_i = m_i^e \pmod{n}$$

Para descifrar el bloque c_i se aplica la clave privada δ de acuerdo con el siguiente cálculo:

$$m_i = c_i^d \pmod{n}$$

2.6.2 Algoritmo de ElGamal

Es un algoritmo asimétrico, basado en el problema del logaritmo discreto. Fue diseñado en 1985, en un principio para producir firmas digitales, pero posteriormente se extendió también para cifrar mensajes.

Para generar un par de claves:

1. Se escoge un número primo n .
2. Se seleccionan dos números aleatorios a y $b < n$.
3. Se calcula entonces $r = a^b \pmod{n}$
4. La clave pública es (a, r, n) .
5. La clave privada es b .





Para cifrar realizamos las siguientes operaciones:

1. Obtener el mensaje simple m .
2. Seleccionar un número aleatorio s primo relativo a $(n-1)$.
3. El valor de s se mantiene en secreto.
4. Calcular $x = a^s \pmod{n}$
5. $y = r^s m \pmod{n}$
6. El mensaje cifrado es (x,y)

Para descifrar $m = y x^{-b} \pmod{n}$

2.6.3 Algoritmo de Rabin

Es similar a RSA, y consiste básicamente en el problema de calcular raíces cuadradas módulo un número compuesto. Este problema es equivalente al de la factorización de dicho número.

Para generar un par de claves:

1. Seleccionamos dos números primos distintos, p y q .
2. Obtenemos el producto $n = pq$
3. La clave pública es n .
4. La clave privada es (p,q)

Para cifrar realizamos las siguientes operaciones:

1. Obtener el mensaje simple m .
2. La función de transformación es: $c = m^2 \pmod{n}$

Para descifrar realizamos las siguientes operaciones:

1. Encontrar las raíces cuadradas de $c \pmod{n}$ por medio de los siguientes pasos:
 - a. Usar el algoritmo extendido de Euclides para $ap + bq = 1$
 - i. Como entrada tenemos dos números enteros positivos a y b con $a \geq b$.
 - ii. Si $b=0$ entonces se pone el valor de $a \rightarrow d, 1 \rightarrow x, y \rightarrow 0$ así tenemos (d, x, y) .
 - iii. Se pone a $1 \rightarrow x_2, 0 \rightarrow x_1, 0 \rightarrow y_2, 1 \rightarrow x_1$
 - iv. Realizamos las siguientes dos operaciones mientras que $b > 0$:
 1. $a/b \rightarrow q, a - qb \rightarrow r, x_2 - q x_1 \rightarrow x, y_2 - q y_1 \rightarrow y$
 2. $b \rightarrow a, r \rightarrow b, x_1 \rightarrow x_2, x \rightarrow x_1, y_1 \rightarrow y_2, y \rightarrow y_1$
 - v. Se pone el valor de $a \rightarrow d, x_2 \rightarrow x, y_2 \rightarrow y$
 - vi. Utilizamos (d,x,y)
 - b. Obtener $r = c^{(p+1)/4} \pmod{p}$.





- c. Obtener $s = c^{(q+1)/4} \pmod q$.
 - d. Obtener $x = (aps + bqr) \pmod n$
 - e. Obtener $r = (aps - bqr) \pmod n$.
 - f. Las cuatro raíces de c modulo n son: $x, -x \pmod n, y$ y $-y \pmod n$.
2. El mensaje tiene ahora cuatro posibilidades m_1, m_2, m_3 o m_4 . De ahí se selecciona un m . [17]

2.6.4 La nueva dirección en criptografía

El más significativo desarrollo para la historia de la criptografía fue desarrollado en 1976 cuando Diffie Hellman publicó la nueva dirección en criptografía. Este papel introduce un revolucionario concepto de clave pública de cifrado y también provee de un nuevo e ingenioso método para el intercambio de claves de cifrado. En 1978 Rivest, Shamir y Adleman descubrieron el primer cifrado de clave pública y esquema de firma, ahora referido como RSA [3][4][45][63][45][41][36]. Otra clase de estos algoritmos es El Gamal descubierto en 1985 [58][40] y Algoritmo de Rabin, DSA (del inglés *Digital Signature Algorithm*, o Algoritmo de Firma Digital), Curvas Elípticas y PGP (Pretty Good Privacy) siendo estos los más utilizados.

Debido a que el presente trabajo está enfocado al cifrado de datos con clave privada, sólo se comentará brevemente sobre algunos datos relevantes de la criptografía de clave pública.

Los sistemas de clave pública se caracterizan por el uso de un tipo de algoritmo criptográfico con dos claves, una no se revela y la otra sí. Dependiendo de la aplicación, el emisor usa su clave privada o la clave pública del receptor, o las dos, para realizar algún tipo de función criptográfica. En términos generales, podemos clasificar el uso de criptosistemas de clave pública en tres categorías:

- Cifrado/descifrado: el emisor cifra un mensaje con la clave pública del receptor.
- Firma digital: el emisor firma un mensaje con su clave privada. Esto se consigue mediante un algoritmo criptográfico aplicado al mensaje o a un pequeño bloque de datos que es una función del mensaje.
- Intercambio de claves: dos partes cooperan para intercambiar una clave de sesión. Hay distintas posibilidades que implican la clave privada de una o de las dos partes.

Algunos algoritmos son adecuados para las tres aplicaciones, mientras otros sólo se pueden usar para una o dos de ellas. Por ejemplo, el algoritmo RSA y el algoritmo de Curvas Elípticas tienen las tres aplicaciones, mientras que Diffie Hellman sólo es útil para el intercambio de claves y DSS (Digital Signature Standard) es sólo para firma digital.





Actualmente PGP se ha convertido en un estándar internacional (*RFC 2440*), lo cual está dando lugar a la aparición de múltiples productos PGP, que permiten desde cifrar correo electrónico hasta codificar particiones enteras del disco duro (PGPDisk), pasando por la codificación automática y transparente de todo el tráfico TCP/IP(PGPnet)[31]. Se basa en algoritmos que han sobrevivido a revisiones exhaustivas y se consideran sumamente seguros. Concretamente el paquete incluye RSA, DSS y Diffie-Hellman para cifrado de clave pública; CAST-128, IDEA y 3DES para cifrado simétrico; y SHA para codificación hash[55]. Además esta Open PGP que fue en julio de 1997 cuando PGP Inc. propone a la *Internet Engineering Task Force* (IETF) que se desarrolle un estándar, llamado OpenPGP, para normalización del protocolo y evitar el potencial caos dada la creciente aparición de diversas implementaciones de PGP.

2.7 Aplicación práctica de cifrado de datos en distintos lenguajes de programación

GnuPG, por GNU Privacy Guard, es la implementación completa y libre del estándar Open PGP por parte del proyecto GNU, tal como se define en la RFC 4880. GnuPG permite cifrar y firmar datos y comunicaciones. GnuPG también conocido como GPG no se basa en algoritmos patentados o restringidos de modo alguno. En lugar de ello, puede utilizar, por ejemplo, los algoritmos CAST5, Triple DES, AES, Blowfish y Twofish. Este es un software de cifrado híbrido[67].

Open PGP es el estándar de cifrado de correo electrónico más ampliamente utilizado en la actualidad. Se define y mantiene por el grupo de trabajo openPGP de la IETF, en el documento RFC 4880.[32][65]

2.8 Patentes de los algoritmos criptográficos

Al hablar de patentes que podrían restringir la disponibilidad de implementaciones de algoritmos criptográficos actuales, empezaremos comentando un caso. En el caso del algoritmo IDEA o *International Data Encryption Algorithm* (*Algoritmo Internacional de Cifrado de Datos*), inventado por Xuejia Lai y James Massey en la Escuela Politécnica Federal de Zúrich, su patente, actualmente, es propiedad de Ascom-Tech.

IDEA fue diseñada en contrato con la Fundación Hasler, la cual se hizo parte de Ascom-Tech AG. IDEA es libre para uso no comercial, aunque fue patentado y sus patentes se





vencerán en 2010 y 2011. El nombre IDEA es una marca registrada y está bajo licencia mundial por MediaCrypt.

Veremos algunas otras patentes registrada en los EE.UU. a manera de ejemplo (Tabla 14) –se notará cuáles están actualmente caducas-[32]:

Tabla 14. Patentes registradas en EE.UU.

Nombre del algoritmo	Año de envío	Año de patente	Nro. De patente	Inventores	Asignación	Patente caduca
DES	1975	1976	3,962,539	Ehram et al	IBM	SI
Diffie-Hellman	1977	1980	4,200,770	Hellman, Diffie, y Merkle	Universidad de Stanford	SI
RSA	1977	1983	4,405,829	Rivest, Shamir, y Adleman	MIT	SI
IDEA	1992	1993	5,214,703	Lai y Massey AG	Ascom Tech	NO
DSA	1991	1993	5,231,668	Kravitz	EE.UU.	NO

2.9 Sistemas distribuidos hoy en día

Dependiendo de las necesidades de la aplicación, el sistema distribuido puede estar compuesto de diversos componentes: computadoras personales, sistemas de tiempo real VME/VXI/PXI, PLCs, dispositivos robustos FieldPoint o cualquier combinación con nodos de cálculo, u otros similares, en la Fig. 12 se muestra un diagrama de un sistema distribuido.



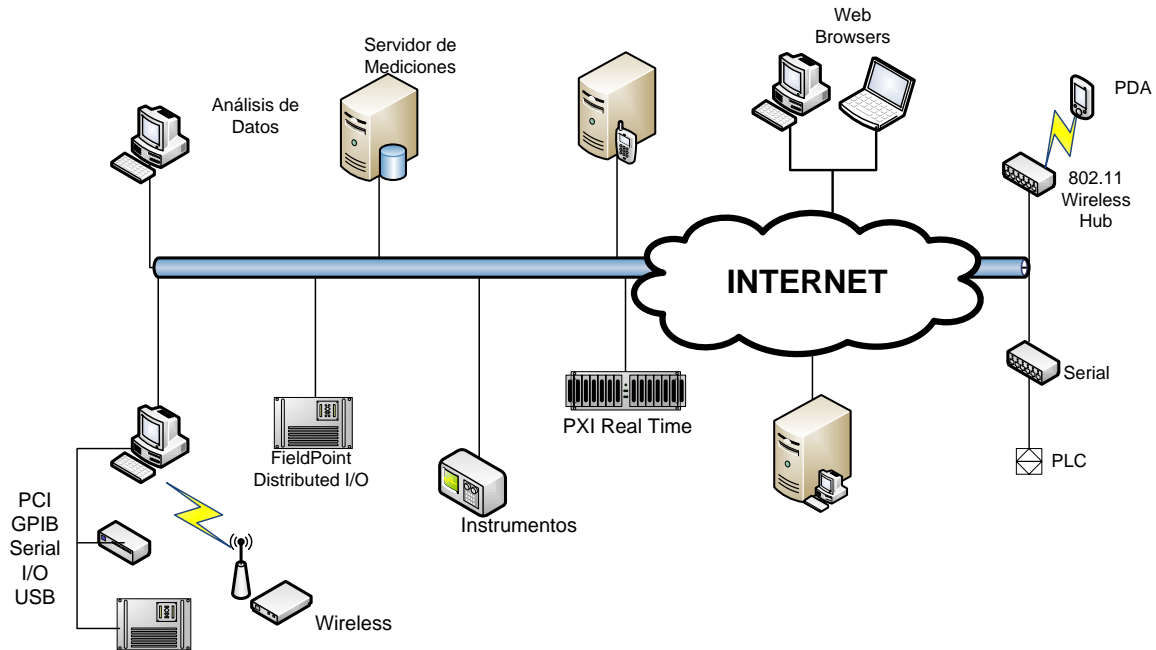


Fig. 12. Red Ethernet de computadoras y otros nodos de cálculo.

Tecnologías de redes inalámbricas, tales como Ethernet inalámbrica y Bluetooth, hacen que sea menos tediosa la instalación de un sistema distribuido. Al final, la clave para integrar el sistema es una fuerte integración del software. Si se necesita escalar el sistema a través del tiempo, el software también debe ser suficientemente abierto para agregar nueva tecnología sin perturbar severamente el sistema o recomenzarlo.

Los sistemas distribuidos hoy en día también incluyen:

- PDAs (asistente digital personal) que reciben datos de un dispositivo con sistema operativo incorporado (embedded) sistema embebido o empotrado.
- Celdas de ensayo que envían datos a un servidor central
- Sensores que transmiten datos a una computadora personal de manera inalámbrica

Un sistema distribuido involucra componentes de hardware que comunican datos o E/S de control que están temporizadas o sincronizadas entre sí, típicamente a altas velocidades.

Ejemplos de estos sistemas son:

- FPGAs (del inglés *Field Programmable Gate Arrays*) que se comunican con microprocesadores.
- Conjunto de nodos de control sobre un bus de tiempo real.





Además, se pueden encontrar sistemas híbridos, como el que se observa en la Fig. 13, donde se ejecuta una lógica en una FPGA que trabaja con un sistema de cálculo de tiempo real, el cual, a su vez, se comunica a una macro red Ethernet.

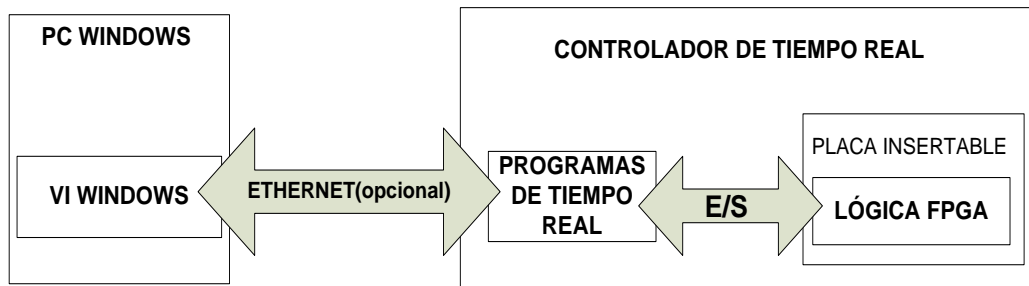


Fig. 13. Ejemplo de una red híbrida ethernet con E/S.

La topología de un sistema distribuido se utiliza en diversos entornos, desde laboratorios de investigación a plantas manufactureras. Ya sea que se utilice en el experimento de un acelerador de partículas o en una eficiente línea de envasado, se encontrarán muchos de los siguientes elementos clave:

- Medición y control de tiempo real a nivel de nodos
- Monitoreo, registro de datos y control del sistema a niveles de supervisión
- Capacidad para integrar protocolos estándar de sistemas distribuidos, tales como EPICS y OPC a fin de enlazar nodos existentes.
- Capacidad para utilizar buses de comunicaciones estándar, tales como Ethernet y CAN.
- Flexibilidad de programación abierta para integrar dispositivos tales como software de terceros, múltiples sistemas operativos, PLCs de varias marcas, cámaras y otros componentes de hardware o software[72].

2.9.1 Implementaciones en sistemas distribuidos con instrumentación virtual:

Con la plataforma LabVIEW se puede desarrollar más rápidamente este tipo de sistema distribuido y mantener a la vez la adaptabilidad en el largo plazo. La programación gráfica de LabVIEW facilita el desarrollo de estos sistemas mediante las siguientes áreas clave:

- Paradigma de desarrollo consistente, que escala desde Windows a Linux, a Solaris,





- a Mac OS X, a RTOSs y a dispositivos con SO incorporados.
- Interfaces a redes abiertas, que incluyen TCP/IP, UDP, ActiveX, .NET Web services y más.
- Herramientas de productividad para construir aplicaciones de redes, tales como:
 - Protocolo cliente/servidor Data Socket para grabación a disco en vivo basado en TCP/IP.
 - Servidor web incorporado que no requiere programación.
 - Asistente de comunicaciones, que genera código para comunicarse con sistemas que van desde tiempo real a sistemas sin restricciones críticas de tiempo.
 - Funcionalidad de temporizado y sincronización, que aprovecha las características de hardware para integrar E/S.

Utilizando las características de red de LabVIEW, se pueden comunicar datos a través de nodos ya sea mediante TCP/IP o UDP para transmisión de datos de bajo nivel o con la tecnología de mayor nivel Data Socket cliente/servidor. Se puede obtener control remoto utilizando ActiveX o también se puede buscar los servicios .NET Web y llamarlos remotamente. Además, los nodos mismos son activados a través de la Web utilizando las herramientas Web de publicación de LabVIEW para presentar datos a otros nodos con un navegador Web. Un nodo también puede exponer su aplicación para control remoto vía el navegador Web en máquinas que son designadas como seguras[72][73].

La industria de la tecnología de la información también tiene herramientas para administrar la conectividad y seguridad de sistemas de control, y adquisición de datos que usan comunicaciones basadas en Web, como el servidor Web incluido en cada sistema de ejecución como LabVIEW Real-Time de National Instruments. Por ejemplo, con herramientas comerciales, puede configurar prioridades de anchos de banda y configurar agrupaciones lógicas en los dispositivos en red a través de VLANs y para seguridad, puede usar herramientas estándar para configurar redes privadas virtuales (VPNs), activar filtros IP, limitar acceso a puertos, y activar el monitoreo de acceso no autorizados[72].

Ahora, por otro lado tenemos los sistemas distribuidos que continúan ganando aceptación en aplicaciones de medición y control distribuidas debido a su bajo costo, operación entre dispositivos, comunicación con la empresa y un mayor ancho de banda. Como ejemplo de estos sistemas está el presentado en el proyecto denominado *Sistema de Monitoreo Ambiental de Ruido* en el Centro Histórico, se colocarán 10 unidades de medición, consistentes en cajas de 56 por 44 centímetros, con un micrófono y un dispositivo de cómputo que enviarán al centro de control de reportes cada cinco minutos mediante una conexión inalámbrica de internet. Los dispositivos se colocarán cuatro metros sobre el suelo, operarán las 24 horas del día y almacenarán la información recabada con un software desarrollado por mexicanos, que cumple con estándares internacionales[47][48][44][49].





2.9.2 Sistema de monitoreo y control en el país

Existen en el país empresas dedicadas al desarrollo de soluciones de control, medición y automatización aplicadas a la industria eléctrica, petroquímica, de la construcción, etc.[74][75][76][77][78][79]. Dentro de los sistemas que se tienen están los siguientes:

2.9.3 Sistema digital de monitoreo y control

Solución modular para atender las necesidades de control de procesos de refinерías, centros procesadores de crudo, centros de procesadores de gas, complejos petroquímicos, y ductos, así aquéllos que tienen que ver con la gestión y venta de productos como son terminales de almacenamiento y centros de reparto.

El sistema digital de monitoreo y control está constituido por 3 subsistemas principales:

1. Subsistema de control supervisorio

Mediante el cual se establece la interface entre el usuario y el sistema, el cual consta de los equipos de cómputo, el software de control que permite mediante pantallas específicas mantener el monitoreo y control de los bloques funcionales del proceso.

2. Subsistemas de control local

Mediante el uso de modernas y poderosas computadoras industriales o procesadores de lógica programable, permiten la configuración de lazos de control específicos, que gobiernan el sistema vital de la operación en la planta a partir de las señales de equipos e instrumentos instalados en campo. La información generada por el subsistema de control local es enviada al subsistema de control supervisorio para su interpretación por parte de los operadores.

3. Subsistemas de comunicación

Incluye switches, ruteadores, equipos de radio o satelitales como alternativas posibles para integrar desde sistemas mínimos hasta sistemas con topologías complejas. Un grupo de protocolos de comunicación, tales como HART, MODBUS RTU, MODBUS PEMEX, complementan de manera adecuada los requerimientos técnicos de estas soluciones.





Para complementar las necesidades en centros atendidos, cuentan con un grupo de subsistemas de aplicación a nivel de campo, que incluyen las interfaces adecuadas para su comunicación con el sistema digital de monitoreo y control: Subsistema de Acceso Vehicular, Subsistema de Telemedición, Subsistema de Operación de Válvulas Eléctricas, Subsistema de Operación de Bombas, Subsistema de Carga y Descarga de Productos.[70]. Entre otros.





2.10 Conclusiones del capítulo

En este capítulo se tratan cuestiones generales relativas a la Criptología y a los Criptosistemas, se detallan características principales de aquellos que por sus propiedades distintivas son considerados como los más importantes.

Los criptosistemas se dividen en dos grandes grupos los de clave privada y los de clave pública. Dentro del primer grupo tenemos los siguientes: monoalfabéticos, polialfabéticos, por sustitución homonofónica, transposición, redes de Feistel, S-Cajas, Cast, Blowfish, Twofish, IDEA, AES y no puede faltar el DES.

Por otro lado tenemos los criptosistemas de clave pública dentro de los más importantes RSA, algoritmo de ElGamal, algoritmo de Rabin y algunos otros que debido al extenso compendio de los algoritmos existentes sólo se ponen estos.

Dentro de este capítulo se expone la importancia del criptosistema DES, sus variantes, y algunas propiedades dentro de la criptografía. Se mencionan algunos criptosistemas parecidos a DES y las aplicaciones en lenguajes de programación existentes.

También se comenta sobre los sistemas distribuidos, en particular sobre su implementación con instrumentación virtual y la utilidad que tienen hoy en día. Tomando en cuenta las características de los sistemas de monitoreo y control y sus subsistemas como son de: control supervisorio, de control local y de comunicación.

Sin embargo, teniendo en cuenta sistemas desarrollados mediante instrumentación virtual u otros, los sistemas de comunicación, en general, se puede tener acceso a los canales con o sin autorización, por lo que, se toman mayores precauciones en el envío de datos entre varios usuarios, que requieren de confidencialidad en sus envíos, para hacer esto es posible hacer uso de la criptografía y por lo tanto tomar un algoritmo estándar para cifrar los datos y que sin importar que tan seguro o inseguro sea un canal, permita enviar información a través de él de manera segura.

Por lo anterior, el aporte de este trabajo de tesis, con respecto a otras implementaciones consiste en haber generado un par de módulos para cifrar y descifrar datos en instrumentación virtual para que sea utilizado en sistemas distribuidos, pues actualmente no existen herramientas y funciones pre-programadas para el uso de criptosistemas dentro de las aplicaciones que realizan funciones de monitoreo o supervisión y adquisición de datos.





3. MARCO TEÓRICO

3.1 Introducción

El constante crecimiento de las redes de comunicaciones genera una gran cantidad de transmisión de información, de manera tal que por medio de la computadora se obtiene o se genera información de diversos formatos, tipos y contenidos. Existen sistemas y datos de suma importancia para las instituciones, empresas y personas, que buscan la manera más segura de mantener la integridad, la autenticidad y la confidencialidad de su información, para esto recurren a la implementación de técnicas o algoritmos criptográficos. Los cuales consisten en una serie de pasos que sustituyen y modifican la información que se codifica para que no cualquier persona la pueda entender.

La seguridad en el manejo de la información está sujeta a diversos factores que pueden llegar a alterar, dañar o incluso a destruir tal información; por lo que es necesario utilizar un programa de criptografía que proporcione mejor seguridad dentro de un mundo informático.

Por otra parte está la función básica del sistema distribuido, consiste en supervisión, y telecontrol de equipo remoto de adquisición de datos, permitiendo visualizar las diferentes variables de estados y variables analógicas que las unidades de transmisión remota adquieren.

En relación con los puntos anteriores debe hacerse hincapié en la falta de personal que diseñe, programe, implemente, administre y mantenga los sistemas de regulación y supervisión que satisfaga esas necesidades[50].

De acuerdo a lo anterior, a continuación se hace una breve recopilación de los avances, tanto científicos como comerciales, que hasta ahora se tienen en criptosistemas y en particular de sus aplicaciones en los sistemas distribuidos, y sistemas de adquisición de datos y monitoreo.





3.2 Selección del algoritmo

A la hora de incorporar criptografía en nuestros desarrollos, nos veremos en la necesidad de optar por alguno de los algoritmos disponibles en nuestro entorno de programación, entornos de trabajo (*frameworks*) o librerías adicionales.

En primera instancia, deberíamos tener en claro que el cifrado asimétrico no reemplaza al cifrado simétrico, que ambos mecanismos proveen soluciones a problemas diferentes y cuando nos será conveniente la utilización de cada una[32].

El presente trabajo se enfoca en DES, por lo que comentaremos algunas de las bondades que tiene:

- Cifra con bloques de 64 bits de longitud, además utiliza una clave del mismo tamaño (salvo por una pequeña diferencia en el manejo de la clave).
- Utiliza el mismo algoritmo para cifrar como para descifrar.
- Se basa en una combinación de confusión y difusión.
- No tiene restricción de uso el algoritmo, su patente está vencida.
- Se encuentra completamente especificado.
- Es fácil de entender.
- La seguridad reside en la clave.
- Utiliza operaciones elementales.
- Se ideó una alternativa para que, manteniendo el mismo algoritmo, se pudiese incrementar el tamaño de la clave. De esta manera, surge este nuevo algoritmo, que en su versión o variante más simple no es otra cosa que la triple aplicación del algoritmo DES, con una clave que correspondería al conjunto de las tres claves DES aplicadas.
- Fue aprobado como estándar federal por el Instituto Nacional Americano de Estándares-de los E.E.U.U- o American National Standards Institute (ANSI), en el año 1981. Antes fue aprobado también como estándar federal del mismo país, FIPS(Federal Information Processing Standard) en el año 1976 (publicado en 1977 como FIPS PUB 46)[32][51][57].

3.3 Sistemas SCADA

Damos el nombre de SCADA (*Supervisory Control And Data Acquisition* o Control con Supervisión y Adquisición de Datos) a cualquier software que permita el acceso a datos remotos de un proceso y permita, utilizando las herramientas de comunicación necesarias en cada caso, el control del mismo.





No se trata de un sistema de control, sino de una utilidad software de monitorización o supervisión, que realiza la tarea de interfaz entre los niveles de control (PLC) y los de gestión, a un nivel superior.

Los objetivos para que su instalación sea perfectamente aprovechada son los siguientes:

- Funcionalidad completa de manejo y visualización en el sistema operativo Windows sobre cualquier PC estándar.
- Arquitectura abierta que permita combinaciones con aplicaciones estándar y de usuario, que permitan a los integradores crear soluciones de mando y supervisión optimizadas (Active X para ampliación de prestaciones, OPC para comunicaciones con terceros, OLE-DB para comunicación con bases de datos, lenguaje estándar integrado como VB(Visual Basic) o lenguaje C, acceso a funciones y datos mediante una API).
- Sencillez de instalación, sin exigencias de hardware elevadas, fáciles de utilizar, y con interfaces amigables con el usuario.
- Permitir la integración con las herramientas ofimáticas y de producción.
- Fácilmente confiable y escalable, debe ser capaz de crecer o adaptarse según las necesidades cambiantes de la empresa.
- Ser independiente del sector y la tecnología. Funciones de mando y supervisión integradas.
- Comunicaciones flexibles para poder comunicarse con total facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión).

3.3.1 Arquitectura de un sistema SCADA

El desarrollo de la computadora personal ha permitido su implantación en todos los campos del conocimiento y a todos los niveles imaginables.

Las primeras incursiones en el campo de la automatización localizaban todo el control en la computadora y tendían progresivamente a la distribución del control en planta. De esta manera, el sistema queda dividido en tres bloques principales:

1. Software de adquisición de datos y control (SCADA).
2. Sistemas de adquisición de datos y mando (sensores y actuadores).
3. Sistemas de interconexión (comunicaciones).



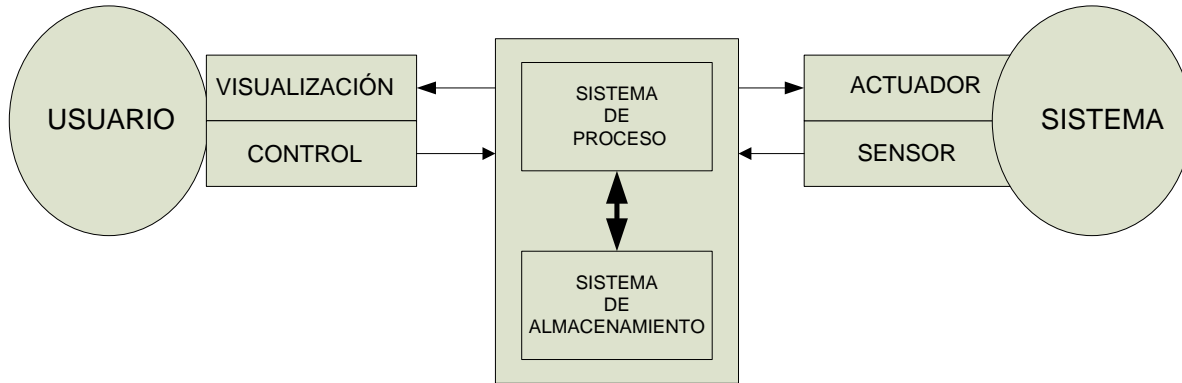


Fig. 14. Estructura básica de un sistema de supervisión y mando.

El usuario, mediante herramientas de visualización y control, tiene acceso al sistema de control de proceso, generalmente una computadora donde reside la aplicación de control y supervisión (se trata de un sistema servidor). La comunicación entre estos dos sistemas se suele realizar a través de redes de comunicaciones corporativas (Ethernet) ver Fig. 14.

El sistema de proceso capta el estado del Sistema a través de los elementos sensores e informa al usuario a través de las herramientas HMI. Basándose en los comandos ejecutados por el usuario, el sistema de proceso inicia las acciones pertinentes para mantener el control del Sistema a través de los elementos actuadores.

La transmisión de los datos entre el sistema de proceso y los elementos de campo (sensores y actuadores) se lleva a cabo mediante los denominados buses de campo. La tendencia actual es englobar los sistemas de comunicación en una base común, como Ethernet industrial. Toda información generada durante la ejecución de las tareas de supervisión y control se almacena para disponer de los datos a posteriori.

Mediante el software de adquisición de datos y control, el mundo de las máquinas se integra directamente en la red empresarial, pasando a formar parte de los elementos que permitirán crear estrategias de empresa globales.

Un sistema SCADA es una aplicación de software especialmente diseñada para funcionar sobre computadoras en el control de producción que proporciona comunicación entre los dispositivos de campo, llamados también RTU (Remote Terminal Units o unidades Remotas), donde se pueden encontrar elementos tales como controladores autónomos o autómatas programables, y un centro de control o unidad central (MTU, Master Terminal Unit), donde se controla el proceso de forma automática desde la pantalla de uno o varias computadoras.





3.3.1.1 El hardware

Un sistema SCADA, a escala conceptual, está dividido en dos grandes bloques:

- Captadores de datos: Recopilan los datos de los elementos de control del sistema (por ejemplo, autómatas, reguladores, registradores) y los procesan para su utilización. Son los servidores del sistema.
- Utilizadores de datos: Los que utilizan la información recogida por los anteriores, como pueden ser las herramientas de análisis de datos o los operadores del sistema. Son los clientes.

Mediante los clientes los datos residentes en los servidores pueden evaluarse, permitiendo realizar las acciones oportunas para mantener las condiciones nominales del sistema.

Mediante los denominados buses de campo, los Controladores de proceso (generalmente autómatas programables o sistemas de regulación) envían la información a los Servidores de datos (Data Servers), los cuales, a su vez, intercambian la información con niveles superiores del sistema automatizado a través de redes de comunicaciones de Área Local.

Estos sistemas están formados por los siguientes elementos básicos:

- Interfaz Hombre-máquina
- Unidad Central
- Unidad Remota
- Sistema de Comunicaciones

3.3.1.2 Interfaz hombre-máquina (HMI, MMI)

Comprende los sinópticos de control y los sistemas de presentación gráfica. La función de un Panel Sinóptico es la de representar, de forma simplificada, el sistema bajo control (un sistema de aprovisionamiento de agua, una red de distribución eléctrica, una factoría).

En un principio los paneles sinópticos eran de tipo estático, colocados en grandes paneles plagados de indicadores y luces. Con el tiempo han ido evolucionando, junto al software, en forma de representaciones gráficas en pantalla de visualización de Datos). En los sistemas complejos pueden aparecer las terminales múltiples, que permiten la visualización, de forma simultánea, de varios sectores del sistema.

3.3.1.3 Unidad central (MTU, Master Terminal Unit)

Centraliza el mando del sistema. Se hace uso extensivo de protocolos abiertos, lo cual permite la interoperabilidad de multiplataformas y multisistemas. Un sistema de este tipo





debe estar basado en estándares asequibles a bajo precio para cualquier parte interesada. De esta manera es posible intercambiar información en tiempo real entre centros de control y subestaciones situadas en cualquier lugar.

En el Centro de Control se realiza, principalmente, la tarea de recopilación y archivado de datos. Toda esta información que se genera en el proceso productivo se pone a disposición de los diversos usuarios que puedan requerirla. Se encarga de:

- Gestionar las comunicaciones.
- Recopilar los datos de todas las estaciones remotas (RTU).
- Envío de información.
- Comunicación con los operadores.
- Análisis.
- Impresión
- Visualización de datos
- Mando
- Seguridad

Estas tareas están encomendadas a equipos informáticos con funciones específicas y exclusivas, tales como:

- **Almacenar datos (Database Server):** se ocupa del archivado de datos para el proceso posterior de los mismos mediante herramientas de representación gráfica o de análisis estadístico.
- **Almacenar archivos (File Server):** almacena los resultados de los análisis de los datos recogidos, guarda los datos concernientes a los eventos del sistema, datos de configuraciones, alarmas, etc.
- **Administración:** permite la gestión y el almacenamiento del sistema SCADA, controlar los sistemas de seguridad, modificar la configuración de las tareas de backup, etc.
- **Comunicaciones:** permite el intercambio de datos en tiempo real con estaciones remotas. Este es un punto de entrada y salida de datos, por tanto, debe prestarse especial atención a la seguridad y protegerlo de accesos no autorizados.

3.3.1.4 Unidad remota (RTU, Remote Terminal Unit)

Por unidad Remota o Estación Remota, podemos entender aquel conjunto de elementos dedicados a labores de control y/o supervisión de un sistema, alejados del Centro de Control y comunicados con este mediante algún canal de comunicación.

Dentro de esta clasificación podemos encontrar varios elementos más o menos diferenciados:

- RTU (Remote Terminal Unit): especializado en comunicación.





- PLC (Programable Logic Controller): tareas generales de control.
- IED (Intelligent Electronic Device): tareas específicas de control.

3.3.1.4.1 RTU

Las unidades remotas se encargaban en un principio de recopilar los datos de los elementos de campo (autómatas reguladores) y transmitirlos hacia la unidad central, a la vez que enviar los comandos de control a estos. Serían los denominados Procesadores de Comunicaciones.

Suelen estar basadas en computadoras especiales que controlan directamente el proceso mediante tarjetas convertidoras adecuadas o que se comunican con los elementos de control (PLC, reguladores) mediante los protocolos de comunicación adecuados. Su construcción es más robusta, son operativos dentro de un rango de temperaturas mayor que las computadoras normales y su robustez eléctrica también es mayor (transitorios de red, variaciones de alimentación, interferencias electromagnéticas).

Con la introducción de sistemas inteligentes aparecen también las funciones de adquisición y proceso de datos, así como de seguridad ante accesos sin autorización o situaciones anómalas que pueden perjudicar al funcionamiento de la estación y provocar daños en sus componentes.

El software de estos elementos suele estar elaborado en lenguajes de alto nivel (C, VisualBasic, Delphi) que permiten interpretar los comandos provenientes de la estación Maestra (Master Terminal Unit).

3.3.1.4.2 PLC

Los controladores lógicos programables o PLC (Programmable Logic Controller) como sistemas de dedicación exclusiva al control de instalaciones, máquinas o procesos. Con el tiempo han ido evolucionando, incorporando cada vez más prestaciones en forma de módulos de ampliación, entre ellos los Procesadores de Comunicaciones, que han hecho desvanecerse la línea divisoria entre RTU y PLC, quedando incluidas todas las prestaciones en el PLC.

3.3.1.4.3 IED

Son los denominados periféricos inteligentes (Intelligent Electronic Devices). Se trata de elementos con propiedades de decisión propias (programas) que se ocupan de tareas de control, regulación y comunicación. Dentro de esta clasificación se pueden encontrar elementos tales como PCL, Reguladores, Variadores de Frecuencia, Registradores, Procesadores de comunicaciones, Generadores de tiempo y frecuencia, Controladores de energía reactiva, Transductores, etc.





3.3.1.4.4 Sistemas remotos

Hoy día una estación remota no es necesariamente un autómatas con capacidades de comunicación controlando una compuerta de un embalse. Puede tratarse de un gran sistema complejo que forme parte, a su vez, de un sistema de control mucho más extenso, como el control de distribución eléctrica de un país, donde las estaciones remotas pueden tener a su cargo una ciudad entera o controlar la distribución regional.

En este caso, la estación remota tiene implementadas funciones de control, interfaz hombre-máquina, adquisición de datos, control de base de datos, protocolos de seguridad y comunicaciones internas entre subsistemas.

En la Fig. 15, por ejemplo, se puede observar una subestación de control de una depuradora dentro del sistema de distribución y gestión de agua para consumo de una región determinada.

La subestación está protegida de dos maneras:

- **Hardware:** funcionan como barreras físicas; desde la valla de protección de los recintos y los sistemas de vigilancia, hasta las claves de las salas de control o de los armarios que contienen los elementos de mando (PLC).
- **Software:** son barreras lógicas. Los accesos desde dentro, no autorizados, se evitan mediante sistemas de contraseñas en los equipos. Los accesos desde fuera, mediante dispositivos especiales que limitan el acceso (cortafuegos o firewalls).



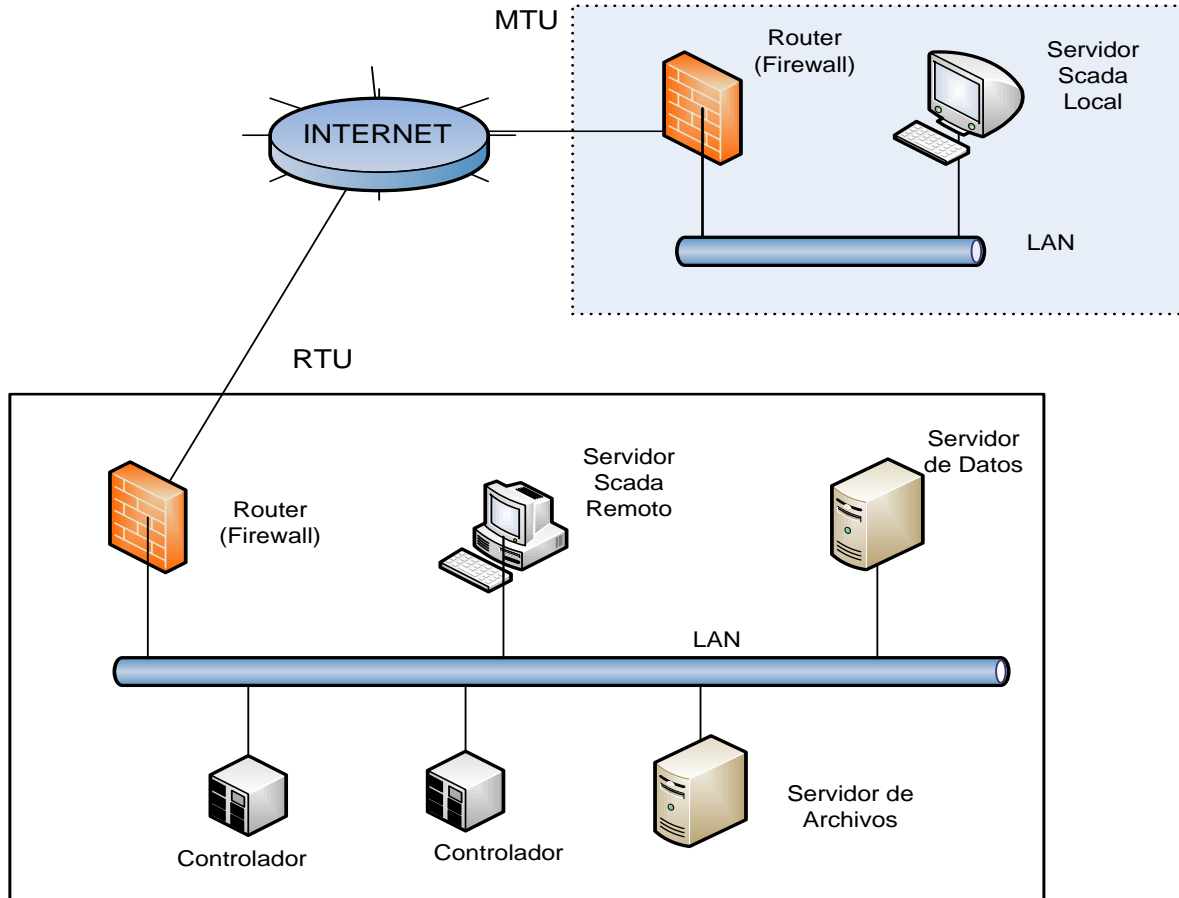


Fig. 15. Arquitectura general de una RTU.

3.3.1.5 Sistemas de comunicación

El intercambio de información entre servidores y clientes se basa en la relación de productor-consumidor.

Los servidores de datos interrogan de manera cíclica a los elementos de campo (polling), recopilando los datos generados por registradores, autómatas, reguladores de proceso, etc.

Gracias a los controladores suministrados por los diferentes fabricantes y a su compatibilidad con la mayoría de estándares de comunicación existentes (léase buses de campo), es posible establecer cualquier tipo de comunicación entre un servidor de datos y cualquier elemento de campo.





