UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"Herramienta criptográfica RAD para el desarrollo de sistemas multiusuarios".

T E S I S

PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

MANUEL ALFREDO ARIAS MARTÍNEZ

A S E S O R:

M.C. FRANCISCO DE ASIS LÓPEZ FUENTES

HUAJUAPAN DE LEÓN, OAXACA. ENERO 2003

DEDICATORIA

Esta tesis no hubiera sido posible sin la ayuda de tres seres fundamentales en mi vida, es a ellos a quien se las dedico:

A Dios por ayudarme a entender el significado de “Primero lo primero”.

A quien más sino a mis Padres,
y qué más, sino decirles que los AMO.

Gracias Infinitas.

AGRADECIMIENTOS

A mi asesor M.C. Francisco de Asís López Fuentes**,**
gracias por tu paciencia y ayuda incondicional.

A ti Ime, por entenderme y ayudarme en todo este tiempo.

A mis hermanos y amigos.

RESUMEN

El actual proyecto de tesis es un conjunto de dos controles ActiveX y una aplicación tipo asistente, los cuales tienen como objetivo el proveer de seguridad a sistemas de software en desarrollo bajo ambiente Windows trabajando con el lenguaje de programación Visual Basic Ver. 5.0 y 6.0. Cada control ActiveX realiza una función en específico permitiendo con esto flexibilidad y eficacia a la hora de su implementación.

Este proyecto está dirigido a todos los desarrolladores que:

- Necesiten implementar seguridad computacional mediante contraseñas.
- Trabajen con el lenguaje de programación Visual Basic Ver. 5.0 y 6.0
- Deseen implementar el algoritmo criptográfico Twofish o niveles de entropía para el encriptamiento de información.
- Necesiten implementar control de accesos y autenticación, administrando una o varias contraseñas de usuarios.

Entre las características generales del actual proyecto de tesis se encuentra el uso del algoritmo de encriptación Twofish, control de accesos con autenticación de usuarios y opción a RBAC (*Role Based Access Control*), se pueden administrar de 1 a n usuarios cada uno de ellos con claves de acceso y privilegios diferentes a nivel aplicación, para cada proyecto que se desarrolle, generador de claves de acceso al activar la opción OTP (*One Time Password*), registro de todos los accesos al sistema ya sean estos efectivos o

rechazados, bloqueo automático de contraseñas, un nivel de entropía (grado de desorden en la información) utilizando el método criptográfico de sustitución y por último, un nivel de entropía utilizando el método criptográfico de permutación.

Todo lo anterior permite implementar seguridad a nivel interfaz de aplicación y abarca los siguientes servicios de seguridad:

- Confidencialidad: Asegura que la información sea accedida para su visualización, o impresión, solo por las personas autorizadas para entrar al sistema mediante contraseñas.
- Autenticación: Permite autenticar a la persona que está tratando de entrar al sistema realizando cuestionamientos personales.
- Integridad: Permite integridad a nivel de interfaz de aplicación, asegurando que sólo los usuarios autorizados puedan cambiar, aumentar o borrar solamente la información que les compete. Permite integridad de información a nivel bases de datos, al identificar si la información ha sido alterada ó borrada. Esto siempre y cuando, la información haya sido encriptada por alguno de los algoritmos del actual proyecto de tesis.
- Control de accesos: Filtra, controla y registra todos los accesos al sistema, ya sean éstos exitosos o rechazados.

El actual proyecto no asegura la integridad física de la información, ni la disposición en todo momento de ésta. Tampoco ofrece ningún servicio de seguridad en la transmisión de la información, excepto que la información viaje encriptada en la red, en caso de así requerirse.

INDICE

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1

Introducción

"La seguridad es una necesidad básica, estando interesada en la prevención de la vida y las posesiones." [1]

La anterior definición de seguridad permite entender que ésta palabra era tan utilizada desde hace tiempo, como lo es hasta ahora, salvo que hoy en día es necesario complementarla y actualizarla al avance tecnológico y forma de vida actual. El concepto de seguridad ha ido adquiriendo distintos significados, dependiendo del lugar, tiempo y circunstancias en que se aplique.

Otra definición interesante de analizar es la siguiente: "Seguridad es la interrelación dinámica (competencia) entre el agresor y protector para obtener (o conservar) el valor tratado, enmarcada por la situación global" [1]. La definición anterior se acerca más al significado que definirá el objetivo central de la presente tesis, pero aún es alejada pues la "situación global" de esta definición, muy posiblemente no es la misma que enmarca el actual proyecto.