Un servidor de datos puede gestionar varios protocolos de forma simultánea, estando limitado por su capacidad física de soportar las interfaces de hardware (las popularmente conocidas tarjetas de comunicación). Estas permiten el intercambio de datos bidireccional entre la Unidad Central y las unidades remotas (RTU) mediante un protocolo de comunicaciones determinado y un sistema de transporte de la información para mantener el enlace entre los diferentes elementos de la red:

- Línea telefónica, dedicada o no.
- Cable coaxial
- Fibra óptica
- Telefonía celular (GPRS, UMTS).
- Radio (enlaces de radio VHF, UHF, Microondas).

La topología de un sistema SCADA (su distribución física) variará adecuándose a las características de cada aplicación. Unos sistemas funcionarán bien en configuraciones de bus, otros en configuraciones de anillo. Unos necesitarán equipos redundantes debido a las características del proceso, etc.[43]

3.4 Sistema distribuido

3.4.1 El modelo cliente/servidor

Las necesidades actuales de las empresas (mayor competitividad, globalización, flexibilidad, calidad, etc.) junto a la evolución de las tecnologías de la información han propiciado la difusión del modelo conocido con el nombre de cliente/servidor.

No existe una definición universal del modelo cliente/servidor, por lo que aquí se exponen algunas de las aproximaciones utilizadas más frecuentemente.

Cliente/servidor es una tecnología distribuida que define el papel que realiza un cliente que requiere un servicio y el papel del servidor que proporciona el servicio.

Ninguna de las dos definiciones expone un modelo o estructura completos, por lo que se hace necesario caracterizar al modelo cliente/servidor de la siguiente manera:

- El servidor debe ofrecer y presentar una interfaz que este bien documentada y que sea pública (es decir, publicada para que pueda ser utilizada por todas las aplicaciones). Esta característica se considera esencial cuando el procesamiento centralizado se sustituye por un conjunto de servidores que son llamados arbitrariamente por un conjunto de clientes.





- El cliente no conoce al servidor lógico, sino su interfaz y valores que el servidor devuelve al cliente. Como consecuencia, el código del cliente no depende de la realización del servidor; no obstante, puede que un servidor presente una serie de restricciones o que no pueda utilizarse en todos los tipos de entornos.
- Los clientes no dependen de la localización física del servidor.
- Las aplicaciones cliente deben ser transparentes al entorno local.

En la Fig. 16. se muestra un esquema típico basado en el modelo cliente/servidor.

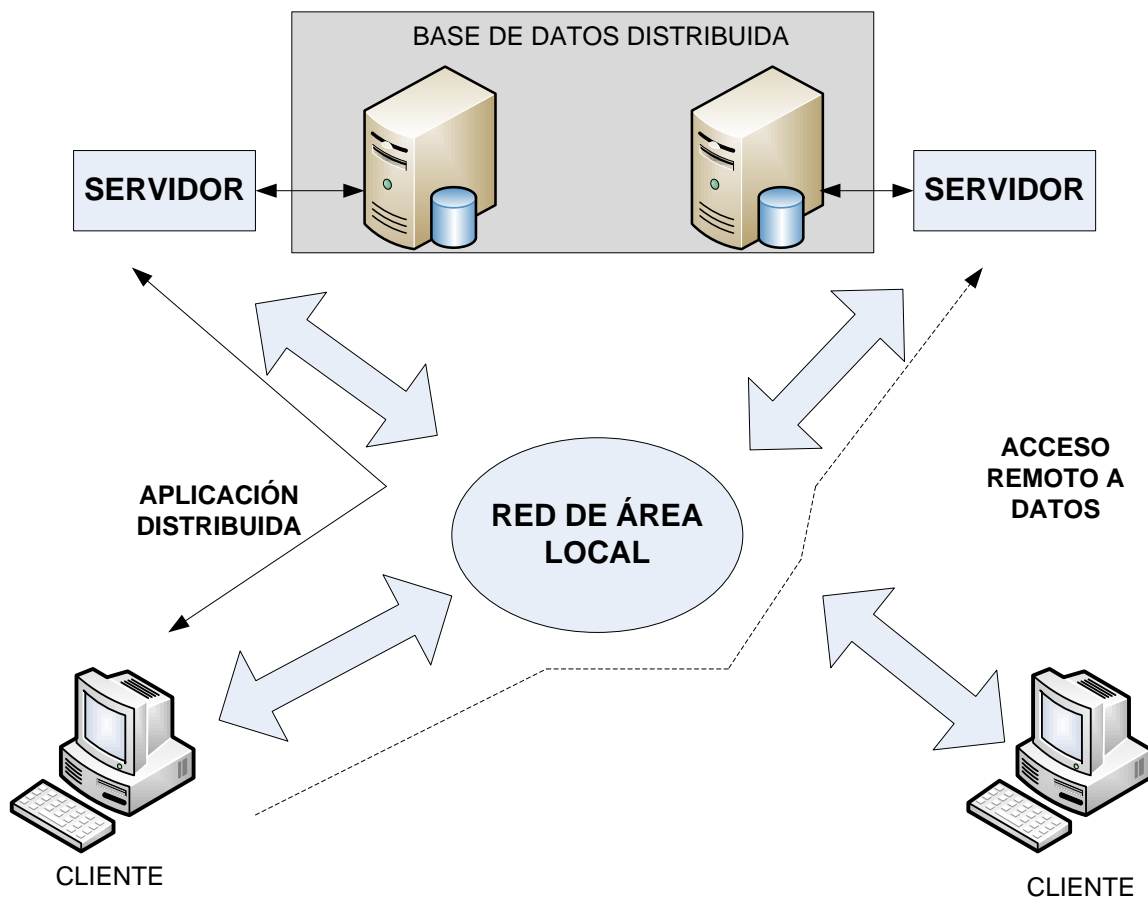


Fig. 16. Esquema del modelo cliente/servidor.

Aunque normalmente se asocian los papeles del cliente y del servidor a los computadores donde se ejecuten las funciones, no hay que confundir el concepto de cliente o de servidor con los equipos, ya que ambas funciones pueden ser desempeñadas por la misma





computadora. Así, por ejemplo, en la misma computadora puede existir una aplicación que este proporcionando servicios solicitados por aplicaciones residentes en otras computadoras (con lo que estaría actuando de servidor), mientras que puede existir otra aplicación que se encuentre demandando servicios a otros servidores (con lo que estaría actuando de cliente).

3.4.2 Elementos básicos del modelo cliente/ servidor

En la Fig. 17 se muestran los componentes básicos del modelo cliente/servidor. En la parte superior de la figura se encuentran las aplicaciones cliente que invocan a los servidores representados en el segundo recuadro. La infraestructura es el conjunto de servicios elementales necesarios para la realización del modelo, mientras que en el recuadro inferior se encuentra la red, con su interfaz de programación (API) que permite que las aplicaciones cliente/servidor utilicen los niveles de transporte e inferiores. En el caso ideal debería ser independiente del sistema de transporte. Debajo de la interfaz están algunos de los protocolos de transporte a los que invocará en cada caso y, finalmente, los niveles de red e inferiores.

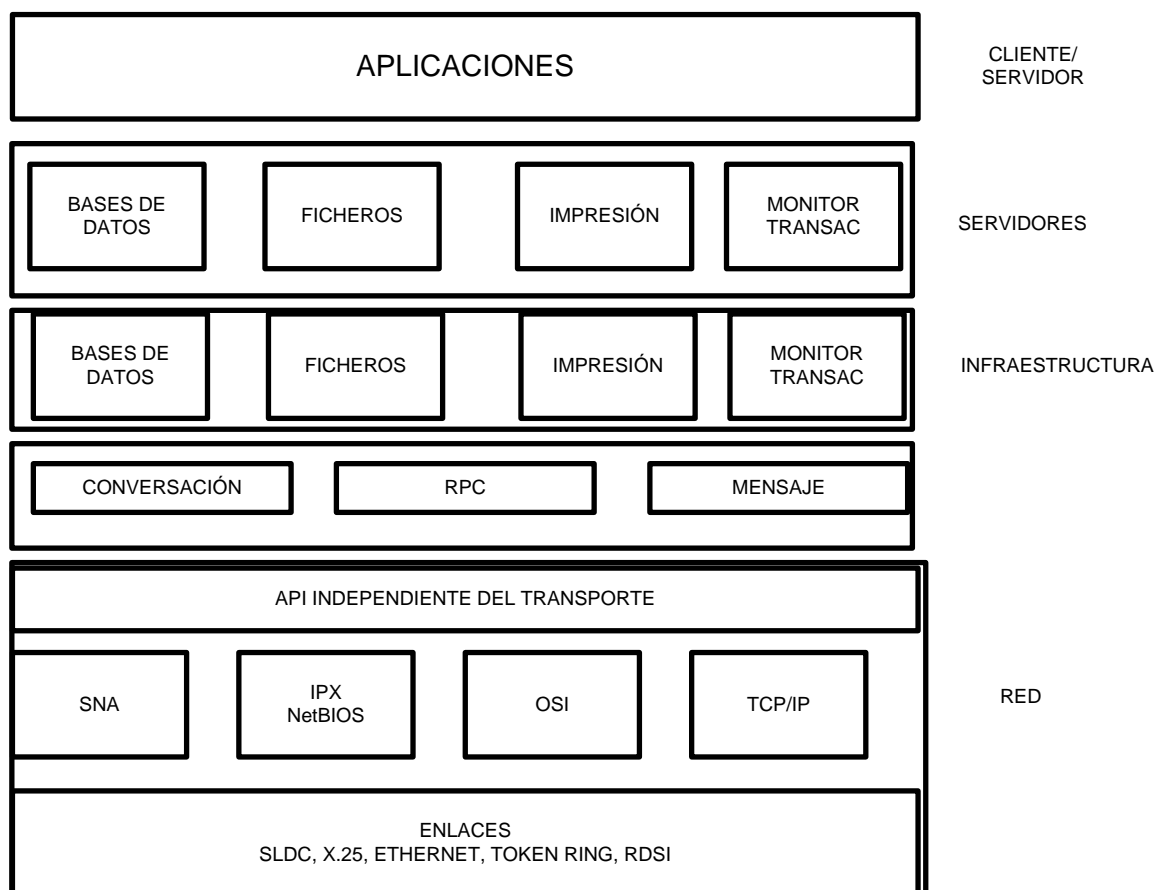


Fig. 17. Componentes básicos del modelo cliente/servidor.





3.4.3 Infraestructura

La infraestructura está constituida por un conjunto de servicios horizontales que pueden ser utilizados por cualquier aplicación.

Entre estos servicios comunes destacan: servicio de identificación de entidades (directorio), servicio de seguridad, servicio de conversión de datos, servicio de gestión de transacciones, servicio de conexión entre redes, servicio de comunicación entre procesos.

3.4.4 Servicios de seguridad

En los sistemas distribuidos con arquitectura cliente/servidor es esencial disponer de un sólido sistema de seguridad, pues los ataques que se produzcan en cualquier elemento del sistema, estación o enlace, pueden afectar a todo el sistema en su conjunto.

En los sistemas cliente/servidor se accede a los servicios mediante peticiones, que pueden ser de diversos tipos, sean llamadas a procedimientos remotos o transaccionales. Es, pues, esencial que cada una de las peticiones sea comprobada y autenticada individualmente. Los ataques a la seguridad de los sistemas distribuidos pueden clasificarse en los siguientes grandes grupos:

- Lectura de información en la red.
- Caballo de Troya: el hecho de que un procesador parezca otro para acceder a algún servicio o recurso no autorizado.
- Falsificación de la transmisión de datos.

Para realizar una adecuada protección contra los ataques es necesario proveer al sistema de mecanismos de seguridad, tales como:

- Autenticación de identidades.
- Verificación de autorización, normalmente mediante listas de control de acceso.
- Verificación de la integridad de los datos.
- Privacidad de datos, mediante métodos criptográficos.

No todos los sistemas distribuidos están necesariamente dotados de todos los mecanismos. Así, los métodos criptográficos pueden emplearse únicamente para informaciones críticas como las contraseñas de acceso.

El control de acceso se basa normalmente en la definición de perfiles de usuario. El supervisor define los derechos de acceso a los recursos de la red, entre los que se incluyen el sistema de archivos, las aplicaciones, las impresoras y otros periféricos. Las





posibilidades de acceso se establecen mediante listas de control de acceso[22].

3.5 Programación gráfica

LabVIEW es un entorno de programación gráfico usado por miles de ingenieros e investigadores para desarrollar sistemas sofisticados de medida, pruebas y control usando íconos gráficos e intuitivos y cables que parecen un diagrama de flujo. Ofrece una integración incomparable con miles de dispositivos de hardware y brinda cientos de bibliotecas integradas para análisis avanzado y visualización de datos, todo para crear instrumentación virtual. La plataforma LabVIEW es escalable a través de múltiples objetivos y sistemas operativos, desde su introducción en 1986 se ha vuelto un líder en la industria [72].

Un instrumento virtual es un módulo software que simula el panel frontal del instrumento y, apoyándose en elementos hardwares accesibles por la computadora (tarjetas de adquisición de datos, tarjetas DSP, instrumentos accesibles vía GPIB, VXI, RS-232, USB, ethernet), realiza una serie de medidas como si se tratase de un instrumento real. De este modo, cuando se ejecuta un programa que funciona como instrumento virtual o VI (Virtual Instrument), el usuario o usuaria ve en la pantalla de su computadora un panel cuya función es idéntica a la de un instrumento físico, facilitando la visualización y el control del aparato. A partir de los datos reflejados en el panel frontal, el VI debe actuar recogiendo o generando señales, como lo haría su homólogo físico.

LabVIEW es hoy una plataforma estándar en la industria de pruebas y medidas para el desarrollo de sistemas de prueba y control de instrumentación, en el campo de la automatización industrial para la adquisición de datos, análisis, monitorización y registro, así como para el control y monitorización de procesos, en el área de visión artificial para el desarrollo de sistemas de inspección en producción o laboratorio[33], entre otras.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) es un ambiente de desarrollo basado en el ambiente gráfico de programación **G**; *LabVIEW* está integrado por un amplio conjunto de funciones matemáticas, de procesamiento digital de señales, protocolos de comunicación de hardware como GPIB, VXI, PXI, RS-232, y también por funciones de comunicación con tarjetas de adquisición de datos, entre otras muchas funciones, como bibliotecas para el uso de protocolos y *software* estándar como *TCP/IP*, *UDP*, *DataSocket* y *WEB SERVER activeX*.

3.6 La arquitectura de protocolos TCP/IP

La arquitectura de protocolos TCP/IP es resultado de la investigación y desarrollo llevados a cabo en la red experimental de conmutación de paquetes ARPANET, financiada por la





Agencia de Proyectos de Investigación Avanzada para la Defensa (DARPA, Defense Advanced Research Projects Agency), y se denomina globalmente como la familia de protocolos TCP/IP. Esta familia consiste en una extensa colección de protocolos que se han especificado como estándares de Internet por parte de IAB (*Internet Architecture Board*).

3.6.1 Las capas de TCP/IP

El modelo TCP/IP estructura el problema de la comunicación en cinco capas relativamente independientes entre sí:

- Capa física
- Capa de acceso a la red
- Capa internet
- Capa extremo-a-extremo o de transporte
- Capa de aplicación

La capa física define la interfaz física entre el dispositivo de transmisión de datos (por ejemplo, la estación de trabajo o el conmutador) y el medio de transmisión o red. Esta capa se encarga de la especificación de las características del medio de transmisión, la naturaleza de las señales, la velocidad de datos y cuestiones afines.

La capa de acceso a la red es responsable del intercambio de datos entre el sistema final (servidor, estación de trabajo, etc.) y la red a la cual está conectada. El emisor debe proporcionar a la red la dirección del destino, de tal manera que ésta pueda encaminar los datos hasta el destino apropiado. El emisor puede requerir ciertos servicios que puedan ser proporcionados por el nivel de red, por ejemplo, solicitar una determinada prioridad. El software en particular que se use en esta capa dependerá del tipo de red que se disponga.

Así, se han desarrollado, entre otros, diversos estándares para la conmutación de circuitos, la conmutación de paquetes (por ejemplo, retransmisión de tramas) y para las redes de área local (por ejemplo, Ethernet). Por tanto si tiene sentido separar en una capa diferente todas aquellas funciones que tenga que ver con el acceso a la red. Haciendo esto, el software de comunicaciones situado por encima de la capa de acceso a la red no tendrá que ocuparse de los detalles específicos de la red a utilizar. El software de las capas superiores debería, por tanto, funcionar correctamente con independencia de la red a la que el conmutador está conectado.

Para sistema finales conectados a la misma red, la capa de acceso a la red está relacionada con el acceso y encaminamiento de los datos. En situaciones en las que los dispositivos estén conectados a redes diferentes, se necesitarán una serie de procedimientos que permitan que los datos atraviesen las distintas redes interconectadas. Ésta es la función de la capa internet. El protocolo internet (*IP, Internet Protocol*) se utiliza en esta capa para ofrecer el servicio de encaminamiento a través de varias redes. Este protocolo se





implementa tanto en los sistemas finales como en los enrutadores intermedios. Un enrutador es un procesador que conecta dos redes y cuya función principal es retransmitir datos desde una red a otra siguiendo la ruta adecuada para alcanzar al destino.

Independientemente de la naturaleza de las aplicaciones que estén intercambiando datos, es usual requerir que los datos se intercambien de forma fiable. Esto es, sería deseable asegurar que todos los datos lleguen a la aplicación destino y en el mismo orden en el que fueron enviados. Los mecanismos que proporcionan esta fiabilidad son esencialmente independientes de la naturaleza intrínseca de las aplicaciones. Por tanto, tiene sentido agrupar todos estos mecanismos en una capa común compartida por todas las aplicaciones; esta se denomina capa extremo-a-extremo, o capa de transporte. El protocolo para el control de la transmisión, TCP (*Transmission Control Protocol*) es el más utilizado para proporcionar esta funcionalidad.

Finalmente, la capa de aplicación contiene toda la lógica necesaria para posibilitar las distintas aplicaciones de usuario. Para cada tipo particular de aplicación, como por ejemplo, la transferencia de archivos, se necesitará un módulo bien diferenciado.

3.6.2 TCP y UDP

La mayor parte de aplicaciones que se ejecutan usando la arquitectura TCP/IP tienen como protocolo de transporte TCP. TCP proporciona una conexión fiable para transferir los datos entre las aplicaciones. Una conexión es simplemente una asociación lógica de carácter temporal entre dos entidades de sistemas distintos. Cada PDU de TCP, denominada *segmento TCP*, contiene en la cabecera la identificación de los puertos origen y destino, los cuales corresponden con los puntos de acceso al servicio (SAP) de la arquitectura OSI. Los valores de los puertos identifican a los respectivos usuarios (aplicaciones) de las dos entidades TCP. Una conexión lógica alude a un par de puertos. Durante la conexión, cada entidad seguirá la pista de los segmentos TCP que vengan y vayan hacia la otra entidad, para así regular el flujo de segmentos y recuperar aquellos que se pierden o dañen.

Además del protocolo TCP, la arquitectura TCP/IP usa otro protocolo de transporte: el protocolo de datagramas de usuario, UDP (*User Datagram Protocol*), UDP no garantiza la entrega, la conservación del orden secuencial, ni la protección frente a duplicados. UDP posibilita el envío de mensajes entre aplicaciones con la complejidad mínima. Algunas aplicaciones orientadas a transacciones usan UDP. Un ejemplo es SNMP (*Simple Network Management Protocol*), el protocolo normalizado para la gestión en las redes TCP/IP. Debido a su carácter no orientado a conexión, UDP en realidad tiene poca tarea que hacer. Básicamente, su cometido es añadir a IP la capacidad de identificar los puertos. [56]

3.6.3 TCP/IP en LabVIEW

LabVIEW dispone de una serie de VIs que implementa el protocolo TCP-IP, que aparecen





en la Fig.18. Algunas de las funciones permiten establecer una comunicación TCP (activa o pasiva), enviar y recibir datos y cerrar la conexión. También se dispone de una función para realizar la conversión de la dirección IP en formato string A.B.C.D a su representación en un valor numérico que después utilizan las funciones de envío y recepción.

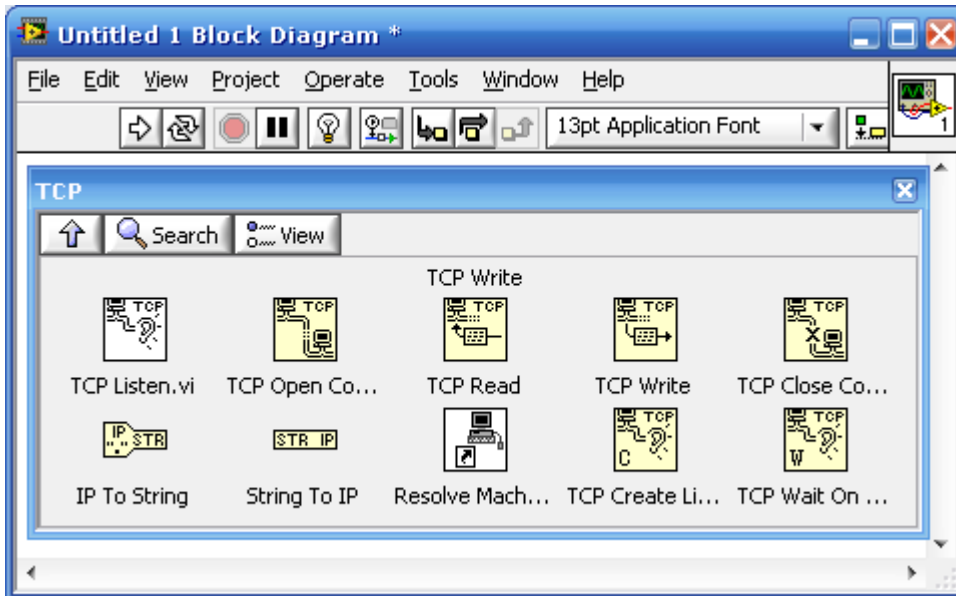


Fig. 18. Interfaz con las funciones de TCP/IP en LabVIEW.

3.6.3.1 Puerto

Es una interfaz entre dos elementos. En TCP/IP, utilizaremos los puertos como enlace entre dos aplicaciones.

3.6.3.2 Dirección IP

La dirección IP es la dirección de nuestro equipo dentro de Internet. La dirección IP está formada por 32 bits agrupados en 4 bloques de un byte cada uno y según las normas del protocolo IP la dirección IP se puede clasificar de cuatro formas diferentes: Clase A, Clase B, Clase C, Clase D.

En la práctica, la notificación utilizada tiene una estructura como la mostrada a continuación: 147.83.4.32, donde los dos primeros bytes definen el dominio (red o subred) y los últimos definen el terminal.

3.6.3.3 Servidor de datos

En este ejemplo se implementa de forma muy simple y muy rápida un servidor de datos.





Podemos ver el diagrama de bloques en la Fig.19.

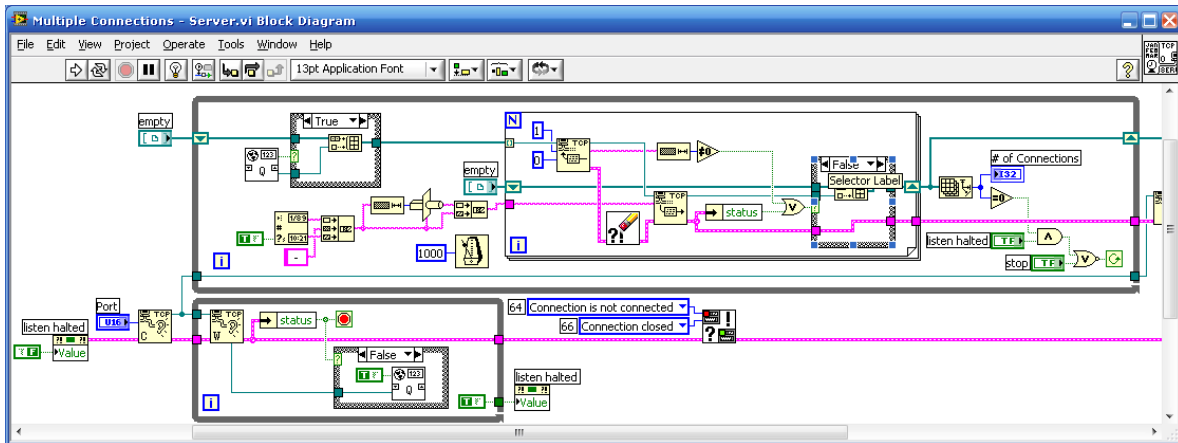


Fig. 19. Diagrama de bloques del servidor de datos en LabVIEW.

El primer paso es establecer una escucha de la línea a la espera de efectuar una conexión TCP (conexión pasiva). Una vez tenemos la conexión TCP el servidor envía una serie de datos al cliente. Para ello, primero le envía la longitud de la trama de datos y luego le envía la trama de datos. Por último lo que hace es cerrar la conexión y volver a comenzar todo el proceso, ponerse a la espera de una nueva conexión.

Los pasos a seguir para generación de un servidor de datos son los siguientes:

1. Escuchar y esperar a que algún cliente pida una conexión.
2. Una vez que se pide conexión, se envían los datos de la siguiente forma:
 - a. Primero se envía la longitud de la trama de datos.
 - b. Segundo se envía la trama de datos.
3. Una vez que se han enviado los datos se cierra la conexión.

3.6.3.4 Cliente de datos

En este ejemplo se implementa de forma simple y rápida un cliente de datos. Como se puede observar en la Fig. 20 lo primero que se hace es abrir la conexión TCP de una dirección y puerto determinado a continuación, lo que se realiza en una lectura de 4 bytes. Estos 4 bytes contienen la longitud de la trama de datos que vamos a recibir, se leen los datos y los mostramos. Una vez mostrados los datos cerramos la conexión.



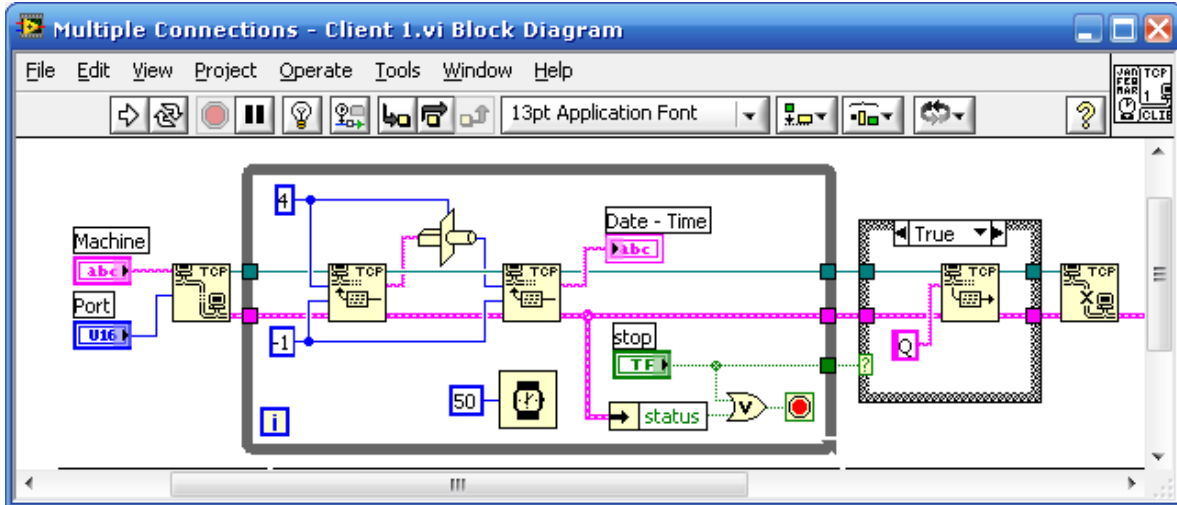


Fig. 20. Diagrama de bloques del cliente de datos en LabVIEW.

Los pasos a seguir para la generación de un cliente de datos son:

1. Abrir la conexión en un puerto y en una dirección del servidor de datos.
2. Hacer una primera lectura de 4 bytes donde se leerá la longitud de la trama de datos.
3. Realizar una segunda lectura de la trama de datos y se leerá tantos bytes como indique la primera lectura.
4. Una vez que se ha finalizado la lectura de datos se cierra la conexión.[33]

3.7 Lenguajes de programación visuales

Visual Basic es un lenguaje de programación de alto nivel que retoma la temprana versión de *DOS* llamada *BASIC*, siglas de *Beginners' Allpurpose Symbolic Instruction Code*. Es un lenguaje de relativamente fácil aprendizaje.

Lenguajes de programación como *Visual Basic*, *Visual C++*, *C++ BUILDER*, *DELPHI*, entre otros, son lenguajes de programación visual y orientados a eventos. En estos lenguajes la programación se hace en un ambiente gráfico. Aquí el usuario puede seleccionar en un cierto objeto aleatoriamente, por eso cada objeto es programado independientemente para responder a cierta acción (evento). Es por esto que un programa hecho en un lenguaje de programación visual está compuesto de muchos subprogramas, y cada uno tiene su código propio.

Measurement Studio es una solución integrada de herramientas de medidas creada específicamente para programadores de Visual Studio .NET. Está diseñado para ingenieros e investigadores que construyen aplicaciones de pruebas, medidas y control en Visual Studio 2010/2008/2005 (Fig. 21). *Measurement Studio* incrementa la productividad del





desarrollador al extender Microsoft .NET Framework y proporcionando automatización, así como controles de formas de Windows y formas Web para Visual Basic .NET y Visual C# [72].

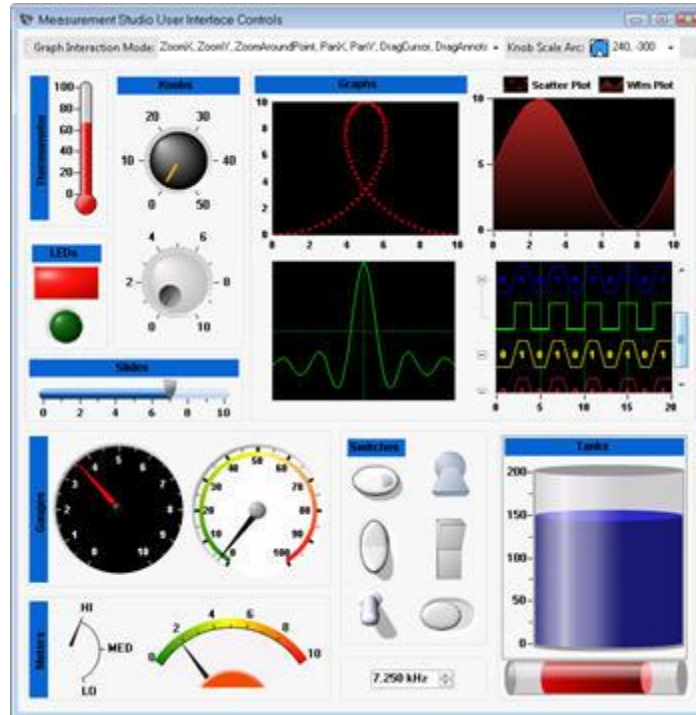


Fig. 21. Interfaz de automatización gráfica de usuario (GUI) para aplicaciones Windows y Web. (Cortesía de National Instruments)





3.8 Propuesta de la tesis

En la presente tesis se desarrolla un sistema de intercambio de datos cifrados entre aplicaciones distribuidas con instrumentación virtual. En la que se comienza por elegir el algoritmo de cifrado de datos más adecuado para ser implementado por primera vez en instrumentación virtual, ya que actualmente no se cuenta con ningún tipo de módulo pre-programado.

Al tener la elección del algoritmo de cifrado de datos, siendo este, DES. Ahora, se realiza la implementación del cifrado de clave simétrica, considerando sus 19 etapas y además del cálculo de las subclaves a partir de los 64 bits de entrada. Se requieren dos etapas para aplicar el algoritmo DES que son: el cifrado de datos y el descifrado, las que se generan en instrumentación virtual.

Se integra un sistema de comunicación basado en el modelo cliente/servidor para comprobar el envío y recepción de datos en un sistema distribuido en TCP/IP con instrumentación virtual.

También, se desarrolla el sistema de adquisición de datos y control supervisorio (SCADA), para que a través de éste, se pueda comprobar que es posible cifrar y descifrar información en un ambiente distribuido con LabVIEW.

En el SCADA se realizan los paneles frontales siguientes: un panel del gráfico histórico de una variable, paneles de gráficos de cuatro variables del sistema, un panel del gráfico de visualización de una variable, un panel para la base de datos de las variables del sistema, así como paneles para modificar valores de la base de datos, paneles de lectura de datos binaria, paneles de recopilación de datos, y por supuesto los dos últimos módulos del sistema, para el intercambio de datos cifrados, como son el cifrado y descifrado puestos en los módulos de ejecución del cliente y también del servidor, para que se ejecuten en el sistema distribuido.





3.9 Conclusiones del capítulo

Se mencionan los componentes de la arquitectura de un SCADA, como son: el hardware, la interfaz hombre-máquina, la unidad central, la unidad remota y el sistema de comunicación del sistema distribuido.

Sobre el modelo cliente/servidor, se consideran los elementos básicos de éste, la infraestructura y los servicios de seguridad que tiene.

Se hace mención sobre los lenguajes de programación gráficos y su utilidad en sistemas de comunicaciones. Además de la arquitectura del protocolo TCP/IP y su interfaz en LabVIEW, ya que este trabajo de tesis utiliza todos estos componentes, en la implementación del criptosistema DES en un sistema distribuido con protocolo TCP/IP utilizando instrumentación virtual.





4. PROGRAMACIÓN DEL CRIPTOSISTEMA

4.1 Introducción

Una de las herramientas automatizada más importante, para la seguridad de redes y comunicaciones es el cifrado simétrico que sólo utiliza una clave para cifrar como para descifrar, resaltando la técnica de cifrado que más se usa, el DES (Estándar de Cifrado de Datos). Considerando esto se generó la implementación en instrumentación virtual del algoritmo DES, el cual requiere una clave de 64 bits para cifrar en bloques de 64 bits, enfatizando que utiliza el mismo algoritmo para cifrar y para descifrar.

Considerando que el algoritmo de cifrado de datos DES, ya ha sido rebasado en lo que a seguridad refiere por otros estándares más confiables. No obstante es de gran utilidad su implementación con la finalidad de introducir la criptografía en sistemas distribuidos mediante instrumentación virtual.

Además se desarrollaron dos paneles, el primero refiere al cifrado de datos y el segundo al descifrado, ya que estos se pueden ejecutar tanto en el servidor como en el cliente o clientes del sistema distribuido.

4.2 Elección del algoritmo de clave simétrica, DES

Las medidas de seguridad en red son necesarias para proteger los datos durante su transmisión y poder garantizar que los datos enviados sean auténticos. Una de estas medidas de seguridad es la herramienta de cifrado de datos con clave simétrica utilizada en la presente tesis dentro del sistema de comunicación descrito e implementado con instrumentación virtual, por lo que se considera necesario describir algunas de las condiciones de la elección del algoritmo de cifrado de datos:

Primero, se verán los motivos del *¿Por qué se elige un algoritmo de clave privada o simétrico sobre uno de clave pública o asimétrico?*





- En la sección 2.4 se puede observar que el cifrado de clave privada utiliza la misma clave para cifrar como para descifrar.
- Uno de los requisitos para aplicar cifrado de datos con clave privada es tener como mínimo:
 - Un emisor y un receptor, deben obtener las copias de la clave secreta de una forma segura y deben mantenerla en secreto. Si alguien puede descubrir la clave y conoce el algoritmo, toda comunicación que utilice esta clave puede ser leída.
 - El algoritmo debe cumplir la condición de que aunque un intruso conozca el algoritmo y tenga acceso a uno o más mensajes cifrados, sea incapaz de descifrar el mensaje o averiguar la clave.
- Empleando cifrado simétrico (utiliza sólo una clave) se consigue privacidad sin perder velocidad en la transferencia de datos.
- Es posible realizar la autenticación mediante el uso de cifrado simétrico. Se supone que solamente el emisor y el receptor comparten una clave, entonces solamente el emisor genuino sería capaz de cifrar un mensaje satisfactoriamente para el otro participante.
- El cifrado de clave privada es más rápido por el tipo de operaciones que ejecuta en comparación con un cifrado de clave pública.

Ahora, se verán los motivos del *¿Por qué se elige el algoritmo de clave simétrica estándar de cifrado de datos (DES)?*

- Normalmente, es más rápido implementar un algoritmo de clave simétrica como DES, pues se utiliza el mismo para cifrar, que para descifrar. Además de que el algoritmo utiliza operaciones binarias, que también permite superar en rapidez a un algoritmo de clave pública.
- De acuerdo a la sección 2.2.2 dentro de los aspectos de robustez sobre el algoritmo DES, se considera que hace uso de una clave de 56 bits, hay 2^{56} claves posibles. Por lo que una máquina que realice un cifrado DES por microsegundo tardaría mucho en romper el cifrado (Tabla 11.). Aunque es solo cuestión de tiempo que la velocidad de procesamiento computacional dejara a DES obsoleto. Sin embargo, para los fines que se tienen en este trabajo el haberlo utilizado es un ejercicio para introducir la criptografía en sistemas distribuidos mediante instrumentación virtual





- Además, la patente del DES está vencida de acuerdo a la Tabla 14 y la sección 2.8.
- Al aplicar confusión (sustitución) y difusión (permutaciones o transposiciones), permite aplicar la teoría básica de criptografía en un sistema de comunicación y evita tener redundancia en el criptosistema. DES se basa en una combinación de confusión y difusión.
- Fue un estándar de cifrado de datos, completamente especificado y publicado en FIPS PUB 46.
- El cifrado simétrico y el cifrado asimétrico se suelen combinar en aplicaciones de red seguras. Por lo que un cifrado simétrico como DES, puede ser parte de un criptosistema más complejo, debido a que puede formar parte de esta serie de combinaciones. Como se puede ver a continuación DES es implementado en instrumentación virtual y con esto se abre una serie de posibilidad para realizar otras implementaciones ya que no existen módulos como este, que permitan hacer las combinaciones del cifrado simétrico y asimétrico con lenguaje G.

4.3 Panel del cifrado de datos

Este criptosistema es desarrollado para tener de forma sencilla y práctica, una herramienta para proteger mejor los datos que son enviados a través del canal dentro de un sistema de monitoreo y adquisición de datos con lenguaje G. Entonces, considerando un sistema SCADA se puede manejar un conjunto de variables de acuerdo a los parámetros adquiridos para seleccionar aquellas *variables muestra* del cifrado de datos llamadas *Variables a Cifrar*.

Las *Variables a Cifrar* son parte del sistema SCADA (*AParAlarma, Historico infinito VceUI, VceUI, VceUC, ParConUI*) y permiten observar los tipos de datos que se pueden cifrar y enviar a través de la red de forma segura; en este caso se seleccionan estas variables, es posible cambiarlas de acuerdo al sistema distribuido que se esté manejando, ya que este sistema nos permite ver el mecanismo de cifrado en ejecución con datos adquiridos en campo.

También se puede observar que se pide introducir la *clave* con la cual se cifrará dentro del criptosistema y se mostrará a su salida los *mensajes cifrados en bloques* y *el mensaje cifrado*, ambos tomados de la misma variable y mostrados de forma distinta, el primero se muestra en bloques de 8 bytes cada uno ya que el criptosistema que se presenta genera bloques cifrados de 64 bits y el segundo muestra el mensaje simple en una cadena. Como se puede observar en la Fig.22.



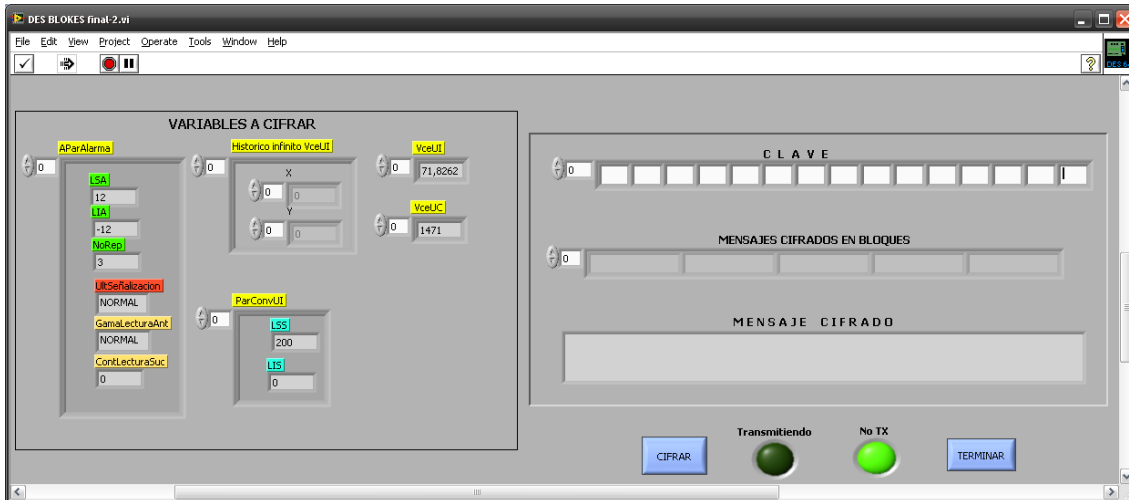


Fig. 22. Panel frontal del cifrado de datos

Además de tener botones para *CIFRAR* el cual comenzará el proceso de cifrado de datos de las *VARIABLES A CIFRAR*, transmitirá al final de este proceso y por lo tanto encenderá el botón de *transmitiendo* que antes sólo encendera el botón de *NoTX* (No transmitiendo). El botón de *TERMINAR* dará por finalizado el instrumento virtual que realiza la parte del cifrado de datos.