Lo anteriormente definido, no tiene otro objetivo sino mostrar cómo ha ido evolucionando el término de seguridad, y al mismo tiempo, mencionar que la seguridad ha sido tema y tarea desde hace mucho tiempo.

Enfocando el término de seguridad al actual proyecto, uno de las definiciones que cubre perfectamente con el objetivo en tiempo y situación específica, es la siguiente: “El objetivo de la seguridad informática, será mantener la integridad, disponibilidad, privacidad, control y autenticidad de la información manejadas por computadora.”[2]

La definición anterior identifica perfectamente las tareas de las que debe de salvaguardar una solución de seguridad, a los datos e información. Como se puede apreciar, hablar de seguridad informática es hablar de un concepto sumamente amplio y no simple de solucionar. Por ello, existen diferentes aplicaciones y dispositivos que ofrecen proveer de algunas o todas las necesidades de seguridad informática, pero antes que nada, se debe tener en cuenta que “la seguridad no es un producto, sino un proceso” [3], por lo tanto, es necesario implementar técnicas, aplicaciones, dispositivos y capacitación, para lograr la seguridad deseada.

1.1. ANTECEDENTES

La seguridad computacional ha ido evolucionando y frecuentemente hace uso de toda la astucia e inteligencia posible para cuidar los datos e información de una organización.

Una rama importante en la seguridad de los datos e información, es la Criptografía. El término Criptografía, significa “escritura escondida” y consiste en enmascarar los datos de forma que al verlos sean completamente incomprensibles en significado. Actualmente existen algoritmos criptográficos muy potentes que son conocidos en todo el mundo y altamente probados. Tal es el caso del algoritmo Twofish, el cual es utilizado en el actual proyecto para el encriptamiento de los datos e información.

Un organismo muy importante en cuanto a seguridad criptográfica se refiere, entre otras actividades que coordina y realiza, es el anteriormente llamado NBS[1], ahora conocido como NIST, este organismo realizó un anuncio público solicitando propuestas de algoritmos para protección de datos durante su transmisión y almacenamiento.

En 1973, el NIST anuncia una competencia internacional para definir el AES (*Advanced Encryption Standard*), como sucesor del DES (*Data Encryption Standard*). El motivo de buscar un algoritmo sustituto es que, DES utiliza una clave de 56 bits, que es demasiado pequeña para la tecnología actual y puede romperse fácilmente por búsqueda exhaustiva de claves en computadoras de alta velocidad. DES ha llegado al final de su vida útil, lo mismo que cualquier programa que se base en él. AES será disponible internacionalmente y sin costo.

NIST publicó las especificaciones para el AES a finales de 1997. Los requerimientos principales son: un algoritmo simétrico con bloques de 128 bits y con llaves de 128, 192 o 256 bits. Los participantes de esta competencia tenían que escribir código en los lenguajes de programación ANSI-C y Java. Como plataforma se especificó el procesador Pentium de Intel (un procesador de 32 bits) y también procesadores de 8 bits como los que se utilizan en tarjetas inteligentes (*smart cards*). NIST explicó que los criterios para escoger el algoritmo ganador serían en orden de prioridad los siguientes: seguridad, velocidad y sencillez del diseño. Finalmente todo participante oficial en el concurso debía garantizar que si ganase renunciaría a todo derecho de propiedad intelectual sobre su algoritmo. [4]

[1] NBS, antecesor del NIST, el NIST (*National Institute of Standards and Technology*, Instituto Nacional de Normas y Tecnología) es una Agencia del Ministerio de Comercio del Gobierno federal norteamericano, fundada en 1901 (como *National Bureau of Standards*, NBS) [7]. Entre sus principales objetivos, el NIST, declara que fue creado por el Congreso para apoyar a la industria en el desarrollo de la tecnología necesaria para mejorar la calidad de los productos, para modernizar los procesos de fabricación, para asegurar la confiabilidad en los productos y para facilitar una rápida comercialización de los productos basados en los recientes descubrimientos de las ciencias.

NIST aceptó 15 algoritmos como candidatos oficiales, los cuales fueron anunciados en la Primera Conferencia del AES celebrada entre el 20 y 22 de Agosto 1998 en Ventura, California. Twofish fue uno de estos 15 algoritmos con muchas posibilidades de quedar entre los finalistas.