4.4 Panel frontal del descifrado de datos

Un criptosistema lo conforman dos partes la primera es el cifrado de datos y la segunda es el descifrado de datos por lo que se puede ver ahora la pantalla generada cuando se elige *DESCIFRAR* en el panel principal del *SCADASERVER* o *SCADACLIENTE*.

La pantalla que se muestra a continuación (Fig.23), tiene dos botones azules del lado izquierdo *DESCIFRAR* y *FINALIZAR*. Al elegir el primer botón el sistema comienza a procesar la instrucción de modo que deberá tener la *CLAVE*, estar recibiendo los *MENSAJES CIFRADOS* y así empieza a mostrar todos los *MENSAJES DESCIFRADOS* para después obtener la descompactación de los mismos y por último serán mostradas las *VARIABLES DESCIFRADAS*. El sistema de entrada recibió las *VARIABLES A CIFRAR* y sólo aquí reciben el nombre de *VARIABLES DESCIFRADAS* por lo que deben coincidir pues son las mismas. Sí se elige la opción de *FINALIZAR*, entonces se desconecta del panel de descifrado de datos. Observamos que *VARIABLES A CIFRAR* y *VARIABLES DESCIFRADAS* son las mismas variables tomadas de la base de datos del SCADA.



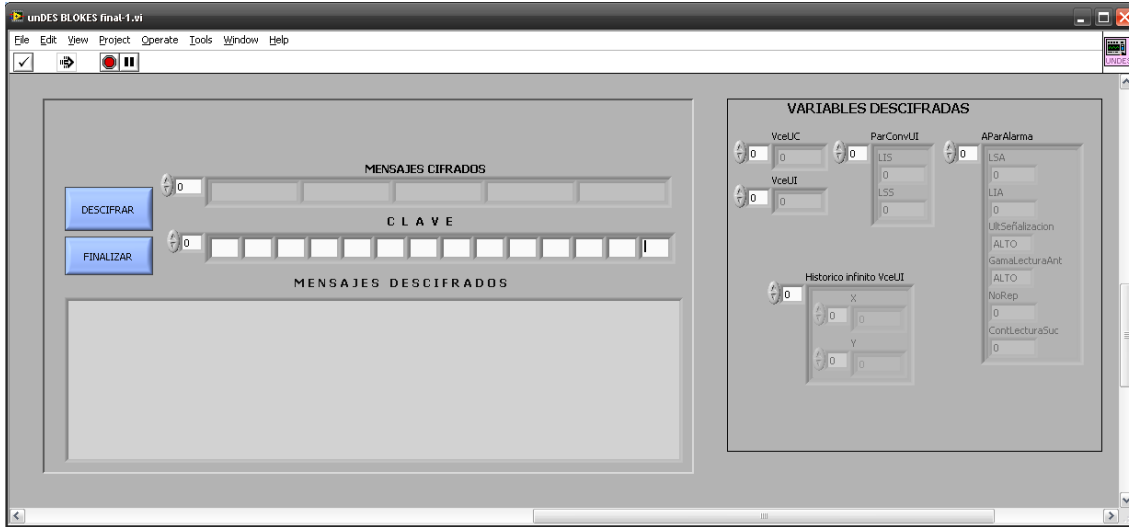


Fig.23. Panel frontal del descifrado de datos.

Dentro de la realización del criptosistema se obtuvo el diagrama jerárquico mostrado en la Fig.24, donde se puede observar que se generaron varios subVIs (ver apéndice) en los que se programó todo el sistema que refiere al desarrollo del cifrado de datos, se consideran las 19 etapas del algoritmo DES para generar más de un módulo y así se hacen otros más pequeños que realicen subprocesos del criptosistema. También pueden ser utilizados en el descifrado. Para entender mejor el funcionamiento de este criptosistemas se tiene a continuación los diagramas de flujo.

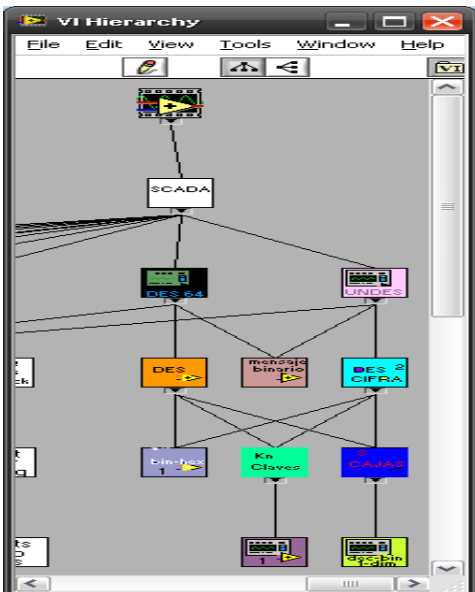


Fig.24. Muestra el VI de la jerarquía del sistema de cifrado de datos





4.5 Descripción de la programación del criptosistema

Da comienzo, la parte del cifrado de datos, por lo que se muestra el diagrama de flujo del algoritmo DES (Fig. 25), que tiene 19 etapas, primero se le aplica una permutación inicial, se dividen en dos partes para aplicar las 16 rondas de las redes de Feistel, las cuales se pueden observar con un bloque llamado Función X (Fig.26) se juntan todos los bits, se realiza un intercambio de bits entre estos y por último se les aplica una permutación final.



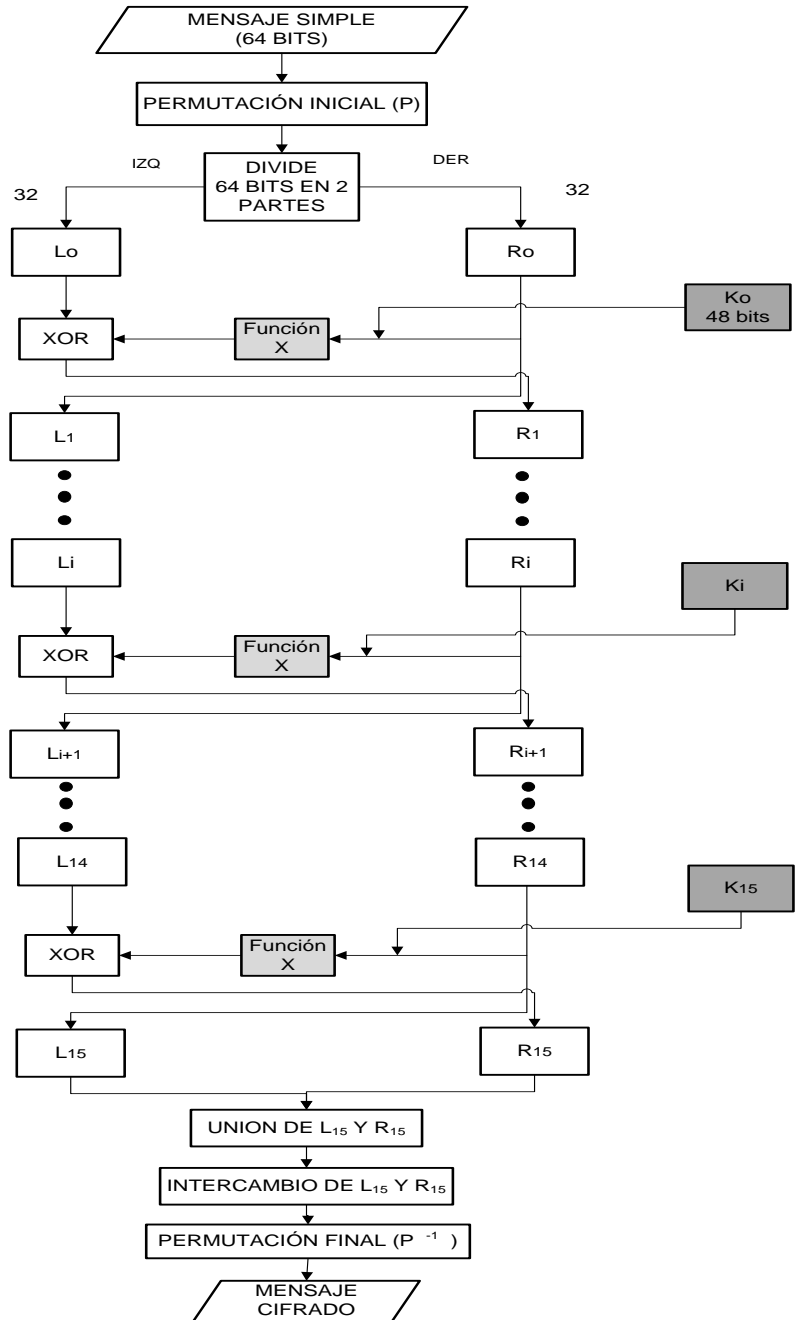


Fig.25. Diagrama de flujo del criptosistema DES

Se tiene la clave de 64 bits a la entrada, se aplica una permutación PC-1 (Tabla 4), luego se procede a dividir en 2 partes del mismo tamaño cada una de 28 bits, a cada parte se le aplica una rotación de un bit o 2 de acuerdo a la iteración dentro de las rondas de las redes de Feistel(Tabla 6), después se juntan y se aplica una permutación PC-2(Tabla 5), para generar una clave distinta (Fig. 26).



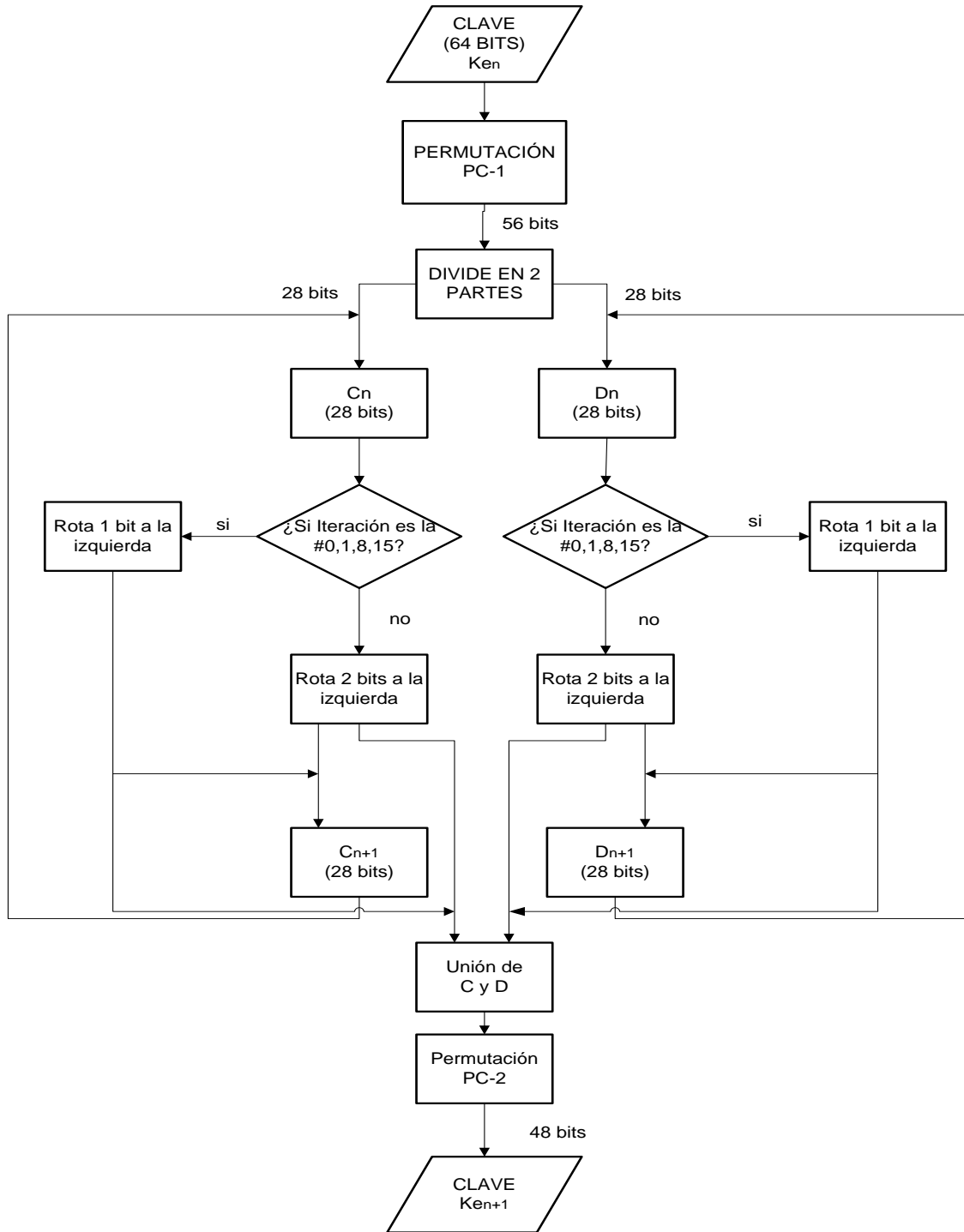


Fig.26. Diagrama de flujo de la obtención de las claves para el DES.

Parte importante del algoritmo es la aplicación de las redes de Feistel (Fig.27) que se explica considerando primero, la *permutación expansiva*, continuando con la *función XOR*





también llamada OR-exclusivo que se implementa entre K_i , el resultado de la *permutación expansiva* para después dividir, aplicar las *S-Cajas* de DES, elegir la S-Caja correspondiente, reducir de 6 bits a 4 bits y desarrollar dicha sustitución de S-Cajas. Por último juntar los bits y dar la *permutación P* (Tabla 8).

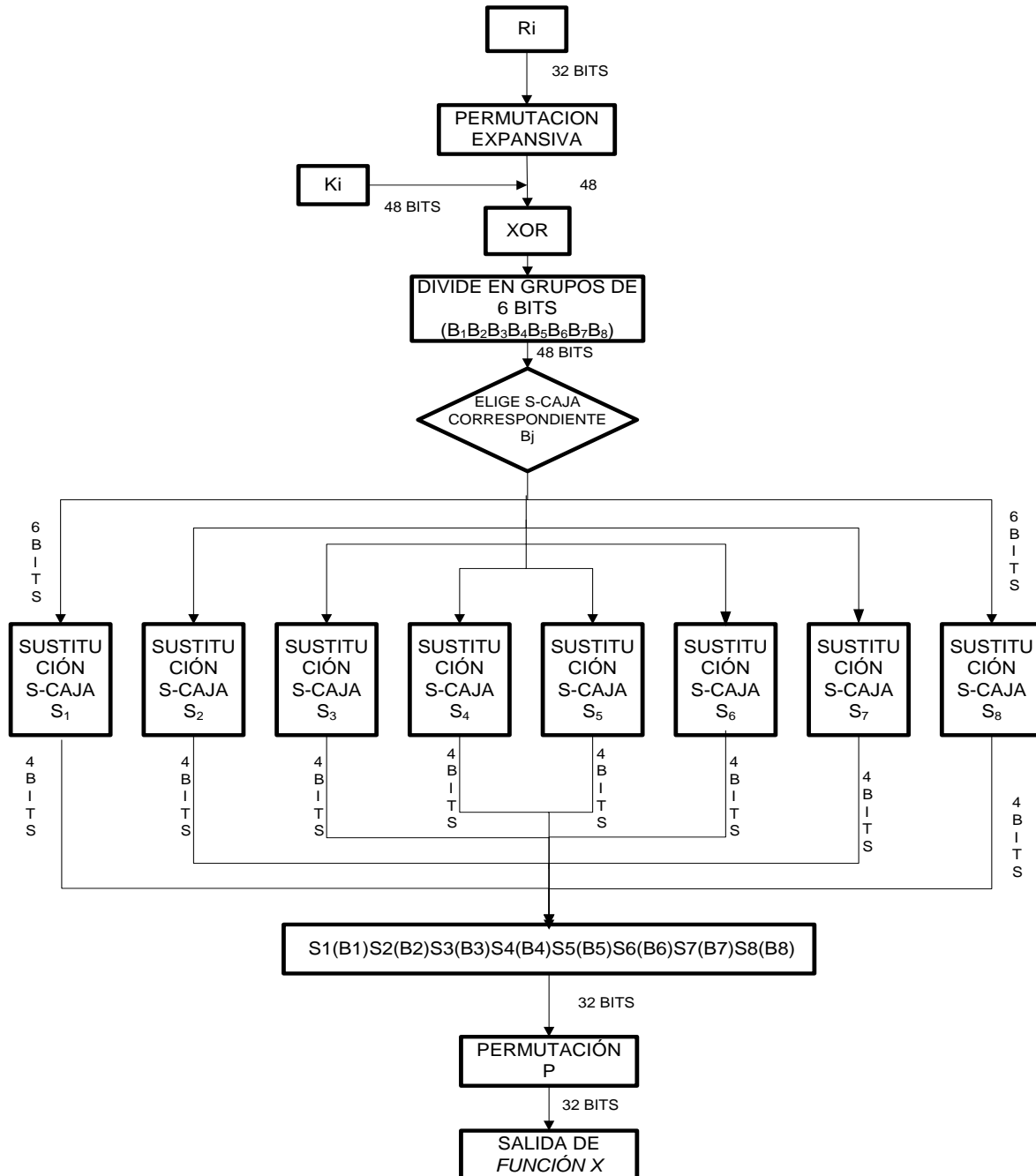


Fig.27. Rondas de las redes de Feistel del DES.





4.6 Conclusiones del capítulo

A lo largo de este capítulo se puede ver la implementación del criptosistema DES, en la parte de *CIFRADO* y *DESCIFRADO de datos*. En la parte del cifrado se explican las 19 etapas que tiene el algoritmo desde un *MENSAJE SIMPLE* que en este caso resultan ser *VARIABLES A CIFRAR* de un sistema y que permite mostrar el funcionamiento del algoritmo dentro de un sistema de adquisición de datos, hasta el descifrado de las *VARIABLES DESCIFRADAS*.

Se puede ver cómo el algoritmo DES funciona a detalle, primero se aplica al mensaje una permutación inicial, después las 16 rondas de las redes de Feistel, la penúltima etapa que consiste en un intercambio de bits y por último la permutación inicial inversa.

Además de obtener la clave, las 16 subclaves para cada una de las rondas de las redes de Feistel.

Por supuesto no se puede omitir el descifrado considerando ingresar en orden inverso las subclaves.





5. INSERCIÓN DEL CRIPTOSISTEMA EN EL MÓDULO DEL SISTEMA SCADA

5.1 Introducción

Un sistema SCADA permite que sin importar la experiencia, los ingenieros y científicos pueden establecer una interfaz rápidamente y de manera rentable con hardware y control, además de poder analizar datos, compartir resultados y distribuir sistemas.

Un sistema SCADA permite conocer los diferentes estados de las variables que son parte de un proceso supervisado; su configuración se realiza mediante las aplicaciones HMI/SCADA, que son programas orientados a objetos y eventos con un amplio manejo de información. Al usar estos sistemas, lo único que se hace es configurar el sistema, sin realizar ningún tipo de desarrollo, y generalmente hay una falta de flexibilidad o un exceso de consideraciones; aunque no se requiere de ningún conocimiento de programación para utilizarlos, además de tener un costo muy elevado. Por otro lado, el desarrollar un sistema SCADA en un entorno de programación gráfico, se tiene la flexibilidad necesaria dependiendo de las necesidades y los algoritmos que se requieran dentro del sistema sin exceso de opciones.

A continuación se tiene la etapa de la integración del cifrado y descifrado de datos para incrustarlo directamente sobre un sistema de comunicación con el modelo cliente/servidor y TCP/IP, así poder conjuntar todo en el módulo del sistema SCADA bajo la plataforma del lenguaje G.

Las funciones que se tomaron del entorno de desarrollo de programación conservan su nombre en inglés y así se presentan en los diagramas de flujo de programación.





5.2 Implementación del sistema cliente/servidor

5.2.1 Descripción de la implementación del cliente

Al realizar un enlace en un cliente de datos es necesario tener una dirección IP y un número de puerto con la que se abre una conexión TCP-IP (Fig. 28), se leen datos, se consideran posibles errores en la comunicación, y una vez que el sistema se encuentra funcionando se envían los mensajes cifrados en bloques como se puede ver en el diagrama de flujo del cliente (Fig. 29). Los textos en inglés se refieren a la función por su nombre, utilizada en el programa.

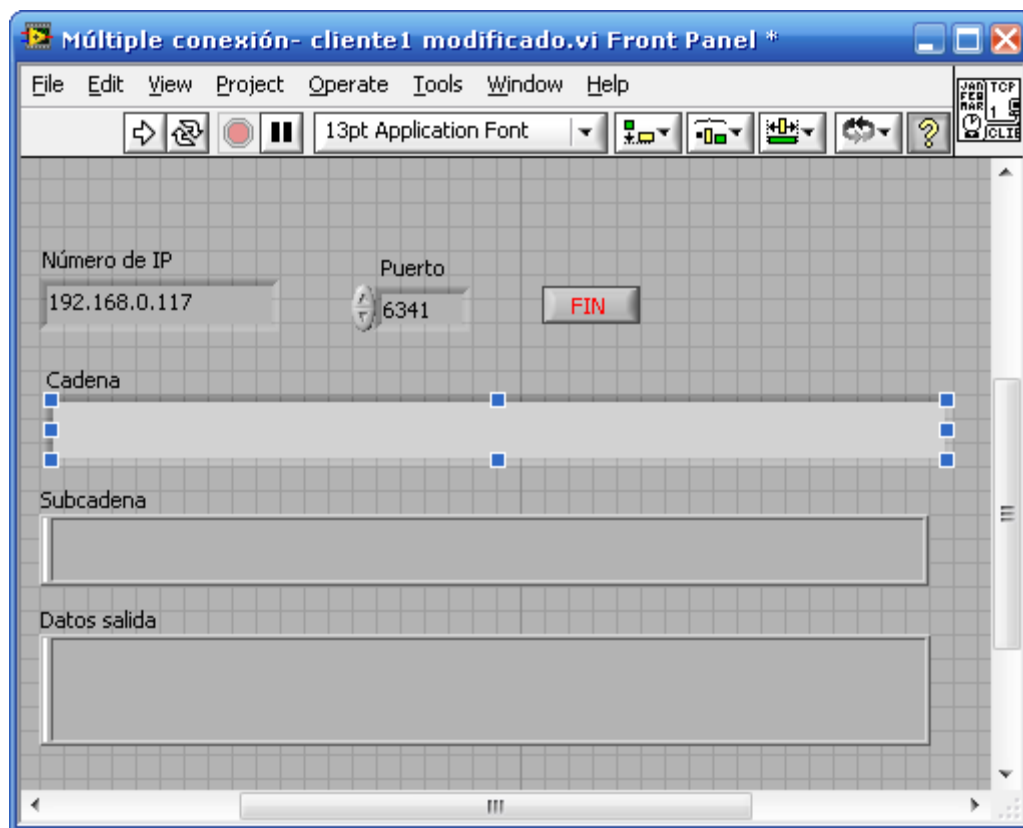


Fig. 28. Panel frontal de la conexión del cliente.



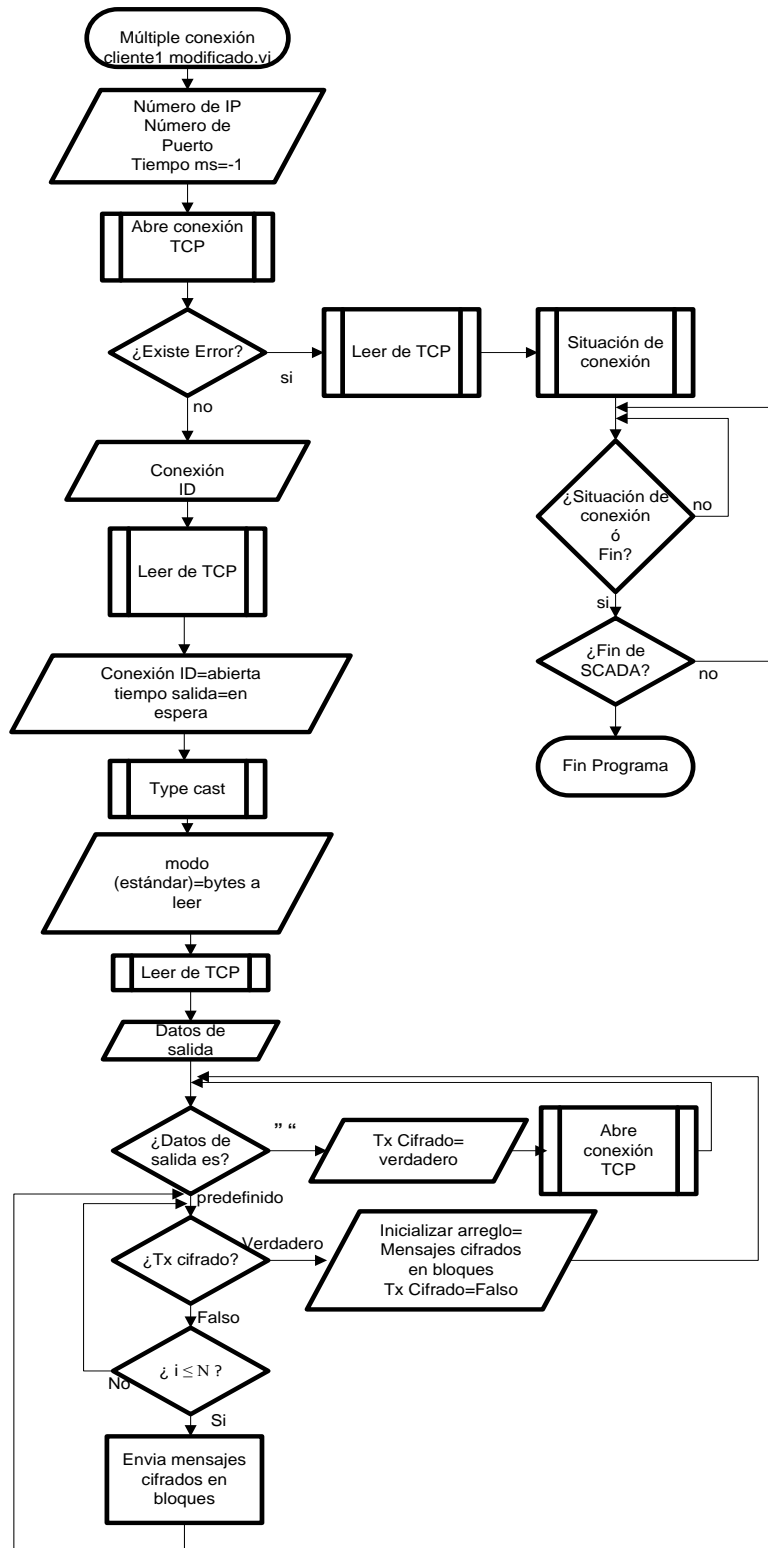


Fig.29. Diagrama de flujo de la conexión del cliente.





A continuación se muestra el panel frontal principal que se utiliza al ejecutar el cliente (Fig. 30).

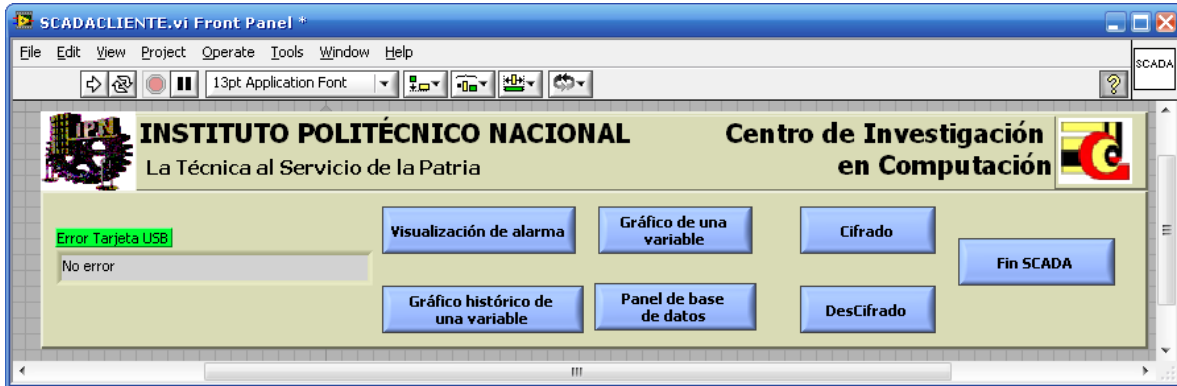


Fig. 30. Panel frontal del SCADA como cliente con los módulos principales.

Ahora se tiene siempre en ejecución la conexión del cliente hacia el servidor ya que es una de las funciones del panel anterior (Fig. 30), además se encuentra a la espera de cualquiera de las opciones: *Panel de base de datos*, *Visualización de alarma*, *Gráfico de una variable*, *Gráfico histórico de una variable*, *Cifrado*, *Descifrado* y *Fin SCADA* e indicador de la tarjeta *Error Tarjeta USB*, enseguida inicializa la base de datos, lee los datos binarios y almacena datos como se muestra el diagrama de flujo de la Fig.31.



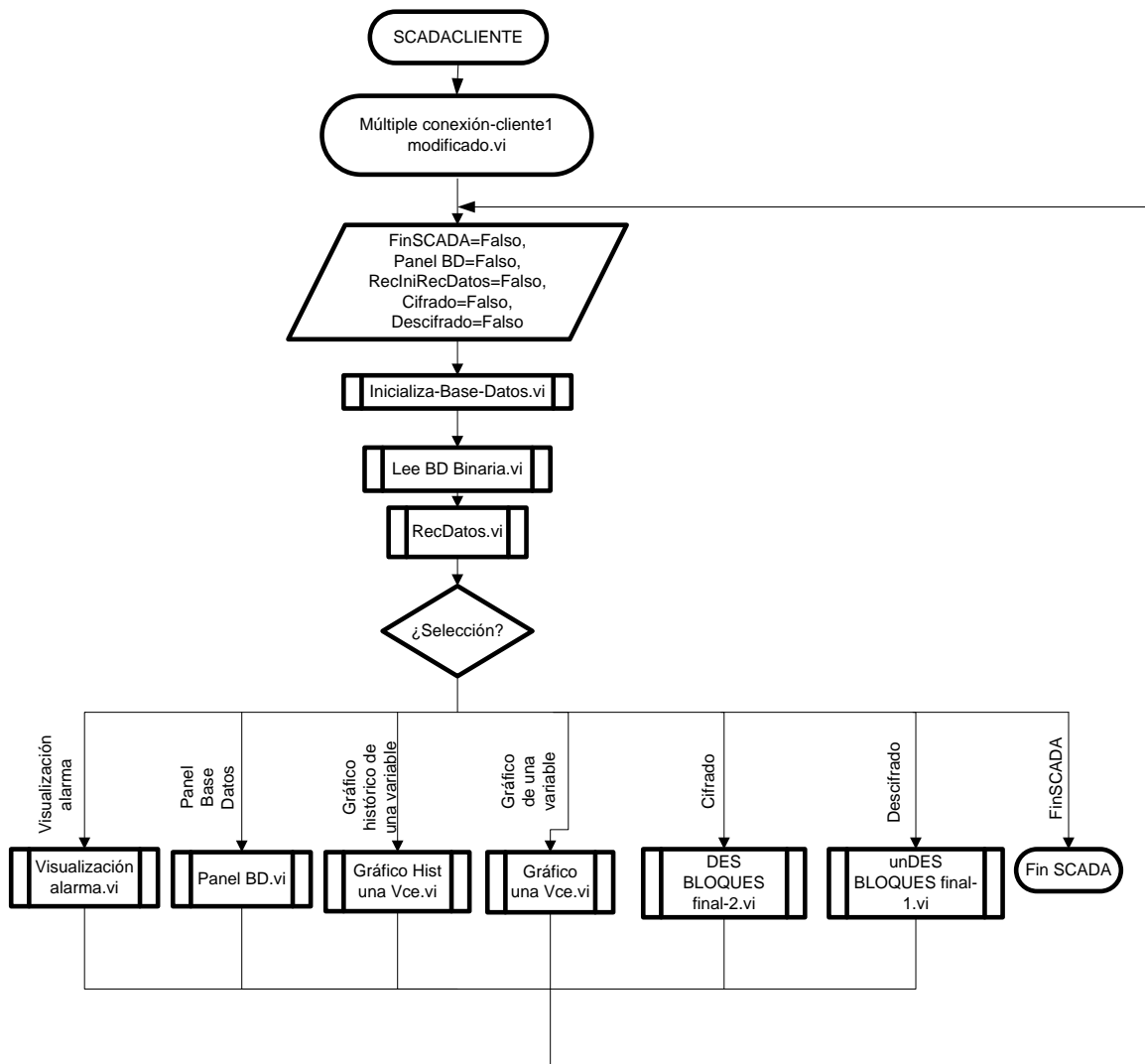


Fig. 31. Diagrama de flujo del SCADA como cliente.

5.2.2 Implementación del modelo servidor

Continuando con el desarrollo de este capítulo, se puede ver como se generó en el SCADA el modelo del servidor con el dato del número de puerto y el sistema comienza a realizar las operaciones (Fig. 32).



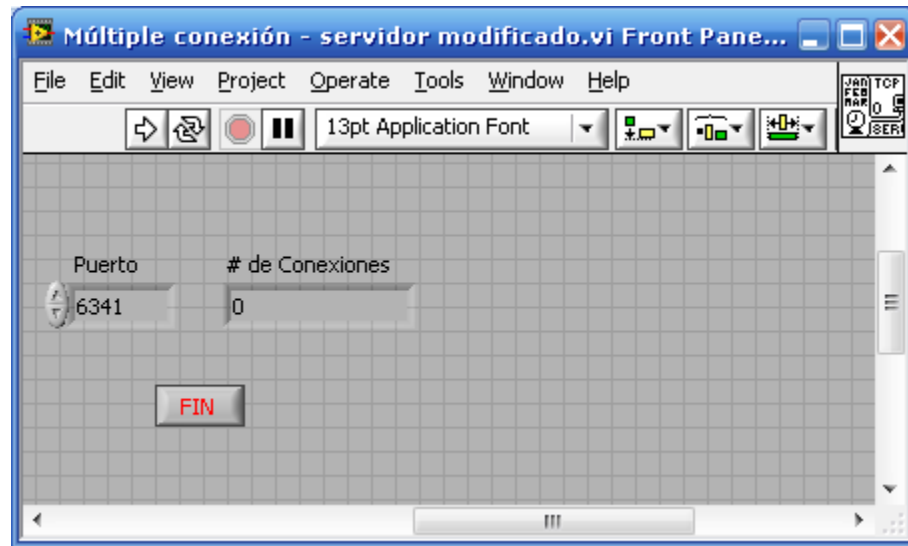


Fig. 32. Panel frontal de la conexión del servidor.

En el caso del servidor se establece para recibir las conexiones de los clientes. Se muestra el diagrama de flujo, en el que se registran las conexiones, el puerto de servicio que tiene activado para la transmisión de datos. Este panel es ejecutado desde el arranque del sistema y además se encuentra en espera de cualquier petición del cliente o clientes que necesiten la comunicación para brindar el servicio (Fig. 33). Los textos en inglés se refieren a la función por su nombre, utilizada en el programa.



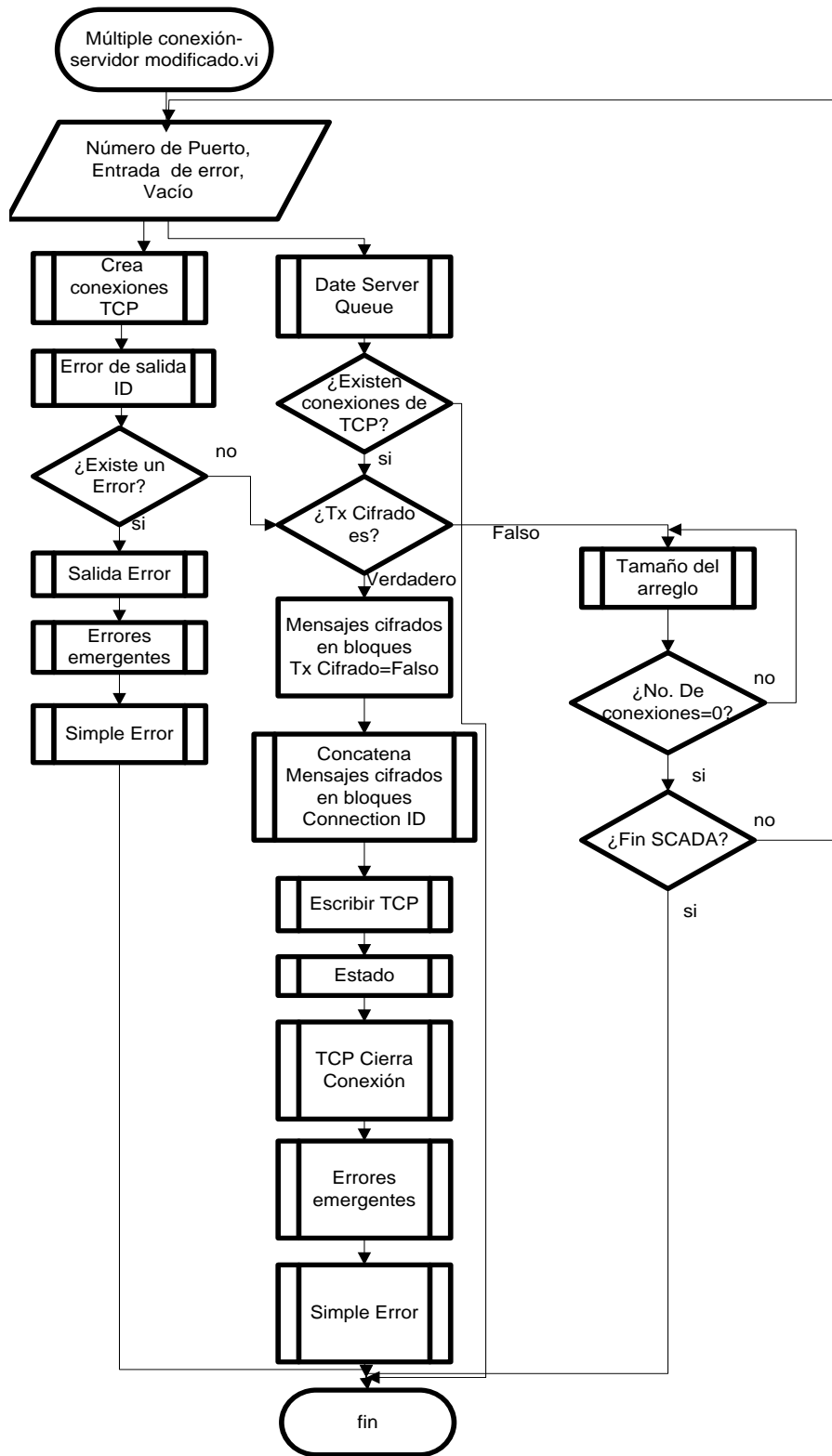


Fig. 33. Diagrama de flujo del SCADA como servidor





Se muestran en la Fig. 34. el panel frontal del operador, cuenta con siete botones: *Panel de base de datos*, *Visualización de alarma*, *Gráfico de una variable*, *Gráfico histórico de una variable*, *Cifrado*, *Descifrado* y *Fin SCADA* e indicador de la tarjeta *Error Tarjeta USB* que da la adquisición de datos por medio de la tarjeta de adquisición que se conecta al *USB* (*Bus Serial Universal*).

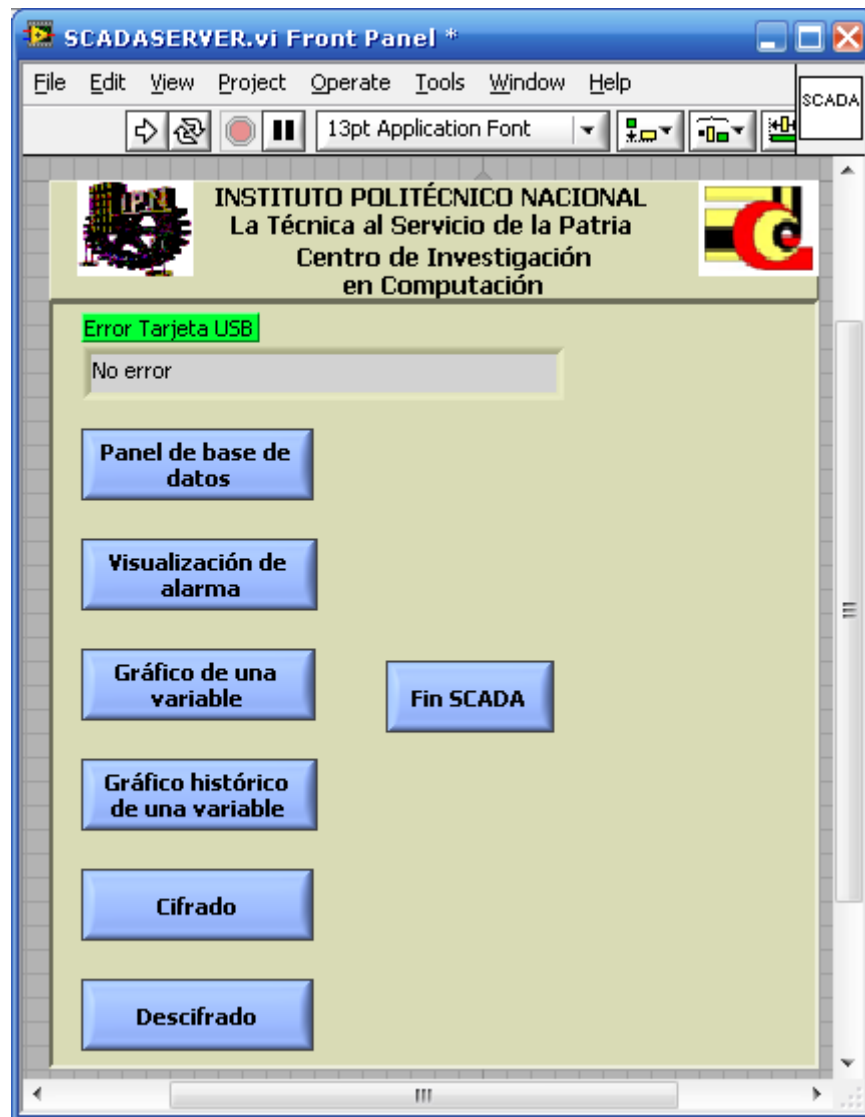


Fig.34. Panel frontal del SCADA como servidor con los módulos principales.

Se hizo el diseño del sistema considerando que el panel frontal tiene varias opciones, este sistema consta de los siguientes módulos de interfaz principales:

- Panel de la Base de Datos.





- Gráfico para visualización de señales.
- Gráfico para visualización de histórico.
- Módulo de recolección de datos.
- Módulo de comunicación de datos.
- Módulo de cifrado de datos.
- Módulo de descifrado de datos.

En el siguiente diagrama de flujo se pueden ver los módulos de interfaz que establecen la conexión de la función: *Múltiple conexión-servidor modificado.vi*, se describe en la Fig. 33. Se mantiene en ejecución y recibe la selección de los módulos principales antes mencionados y continúa hasta que es seleccionada una opción (Fig.35) (Ver *programación mediante diagrama a bloques* en el apéndice).

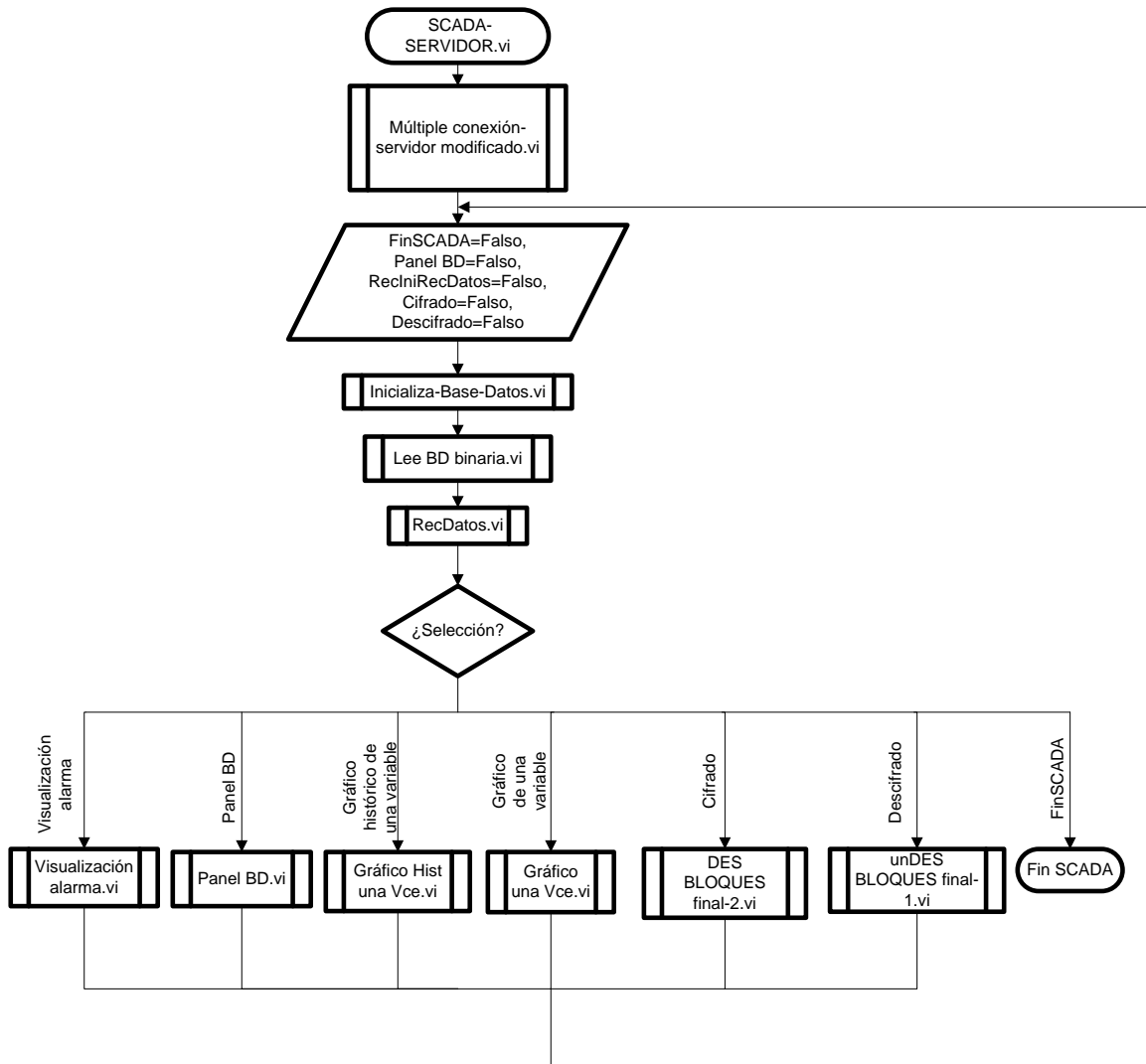


Fig.35. Diagrama de flujo del servidor





La jerarquía de los módulos principales dentro del SCADA, se muestran en la Fig. 36, la figura se generó dentro del lenguaje de instrumentación virtual.

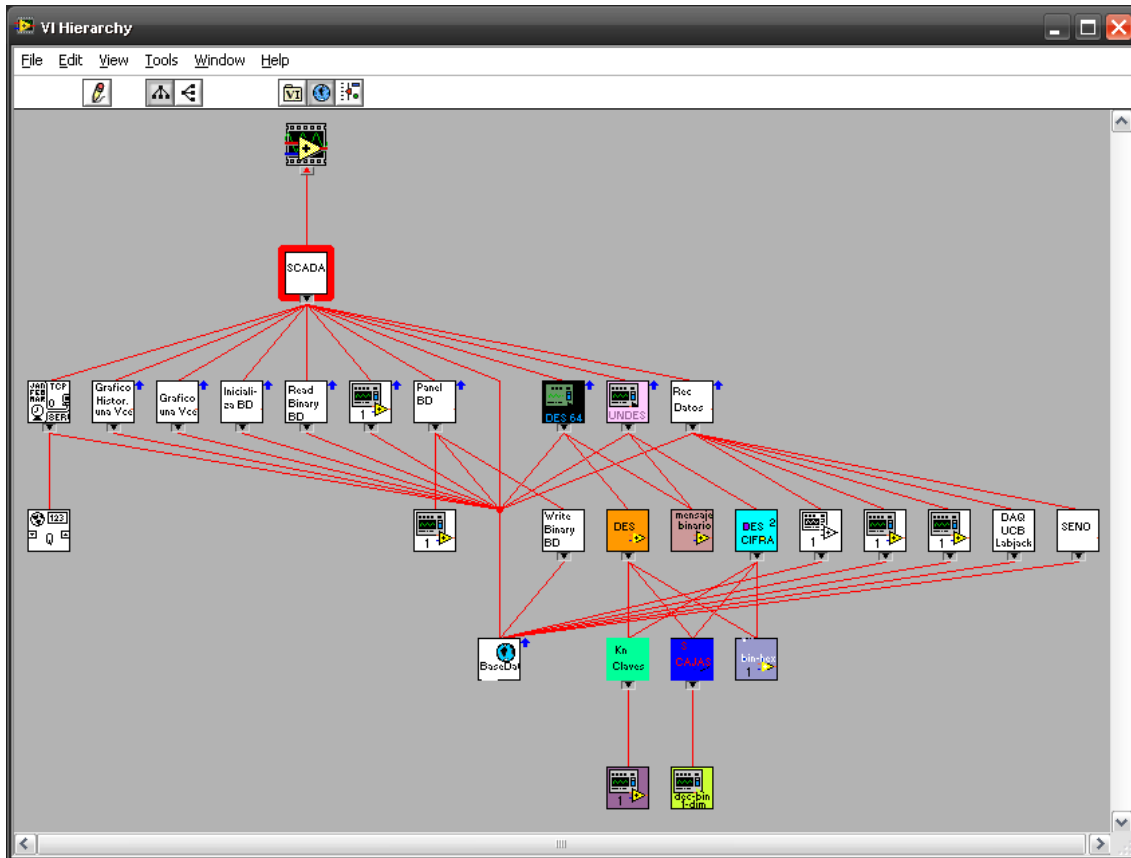


Fig. 36. Jerarquía de los módulos principales

Se explica a continuación cada uno de los módulos principales del SCADA.

5.3 Panel de cambio de parámetros de la base de datos

En el panel de cambio de parámetros de la base de datos (Fig. 37) se muestran las opciones de *Entrada de* con las opciones siguientes: GEN, TCP/IP, USB, PCI y *Reprod*. Están las *Vce conectadas* se definen el número total de variables que se van a monitorear, *Período de Muestreo de las VCE* se define el período de muestreo, *Parámetros para la conversión a UI* se dan los parámetros para la conversión a unidades de ingeniería, *Unidades de Medición*, *AParAlarma* se definen los límites de la alarma por limite de repeticiones y el número de las mismas, *Maximas Vce* indica el número máximo de variables a monitorear, *Base del Reloj* y los botones de *Actualizar*, *Guardar* y *Salir* para todos los parámetros de la base de datos. Lo anteriormente descrito se realiza con el diagrama de flujo correspondiente a la Fig.40 (Ver programación mediante diagrama a bloques en el apéndice.)



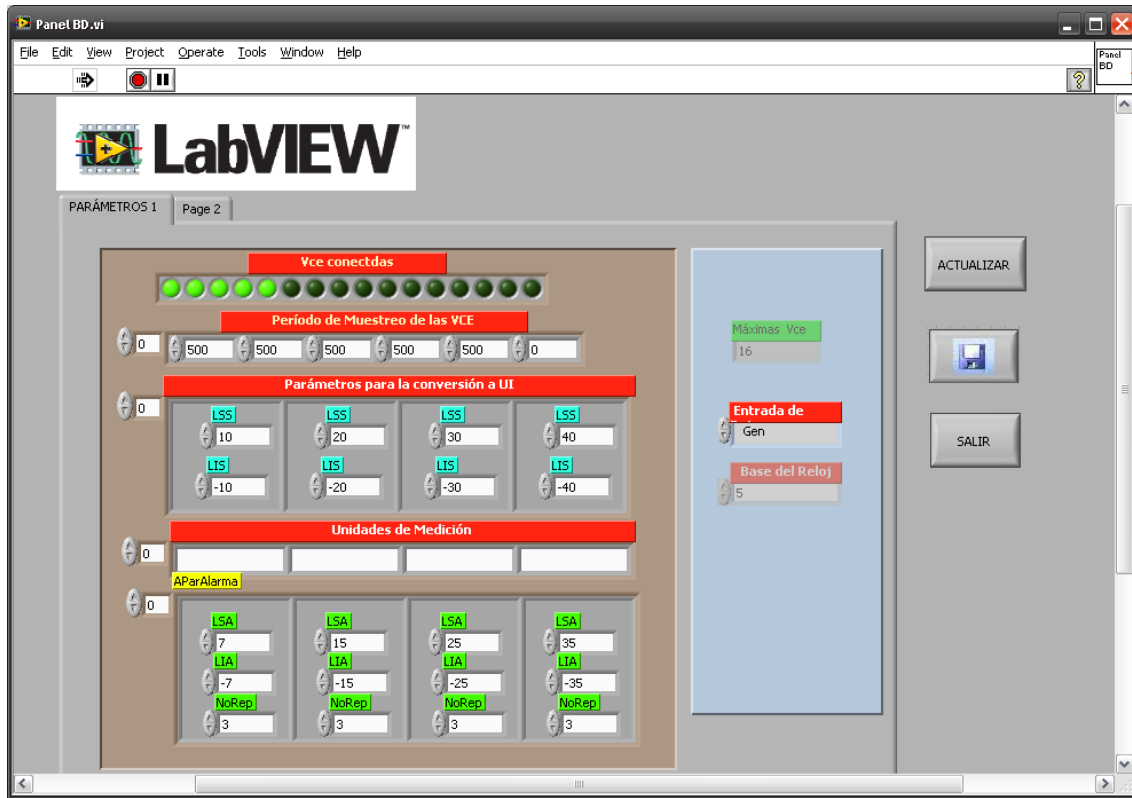


Fig. 37. Pantalla del panel de cambios de parámetros de la base de datos.

Si se desea guardar los parámetros de la base de datos aparecerá la siguiente pantalla Fig. 38.

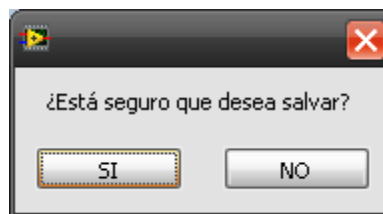


Fig. 38. Pantalla para decidir guardar los cambios de los parámetros del panel de la base de datos.





A continuación se muestra la pantalla (Fig. 39.) para guardar el archivo de la base de datos.

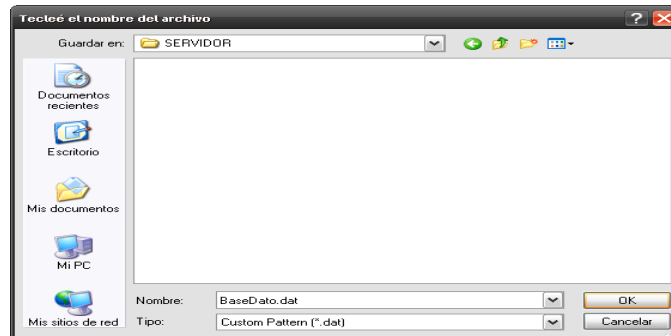


Fig. 39. Pantalla para teclear el nombre con el que se almacenará el archivo de la base de datos.

Se muestra el diagrama de flujo utilizado para realizar las operaciones anteriores, referidas al panel de cambio de parámetros de la base de datos (Fig. 40).



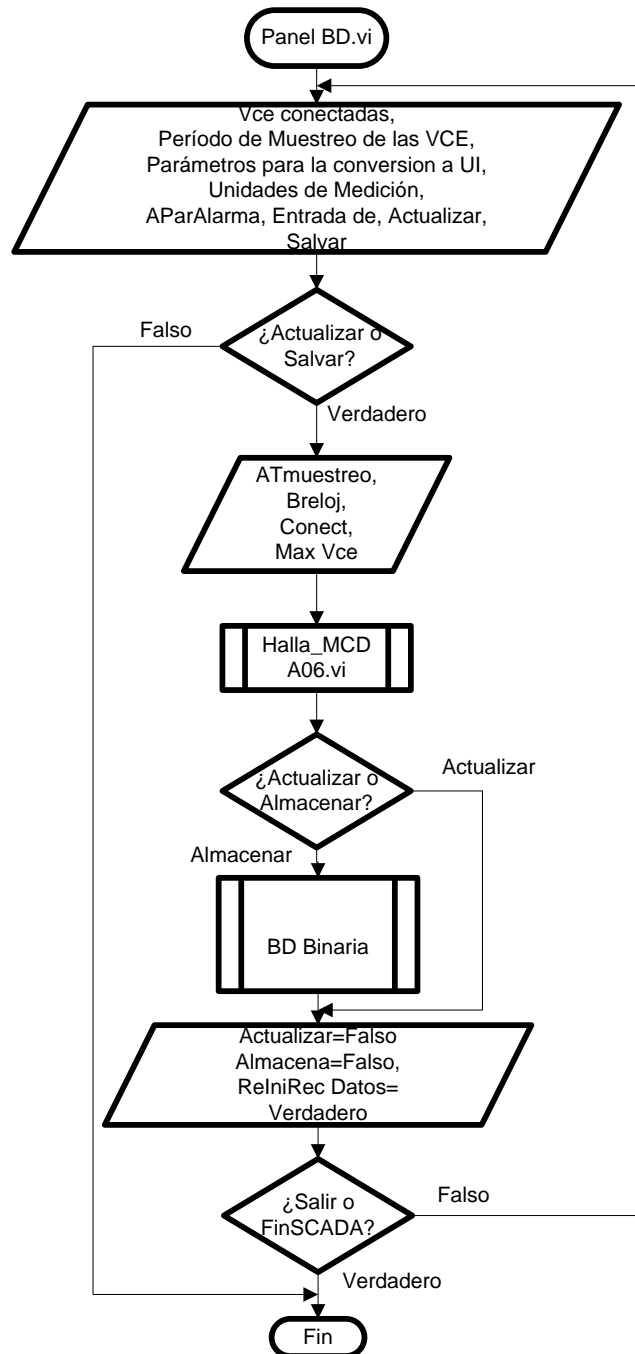


Fig.40. Diagrama de flujo del panel de cambio de parámetros de la base de datos

5.3.1 Panel de la base de datos del sistema

Se muestra a continuación la base de datos del SCADA (Fig. 41), se tienen todas las variables del sistema, como son: *MaxVce*, *BReloj*, *MCD*, *Conect*, *VceUC*, *VceUI*,





ATmuestreo, ParConvUI, UM, MENSAJES CIFRADOS EN BLOQUES, EntDatos, MaxUC, Contador de TR Global, FinSCADA, Error Tarjeta USB, Histórico infinito VCEUI, Tx Cifrado, Tipo String, Cluster y ReIniRecDatos.

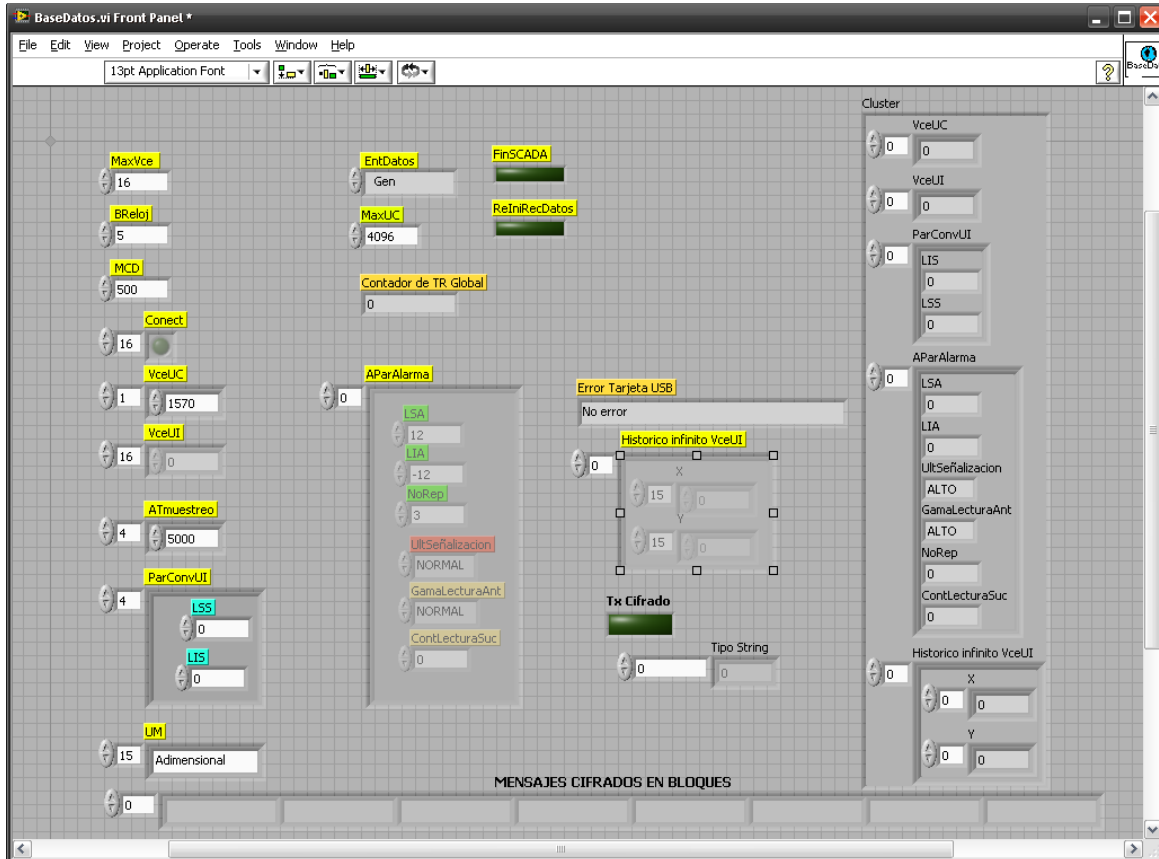


Fig. 41. Base de datos para las variables del sistema.

5.3.2 Panel de gráfico de la variable Vce.

En este gráfico (Fig. 42) se puede apreciar el comportamiento de la variable que se está adquiriendo. En el panel se puede seleccionar la variable Vce que se desea ver graficada y además en cualquier momento se cierra el panel con el botón de FIN.



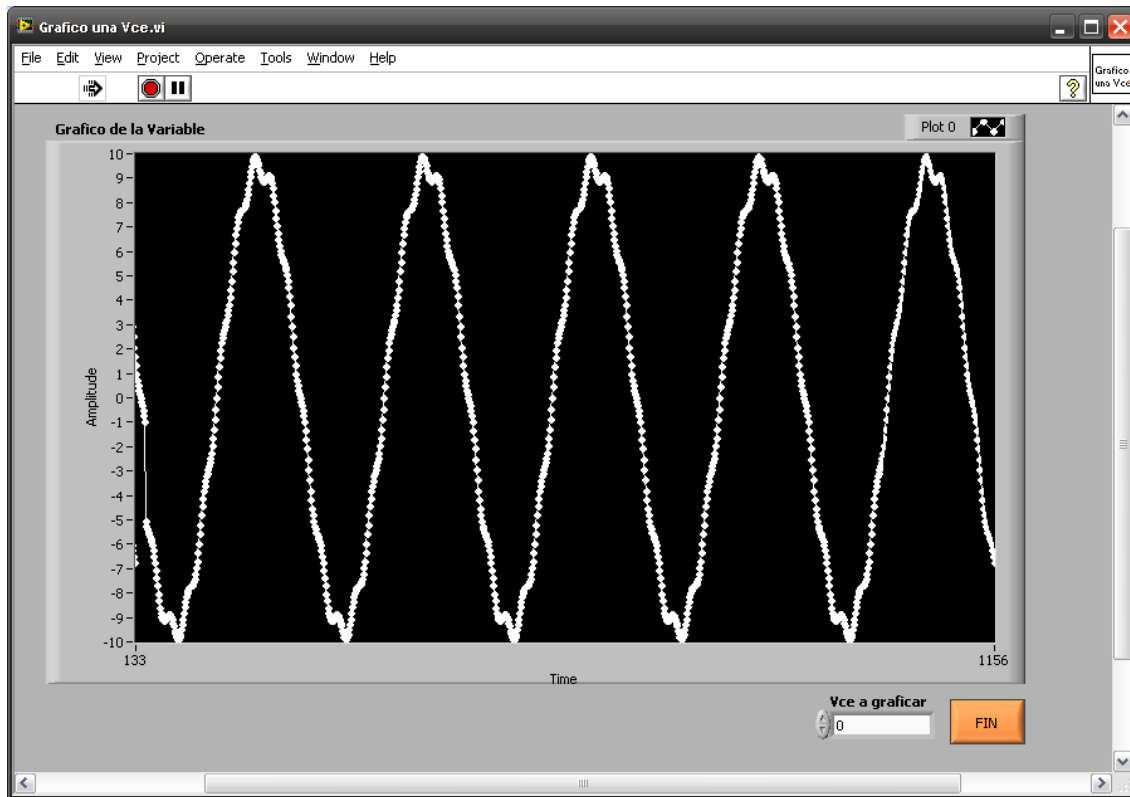


Fig. 42. Sub-VI frontal del gráfico de la variable Vce.

Se gráfica hasta que se presiona el botón FIN, considerando un tiempo de muestreo y como se menciono monitorea la variable Vce, como se puede ver el diagrama de flujo de la Fig. 43 (Ver programación mediante diagrama a bloques en el apéndice).



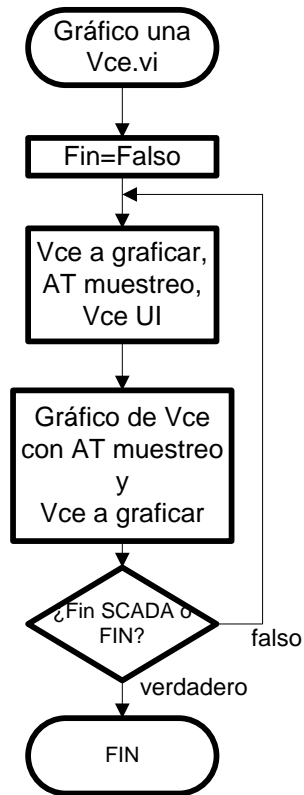


Fig. 43. Diagrama de flujo del gráfico de la variable Vce.





5.3.3 Panel de gráficos de las variables Vce del sistema

Es para visualizar el Gráfico de señales de las variables Vce del sistema (Fig. 44). Además permite realizar cambios de parámetros para la alarma y señalizaciones de alarma.



Fig. 44. Panel de gráficos de las variables Vce del sistema.

Se pueden observar los cambios de cuatro variables Vce del sistema considerando los valores del termómetro, la forma de onda de la variable, la última señalización, la variable Vce seleccionada y de acuerdo a estos datos se muestra la gráfica y se enciende el botón correspondiente a la señalización de la alarma en ALTO, BAJO o NORMAL, de acuerdo al diagrama siguiente (Fig.45) (Ver programación mediante diagrama a bloque en el apéndice).



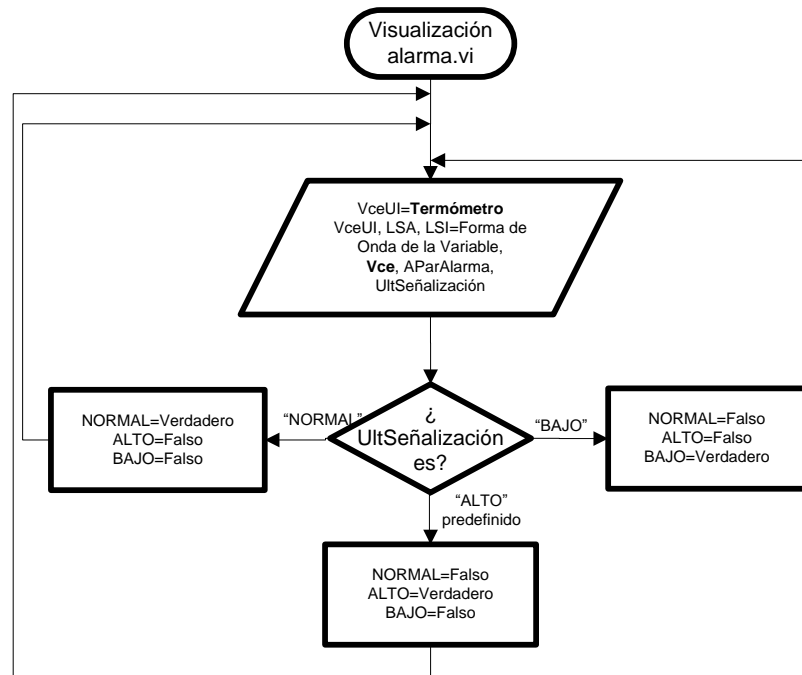


Fig. 45. Diagrama de flujo del panel de gráfico de la visualización de las variables Vce del sistema.

5.3.4 Panel del gráfico de la variable histórico del Vce.

Dentro de un sistema de adquisición de datos puede llegar a ser importante almacenar o llevar una bitácora del gráfico de alguna de las variables, en particular para este sistema se maneja la variable Vce que además se puede seleccionar en el panel, se tiene un botón de refrescar o actualizar la variable y considerando que se puede necesitar finalizar en cualquier momento se pone el botón de FIN (Fig.46).



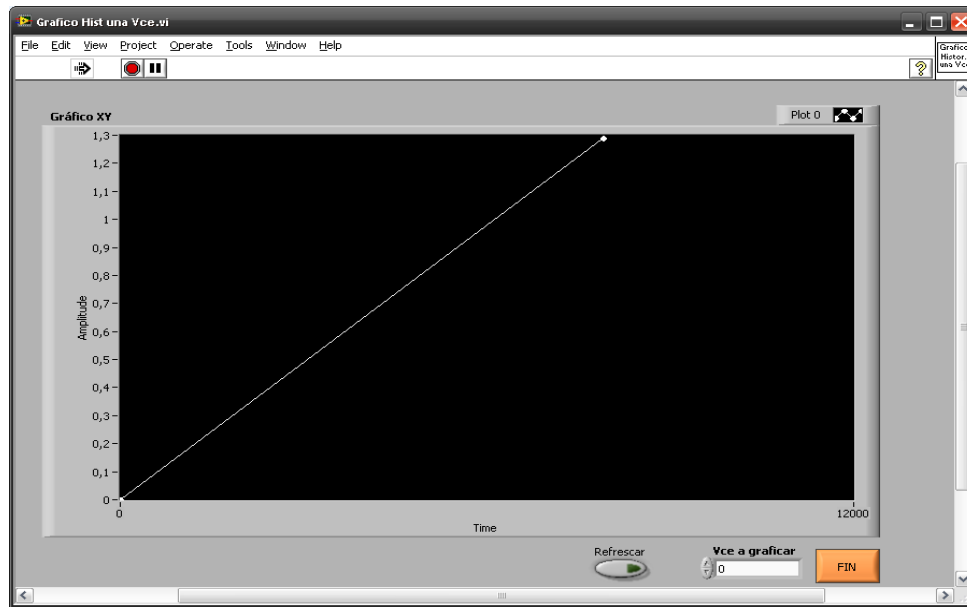


Fig. 46. Panel del gráfico histórico de la variable Vce.

Para generar el anterior panel del gráfico histórico se realizó un diagrama de flujo que permite ver las actividades y etapas de este gráfico (Fig. 47). Además la programación de cada una de los paneles se encuentra en el apéndice (Ver *programación mediante diagrama a bloque*).



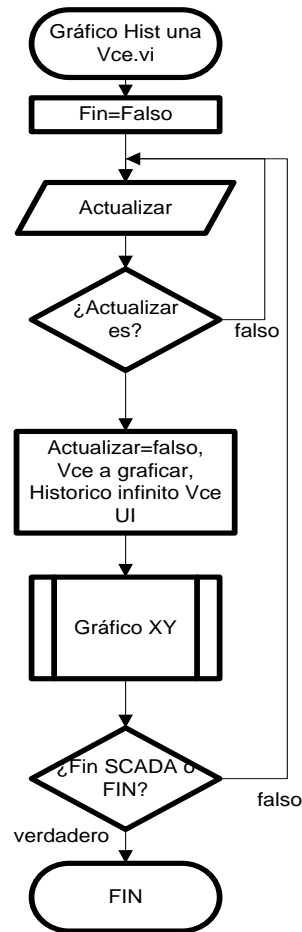


Fig. 47. Diagrama de flujo del panel de gráfico histórico de una variable Vce.





5.3.5 Panel de la recopilación de datos

Muestra la recopilación de los datos de las variables Vce y el número de la variable. Además se incluye un contador en tiempo real de las mismas (Fig. 48).

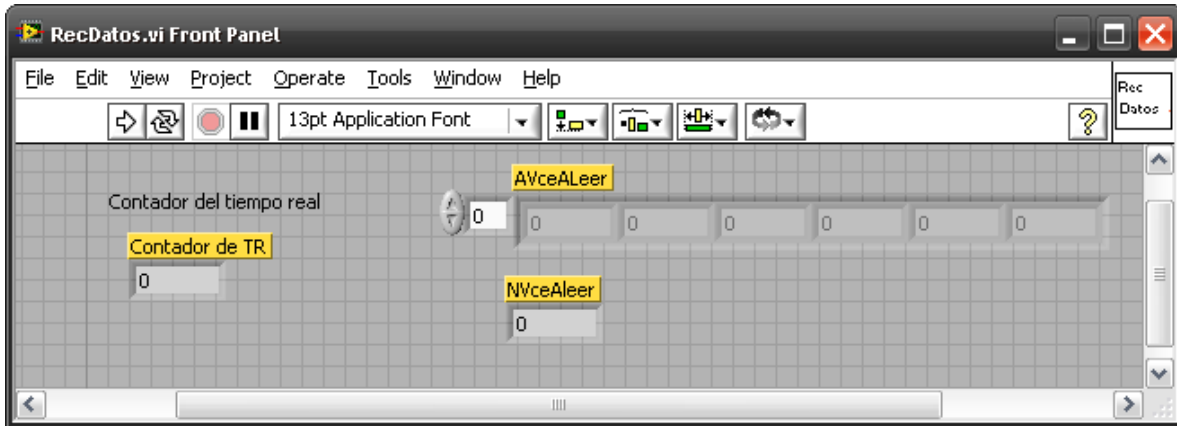


Fig. 48. SubVI de la recopilación de los datos.

Dentro del SubVI llamado RecDatos para abreviar recopilación de los datos, se tiene el diagrama de flujo (Fig. 49) donde se toman los valores de algunas variables de la base de datos, se realiza un pequeño proceso en el cual se pregunta ¿Cuántas Vce se leen? y ¿Cuáles? De acuerdo a esta respuesta se ejecuta la acción del siguiente instrumento virtual.



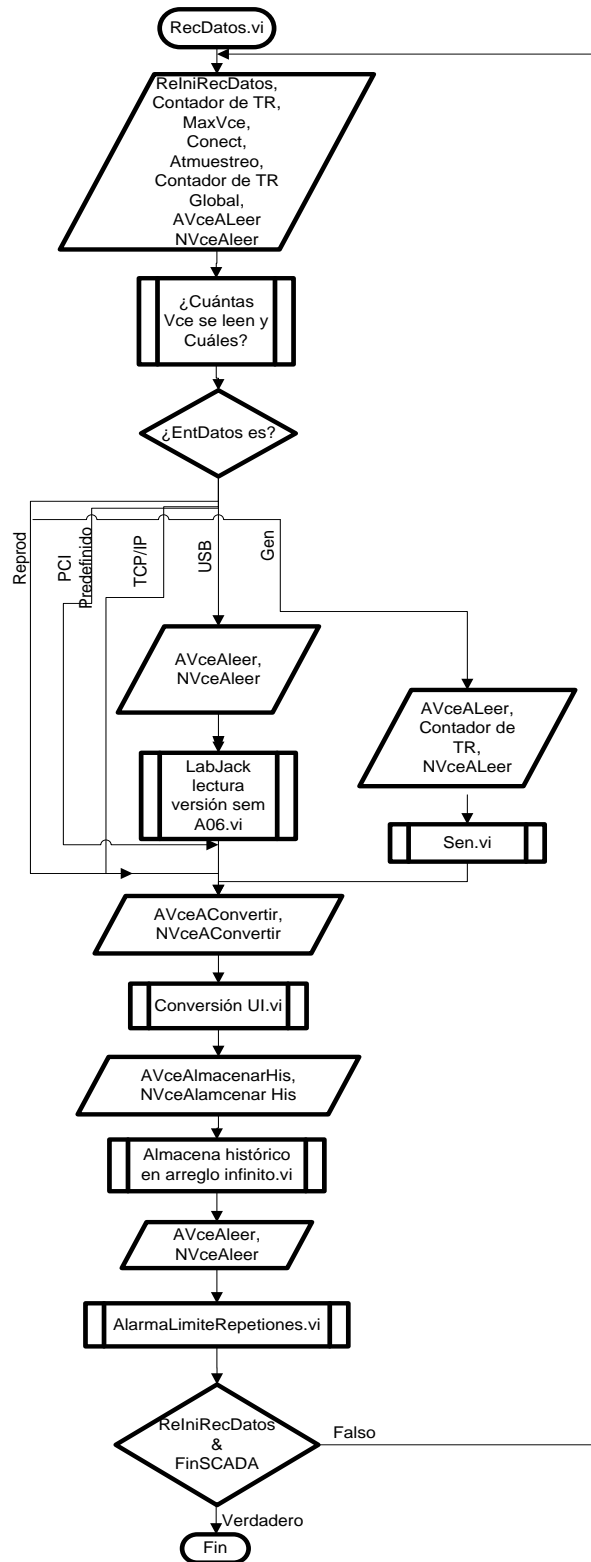


Fig. 49. Diagrama de flujo del subVI recopilación de datos.





5.3.6 Panel de la lectura de datos binaria

En el siguiente subVI del sistema se obtiene la lectura binaria de los datos a partir de la ruta de lectura hacia el archivo con extensión *dat* (Fig. 50).

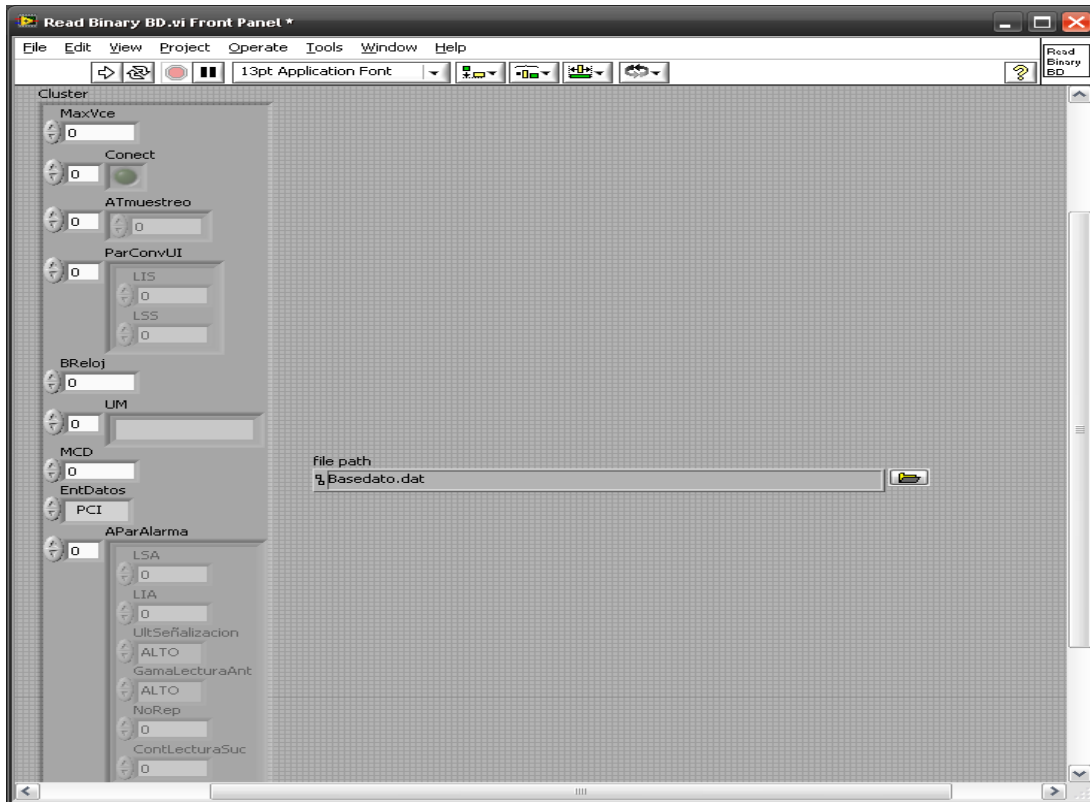


Fig. 50. SubVI que realiza la lectura binaria de los datos.





5.4 Conclusiones del capítulo

En general, este sistema cuenta con algoritmos para hacer la adquisición de los datos, acondicionamiento de los mismos y se complementa con una herramienta automatizada importante para la seguridad de redes y comunicaciones que es el cifrado de datos. De tal forma que el capítulo que termina contiene la descripción de la implementación de cada una de estas partes como son: panel de la base de datos, gráfico para visualización de señales, gráfico para visualización de la variable histórico y el módulo de recolección de datos. Para tal fin se ponen las figuras de los paneles frontales, los diagramas de flujo, algunas otras pantallas secundarias que el sistema contiene y obtener todas las entradas y salidas de este sistema.

No olvidando que el capítulo anterior contiene los módulos implementados de cifrado y descifrado de datos. Con esto se tiene la descripción completa del sistema ahora sólo resta el capítulo de pruebas y resultados, que a continuación se presenta.

.





6. PRUEBAS Y RESULTADOS

6.1 Funcionamiento del criptosistema

DES es un cifrado en bloques, lo que quiere decir que trabaja en bloques de texto de determinado tamaño (en este caso 64 bits) y regresa bloques del mismo tamaño. De esta manera DES resulta en una permutación de las 2^{64} posibles distribuciones de esos 64 bits. Cada bloque de 64 bits es dividido en dos bloques de 32 bits cada uno denominados **L** y **R** (izquierdo y derecho, por las siglas en ingles.) Esta división se utiliza en ciertas operaciones. A continuación se muestra como se desarrollaron las pruebas a las etapas de algoritmo DES.