En la segunda conferencia sobre el AES celebrada del 22 al 23 de Marzo 1999 en Roma, Italia, el enfoque primordial ha sido sobre la velocidad de los candidatos. Los candidatos más rápidos fueron RC6, Twofish, Mars, Rijndael y E2. Twofish fue uno de los 5 finalistas del NIST, y por muchos considerado como el segundo lugar indiscutible después del algoritmo Rijndael y la mejor opción para sustituir a AES en caso de que éste sea quebrantado. [4]

Por lo anterior, se ha decidido usar el algoritmo Twofish como algoritmo de encriptamiento para el actual proyecto de tesis.

1.2 OBJETIVO

Objetivo general

Construir un conjunto de herramientas criptográficas RAD (*Rapid Application Development*, Desarrollo Rápido de Aplicaciones) para dotar de seguridad computacional a sistemas de software en desarrollo por medio de una interfaz para el control de accesos a usuarios y controles ActiveX basados en la encriptación con el algoritmo Twofish y dos niveles de entropía[2].

Para alcanzar este objetivo general se consideran los siguientes objetivos específicos:

Objetivos específicos

- Desarrollar herramientas criptográficas RAD bajo el sistema operativo Windows95 o mayor y como plataforma de desarrollo MS Visual Basic Ver. 5.0 en adelante.

---

[2] El término entropía es utilizado en medicina para definir una disfunción del cristalino que consiste en confundir una letra por otra [5]. En la presente tesis, el término entropía se manejará, en sentido figurado como sinónimo de desorden en la información.

- Reutilizar código por medio del uso de tecnología ActiveX[3].
- Implementar en un control OCX (*Object Control ActiveX*)[4] el encriptamiento a través del algoritmo TwoFish, logrando con esto, encriptar cualquier dato o información mediante unas cuantas instrucciones de código.
- Implementar en un control OCX, dos niveles de entropía.
- Al combinar las características anteriores, se buscará que el proyecto de tesis pueda ofrecer los siguientes servicios de seguridad: identificación y autenticación, roles, transacciones, y dentro de las características del control de acceso interno, palabras claves, encriptación, listas de control de accesos, límites sobre la interfaz de usuario.

1.3 JUSTIFICACIÓN

Actualmente la seguridad no se da solo físicamente, sino lógicamente, es por eso que los sistemas de seguridad son cada vez más necesarios y cada vez más empresas incorporan seguridad a sus sistemas.

Uno de los problemas con los que se enfrentan las empresas al tratar de incorporar seguridad en sus sistemas, es la falta de conocimiento en el tema, por lo cual, la investigación y desarrollo de sus propias soluciones les toma mucho tiempo, y esto se refleja en pérdidas de dinero.

El actual proyecto de tesis propone un sistema de seguridad computacional con las características de las herramientas RAD, las cuales podrán proveer de seguridad a los sistemas que se desarrollen con esta solución en el menor tiempo posible y con el mínimo de conocimientos sobre el tema, lo anterior se debe a que el sistema contendrá módulos completos de seguridad y con solo incorporar éstos al desarrollo en proceso, se podrá dar solución al problema de seguridad a nivel de interfaz.

---

[3] Un componente ActiveX es una unidad de código ejecutable, como un archivo .exe, .dll u .ocx, La tecnología ActiveX permite a los programadores ensamblar estos componentes software reutilizables en aplicaciones y servicios [6].

[4] Componente ActiveX con extensión .ocx. Permite incorporar a la aplicación un componente visible en tiempo de diseño [8].

Por otro lado, el desarrollo de sistemas para las pequeñas y medianas empresas se realiza en plataformas de desarrollo que permitan una rápida y fácil programación, tal es el caso de MS Visual Basic, el cual permite desarrollar soluciones con bases de datos de forma rápida y con un mínimo de codificación.

1.4 ESTADO DEL ARTE

Hoy en día existen muchas empresas y organismos gubernamentales dedicados a la seguridad computacional, pero todos desarrollan pequeñas soluciones de uso inmediato tales como programas que encriptan y desencriptan un texto determinado utilizando un algoritmo conocido. Existen también pequeños programas que protegen archivos con claves de acceso en el momento de intentar abrirse, e incluso, hay empresas que viendo las necesidades de los departamentos internos de informática en micro y pequeñas empresas, han desarrollado controles ActiveX para la plataforma de desarrollo Visual Basic, los cuales, proveen el servicio de encriptación de algún algoritmo criptográfico simétrico, pero solo realizan la encriptación y desencriptación de un texto determinado.