Lo primero que se realiza es la conversión de datos de entrada. Sea **M** el mensaje simple **M**= 0123456789ABCDEF donde **M** se encuentra en formato hexadecimal. Al cambiar **M** a formato binario se tiene el bloque de 64 bits:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
L = 0000 0001 0010 0011 0100 0101 0110 0111
R = 1000 1001 1010 1011 1100 1101 1110 1111

Esto mismo se ve útil en la Fig. 51 con la cual se desarrollo la conversión *cadena* que está en hexadecimal a binario.



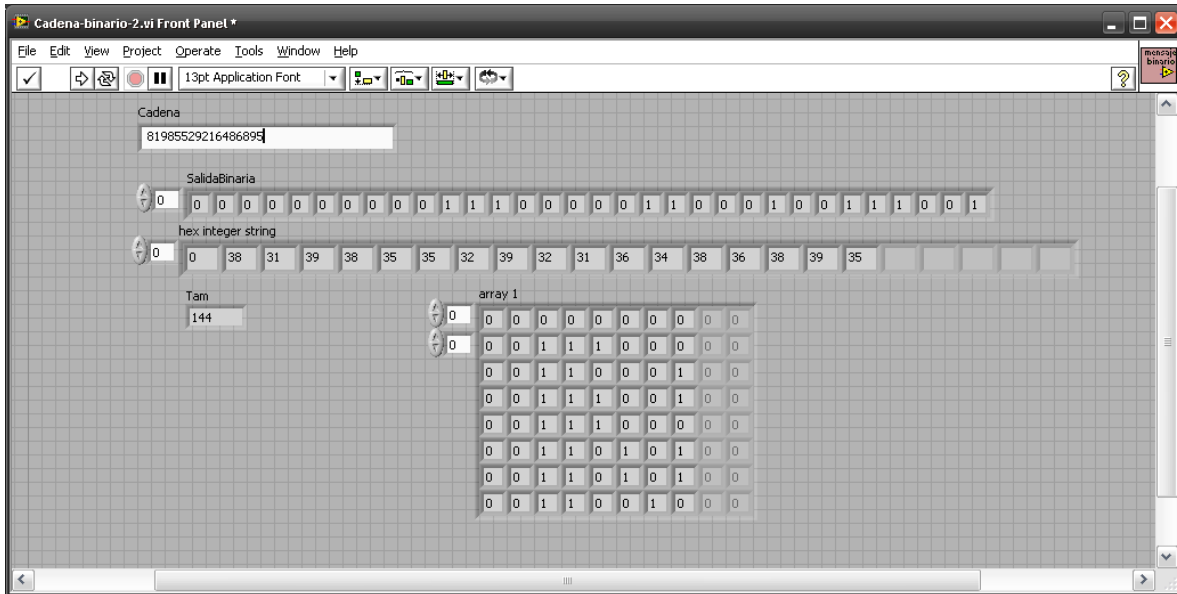


Fig.51. Conversión del mensaje de entrada a un formato binario.

DES opera sobre los bloques de 64 bits utilizando claves de 56 bits. Las claves son guardadas como si utilizaran 64 bits pero cada octavo bit es ignorado. Es decir, los bits 8, 16, 24, 32, 40, 48, 56 y 64. Sin embargo, se seguirá denominando a los bits del 1 al 64. Los bits sobrantes serán eliminados cuando se creen las subclaves.

Se supone que **K** es la *clave* hexadecimal $K = 133457799BBCDFF1$, transformándola a notación binaria (1 = 0001, 3 = 0011, etc.) y agrupando cada 8 bits, se puede ver que el último bit no será utilizado:

$K = 00010011\ 00110100\ 01010111\ 01111001\ 10011011\ 10111100\ 11011111\ 11110001$

La Fig. 52 muestra el SubVI que se utilizó para generar las claves, iniciando con **K** en la clave hexadecimal $K = 133457799BBCDFF1$ transformándola a notación binaria (la programación mediante diagrama a bloques se encuentra en el apéndice). El siguiente paso es generar 16 subclaves de **K**, cada una de las cuales mide 48 bits. Entonces, la clave original de 64 bits es permutada de acuerdo a la siguiente tabla, **PC-1**. Puesto que la primera entrada en la tabla es "57", esto quiere decir que el bit de la posición 57 de la clave original se convierte en el primer bit de la clave permutada **K+**. El bit 49 se convierte en el segundo bit, etc. Se puede notar, que sólo se utilizan 56 bits y faltan precisamente las posiciones múltiplos de 8.[34][51][69]



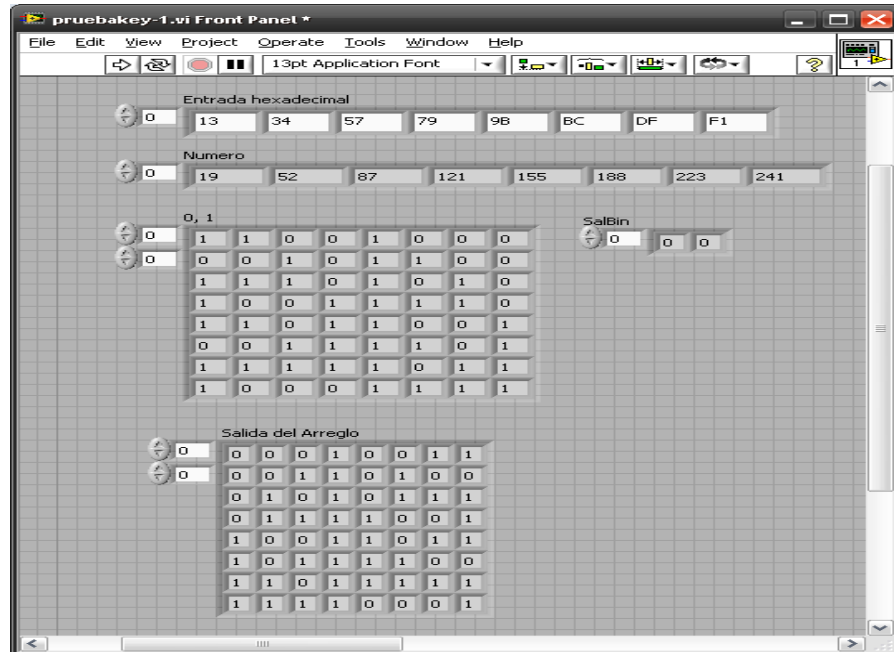


Fig.52. Muestra el subVI que se utilizó como prueba de las claves, iniciando con K en hexadecimal.

De la clave original de 64 bits

$K = 00010011\ 00110100\ 01010111\ 01111001\ 10011011\ 10111100\ 11011111\ 11110001$

Obtenemos la permutación:

$K_+ = 1111000\ 0110011\ 0010101\ 0101111\ 0101010\ 1011001\ 1001111\ 0001111$

Ahora se separa la clave en sus dos mitades, C_0 y D_0 , donde cada mitad tiene 28 bits.

$C_0 = 1111000\ 0110011\ 0010101\ 0101111$

$D_0 = 0101010\ 1011001\ 1001111\ 0001111$

Con C_0 y D_0 definidos, ahora se pueden crear 16 bloques C_n y D_n , $1 \leq n \leq 16$. Cada par de bloques C_n y D_n esta formado del par previo C_{n-1} y D_{n-1} , respectivamente, para $n = 1, 2, \dots, 16$, utilizando el siguiente esquema de corrimientos a la izquierda del bloque previo. Para hacer un corrimiento a la izquierda, mover cada bit un lugar a la izquierda, excepto por el primer bit, el cual es ciclado al final del bloque.

Esto quiere decir que, por ejemplo, C_3 y D_3 son obtenidos de C_2 y D_2 , respectivamente, por dos corrimientos izquierdos, y C_{16} y D_{16} son obtenidos de C_{15} y D_{15} , respectivamente por





un corrimiento izquierdo. En todos los casos, por un solo corrimiento izquierdo se quiere decir una rotación de los bits un lugar a la izquierda, de tal manera que después de un corrimiento izquierdo en las 28 posiciones los bits quedan en las posiciones 2, 3,..., 28, 1.

$$C_0 = 1111000011001100101010101111$$

$$D_0 = 0101010101100110011110001111$$

$$C_1 = 1110000110011001010101011111$$

$$D_1 = 1010101011001100111100011110$$

$$C_{16} = 1111000011001100101010101111$$

$$D_{16} = 0101010101100110011110001111$$

Ahora, son formadas las claves K_n , para $1 \leq n \leq 16$, aplicando la siguiente tabla de permutaciones a cada uno de los pares concatenados $C_n D_n$. Cada par tiene 56 bits, pero **PC-2** (Tabla 5) sólo utiliza 48 de estos.

Por lo tanto, el primer bit de K_n es el 14avo bit de $C_n D_n$, el segundo bit el 17avo, y así en adelante, terminando con el 48avo bit de K_n siendo el 32avo bit de $C_n D_n$.

Para la primera clave se tiene:

$$C_1 D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110$$

La cual, tras aplicar la permutación **PC-2**, se convierte en

$$K_1 = 000110 110000 001011 101111 111111 000111 000001 110010$$

Para las demás claves tenemos:

$$K_2 = 011110 011010 111011 011001 110110 111100 100111 100101$$

$$K_3 = 010101 011111 110010 001010 010000 101100 111110 011001$$

$$K_4 = 011100 101010 110111 010110 110110 110011 010100 011101$$

$$K_5 = 011111 001110 110000 000000 111111 110101 001110 101000$$

$$K_6 = 011000 111010 010100 100111 110010 000111 101100 101111$$

$$K_7 = 111011 001000 010010 010110 111111 100001 100010 111100$$



 $K_8 = 111101\ 111000\ 101000\ 000111\ 010110\ 010011\ 101111\ 111011$ $K_9 = 111000\ 001101\ 101111\ 111101\ 011111\ 011110\ 011110\ 000001$ $K_{10} = 101100\ 011111\ 001101\ 101000\ 111101\ 100100\ 011001\ 001111$ $K_{11} = 001000\ 010101\ 111111\ 010011\ 110111\ 101101\ 001110\ 000110$ $K_{12} = 011101\ 010111\ 000111\ 110101\ 100101\ 000110\ 011111\ 101001$ $K_{13} = 100101\ 111100\ 010111\ 010001\ 111110\ 101011\ 101001\ 000001$ $K_{14} = 010111\ 110100\ 001110\ 110111\ 111100\ 101110\ 011100\ 111010$ $K_{15} = 101111\ 111001\ 000110\ 001101\ 001111\ 010011\ 111100\ 001010$ $K_{16} = 110010\ 110011\ 110110\ 001011\ 000011\ 100001\ 011111\ 110101$

Al obtener la K_{16} terminamos con las subclaves necesarias para el cifrado de datos y con esta información se desarrolla el SubVI (Fig. 53) que ayuda con el paso 1 que crea las 16 subclaves, aplicando el diagrama de flujo de la obtención de las claves para el DES. Se realizan los corrimientos de cada ronda.



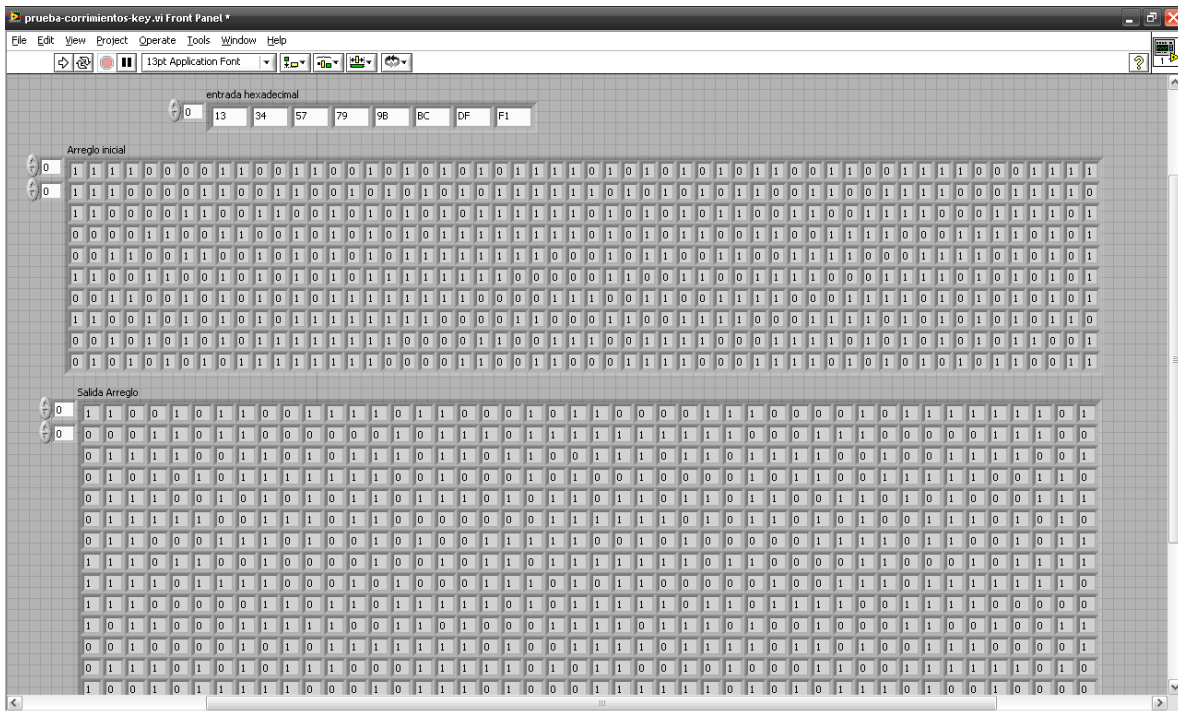


Fig.53. La prueba de los corrimientos de las claves.

Para realizar la etapa que corresponde a las S-Cajas se puede observar en la Fig.54 las 8 S-Cajas que aparecen desde B1 hasta B8 con los valores que se generan en cada una. Aquí se ven las pruebas que se realizaron para que esta etapa pudiera tener las distintas combinaciones y los arreglos que se obtienen al aplicar este algoritmo.



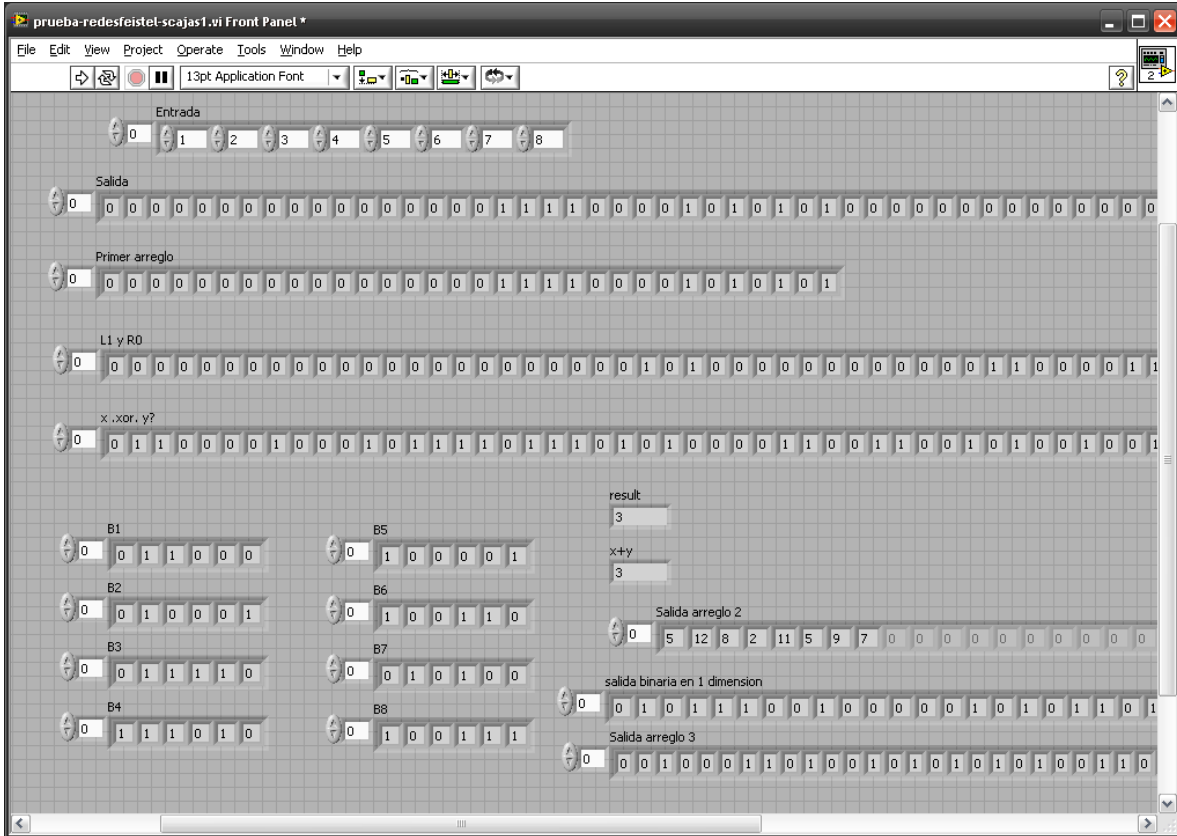


Fig.54. Pruebas de las S-Cajas

A continuación se muestra el arreglo que se obtiene al aplicar la permutación inicial IP (Tabla 3) (Fig.55).

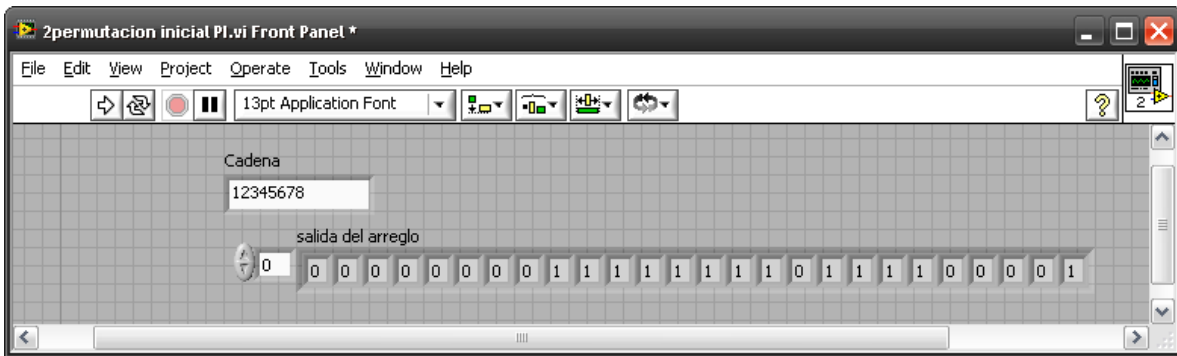


Fig.55. Prueba de la permutación inicial IP

Para poder ver los resultados de las pruebas, se generó el siguiente panel frontal (Fig. 56), se le llamó *DES* y *unDES.vi* con la entrada de *MENSAJE SIMPLE* y la clave, se muestra *MENSAJE CIFRADO* y *MENSAJE DESCIFRADO*. Observando que *MENSAJE SIMPLE Y MENSAJE DESCIFRADO* coinciden, que es lo que se buscaba con este panel.



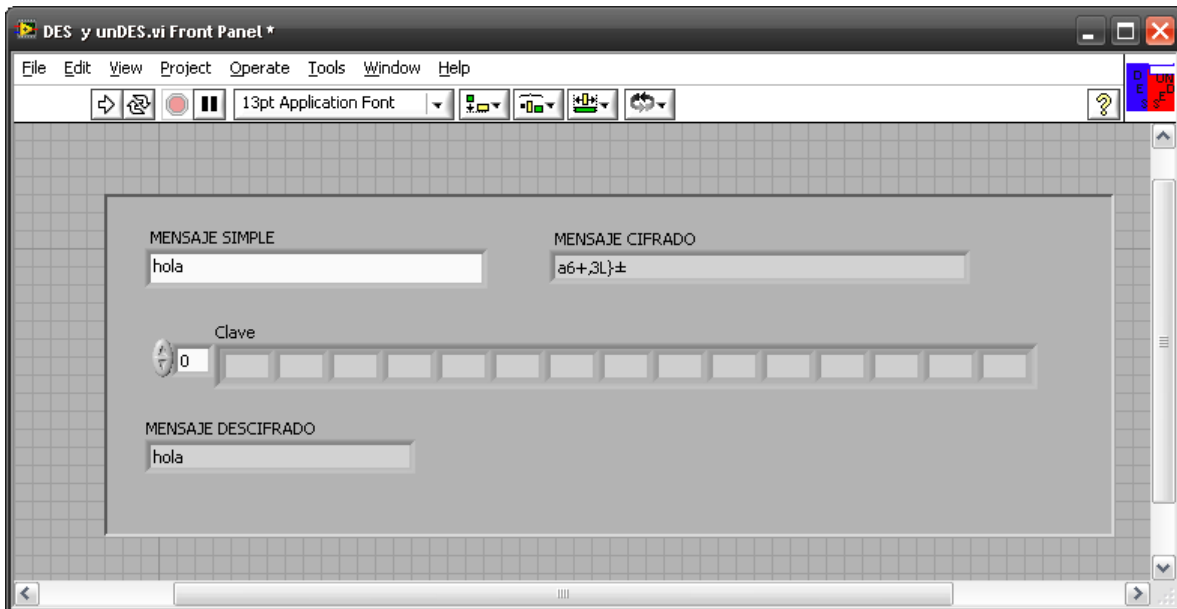


Fig.56. Pruebas de cifrado y descifrado con DES.

El SubVI que se muestra en la Fig.57 es parte de la implementación, corresponde a las pruebas que se realizaron para comprobar que el algoritmo DES funciona en la parte de cifrado y descifrado de un mensaje llamado *Cadena*. Las siguientes etapas de la implementación del algoritmo DES, las pruebas se describen en el apéndice.



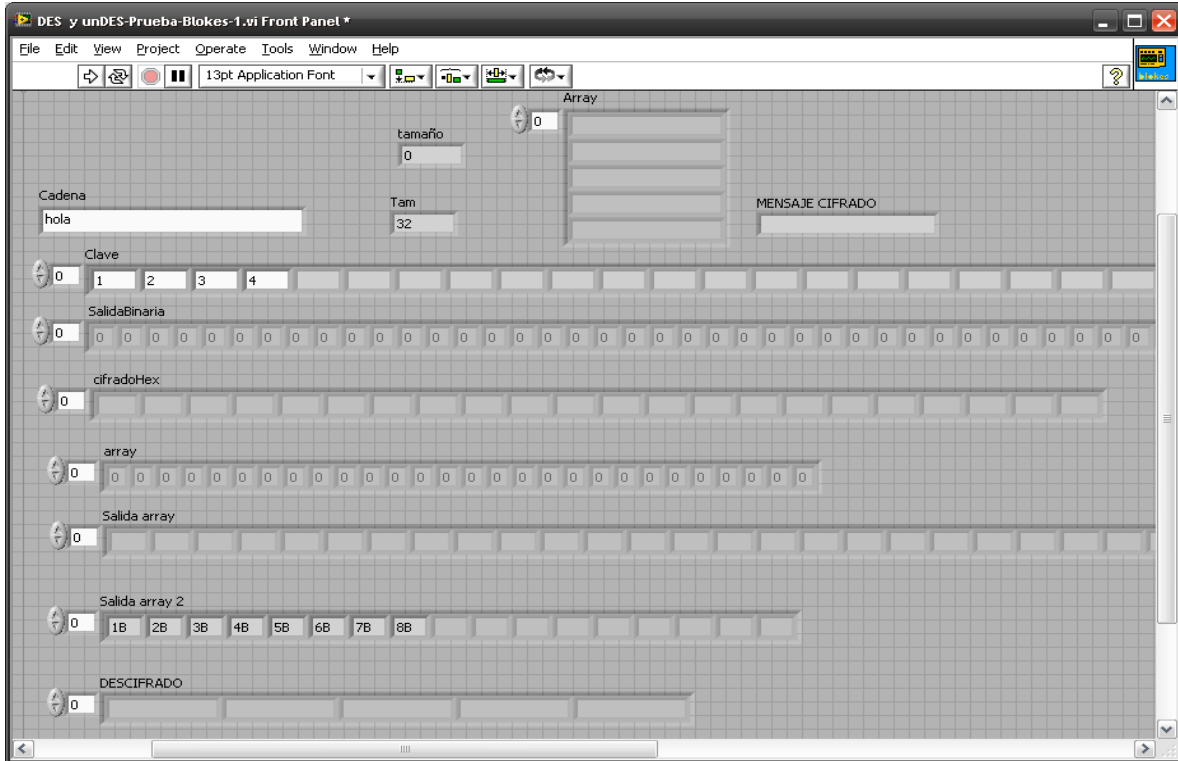


Fig.57. Prueba de cifrado y descifrado en bloques.





6.2 Valores de prueba en el criptosistema

Una vez que se concluyo con la implementación del algoritmo DES, conviene asegurar que funcione adecuadamente. Para esto se incluyen algunos valores de prueba, que contienen todos los datos intermedios que se emplean en el algoritmo. En la figura (Fig. 58) se observa una versión del *DES BLOKES final-1 versión prov.vi* en la que se introducen valores de prueba en *MENSAJE SIMPLE* y en *CLAVE*, haciendo énfasis en que se introducen claves débiles. En la salida de *MENSAJES CIFRADOS EN BLOQUES* y en *MENSAJE CIFRADO* se obtienen los resultados al aplicar el algoritmo de cifrado.

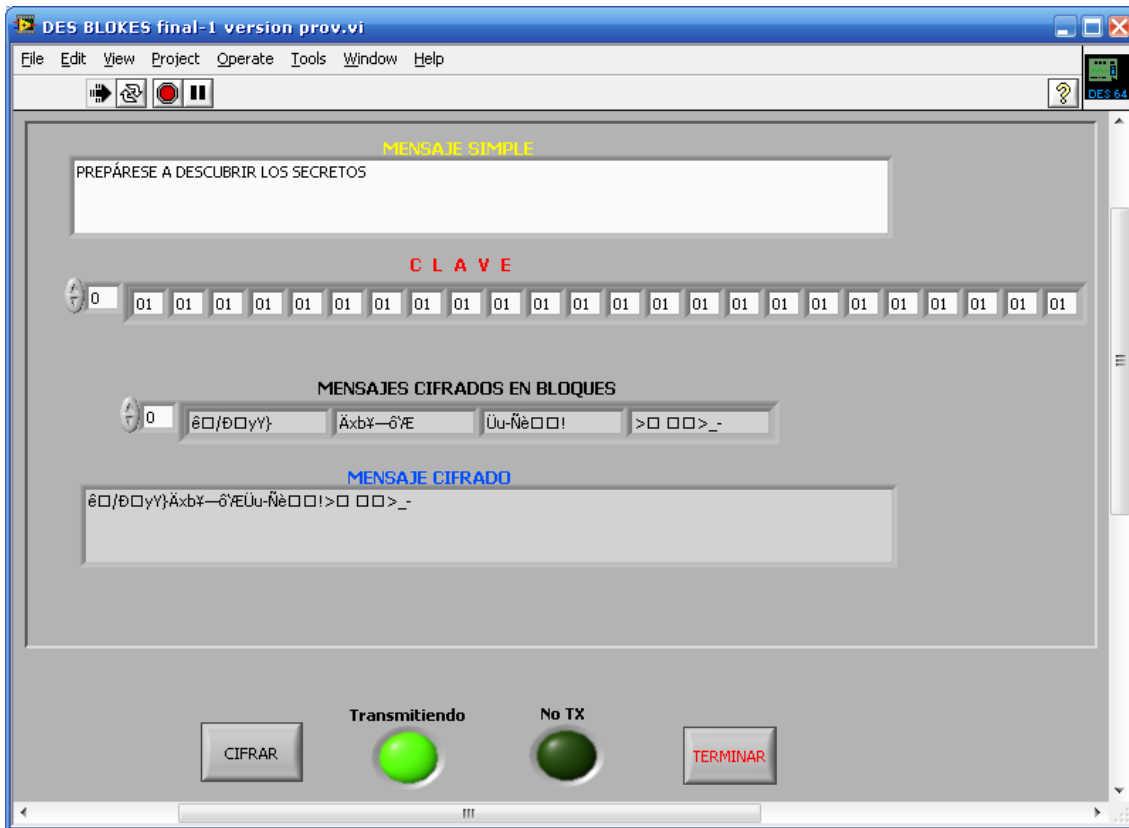


Fig. 58. Pruebas del cifrado de datos aplicando una clave.

6.2.1 Valores de prueba en el criptosistema aplicando clave débil.

Estos valores de prueba son los presentados en la tabla siguiente (Tabla 15). Contiene los valores de claves débiles, clave tras aplicar PC1, mensaje simple y el mensaje cifrado obtenido con el criptosistema DES, como se puede ver en el panel de la Fig. 59.





Tabla 15. Pruebas de distintas claves débiles y el resultado obtenido en los mensajes cifrados.

Claves débiles	Clave tras aplicar PC1	Mensaje simple	Mensaje cifrado
0101010101010101	0000000 0000000	PREPÁRESE A DESCUBRIR LOS SECRETOS	ê ----- /ĐyY }ÄxbŸ— ô‘ÆÜu-Ñè•!> >_- KÙ>"*o®
1F1F1F1F0E0E0E0E	0000000 FFFFFFFF	PREPÁRESE A DESCUBRIR LOS SECRETOS	^{a-†} =±ft4âÁD~ç^Vy<áŽ £> >_-
E0E0E0E0F1F1F1F1	FFFFFFF 0000000	PREPÁRESE A DESCUBRIR LOS SECRETOS	ÛpÔ¼£?²JXW ----- Û4_7B ----- Œ¹ë> >_-
FEFEFEFEFEFEFEFE	FFFFFFFFFFFFFFF	PREPÁRESE A DESCUBRIR LOS SECRETOS	fz½nžWÀüô\$Rú*²?ñĐ !Mê> >_-



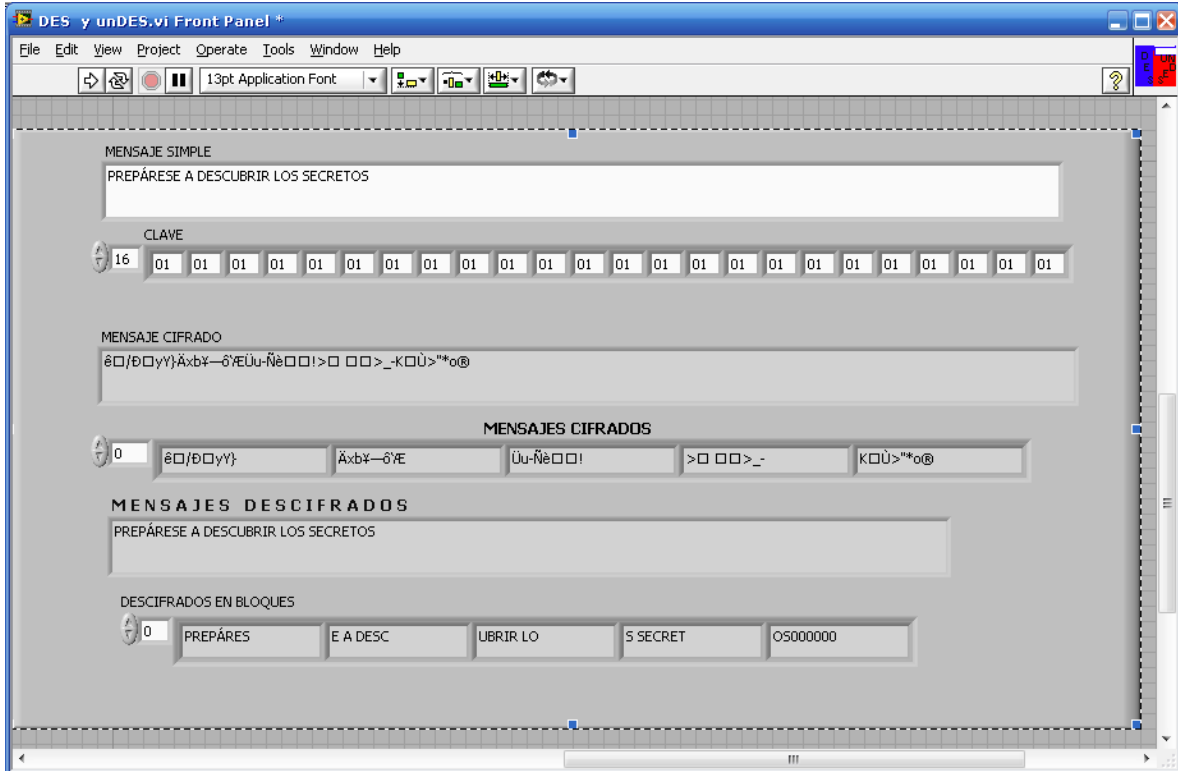


Fig. 59. Prueba del criptosistema con una clave débil.

6.2.2 Valores de prueba en el criptosistema aplicando clave semi-débil.

A continuación se muestra la ejecución del criptosistema aplicando una clave semi-débil (Fig. 60 y 61). Se puede observar los datos introducidos como son mensajes simple, clave y el resultado de este proceso en mensajes cifrados, además se agrega para ver el funcionamiento en un mismo panel el descifrado de los datos.

Dentro de los paneles de las Fig. 60 y 61 se introducen los valores de las claves semi-débiles y los mensajes simples, y se ejecuta el criptosistema para así obtener los mensajes cifrados de la Tabla 16.



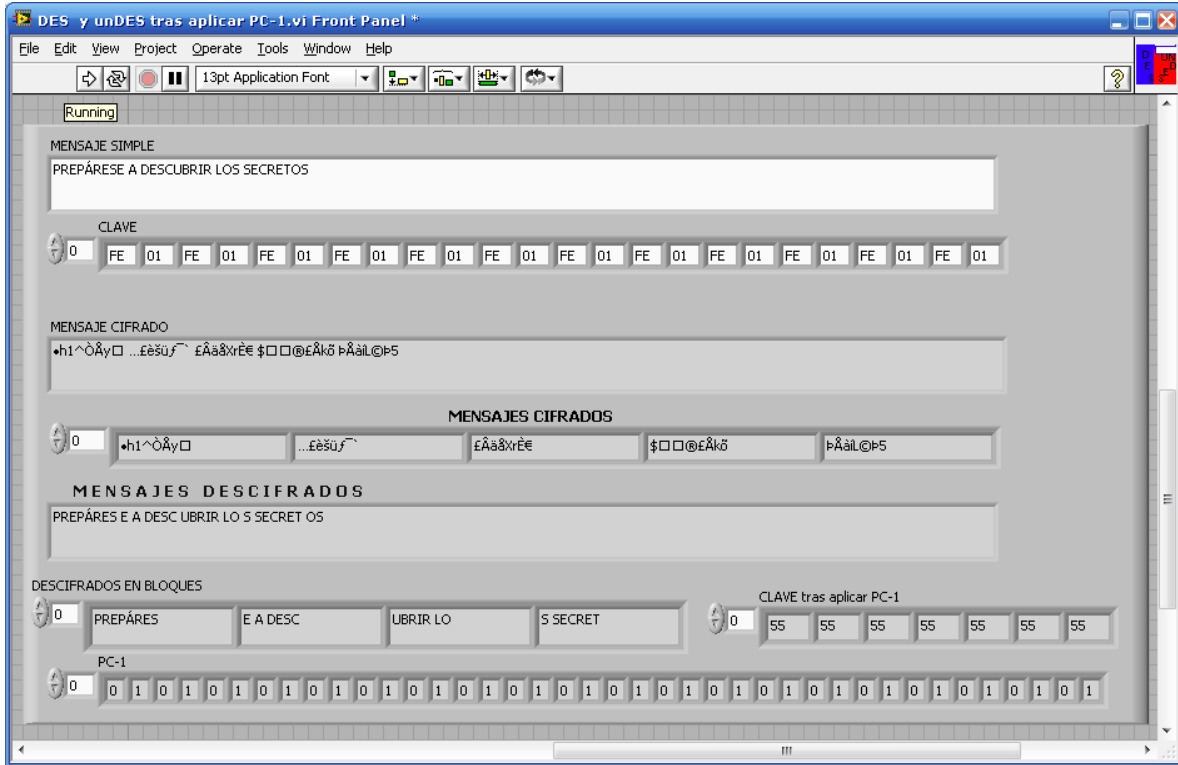


Fig. 60. Pruebas del criptosistemas tras aplicar claves semi-debiles.



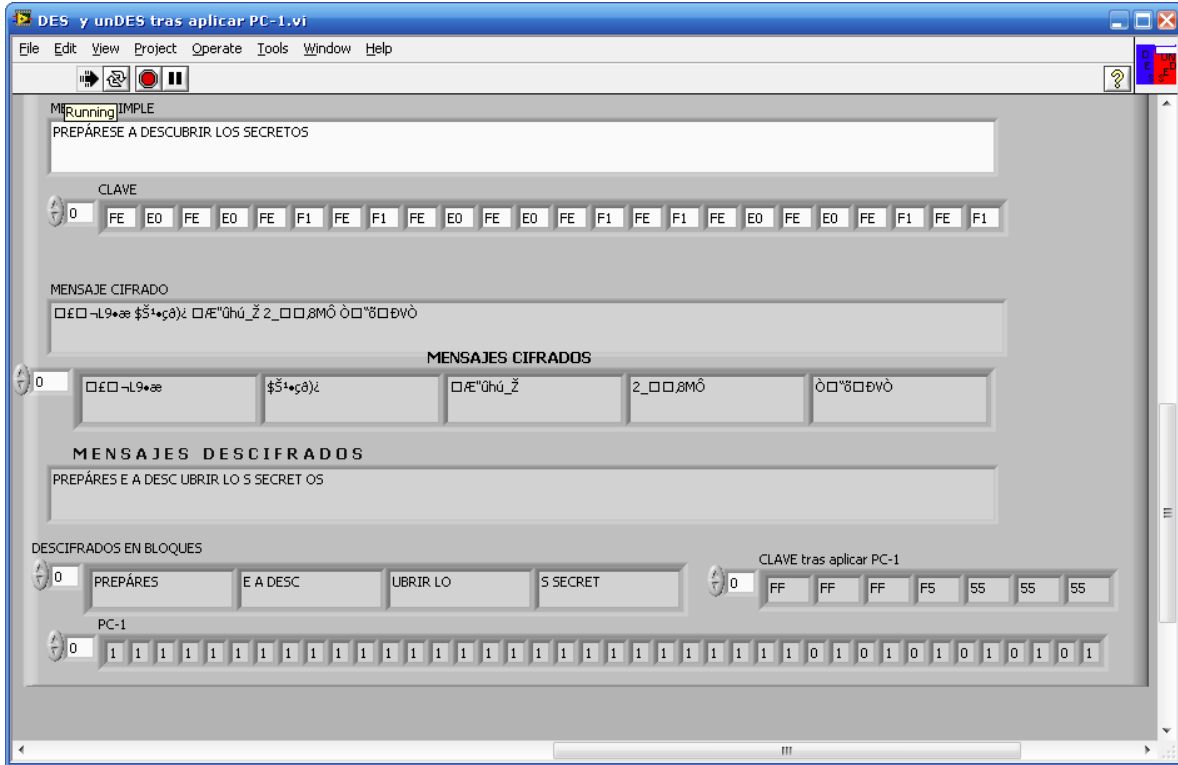


Fig. 61.Pruebas del criptosistema tras aplicar claves semi-débiles

Tabla 16. Pruebas de distintas claves semi-debiles y el resultado obtenido en los mensajes cifrados.

Claves semidébiles	Clave tras aplicar PC1	Mensaje simple	Mensaje cifrado
01FE01FE01FE01FE	AAAAAAAAAAAA AA	PREPÁRESE A DESCUBRIR LOS SECRETOS	jmÜ9œå □ IBL< â“”]£!×NÄ-xu9C),t!à(•h1^ÔÅy□ ...fèšüf~`fÅ ääXrÈ€\$@£ÅkôPÅâiL© p5
FE01FE01FE01FE01	5555555 5555555	PREPÁRESE A DESCUBRIR LOS SECRETOS	¾wp]□ AšfäYw>iXQ@ %B • íxX *oÀá!P‡}¶©Iš!½^
1FE01FE00EF10EF1	AAAAAAA 5555555	PREPÁRESE A DESCUBRIR LOS SECRETOS	9Dp°!•4 ~<ž >l*!]€O*Z~ ×‘VH • # O+ä©:i
E01FE01FF10EF10E	5555555 AAAAAAA	PREPÁRESE A DESCUBRIR LOS SECRETOS	ó’sÔ¼ß×ù¥?ÿžâ□ ŽØ’ø< p/³~ô=iØEBîÝQ}·oÓO





E001E001F101F101	555555 000000	SECRETOS PREPÁRESE A DESCUBRIR LOS SECRETOS	îËËt}ÁCr<^,ä} □ AfÍá% Á, }pç-k Ùâ M~ûÁIŽSI
1FFE1FFE0EFE0EFE	AAAAAAA FFFFFFFF	PREPÁRESE A DESCUBRIR LOS SECRETOS	ù#]‘- +Û]óæ* ùgq²R[çvE=COPÃ\hGn Œå39
FE1FFE1FFE0EFE0E	555555 FFFFFFFF	PREPÁRESE A DESCUBRIR LOS SECRETOS	Öð4Æ8]úü+±;¾ç;zns©X 6¿suÆm • ©fE"à®·Ý
011F011F010E010E	000000 AAAAAA	PREPÁRESE A DESCUBRIR LOS SECRETOS	Uæ1šöl^=Ô(î `ãÈ @‘/ÿ¼pð ÛÜeÓûHÍ ;ªp£Æ50, =ðÛUPÿé@1 • 0eèQö ;] - { • ëÝ>n7K
1F011F010E010E01	000000 555555	PREPÁRESE A DESCUBRIR LOS SECRETOS	‘²\$]† ,>^^} ‡[!-bÒ° -û{×Ñí´ór—ðP~ È}=& 2^
E0FEE0FEF1FEF1FE	FFFFFFF AAAAAA	PREPÁRESE A DESCUBRIR LOS SECRETOS	£-L9•æ\$Š¹•çð)¿Æ"ùhú_ Ž2_,8MÔÒ“ðĐVÒ

6.3 Pruebas de comunicación del sistema

Ahora, se tiene la parte del sistema que refiere a la comunicación entre el cliente y el servidor. Con el servidor en un principio fue difícil ya que sólo se cuenta con el VI de comunicación general proporcionado por LabVIEW, se adaptó a la aplicación de envío de información cifrada y como se puede ver es posible que ésta información viaje a través de cualquier canal.