Ahora bien, las grandes empresas que se dedican a la seguridad computacional, tienen grandes soluciones que difícilmente podrían adquirir las pequeñas y medianas empresas, ya sea por el precio de las soluciones o por el precio de la infraestructura con la que deben contar (servidores AS/400, Oracle, Informix, etc.). Por lo anterior, al investigar soluciones de software que ofrezcan seguridad en los datos o información, únicamente se encontró con soluciones que permiten encriptar y desencriptar, otros ofrecen controlar accesos por contraseñas y encriptación de la información, y las aplicaciones más completas, constan de controles ActiveX que permiten solo encriptación y desencriptación de datos.

En la Tabla 1.1 se indican los programas más populares sobre seguridad computacional y criptografía encontrados en Internet.

Tabla 1.1 - Aplicaciones de Criptografía más comunes

Nombre del programa	Descripción
Speedsafe Pro	Herramienta de seguridad mediante encriptación de datos con el algoritmo Blowfish (448 bit) y Safer SK-128 (128 bit), encripta texto plano, interfaz fácil de utilizar, y genera passwords seguros. Correo Electrónico: info@od-asd.com, Sitio Web: http://www.od-asd.com
Crypt	Encripta y desencripta archivos restringiendo su utilización a terceras personas, pide un password para poder abrirse. Sitio Web: http://jksoft.virtualave.net
Kryptos	Provee de controles OCX para Visual Basic y Delphi con los algoritmos de encriptación Blowfish, Twofish y Base64. Sitio Web: http://www.psd.es
SmartCript-it	Permite encriptar y desenciptar texto plano y archivos con 5 de los algoritmos de encriptación más populares. Correo Electrónico: smart@novasdi.com, Sitio Web: http://www.novasdi.com

Por lo anterior se pensó en un proyecto que brindara una solución a través de:

- Controles ActiveX que permitan encriptar y desencriptar datos pensados para programadores y que éstos sean de fácil implementación en proyectos de desarrollo.

- Un algoritmo de encriptación conocido y probado (Twofish) y un algoritmo propio de encriptación, permitiendo así, generar un mayor nivel de seguridad criptográfica.

- Complementación de los algoritmos de encriptamiento con la administración del control de accesos y autenticación de usuarios.

- Seguridad computacional al alcance de pequeñas y medianas empresas, o incluso para desarrolladores independientes.

Capítulo 2

Servicios de Seguridad y Control de Accesos

2.1 SERVICIOS DE SEGURIDAD

Toda organización, el administrador responsable de la seguridad en específico, necesita definir los requerimientos de seguridad y las características de los productos y servicios que satisfagan esos requerimientos de seguridad.

En la seguridad de la información se consideran tres aspectos fundamentales [9]:

- Ataques de seguridad: Una acción que comprometa la seguridad de la información propiedad de una organización.
- Mecanismos de seguridad: Un mecanismo de seguridad es diseñado para detectar, prevenir o recobrarse de un ataque de seguridad.
- Servicios de seguridad: Los servicios de seguridad son el uso y administración de uno o más mecanismos de seguridad para proveer el servicio.

Ya que el objetivo de la presente tesis es desarrollar un sistema que dote de seguridad, se seguirá tratando los servicios de seguridad de la información.

Una clasificación usual[5] de los servicios de seguridad es la siguiente [9]:

- Confidencialidad: asegura que la información en un sistema de computadora y la información trasmitida, son accesibles solo para su lectura por las partes autorizadas para ello.
- Autenticación: se encarga de que el mensaje o documento electrónico es correctamente identificado, asegurándose que la identidad no es falsa.
- Integridad: asegura que solo las partes autorizadas están habilitadas para modificar los sistemas de la computadora y transmitir la información. Las modificaciones incluyen: escritura, modificaciones, cambios de privilegios, eliminación y creación de la información.
- Rechazo de servicio: asegura que tanto al emisor como el receptor de un mensaje, se les niegue el derecho de transmisión ó acceso a la información.
- Control de acceso: protege la información contra accesos no deseados, tanto físicos como lógicos.
- Disponibilidad: permite que la información esté disponible a las personas autorizadas en el momento en el que se necesite.