El sistema de comunicación hace uso de TCP como el protocolo de comunicación y permite que el servidor reciba varias conexiones y en este caso todas se pueden descifrar (Fig. 62).



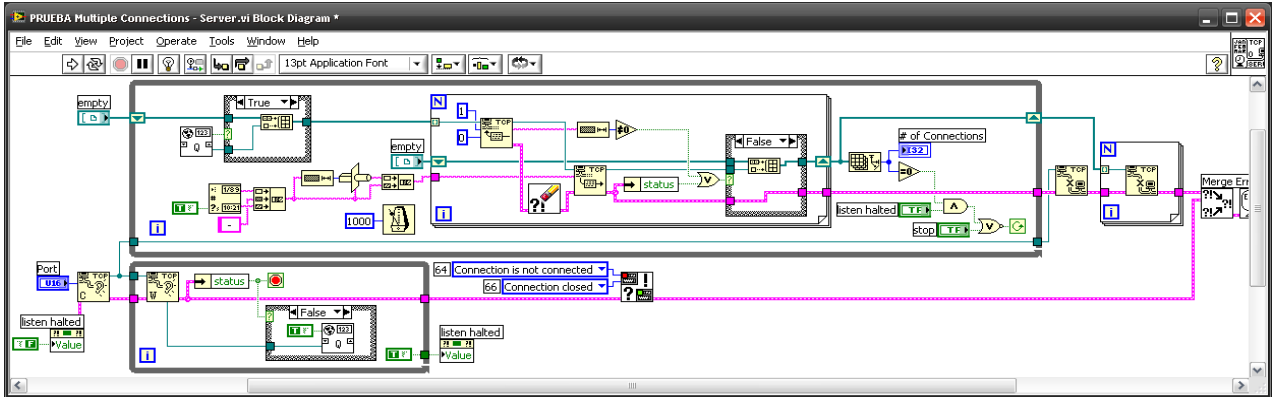


Fig.62. Pruebas de la comunicación del sistema en la conexión del servidor.

Las pruebas de comunicación con el cliente se realiza en otra computadora y en esta se puede hacer el cifrado de datos para enviarlo al servidor, mediante la prueba de la Fig.63.

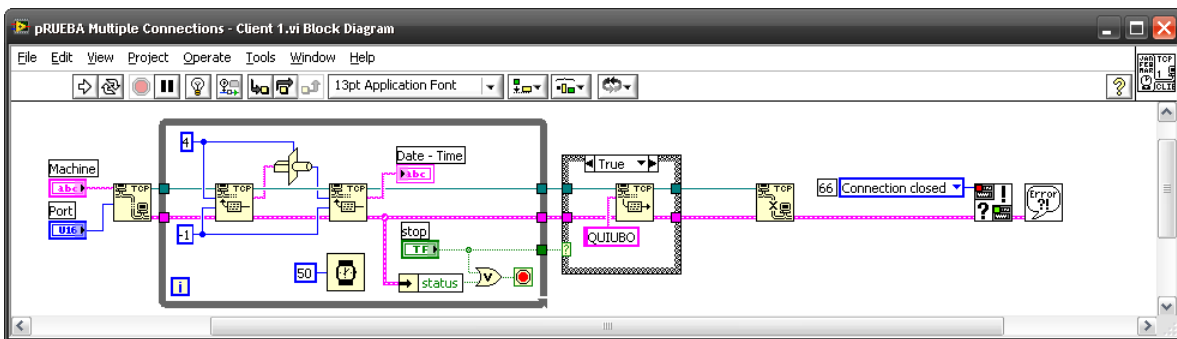


Fig.63. Pruebas de la comunicación del sistema en la conexión de cliente

6.4 Pruebas de tiempo de procesamiento del criptosistema

En la serie de pruebas que se aplicaron al criptosistema, se realizaron las de tiempo de procesamiento que permite visualizar el tiempo de ejecución del cifrado de datos (Fig. 64) con el sistema operativo Windows XP, el software labVIEW versión 8.0., y las siguientes características del procesador de la computadora que se utilizaron para dichas pruebas:

- Intel pentium dual-core processor T2370.
- 1024 MB PC5300 DDR2 SDRAM.
- 120 Gb (5400 RPM) HDD.



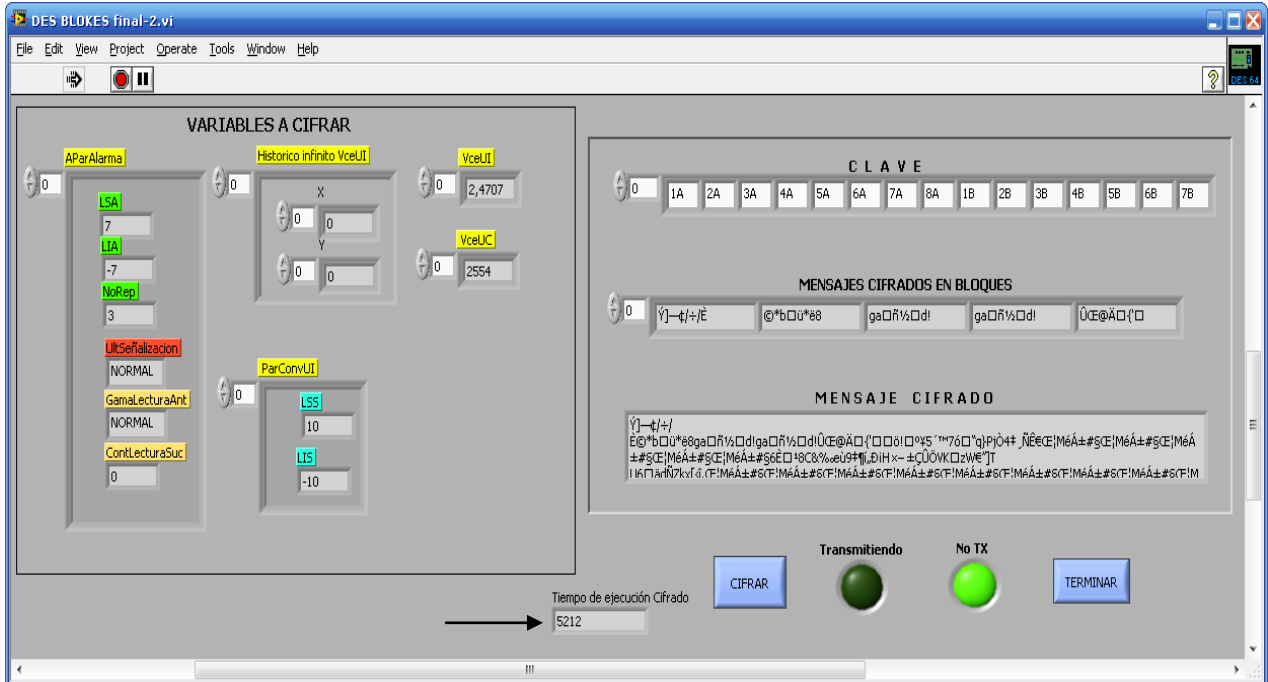


Fig. 64. Prueba del tiempo de ejecución de procesamiento en el cifrado de datos.

Además, se realizaron las de tiempo de procesamiento que permite visualizar el tiempo de ejecución del descifrado de datos (Fig. 65), se consideran las mismas características del procesador de la computadora antes mencionada.



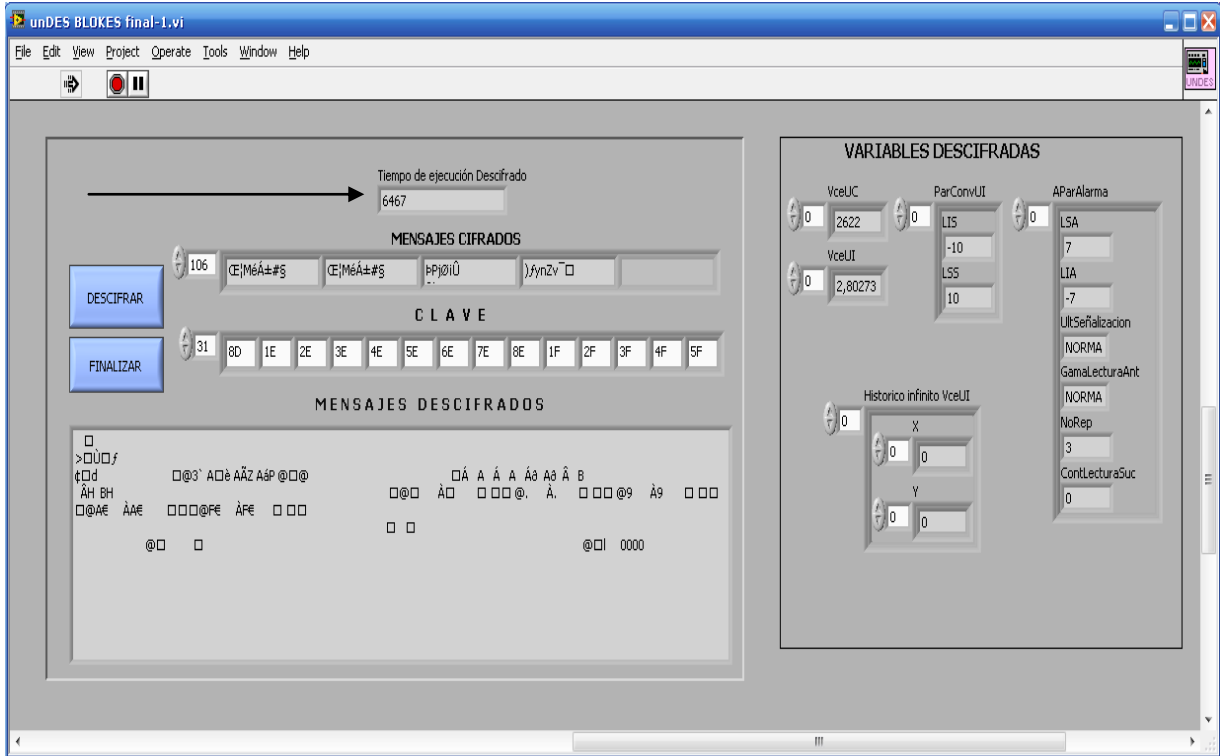


Fig. 65. Prueba del tiempo de ejecución de procesamiento del descifrado.

Se obtienen las pruebas con los tiempos del cifrado de datos y descifrado del algoritmo DES, que se implementó en la presente tesis, se muestra en la Tabla 17. :

Tabla 17. Tiempos de ejecución del cifrado y descifrado de datos.

No. de prueba	Tiempo en el cifrado (ms)	Tiempo en el descifrado (ms)
1	5348	6456
2	5247	6256
3	5165	6333
4	5158	6317
5	5228	6467





6.5 Conclusiones del capítulo

Recapitulando, lo presentado hasta éste momento se tiene al sistema de adquisición de datos y monitoreo implementado y considerando también el desarrollo del criptosistema DES, todo esto en un lenguaje de programación gráfico para el manejo de información a través de un canal inseguro.

Las pruebas realizadas al criptosistema desde cada una de las partes que lo conforman, primero la del cifrado y después el descifrado de datos, como se fueron generando las 19 etapas del algoritmo DES y como se puso en funcionamiento dentro del sistema de comunicaciones, para enviar y recibir información dentro de un sistema cliente/servidor.

Una de las pruebas a las que se sometió el algoritmo de cifrado y descifrado es con las tablas de claves débiles y claves semi-débiles con lo cual se puede ver que el resultado es el que se esperaba en los mensajes cifrados y con resultados satisfactorios.

Por último, se puede observar cómo se llegó al sistema completo y las etapas que se requirieron para utilizar un cifrado de datos en un ambiente distribuido.





CONCLUSIONES

1. Se cumplieron los objetivos planteados en la tesis y resultó demostrada la hipótesis, pues mediante la implementación de un criptosistema, fue posible enviar datos por un canal en un ambiente distribuido utilizando instrumentación virtual con un desempeño satisfactorio.
2. Se seleccionó el algoritmo de cifrado de datos, DES, para intercambio de información entre aplicaciones distribuidas de instrumentación virtual.
3. Se programó el algoritmo seleccionado para el cifrado de datos y descifrado de datos.
4. Se desarrolló un sistema para el intercambio de datos cifrados entre aplicaciones distribuidas desarrolladas mediante herramientas de instrumentación virtual.
5. Se diseñó y programó una aplicación de adquisición de datos y control supervisorio (SCADA), que integra un sistema Cliente/Servidor con TCP/IP en *LabVIEW*.
6. A pesar de que el algoritmo DES está rebasado por otros de mayor seguridad, el haberlo utilizado es un buen ejercicio para introducir la criptografía en sistemas distribuidos mediante instrumentación virtual.





RECOMENDACIONES Y TRABAJO FUTURO

1. Este trabajo puede adaptarse a otras aplicaciones industriales, realizando los ajustes pertinentes.
2. Generar un conjunto de módulos con distintos algoritmos criptográficos, en particular el 3DES o alguna de sus variantes.
3. Implementar un generador de claves con números pseudoaleatorios.
4. Utilizar esta implementación para cifrado de flujo.
5. Cambiar el tamaño de la clave que se introduce al algoritmo.
6. Implementar otro estándar de cifrado como AES.
7. Implementación del algoritmo en componentes de hardware.
8. Utilizar distintos modos de operación del cifrado por bloques.
9. Adicionalmente, sólo se podrían programar aquellos algoritmos de patente vencida.





REFERENCIAS BIBLIOGRÁFICAS

- [1] Alexi, W., B. Chor, O. Goldreich, and C. Schnorr. 1984. RSA/Rabin Bits are $1/2 + 1/\text{poly}(\log N)$ Secure (Extended Abstract). *25th IEEE Symposium on the Foundations of Computer Science*. 449-457.
- [2] Allender, E. 1987. Some Consequences of the Existence of Pseudorandom Generators. *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. 151-159.
- [3] ANSI X9.31 (PART 1), American National Standard for Financial Services – Public key cryptography using RSA for the financial services industry – Part 1: The RSA signature algorithm, draft, 1995
- [4] ANSI X9.31 (PART 2), American National Standard for Financial Services – Public key cryptography using RSA for the financial services industry – Part 2: Hash algorithms for RSA”, draft, 1995.
- [5] ANSI X9.44, Public key cryptography using reversible algorithms for the financial services industry: Transport of symmetric algorithm keys using RSA, draft, 1994.
- [6] *Block Ciphers – Analysis, Design and Applications*, PhD thesis, Computer Science Department, Aarhus University (Denmark), 1994.
- [7] Blum, L., M. Blum and M. Shub. 1983. Comparison of Two Pseudo-Random Number Generators. *Advances in Cryptology: CRYPTO '82 Proceedings*. Plenum Press: New York. 61-78.
- [8] Blum, M. and S. Micali. 1984. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*. 13: 850-864.
- [9] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, July–Oct. 1948
- [10] C. E. Shannon, A mathematical theory of cryptography, Tech. Rep.MM 45-110-02, Bell Labs. Tech. Memo., Sept. 1, 1945.
- [11] C. E. Shannon, Communication theory of secrecy systems, *Bell Syst. Tech. J.*, vol. 28, pp. 656–715, Oct. 1949.
- [12] Carl H. Meyer and Stephen M. Matyas, *Cryptography: A New Dimension in Computer Data Security*, John Wiley & Sons, New York, 1982.
- [13] Chor, B., O. Goldreich and S. Goldwasser. 1985. The Bit Security of Modular Squaring Given Partial Factorization of the Modulos. *Advances in Cryptology: CRYPTO '85 Proceedings*. 448-457. Springer-Verlag: Berlin / New York.
- [14] D. Coppersmith, M. Franklin, J. Pata-Rin, And M. Reiter, Low-exponent RSA with related messages, *Advances in Cryptology–EUROCRYPT '96 (LNCS 1070)*, 1–9, 1996.
- [15] D.W. Davies and W.L. Price, *Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronics Funds Transfer*, Second Edition, John Wiley & Sons, New York, 1984, 1989.
- [16] Data Encryption Standard, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C. (January 1977).





- [17] Díaz de León Santiago, J.L., Yáñez Márquez, C. & **Salinas Salinas, M.**, (2003). *Criptografía, Criptología y Criptoanálisis*, IT-178, Serie Azul, ISBN 970-36-0037-9, CIC-IPN, México.
- [18] Dorthy Elizabeth Robling Denning, *Cryptography and Data Security*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.
- [19] Douglas R. Stinson, *Cryptography: Theory and Practice*, CRC Press, Boca Raton, 1995.
- [20] Fúster Sabater, Amparo., De La Guía Martínez, Dolores., Hernández Encinas, Luis., Montoya Vitini, Fausto. & Muñoz Masqué, Jaime., (2001) *Técnicas Criptográficas de Protección de Datos*. ISBN: 970-15-0602-2. 2ª Edición, Editorial Alfaomega-Ra-Ma
- [21] Galaviz Casas J. y Magidín Arturo. *Introducción a la Criptología*, Universidad Nacional Autónoma de México.
- [22] García Tomás, Jesús., Ferrando, Santiago. & Piattini, Mario. *Redes para Proceso Distribuido*. 2ª. Edición. Ed. Alfaomega Ra-ma. 2001. ISBN 970-15-0706-1.
- [23] Goldreich, O., H. Krawczyk and M. Luby. 1988. On the Existence of Pseudorandom Generators (Extended Abstract). *29th IEEE Symposium on the Foundations of Computer Science*. 12-24.
- [24] Gómez Vietes, A., (2007). *Enciclopedia de la Seguridad Informática*, Alfaomega Grupo Editor, Primera Edición, ISBN 978-970-15-1266-1, México.
- [25] H. Feistel, Block cipher cryptographic system, U.S. Patent # 3,798,359, 19 Mar 1974. [386], "S.
- [26] H. Feistel, W.A. Notz, And J.L. Smith, "Some cryptographic techniques for machine-to-machine data communications", *Proceedings of the IEEE*, 63 (1975), 1545–1554.
- [27] Hankerson, D., Menezes, A. J. y Vanstone, S., *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2004.
- [28] Kahn, David. *The Codebreakers*, Macmillan Publishing Company, New York, 1967.
- [29] Levin, L. 1985. One-Way Functions and Pseudorandom Generators. *17th ACM Symposium on Theory of Computing*. 363-365.
- [30] Long, D. and A. Wigderson. 1983. How Discreet is the Discrete Log? *15th ACM Symposium on Theory of Computing*. 413-420.
- [31] Lucena López, Manuel José. *Criptografía y Seguridad en Computadores*. Dpto. de Informática Universidad de Jaén. Edición virtual. España. 2002. (en línea) <http://www.kriptopolis.com>
- [32] Maiorano, Ariel H., (2009). *Criptografía: técnicas de desarrollo para profesionales*. Alfaomega Grupo Editor Argentino, Primera Edición, ISBN 978-987-23113-8-4, Argentina.
- [33] Manuel Lázaro, Antoni. & Del Río Fernández, Joaquín. (2005) *LabVIEW 7.1. Programación Gráfica para el Control de Instrumentación*. ed. Thomson. ISBN 84-9732-391-2.
- [34] Menezes, A., Van Oorschot, P. & S. Vanstone., *Handbook of Applied Cryptography*, CRC Press, 1996.
- [35] Miles E. Smid and Dennis K. Branstad, (1992) *The Data Encryption Standard: Past and Future*, in Gustavus J. Simmons, ed., *Contemporary Cryptography: The Science of Information Integrity*, IEEE Press.





- [36] N. Demytko, A new elliptic curve based analogue of RSA, *Advances in Cryptology—EUROCRYPT '93* (LNCS 765), 40–49, 1994.
- [37] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 26 de noviembre de 2001. Disponible en <http://csrc.nist.gov/cryptval/>.
- [38] National Institute of Standards and Technology. Data Encryption Standard (DES). Federal Information Processing Standards Publication 146-3, 25 de octubre de 1999. Disponible en <http://csrc.nist.gov/cryptval/> en línea en: <http://www.itl.nist.gov/fipspubs/fip46-2.htm>.
- [39] Nisan, N. and A. Wigderson. 1988. Hardness vs. Randomness (Extended Abstract). 29th IEEE Symposium on Foundations of Computer Science. 2-11.
- [40] P. Horster And H. Petersen, Generalized ElGamal signatures (in German), *Sicherheit in Informationssystemen, Proceedings der Fachtagung SIS'94*, 89–106, Verlag der Fachvereine Zürich, 1994.
- [41] R. Anderson, Practical RSA trapdoor, *Electronics Letters*, 29 (May 27, 1993), 995.
- [42] Reif, J. and D. Tygar. 1988. Efficient Parallel Pseudorandom Number Generation. *SIAM Journal on Computing*. 17: 404-411.
- [43] Rodríguez Penin, Aquilino. (2007) *Sistemas SCADA*. 2ª. Edición. ISBN 978-84-267-1450-3
- [44] Rojo, A. et. al.: *Computational Model for Aircraft's Takeoffs Pattern Recognition*, *Lecture Notes in Computer Science*, Springer-Verlag, Germany Vol. 5197, pp. 14-21, sep. 2008.
- [45] *RSA and Rabin functions: Certain parts are as hard as the whole*, *SIAM Journal on Computing*, 17 (1988), 194–209. An earlier version appeared in [63].
- [46] RSA Laboratories, *The Public-Key Cryptography Standards – PKCS #11: Cryptographic token interface standard*, RSA Data Security Inc., Redwood City, California, April 28 1995.
- [47] Sánchez, L.P., et al.: “Sistema Distribuido de Monitoreo Permanente Experimental, de Ruidos Ambientales en Puntos Críticos del Centro Histórico de La Ciudad de México (SIMAR-CH)”. 2008-2009. IPN-ICYTDF
- [48] Sánchez, L.P., et al.: Noise pattern recognition of airplanes taking off task for a monitoring system. *Lecture Notes in Computer Science*, Springer-Verlag. Vol. 4756, pp. 831-840, 2007.
- [49] Sánchez, L.P., Rojo, A. and Pogrebnnyak, O. Noise Monitoring of Aircrafts Taking off based on Neural Model, *ETFA 2009 - 14th IEEE International Conference on Emerging Technologies and Factory Automation*, ISBN 978-1-4244-2728-4, sep 22-26. 2009, Palma de Mallorca, España.
- [50] Saucedo Flores S; *Sistemas de Control Distribuido*; Escuela Superior de Ingeniería Mecánica y Eléctrica. IPN; 1998.
- [51] Schneier Bruce. *Applied Cryptography*. John Wiley and Sons. 2º Edición. ISBN: 047-1117-09-9. EE.UU. Marzo 1995.
- [52] Shamir, A. 1981. On the generation of cryptographically strong pseudo-random sequences. *8th International Colloquium on Automata, Language, and Programming*. 544-550.





- [53] Simon Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*. Doubleday, 1999.
- [54] Sklavos, Nicolas. & Zhang, Xinmiao.(2007) *Wireless Security and Cryptography. Specifications and Implementations*. CRC Press. ISBN-13: 978-0-8493-8771-5.
- [55] Stallings, William, (2005). *Fundamentos de Seguridad en Redes. Aplicaciones y Estándares*, Pearson Prentice Hall, Segunda edición, ISBN 84-205-4002-1, España.
- [56] Stallings, William. *Comunicaciones y Redes de Computadores*. Editorial Pearson Prentice Hall. Séptima edición. ISBN: 84-205-4110-9
- [57] Stallings, William. *Network And Internetwork Security*. 2º Edición. Prentice Hall. ISBN 0-13-869017-0. EE.UU. 1998.
- [58] T. ElGamal, *Cryptography and logarithms over finite fields*, PhD thesis, Stanford University, 1984.
- [59] Tanenbaum, Andrew. *Redes de Computadoras*. Prentice Hall. 1990
- [60] The Public-Key Cryptography Standards (PKCS), RSA Data Security Inc., Redwood City, California, November 1993 Release.[1073] A.D. R.
- [61] Vazirani, U. and V. Vazirani. 1983. Trapdoor Pseudo-random Number Generators with Applications to Protocol Design. 24th IEEE Symposium On The Foundations Of Computer Science. 23-30.
- [62] Vazirani, U. and V. Vazirani. 1985. Efficient and Secure Pseudo-Random Number Generation. (extended abstract) *Advances in Applied Cryptology: Proceedings of CRYPTO 84*. 193-202. Springer-Verlag: Berlin / New York
- [63] W. Alexi, B. Chor, O. Goldreich, And C.P. Schnorr, RSA/Rabin bits are $1/2 + 1/\text{poly}(\log n)$ secure, *Proceedings of the IEEE 25th Annual Symposium on Foundations of Computer Science*, 449–457, 1984.
- [64] Winternitz, R. 1984. A Secure One-Way Hash Function Built from DES. 1984 IEEE Symposium on Secrecy and Privacy. 88-90.

Referencias a recursos electrónicos

- [65] Sitio Web oficial de internacional PGP: <<http://www.pgpi.org>> [Consulta: enero 2011]
- [66] Sitio Web oficial de la Alianza OpenPGP en la siguiente dirección: <<http://www.openpgp.org/>> [Consulta: enero 2011]
- [67] Sitio Web oficial de The GNU Privacy Guard en: <<http://www.gnupg.org/>> [Consulta: enero 2011]
- [68] Descripción ilustrada del algoritmo simétrico DES: <<http://www.aci.net/kalliste/des.htm>> [Consulta: enero 2011]
- [69] Descripción ilustrada del algoritmo simétrico DES: *The DES Algorithm Illustrated by J. Orlin Grabbe* - <<http://orlingrabbe.com/des.htm>> [Consulta: 2010]
- [70] <http://www.sensacd.com/sistema_digital_monitoreo_control.asp> [Consulta: agosto 2010]
- [71] <<http://ni.com/legal/termsofuse/unitedstates/us/>> [Consulta: enero 2011]
- [72] Sitio Web de National Instruments: <<http://ni.com>> [Consulta: enero 2011]
- [73] <<http://www.tracnova.com>> [Consulta: enero 2011]





- [74] <<http://www.abb.com.mx>> [Consulta: enero 2011]
- [75] <<http://www51.honeywell.com/honeywell/>> [Consulta: enero 2011]
- [76] <<http://www.foxboro.com> o <http://iom.invensys.com/EN/Pages/Foxboro.aspx>> [Consulta: enero 2011]
- [77] <<http://www.emerson.com>> [Consulta: enero 2011]
- [78] <<http://www.yokogawa.com>> [Consulta: enero 2011]
- [79] <<http://www.siemens.com.mx>> [Consulta: enero 2011]





APÉNDICES

Programación mediante diagrama a bloques

A continuación se presentan la parte inicial del sistema en la que se toman los valores de las *Variables a Cifrar* desde la base de datos del sistema y con estas se genera una cadena que se llamará *MENSAJE SIMPLE* (Fig. 66) el cual es la entrada inicial del DES. Para generar dicha cadena se tienen que juntar la variables *VceUI*, *VceUC*, *ParConvUI*, *Historico infinito VceUI* y *AParAlarma*, siendo que se seleccionaron estas variables como muestra para ser enviadas por el canal de información considerando que pueden ser más o menos u otras.

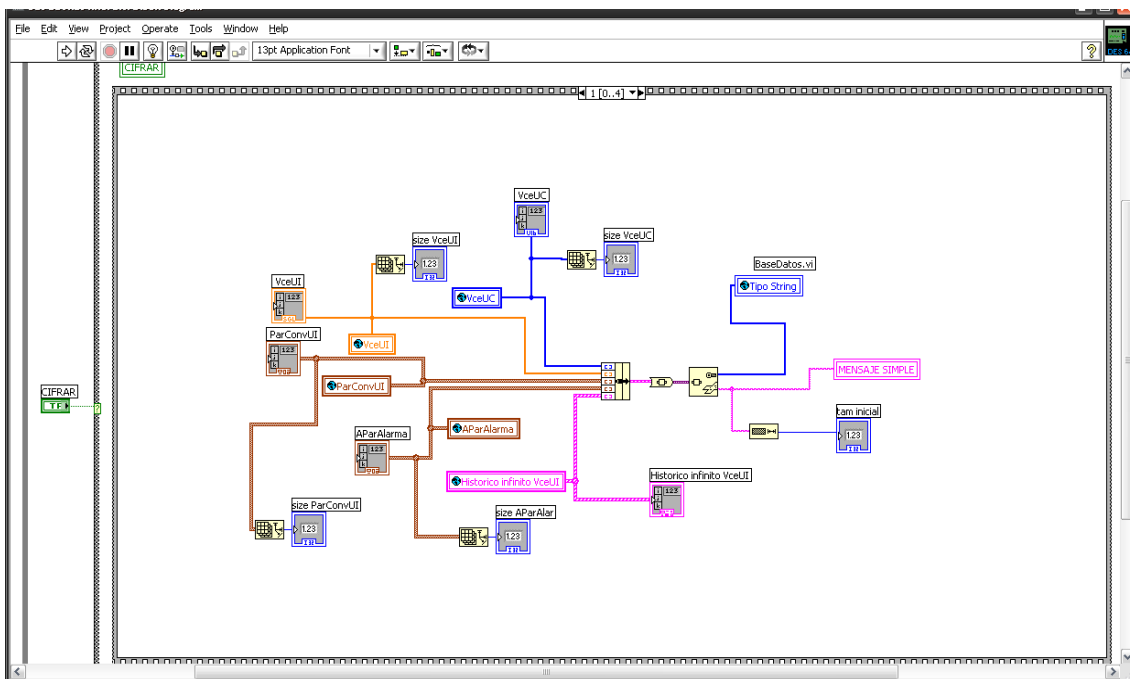


Fig.66. Diagrama encargado de obtener las variables a cifrar y convertirlas a la cadena mensaje simple.

Entonces es necesario dividir el *MENSAJE SIMPLE* en bloques de tamaño fijo de 64 bits, por lo tanto cuando la cadena no sea divisible entre 8, la cadena será completada a 8 bytes, considerando que la cadena sea contada y se agrega 1byte a la vez hasta completar la cadena para tener solo octetos y dando a la salida del ciclo una *cadena de octetos*, como se muestra en la Fig.67.



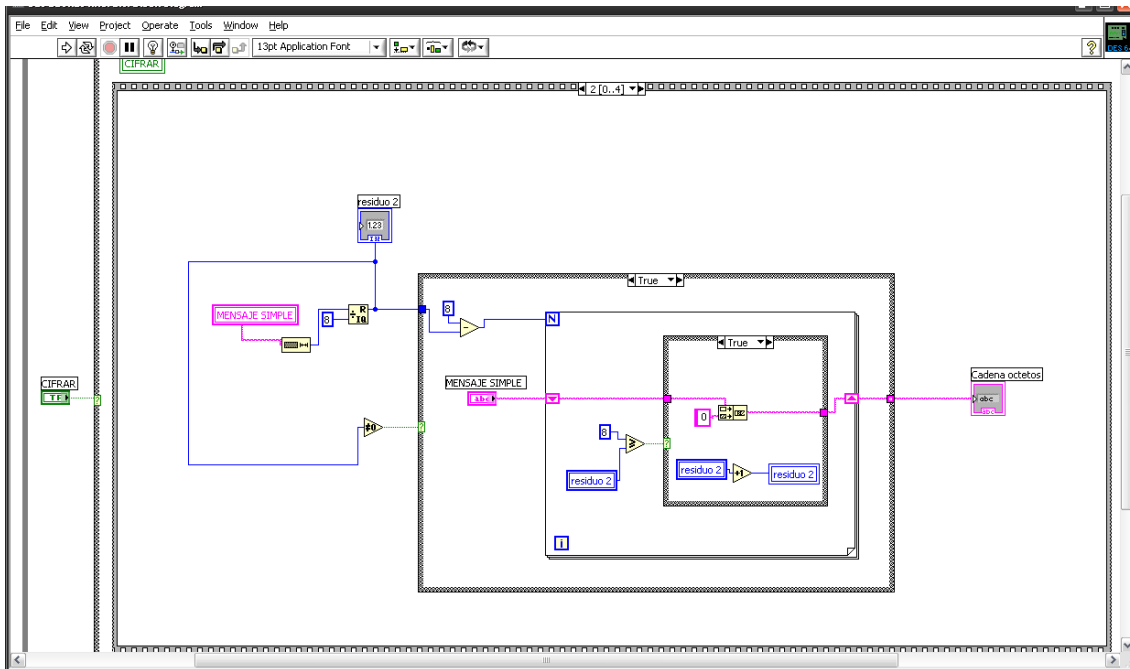



Fig.67. SubVI para dividir el mensaje simple en bloques de tamaño fijo de 64 bits.

A continuación se observan las variables requeridas para que comience a operar el DES, ya que por un lado se proporciona la *clave en bloques de 64 bits* y también la *cadena octetos* en bloques de 64 bits para que a su salida nos dé *MENSAJES CIFRADOS EN BLOQUES* de 64 bits cada uno, con esto se da comienzo al proceso de cifrado.

Por un lado se maneja la parte de la *CLAVE* hexadecimal que se introduce en octetos y por otro lado se introduce la cadena de octetos pero antes son convertidos a binario y se toman

en bloques de 64 bits que se introducen en el subVI que tiene este ícono  al final se tiene el *MENSAJE CIFRADO* y el arreglo de todos los *MENSAJES CIFRADOS EN BLOQUES*, como se muestra en la Fig.68.





MENSAJES SIMPLES DE 64 BITS A LA ENTRADA DE DES

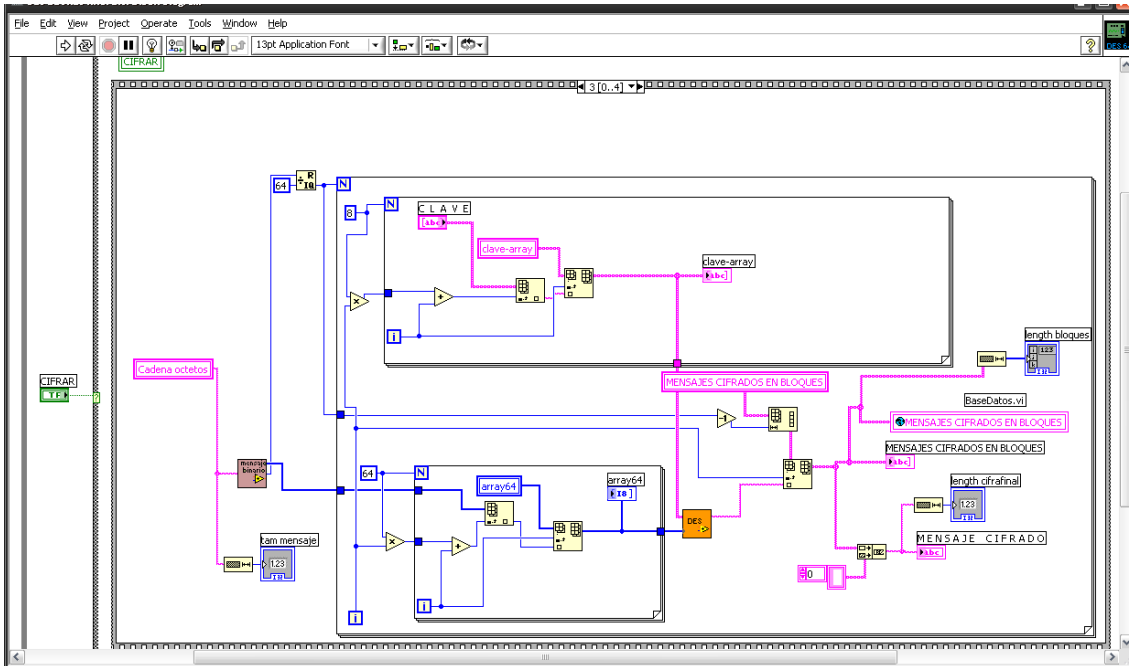


Fig.68. SubVI general del DES con sus entradas por un lado el mensaje simple y la clave y su salida se obtienen los mensajes cifrados en bloques.

Se pueden ver detalles de cómo funciona DES en cada una de sus 19 etapas, mostrando la primera etapa, la permutación inicial IP con la cual son cambiados de posición los bits de la *Entrada Binaria* obteniendo un arreglo llamado IP (Fig.69).

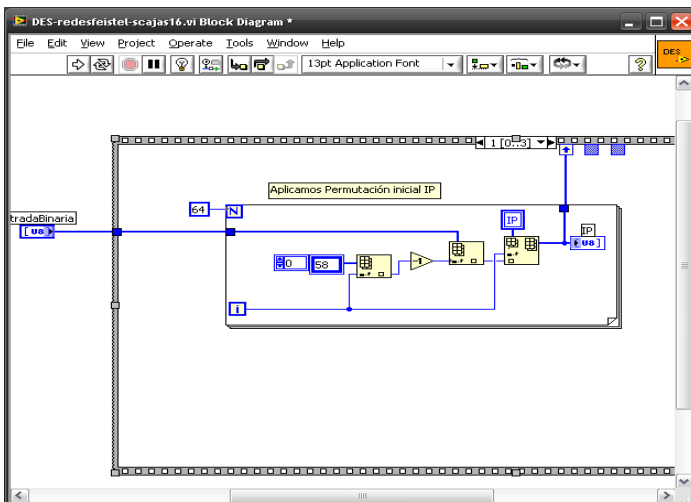



Fig.69. Permutación inicial IP a partir de la entrada de datos binarios.





En la siguiente etapa se aplicaran las 16 iteraciones de las Redes de Feistel (Fig. 70). La variable de entrada es *entradaBinaria* a la que dividimos en dos partes de 32 bits cada una, a la primera le llamaremos L_n y la segunda R_n . La función f de la red de Feistel es aplicada a R_n , se compone de una permutación de expansión (E) en la Tabla 7, que convierte el bloque correspondiente de 32 bits en uno de 48. Después realiza una OR-Exclusiva con el valor K_i (clave correspondiente a cada iteración) también de 48 bits, se aplica ocho S-Cajas de 6×4 bits de acuerdo con la Tabla 10 mostrada en capítulos anteriores y su

implementación se pone a continuación en la Fig.71 y su ícono es , y lo último de las iteraciones consiste en efectuar una permutación (P) mostrada con anterioridad en la Tabla 8, su implementación está en la Fig.72.

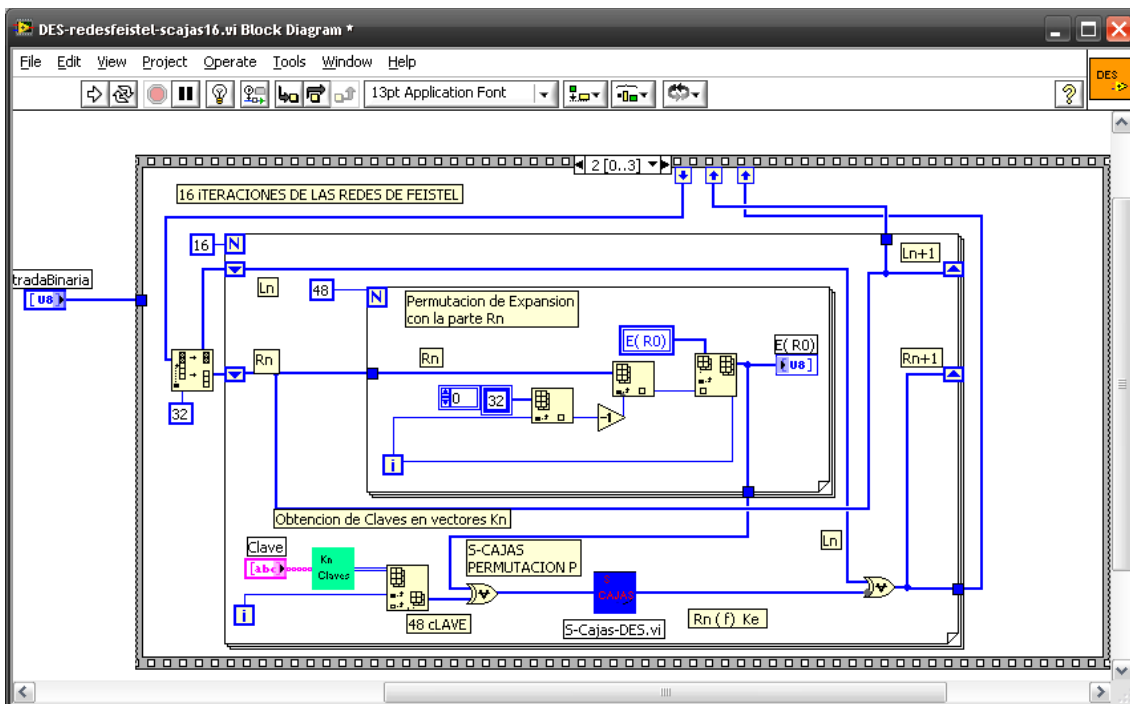


Fig.70. SubVI para realizar las redes de Feistel.