2.2 CONTROL DE ACCESOS

Los controles de acceso se pueden implementar en el sistema operativo, en sistemas de aplicación, en bases de datos, en un paquete específico de seguridad, o en cualquier otro utilitario [10].

El NIST ha resumido los siguientes estándares de seguridad que se refieren a los requisitos mínimos de seguridad en cualquier sistema [10].

[5] No existe un acuerdo universal acerca de los términos utilizados en la literatura de la seguridad de la información. Por ejemplo, el término *integridad* es algunas veces usado para referirse a todos los aspectos de la seguridad [10].

- Identificación y autenticación: se denomina identificación al momento en que el usuario se da a conocer en el sistema, y autenticación a la verificación que realiza el sistema a esta identificación.
- Roles: el acceso a la información también puede controlarse a través de la función o rol del usuario que requiere dicho acceso.
- Transacciones: también pueden implementarse controles a través de las transacciones, por ejemplo, solicitando una clave al requerir el procesamiento de una transacción determinada.
- Limitaciones a los servicios: estos controles se refieren a las restricciones que dependen de parámetros propios de la utilización de la aplicación o preestablecidos por el administrador del sistema.
- Modalidad de acceso: se refiere al modo de acceso que se permite al usuario sobre los recursos y la información. Esta modalidad puede ser: lectura, escritura, ejecución, borrado y todas las anteriores. Existen dos especiales que son creación y búsqueda.
- Ubicación y horario: el acceso a determinados recursos del sistema, puede estar basado en la ubicación física o lógica de los datos o personas.
- Control de acceso interno:
    - Palabras claves (passwords): generalmente se utilizan para realizar la autenticación del usuario y sirven para proteger los datos y aplicaciones.
    - Encriptación: la información encriptada solamente puede ser desencriptada por las personas que posean la clave apropiada.
    - Listas de control de accesos: se refiere a un registro donde se encuentran los nombres de los usuarios.
    - Límites sobre la interfaz de usuario: Estos límites generalmente son utilizados en conjuntos con las listas de control de accesos y restringen a los usuarios a funciones específicas.
    - Etiquetas de seguridad: Consiste en designaciones otorgadas a los recursos que pueden utilizarse para varios propósitos como control de accesos, especificación de medidas de protección, etc. Estas etiquetas no son modificables.

- Control de acceso externo:
    - Dispositivos de control de puertos: estos dispositivos autorizan el acceso a un puerto determinado y pueden estar físicamente separados.
    - Firewalls o puertas de seguridad: Permiten filtrar o bloquear el acceso entre dos redes, usualmente una privada y otra externa
    - Acceso de personal contratado o consultores: debido a que este tipo de personal en general presta servicios temporarios, debe de ponerse especial consideración en la política y administración en sus perfiles de acceso.
    - Accesos públicos: para los sistemas de información consultados por el público en general, se debe tener en cuenta medidas especiales de seguridad.
- Administración: una vez establecidos los controles de acceso sobre los sistemas, es necesario realizar una administración de estas medidas de seguridad lógica, lo que involucra la implementación, seguimientos, pruebas y modificaciones sobre los accesos de los usuarios a los sistemas.

Respecto a los requisitos anteriores, el actual proyecto de tesis manejará: identificación y autenticación, roles, transacciones y todas las características del control de acceso interno, excepto etiquetas de seguridad.

Otro tipo de clasificación del control de accesos por usuarios es atendiendo a quién organiza este control, habiendo tres tipos de organización:

1.- DAC (Discretionary Access Controls)[6] [9]

Se refiere a que el creador del archivo es el que define los permisos de los objetos. Este tipo de control, es el utilizado en los sistemas operativos del mercado tales como Windows de Microsoft y Unix.

[6] Discretionary Access Controls: Control de accesos a discreción, controlado por el creador del archivo.

2.- MAC (Mandatory Access Controls)[7] [9]

En este tipo de control de accesos, el sistema operativo administra los permisos a los objetos. El sistema operativo crea etiquetas con derechos de acceso asignados y cada objeto tiene su etiqueta. Los usuarios se agrupan en sujetos que tienen unos permisos definidos para cada etiqueta.