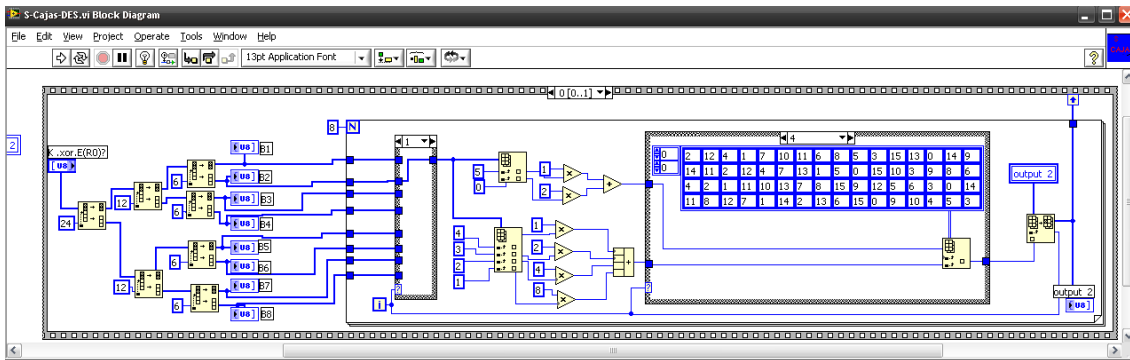


Fig.71. SubVI que muestra la sustitución de S-Cajas.

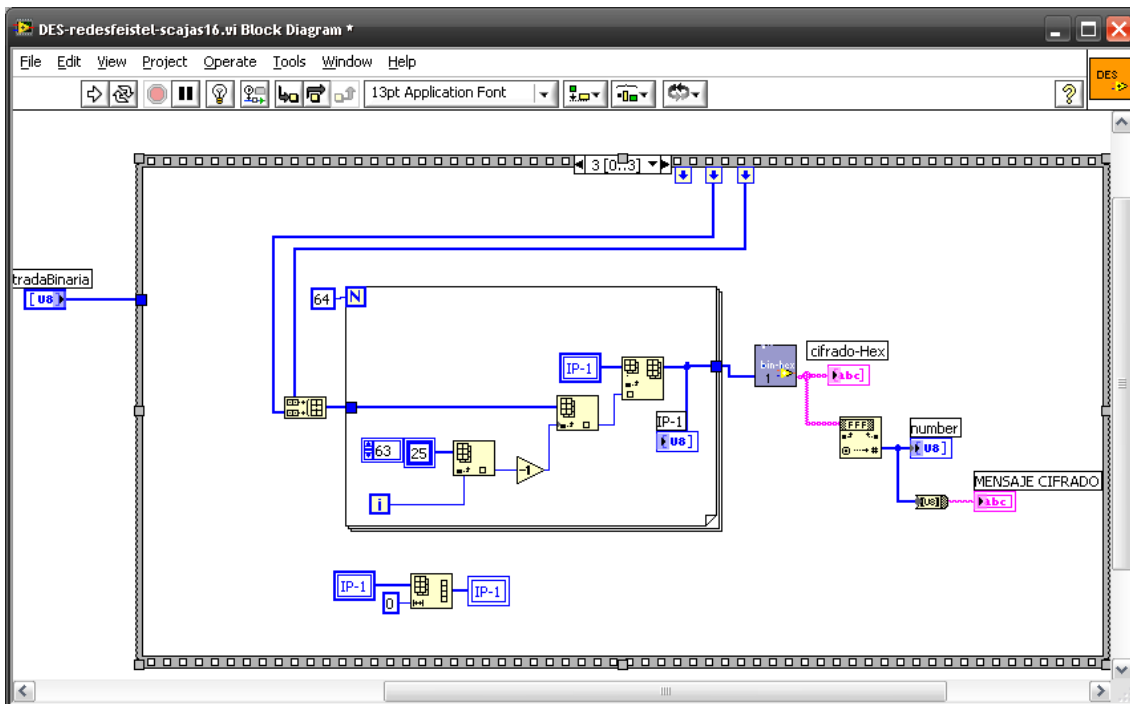


Fig.72. Última permutación IP-1 dentro de las rondas de las redes de Feistel.

Cálculo de las subclaves ke

La otra entrada que necesita DES para funcionar es la clave, Ke, de acuerdo a lo descrito del algoritmo se realiza una permutación inicial (PC-1) en la Tabla 4 sobre la clave. (Fig.73).



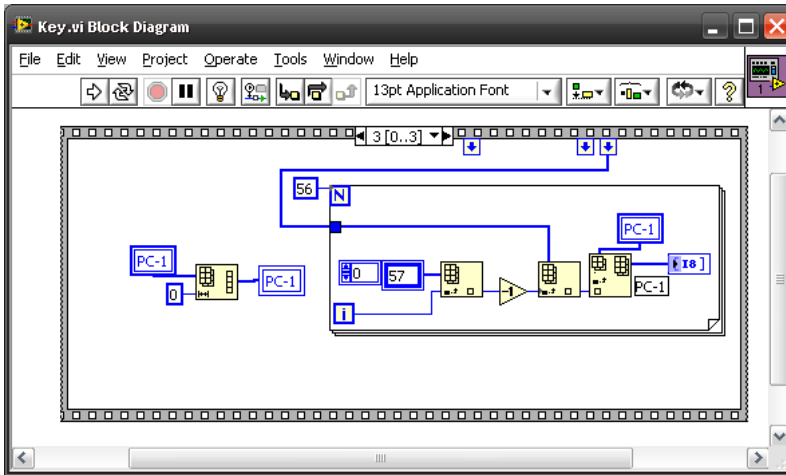


Fig.73. SubVI que aplica la permutación PC-1 a la clave Ke.

A continuación la clave obtenida, K_e , se divide en dos mitades de 28 bits, cada una de las cuales se rota a la izquierda un número de bits determinado que no siempre es el mismo, entonces K_{e_i} que se deriva de la Tabla 6, en que se observan dos casos en la cantidad de bits a ser desplazados, el primero es cuando sólo son rotados en un bit o el otro caso, desplaza o rota 2 bits a la izquierda (Fig.74).



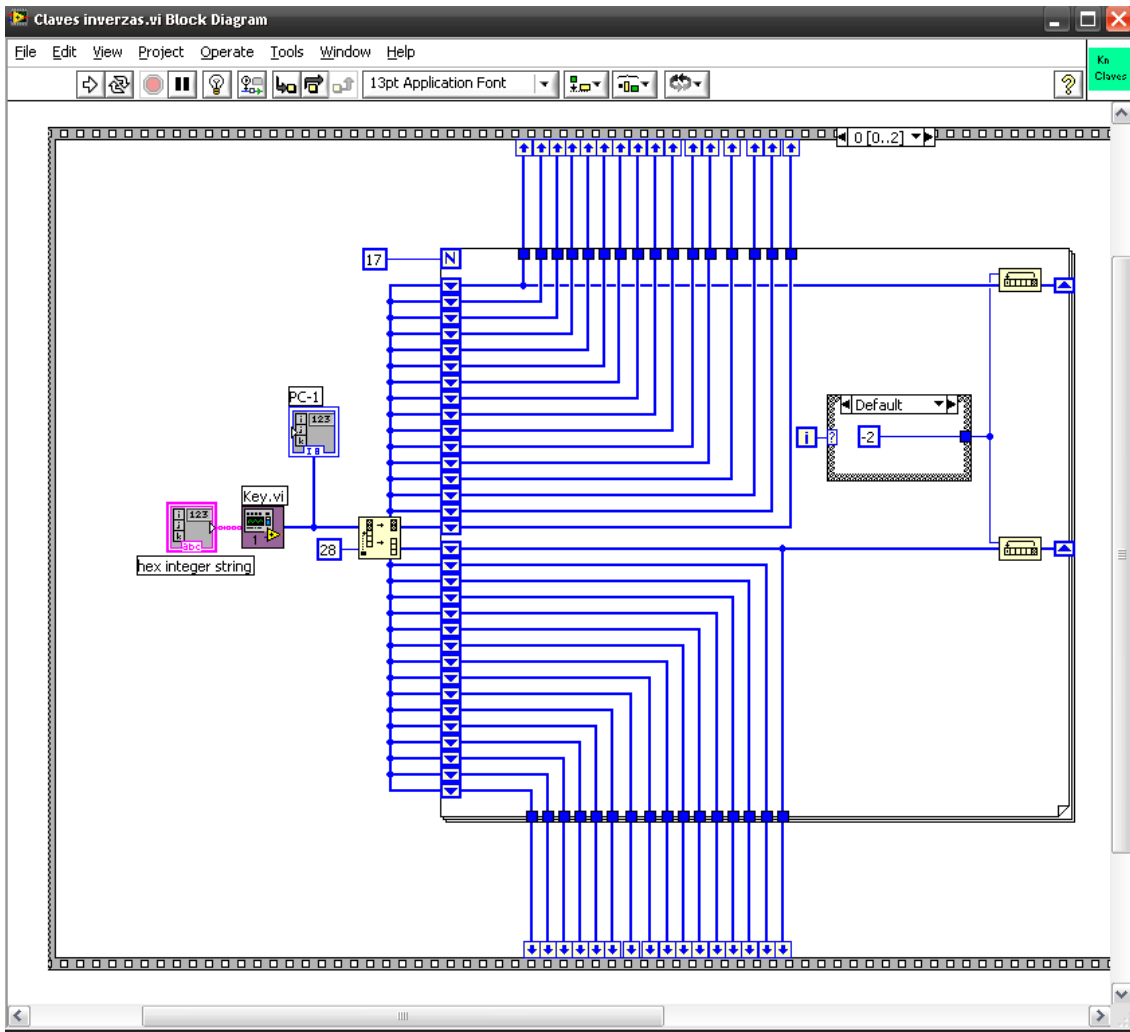


Fig.74. Rotación de un bit o dos para las claves.

Por último, para la obtención de las claves para cada ronda de las redes de Feistel, son unidas las dos mitades, para aplicarles la permutación 2 llamada PC-2, en la tabla 5, entonces observaremos que tiene de entrada 56 bits y a su salida se obtienen sólo 48 bits (Fig. 75) y (Fig. 76).



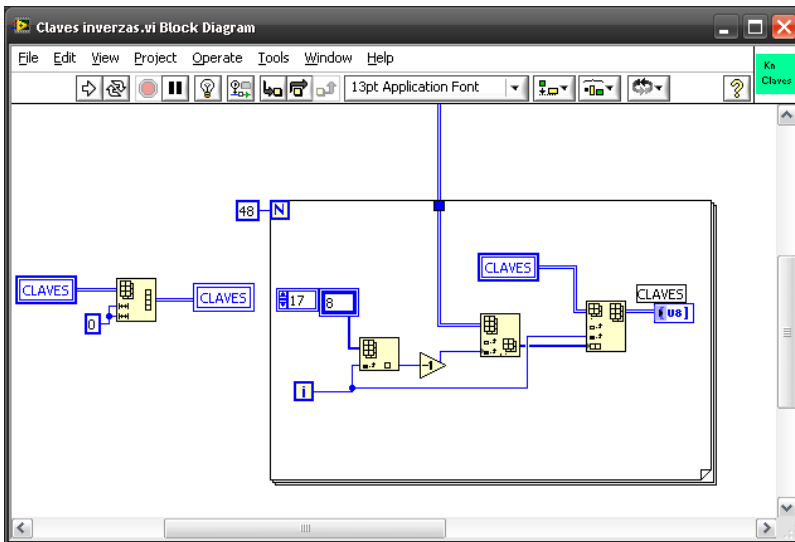


Fig.75. Permutación PC-2 aplicada al final de cada subclave.

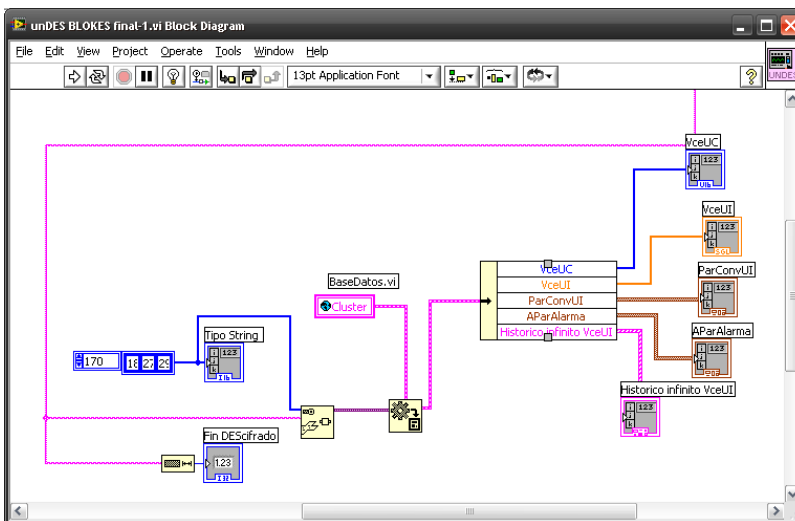


Fig.76. SubVI que muestra como se tienen los mensajes descifrados y descompactados en las variables descifradas.

Ahora sólo resta realizar la transformación inversa del algoritmo Estándar de Cifrado de Datos (DES) que consiste en utilizar el mismo algoritmo con la condición de proporcionar las claves en orden inverso. Como se muestra en la Fig.77.



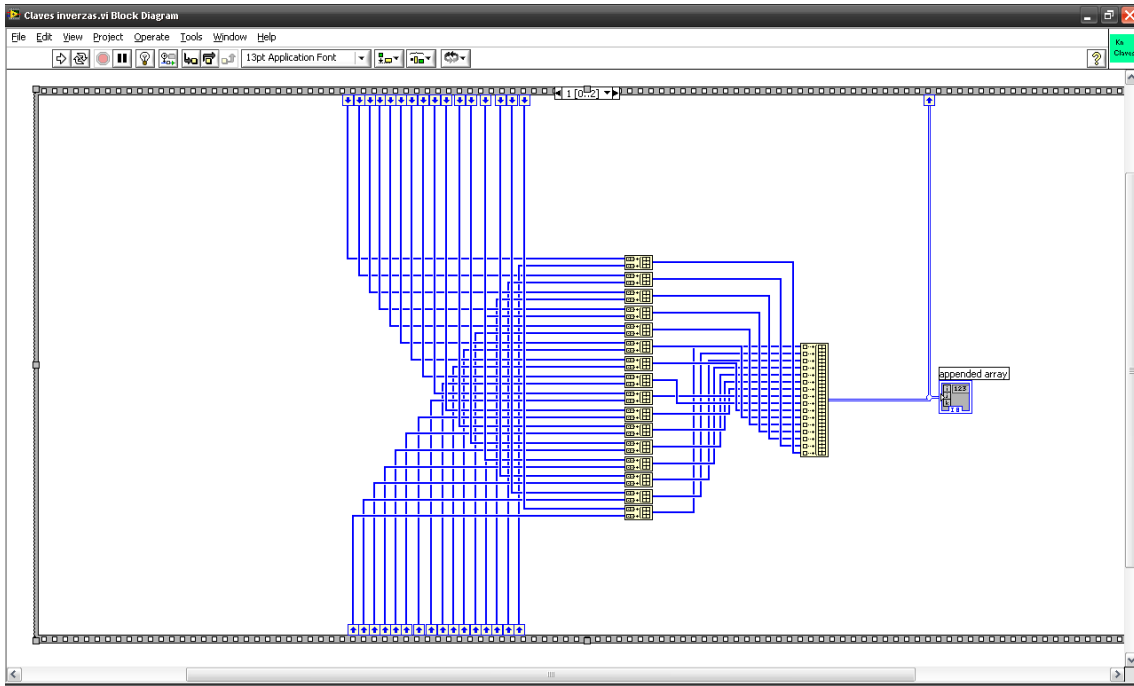


Fig.77. Generación de las claves en orden inverso para ocuparlas en el descifrado.





Archivo panel BD.vi

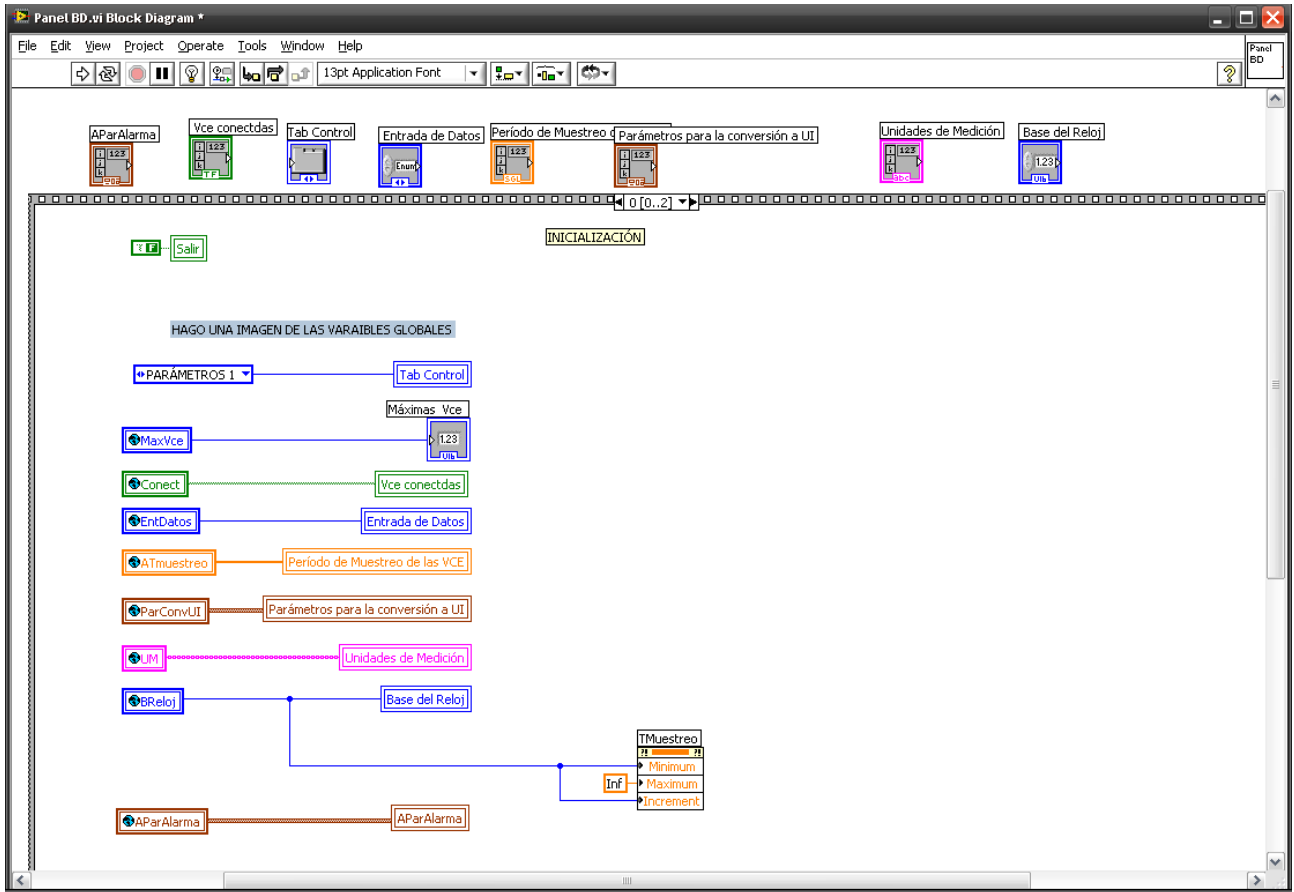


Fig. 78. Inicialización de las variables del panel de la base de datos.



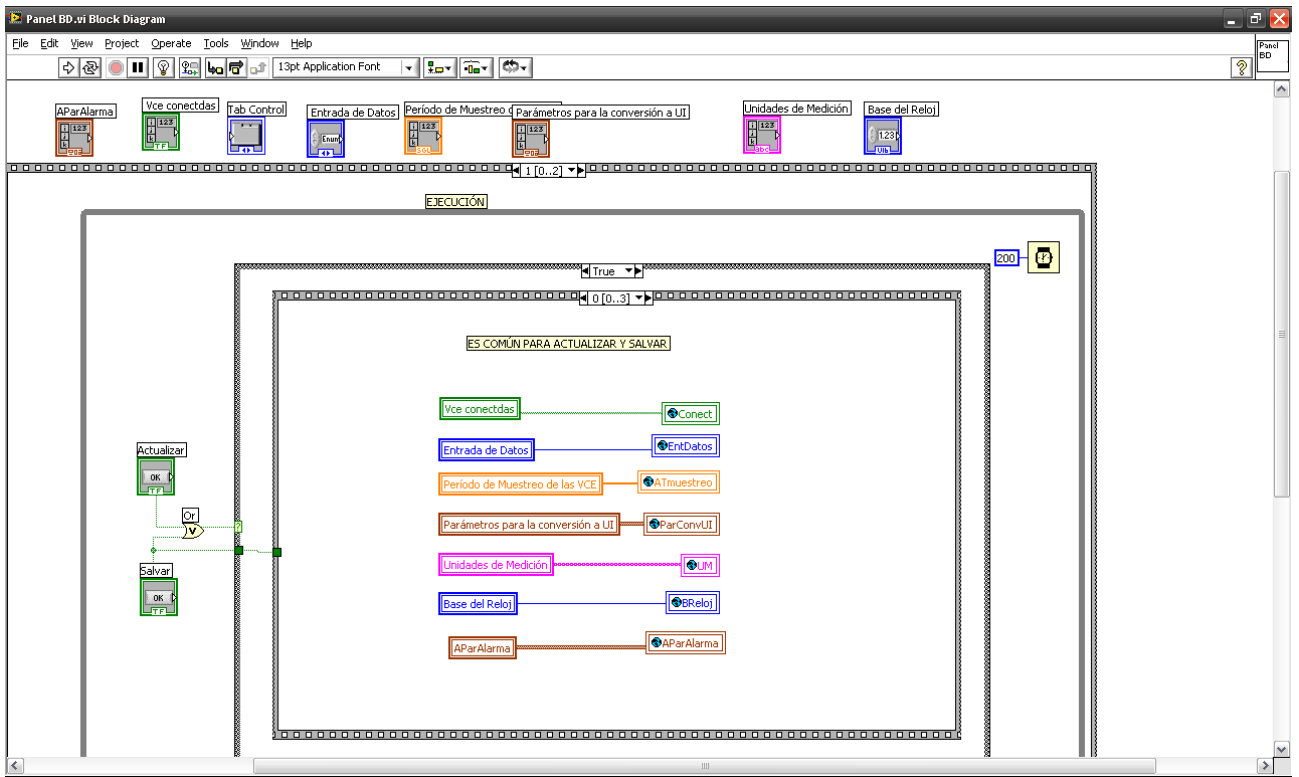


Fig.79. Actualización y guardar las variables del panel de base de datos.



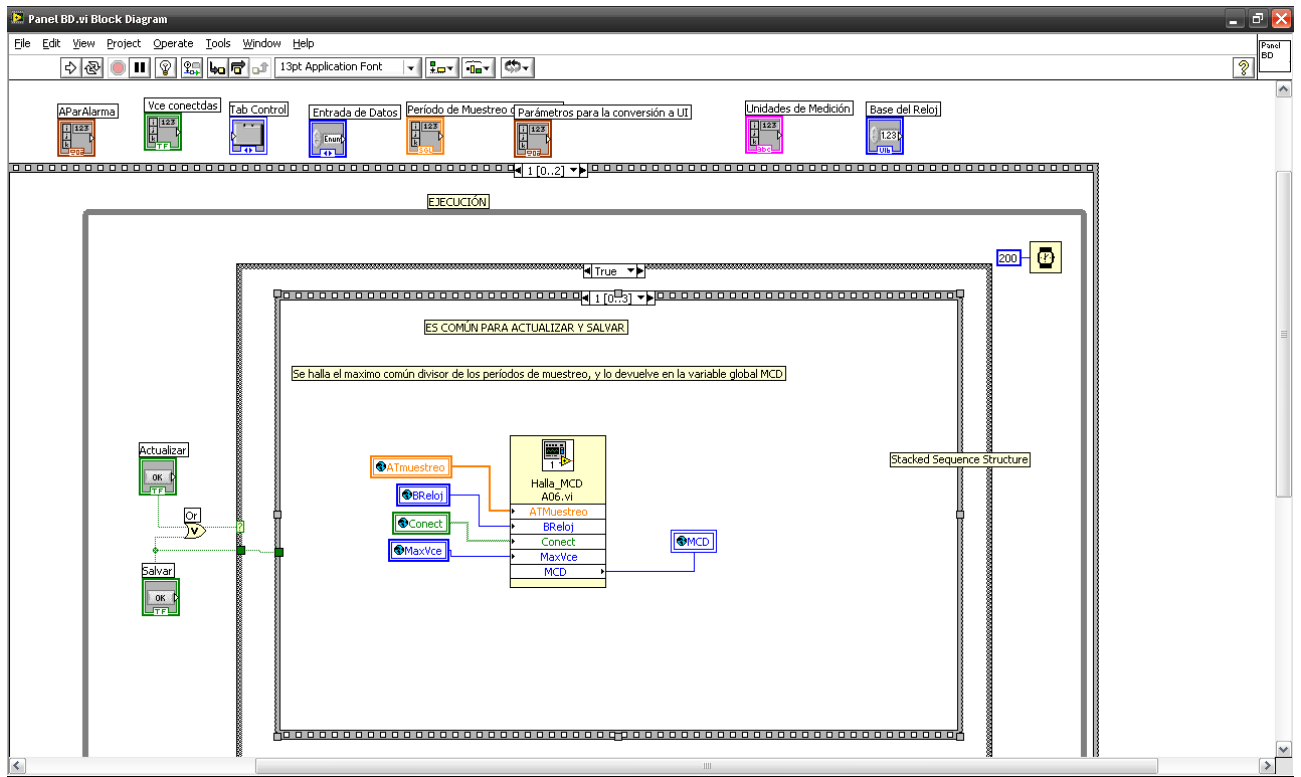


Fig. 80. Obtiene el máximo común divisor de los periodos de muestreo.



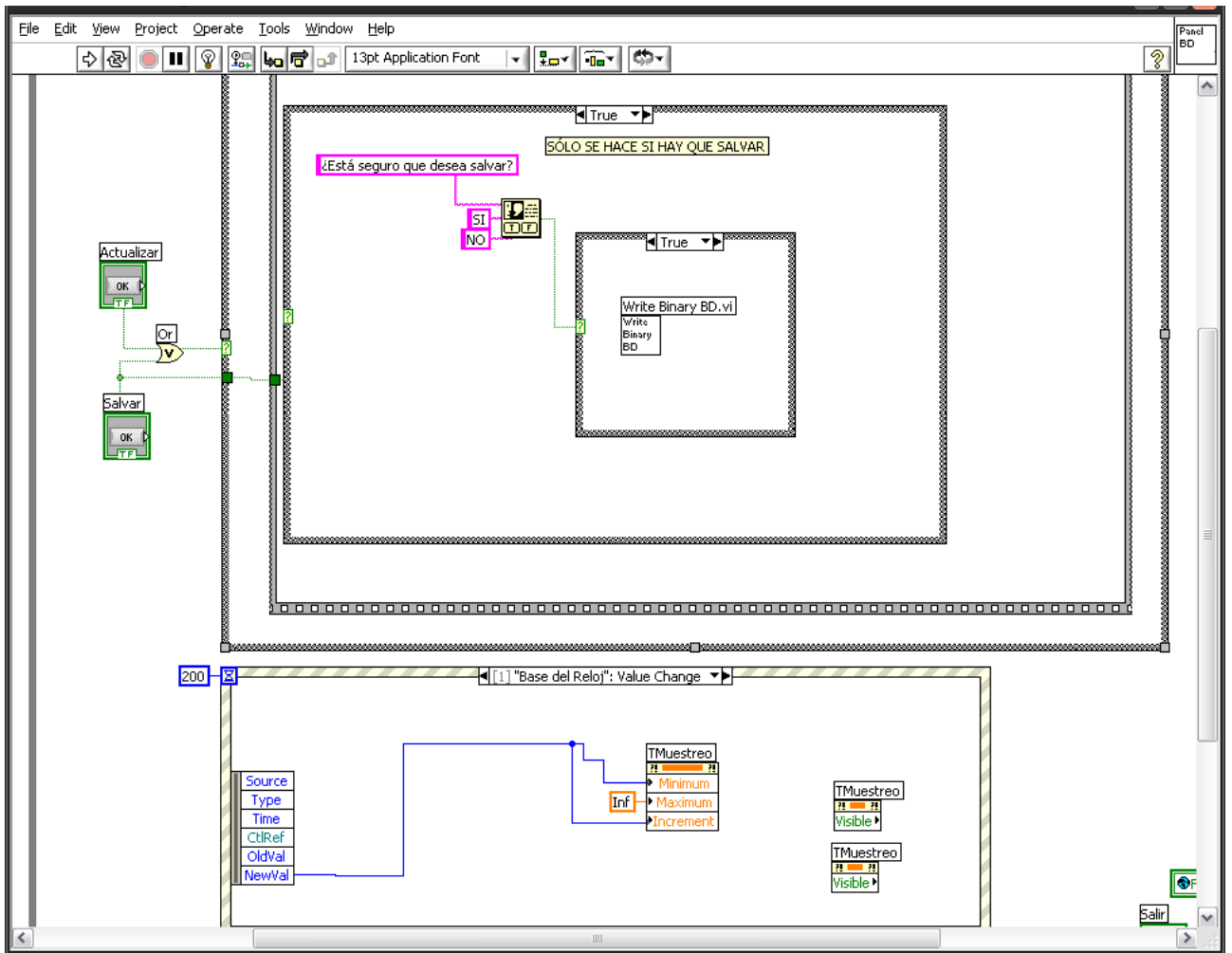


Fig. 81. Almacena el archivo de la base de datos.



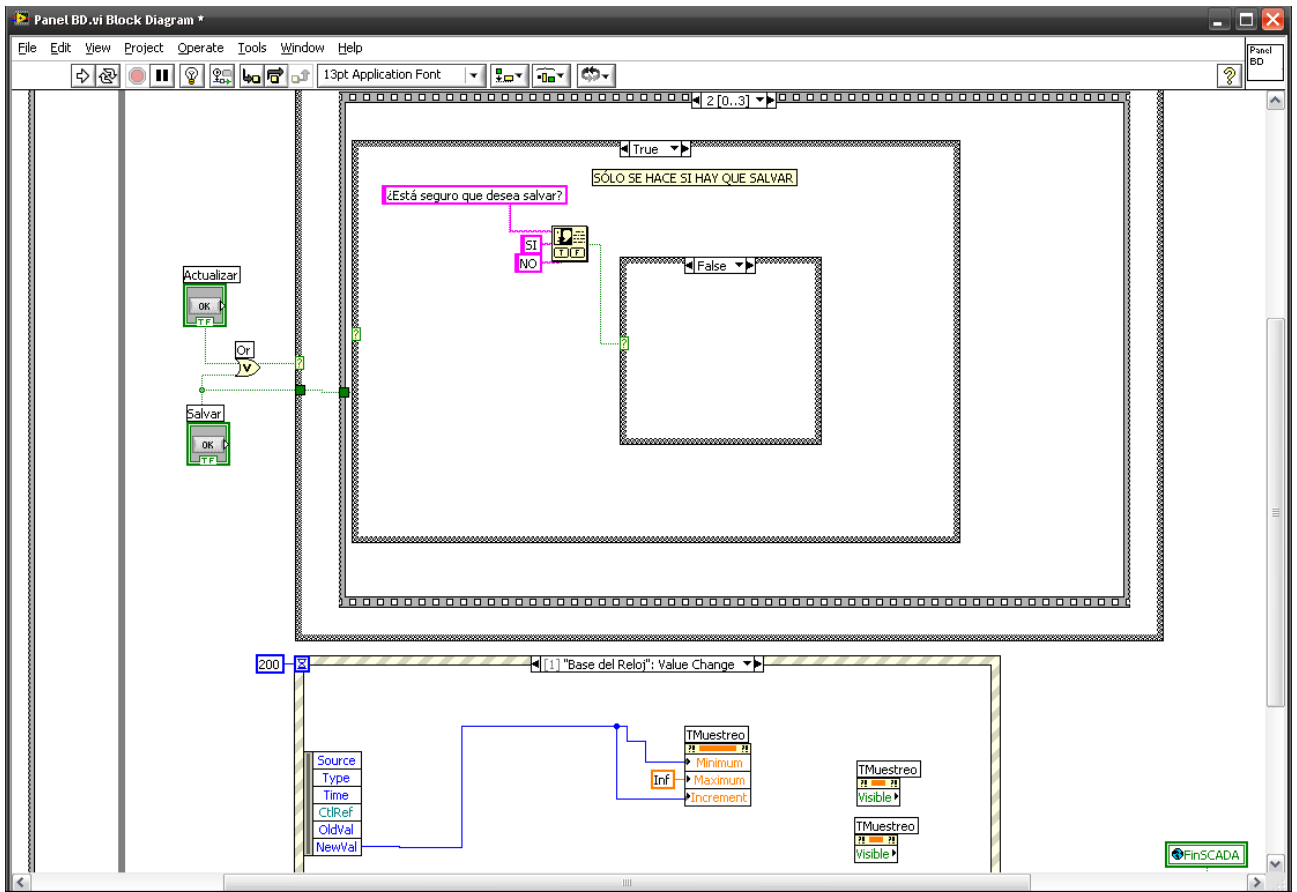


Fig. 82. Ejecuta la escritura del archivo para almacenar la base de datos



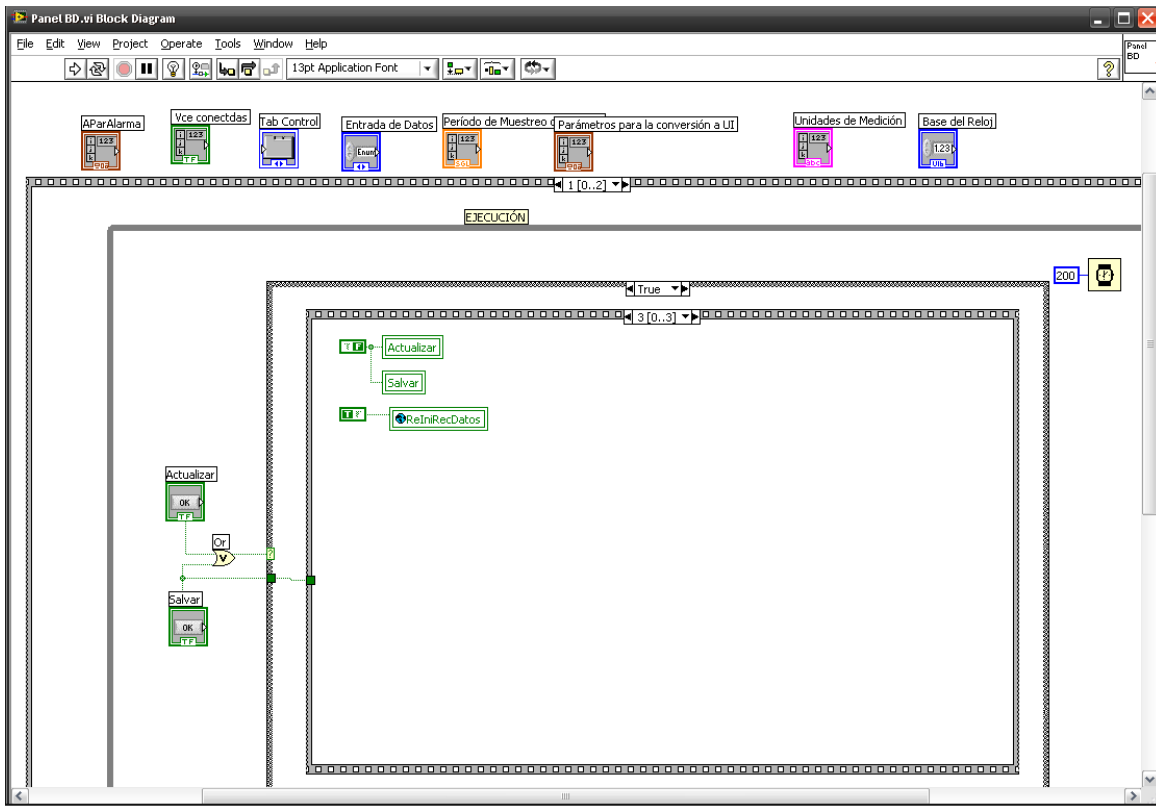


Fig. 83. Reinicializa la recolección de datos para la base de datos.





Archivo del gráfico Vce

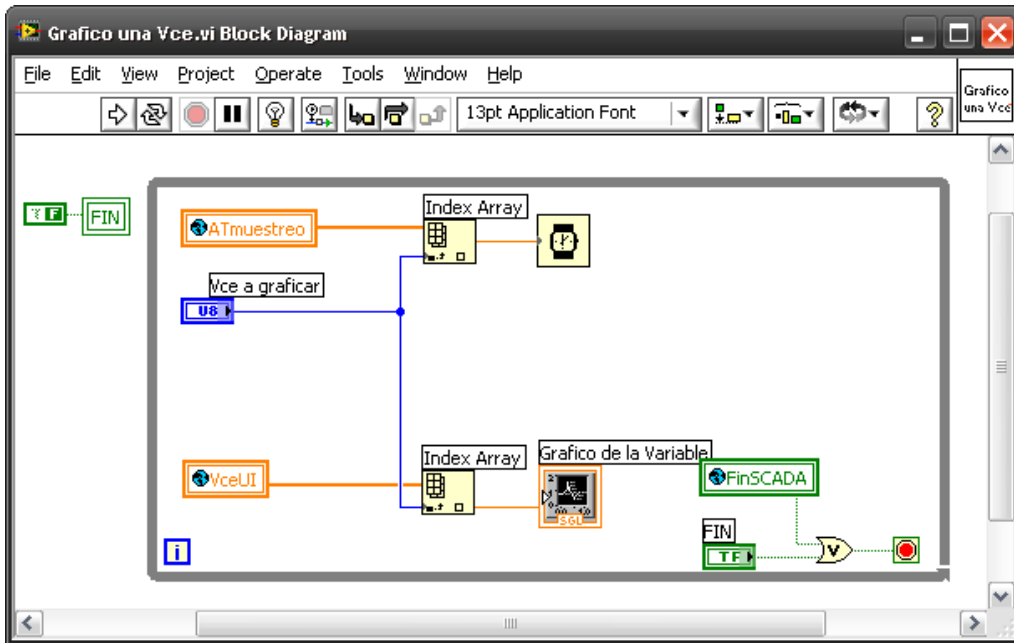


Fig. 84. Realiza la gráfica de una variable Vce.





Archivo de la visualización de la alarma

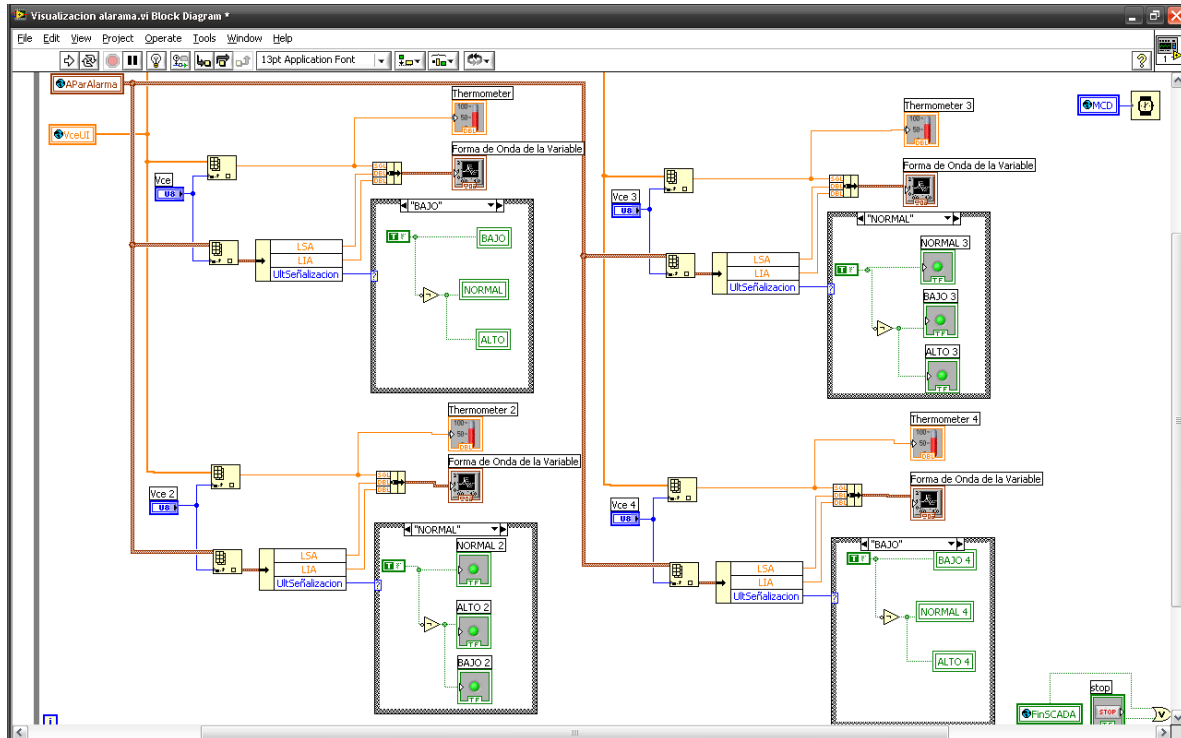


Fig. 85. Realiza las graficas para visualización de alarmas.