3.- RBAC (Role-Based Access Controls)[8] [10]

Intenta tener las ventajas de los anteriores sistemas y evita la rigidez del MAC y la inseguridad del DAC. Los roles son funciones concretas que realiza un usuario dentro de la empresa durante un tiempo determinado, así a los usuarios se les asigna unos roles y cada rol tiene unos permisos sobre los objetos.

Otra forma de aumentar seguridad en el control de accesos a los sistemas es utilizar unos sistemas llamados OTP (One Time Password) [11] donde la contraseña de un usuario cambia cada vez que se usa. De esta forma, si algún atacante descubre una contraseña, no le servirá por que para el siguiente acceso se necesitará otra.

2.3 FORMAS COMUNES DE OCULTAR INFORMACIÓN EN ARCHIVOS

Los sistemas más utilizados actualmente son de contraseña, con diferentes variantes, se aplican a casi todos los aspectos de la seguridad de la información.

Con base en entrevistas con programadores realizadas a través de Internet y experiencias vividas como programador, puedo mencionar que existen tres formas en general de guardar la información en un archivo, en el cual, los programas de interfaz controlan el acceso de usuarios a través de contraseñas o passwords:

[7] Mandatory Access Controls: Control de accesos controlado por el administrador.
[8] Role-Based Access Controls: Control de accesos con base en el rol del usuario.

1.- Ocultamiento del archivo: al instalarse el sistema, todos sus archivos se alojan en un lugar definido del medio de almacenamiento, excepto el archivo que contiene las claves de acceso o passwords. Este archivo se trata de ocultar con una extensión diferente a la del formato utilizado en bases de datos y en otro directorio que sea seguro en el medio de almacenamiento utilizado. Las formas más comunes de ocultar el archivo de contraseñas son: cambiando las propiedades del mismo (oculto, solo lectura), dándole un nombre y extensión tradicional del sistema operativo que se esta utilizando, cambiando el nombre y extensión iguales a un archivo de programa conocido, instalado en la computadora.

2.- Archivo con apertura de contraseña: en este caso, el archivo que contiene las contraseñas, se encripta, pero la encriptación solamente se realiza en la clave de acceso al propio archivo, dándole así, un formato único a éste. La información que contiene el archivo permanece intacta, no experimenta encriptación alguna. La desventaja es que si una aplicación desea utilizar este archivo, necesitará entender también el formato del mismo, motivo por el cual no es muy popular para el desarrollo de sistemas, pues dificulta su migración y mantenimiento.

3.- Archivo con la información encriptada: esta forma de seguridad de información guardada en un archivo es la más utilizada en la actualidad, pues elude la posibilidad de poder conocer las contraseñas, aún sabiendo la ubicación física del archivo en el medio de almacenamiento. Para dar más seguridad a la información, es recomendable utilizar una combinación de estos tres métodos.

Capítulo 3

Técnicas de Criptografía

3.1 DEFINICIONES

Al hablar de criptografía, es necesario mencionar que ésta forma parte de la criptología, que a su vez, la criptología abarca también al criptoanálisis.

Etimológicamente hablando, criptología quiere decir “estudio o tratado de la escritura secreta”, ya que proviene de las palabras griegas *kryptos* que significa esconder, oculto ó secreto y de *logia,as* que significa estudio ó tratado. Actualmente tiene el significado de ciencia de la comunicación segura; su objetivo es que dos partes puedan intercambiar información sin que una tercera parte no autorizada, a pesar de que capte los datos, sea capaz de descifrar la información [13].

La palabra criptografía proviene del griego *kryptos*, que significa esconder y *gráphein*, escribir, es decir, escritura escondida [12]. La criptografía es la técnica de convertir un texto inteligible, texto en claro *(plaintex*t), en otro, llamado texto cifrado ó criptograma *(ciphertex*t), cuyo contenido de información es igual al anterior pero sólo lo pueden entender las personas autorizadas [14]. Cuando se desea mandar información confidencial, se aplican técnicas criptográficas para poder “esconder” el mensaje (a esta acción se le llama cifrar o encriptar), se manda el mensaje por una línea de comunicación que se supone insegura, para que después, solo el receptor autorizado pueda leer el mensaje “escondido” (a esta acción se le conoce como descifrar o

descencriptar). Por lo anterior, se puede decir que la criptografía es la técnica encargada de crear criptosistemas también llamados algoritmos criptográficos [12].