Archivo del gráfico histórico de una Vce

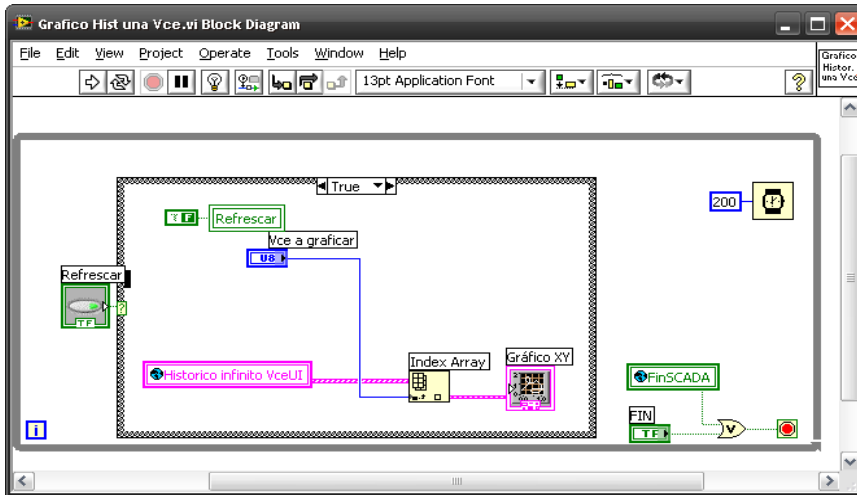


Fig. 86. Genera el archivo para el gráfico histórico de una Vce.

Archivo del RecDatos

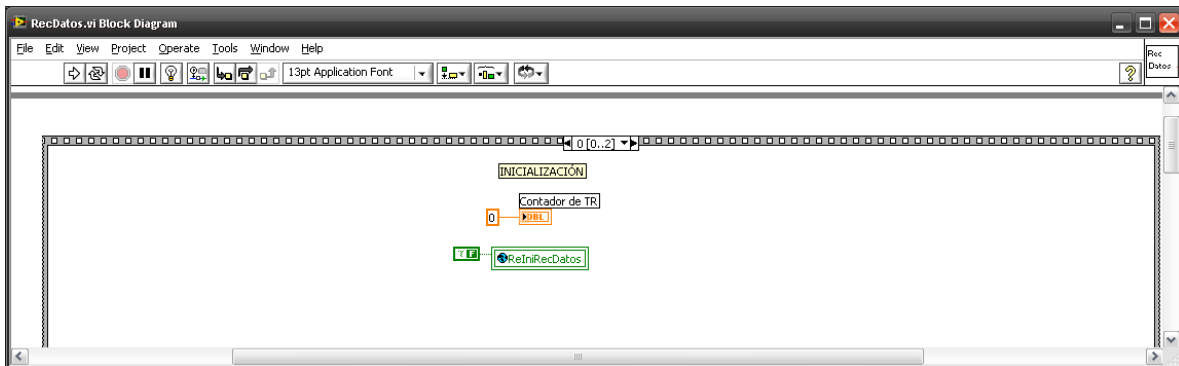


Fig.87. Realiza la recopilación de datos



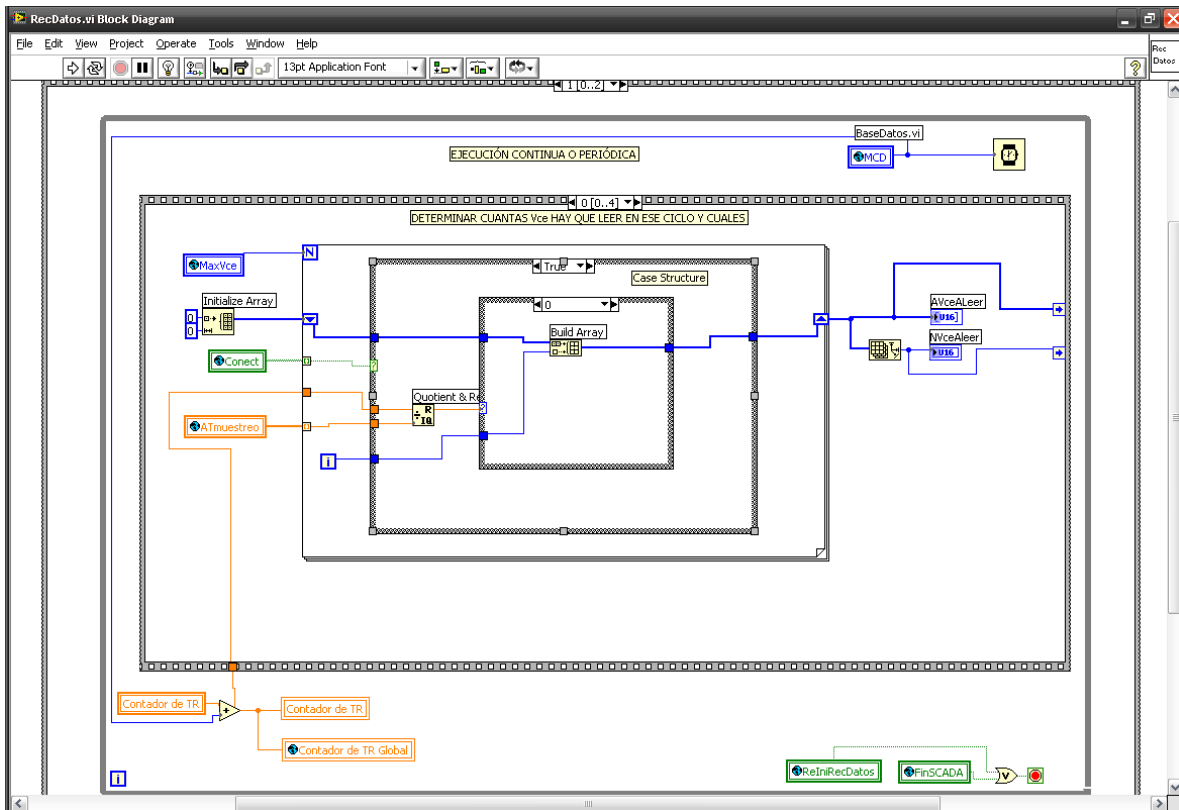


Fig. 88. Determina ¿Cuántas Vce hay que leer? y ¿Cuáles?



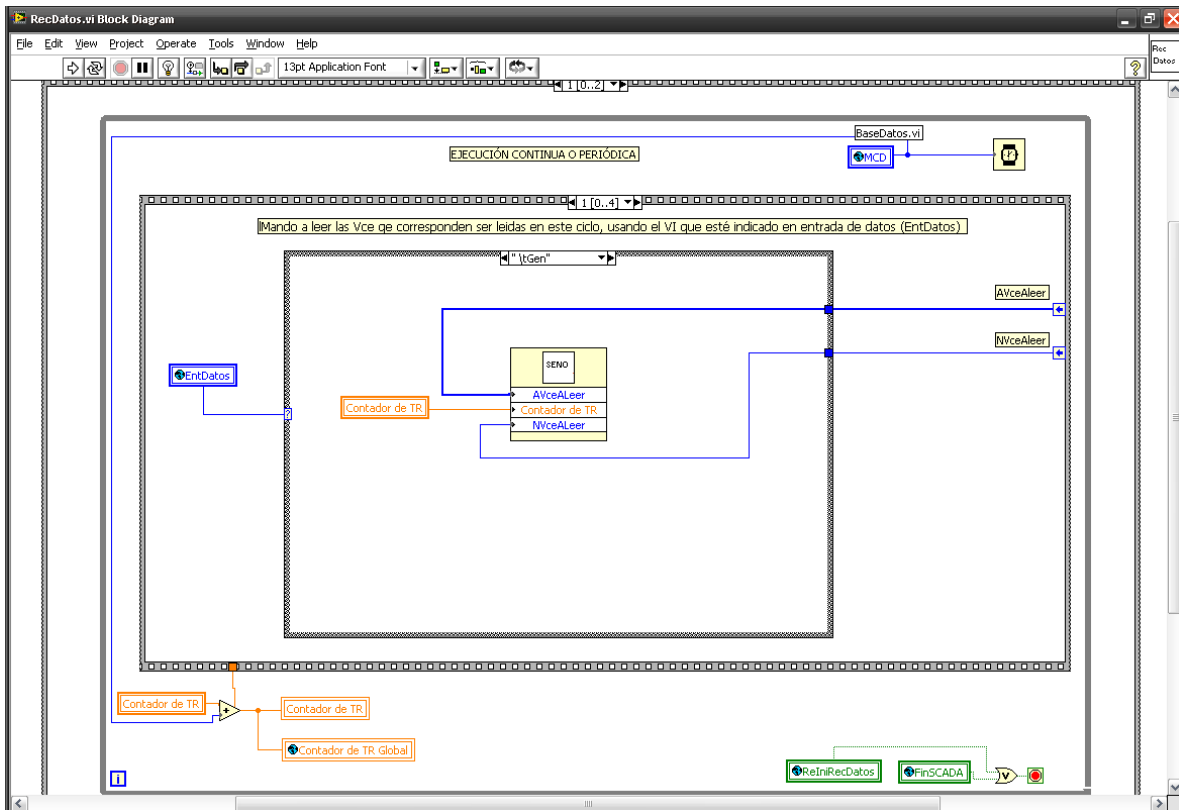


Fig. 89. Ejecuta SubVI de la opción Gen



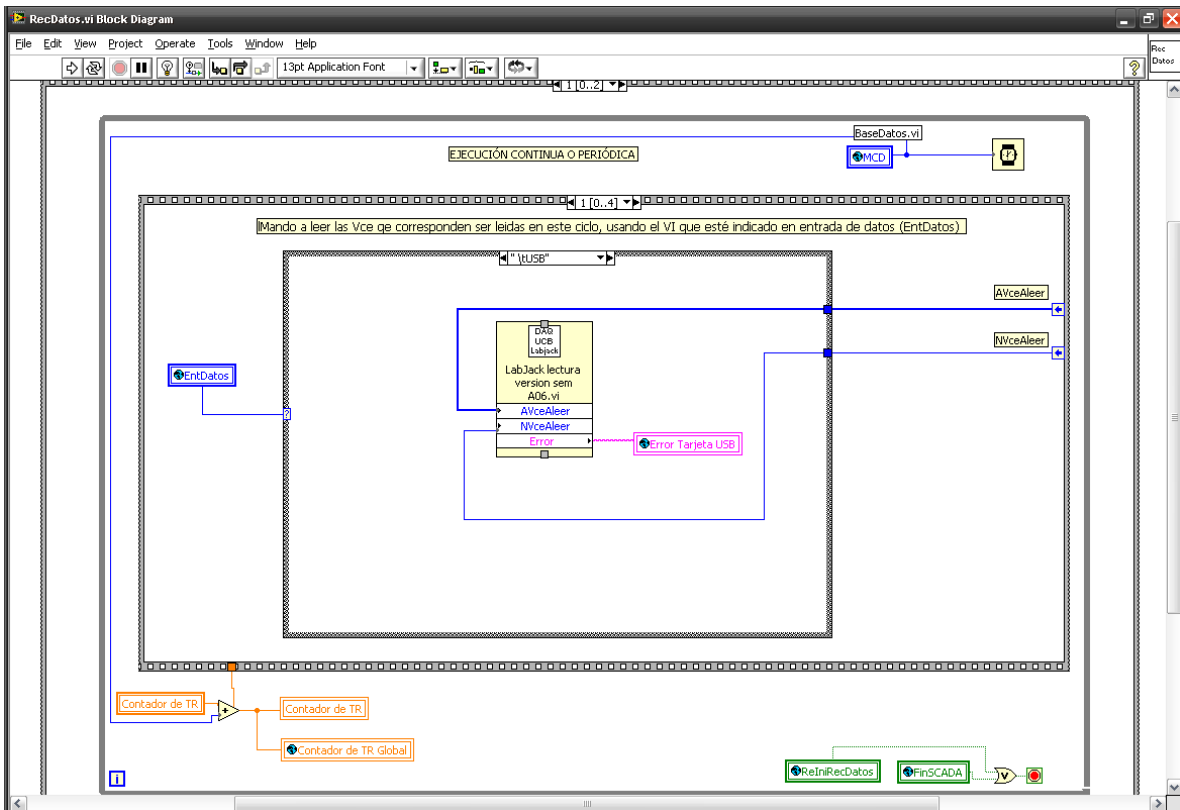


Fig. 90. Lee datos de la tarjeta USB



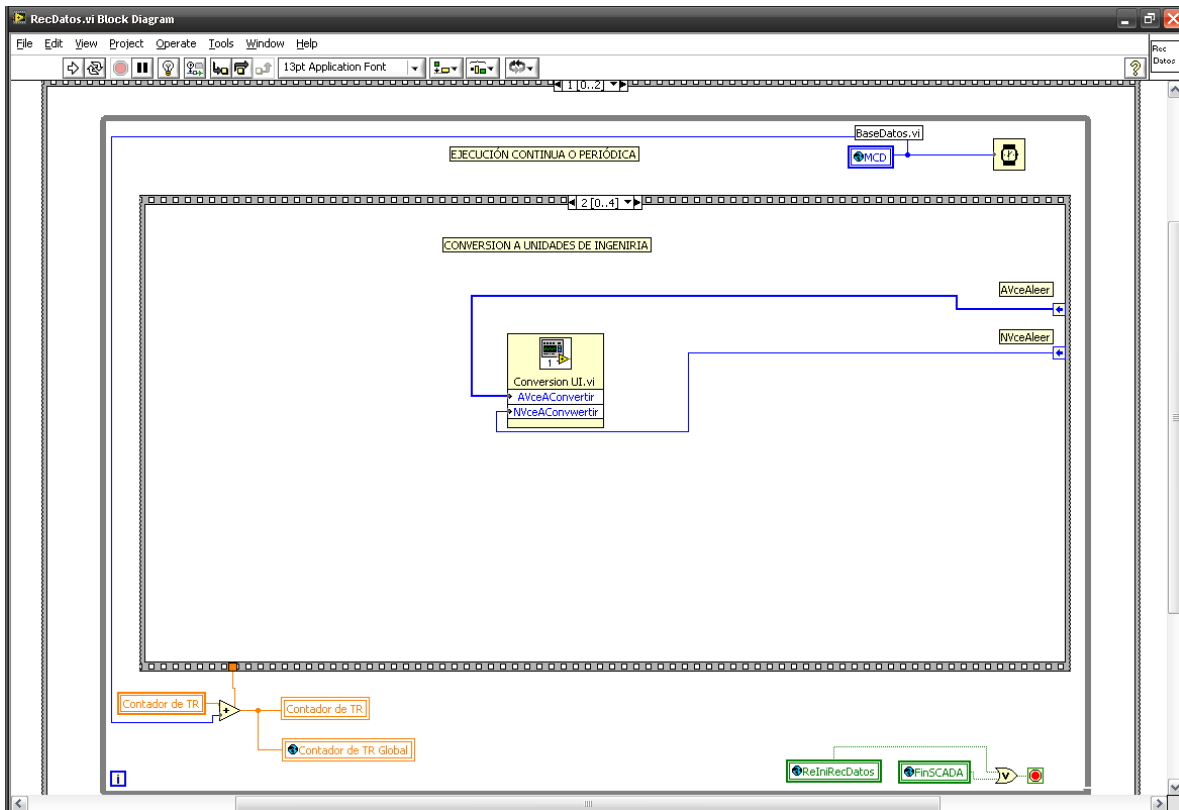


Fig. 91. Realiza la conversión a unidades de ingeniería.



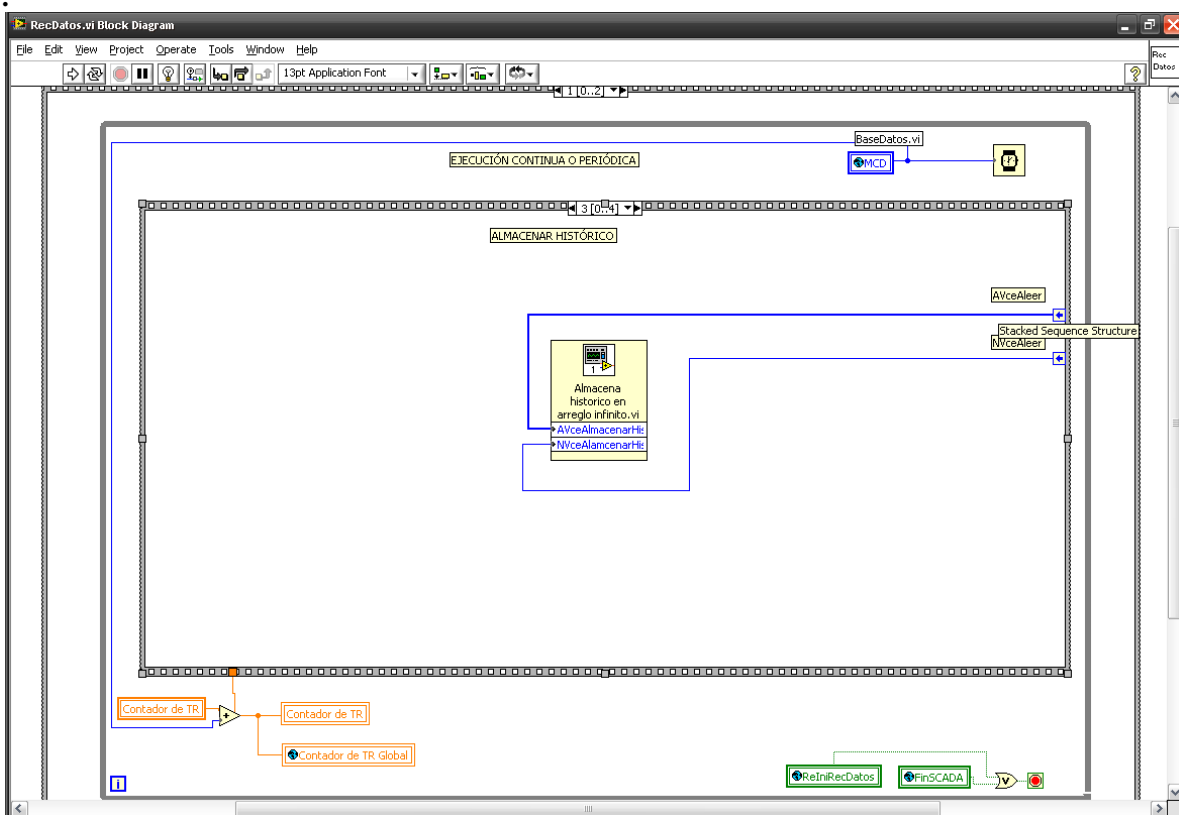


Fig. 92. Almacena el histórico en un arreglo.



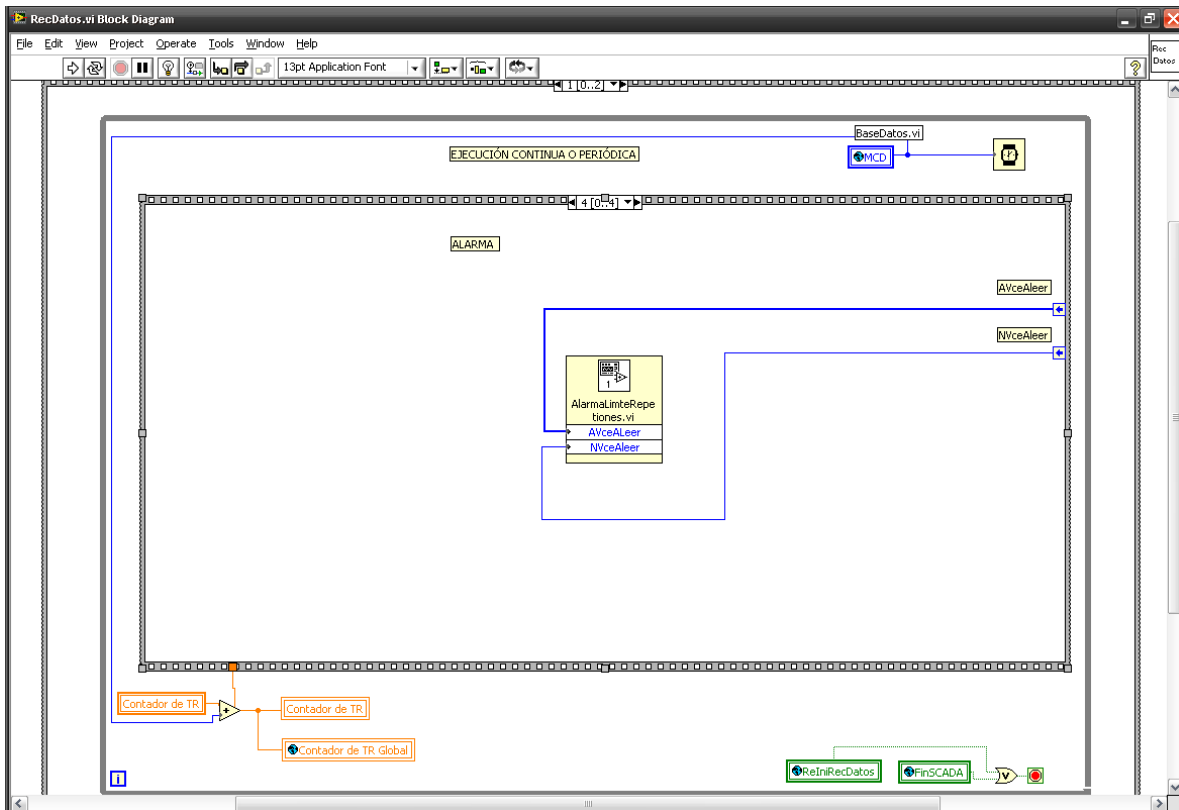


Fig. 93. Realiza el proceso de la alarma.





Archivo que inicializa la base de datos

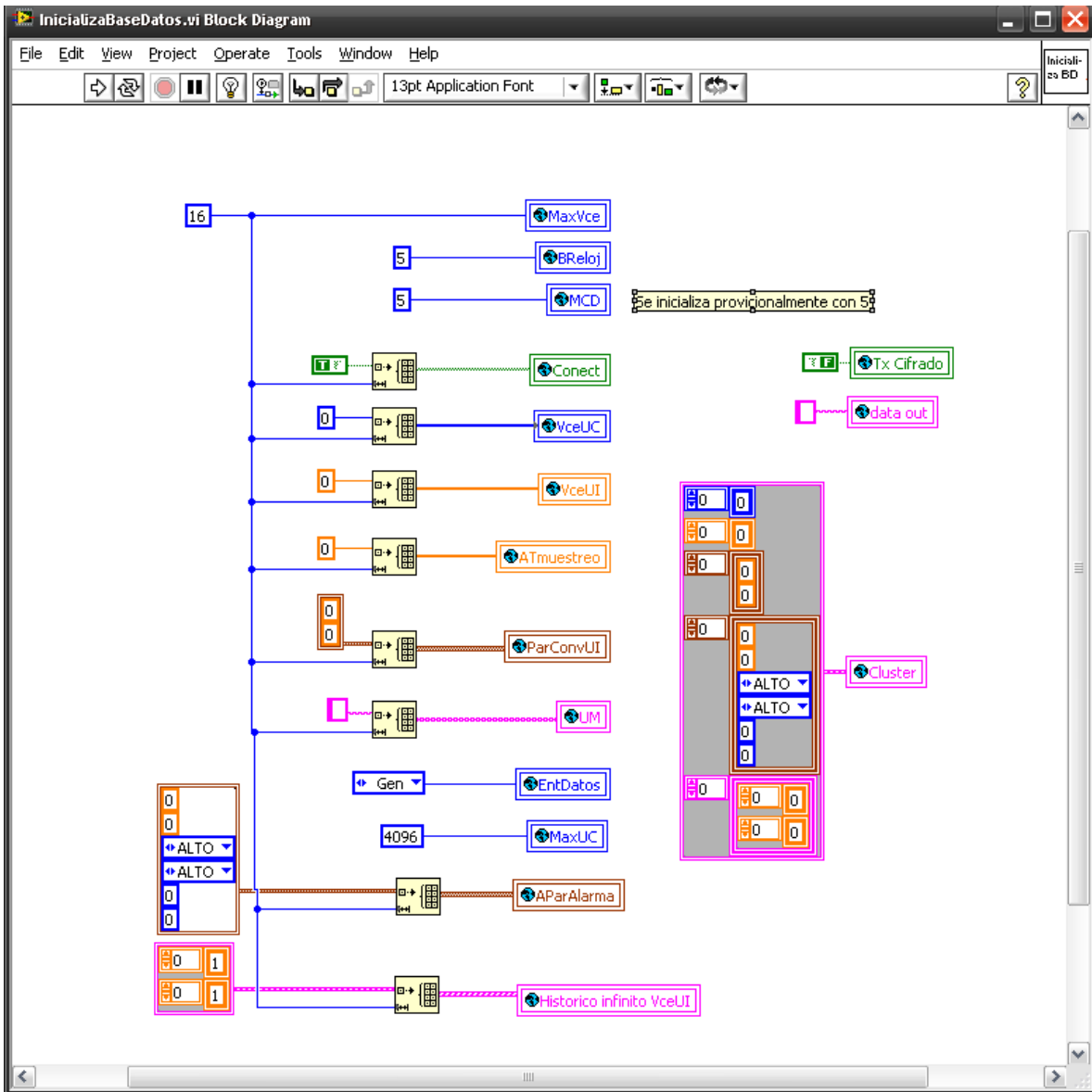


Fig. 94. Inicializa la base de datos.





Archivo base de datos binaria

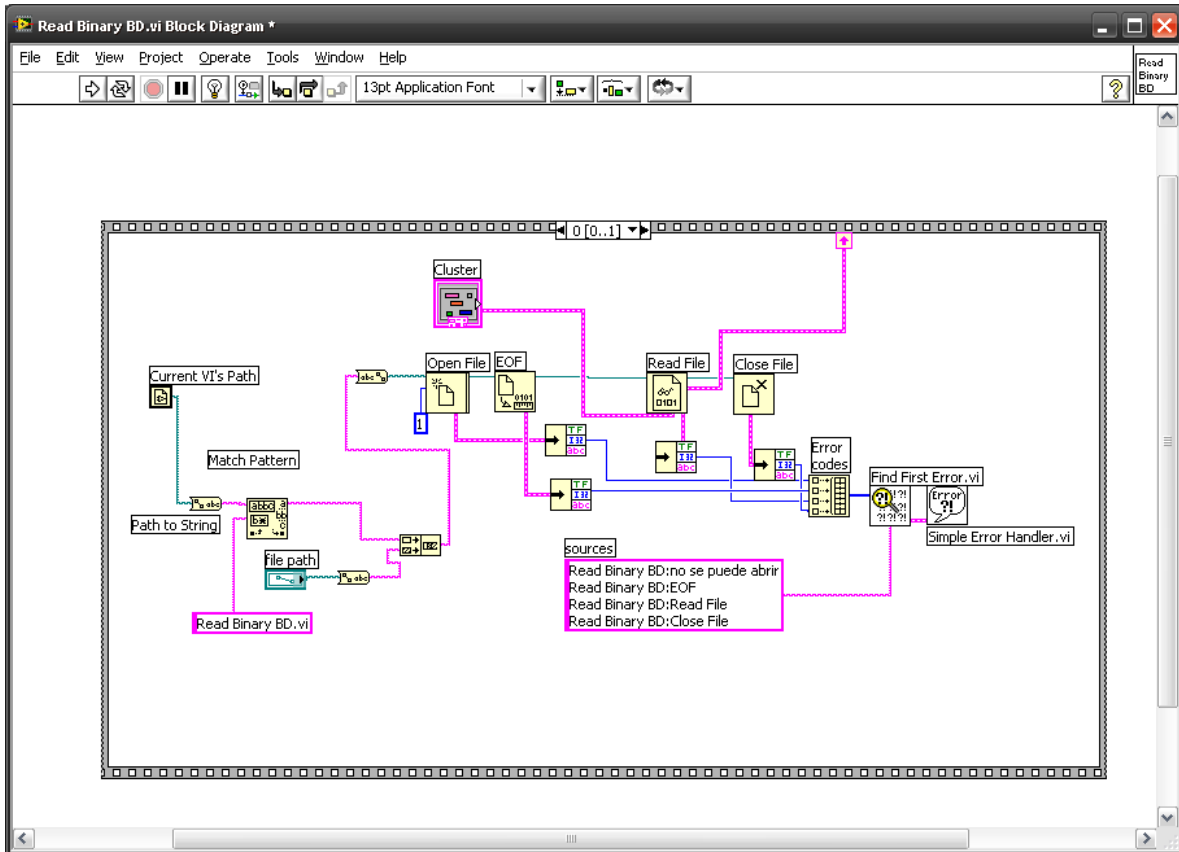


Fig. 95. Almacena los datos en un archivo binario.



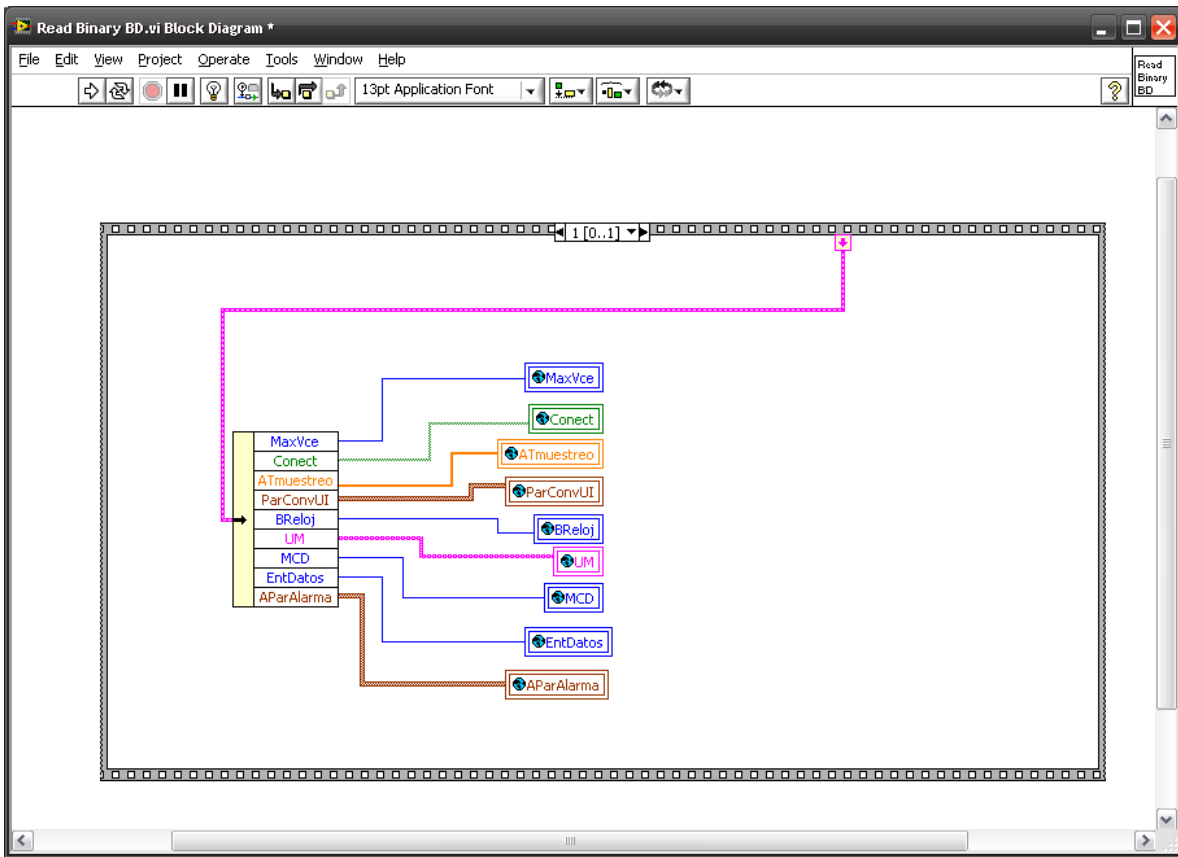


Fig. 96. Lee los datos de la base de datos.





Archivo de transmisión de datos

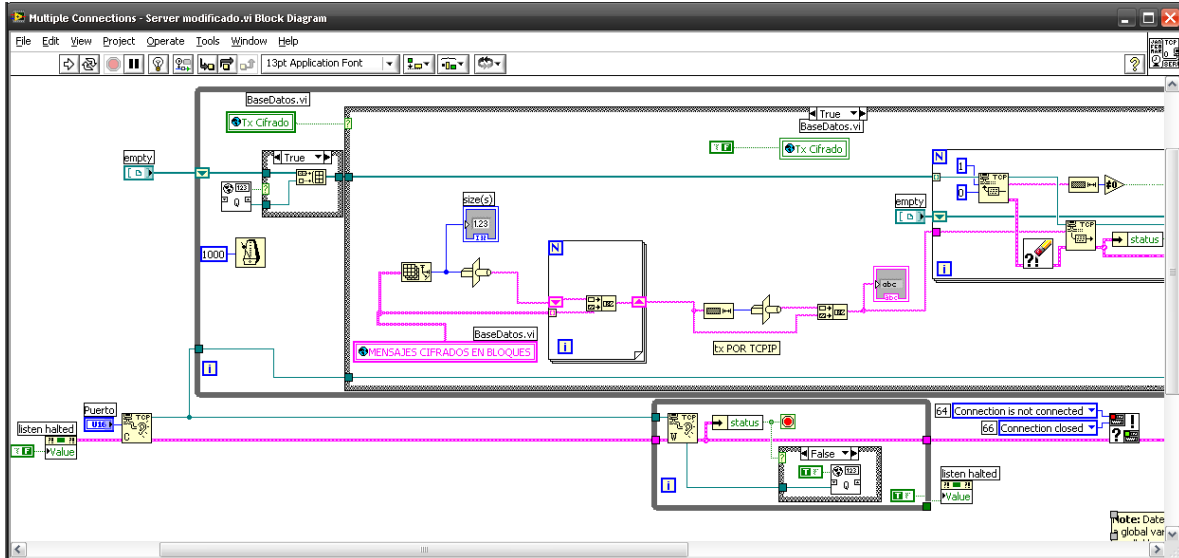


Fig. 97. Realiza la comunicación del servidor.

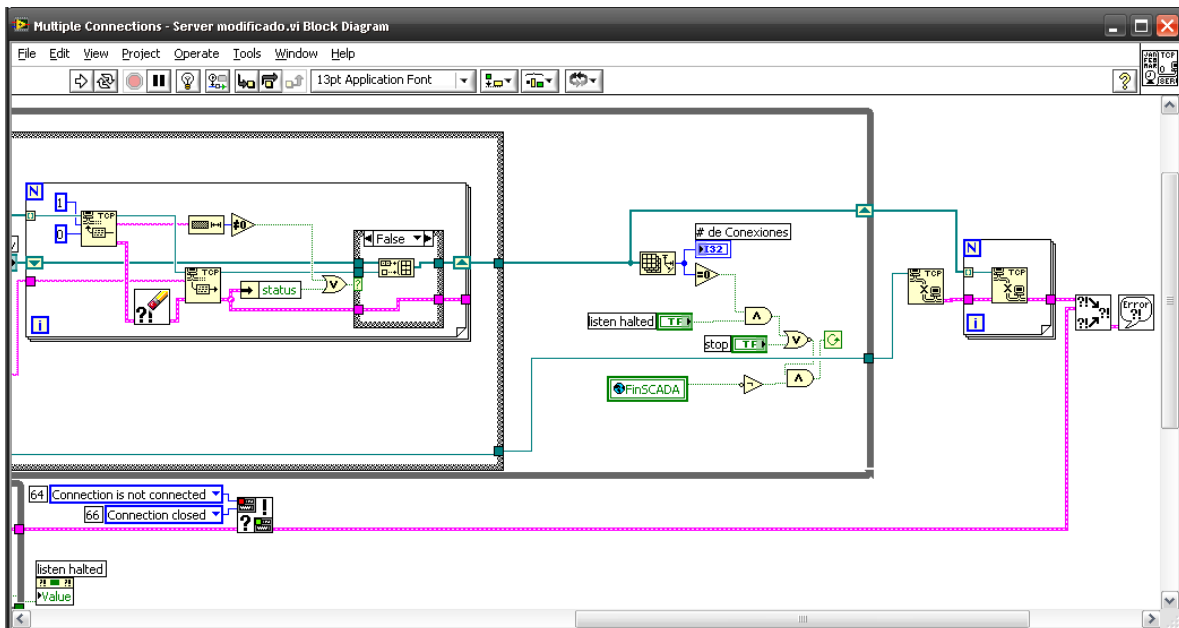


Fig. 98. Maneja los errores en la comunicación del servidor.



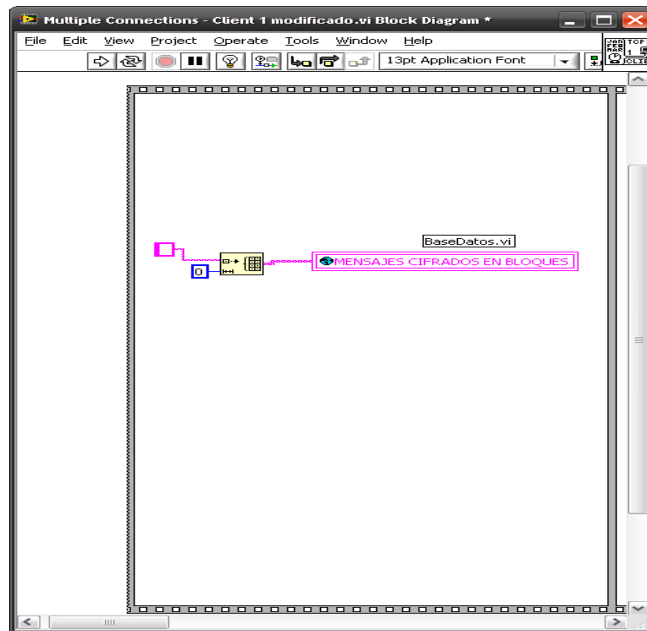


Fig. 99. Inicializa el arreglo de la variable MENSAJES CIFRADOS EN BLOQUES.

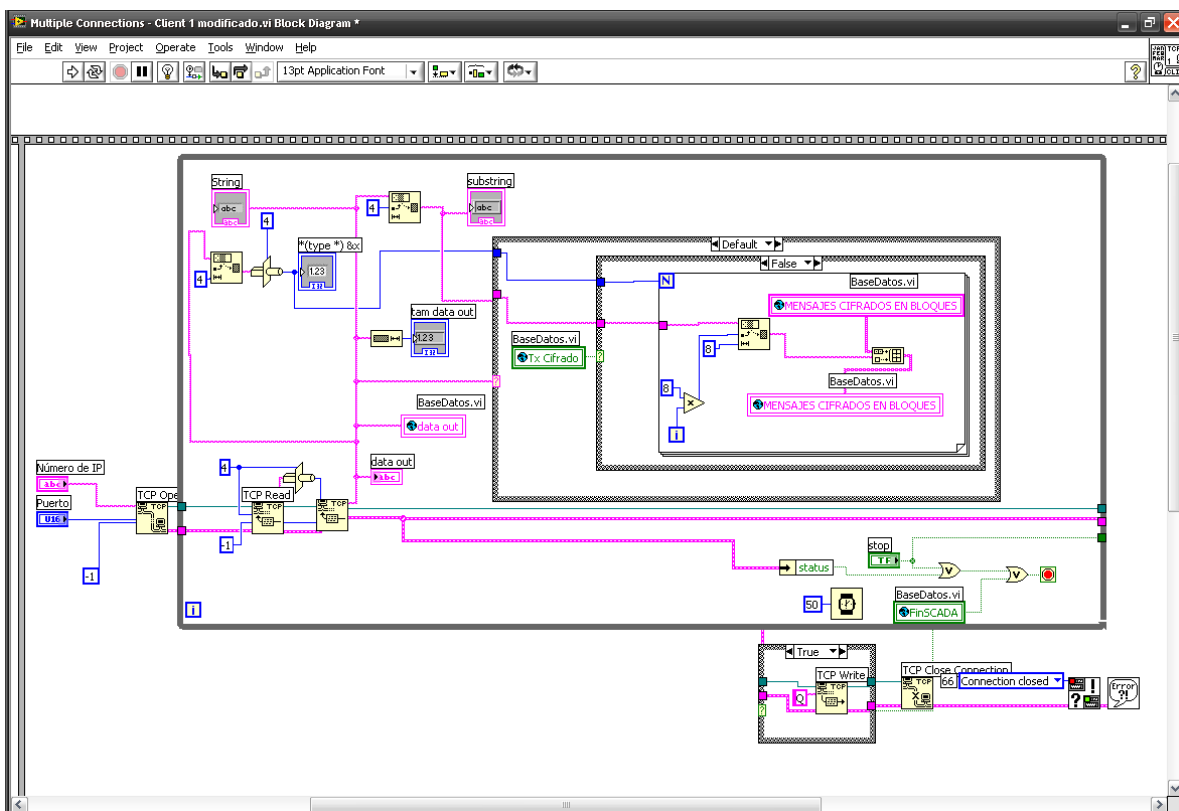


Fig. 100. Transmite los MENSAJES CIFRADOS EN BLOQUES.