El criptoanálisis es la técnica de descifrar un criptograma sin tener la autorización para hacerlo. El criptoanálisis trata de romper los criptosistemas para apoderarse de la información cifrada [13].

Tema importante del actual proyecto de tesis es la criptografía, pues en él, se tiene como objetivo implementar el algoritmo criptográfico Twofish como parte del desarrollo de tesis. No es el objetivo de la presente tesis tratar a la criptología y criptoanálisis a fondo, por lo tanto, se continuará estudiando únicamente a la criptografía.

3.2 CLASIFICACIÓN DE LA CRIPTOGRAFIA

En los sistemas criptográficos se consideran tres aspectos fundamentales [9]:

1. El tipo de operaciones utilizadas para transformar el texto en claro en texto cifrado. Basados en dos principios: sustitución y transposición o permutación[9].
2. El número de llaves utilizadas en el proceso de encriptar y desencriptar un texto. Se puede utilizar una o más llaves para éste proceso y de aquí su clasificación.
3. La manera en la que el texto a encriptar es procesado. El proceso de puede realizar mediante bloques, bytes, bits, etc.

La criptografía se puede clasificar en dos grandes grupos, dependiendo del tipo de clave que se utilice [9]:

- Criptografía convencional, también llamada simétrica o de clave privada.
- Criptografía de clave pública o asimétrica.

[9] En la presente tesis, se utilizará el término permutación en lugar de transposición.

Se seguirá tratando solamente a la criptografía simétrica pues es en esta rama en donde se encuentra ubicado el algoritmo Twofish y los dos niveles de entropía desarrollados en la presente tesis.

3.2.1 Criptografía de clave privada ó simétrica

La criptografía simétrica se refiere al conjunto de métodos que permiten tener comunicación segura entre las partes siempre y cuando anteriormente se hayan intercambiado la clave correspondiente que se denomina como clave simétrica. La simetría se refiere a que las partes tienen la misma llave tanto para cifrar como para descifrar [12]. En la Figura 3.1 se puede apreciar el proceso de pasar un mensaje a través de un algoritmo de encriptación y obtener como resultado un criptograma o texto cifrado. Posteriormente, al aplicársele el algoritmo de desencriptación (acción inversa del mismo algoritmo de encriptación), se puede obtener el mensaje descifrado, el cual puede ser entendido por la persona autorizada para ello.

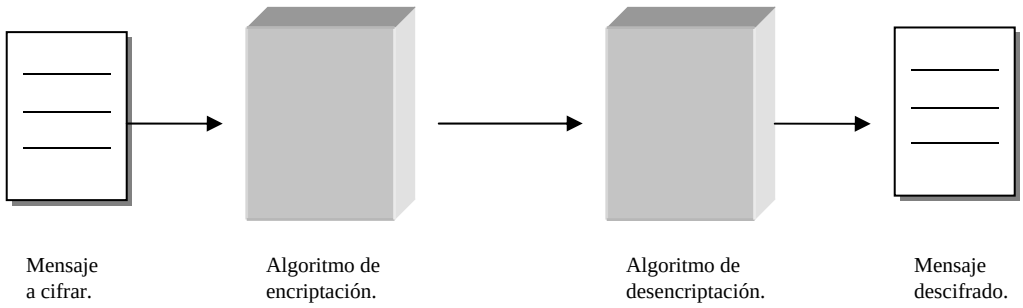


Figura 3.1 Criptografía de clave privada ó simétrica

Los algoritmos simétricos son más rápidos en sus procesos en comparación con los algoritmos asimétricos, por ese motivo, son más utilizados en la mayoría de los dispositivos y procesos de seguridad en la actualidad.

Entre los algoritmos simétricos más conocidos, se encuentran los siguientes:

- DES (*Data Encryption Standard*).
- IDEA (*International Data Encryption Algorithm*). Se utiliza mucho en sistemas nuevos europeos. No está sujeto a las leyes de ningún país.
- RC5. Algoritmo adoptado por *Netscap*e, no está probada completamente su seguridad.
- Blowfish y Twofish. Ambos de uso libre. Entre sus características sobresalen su velocidad de encriptación y el mínimo espacio utilizado en memoria.

3.3 EVOLUCIÓN DE LA CRIPTOGRAFIA SIMÉTRICA

La criptografia ha evolucionado a través del tiempo y han surgido muchas formas de cifrar mensajes, entre los cuales podemos mencionar [9]:

- Método Julio Cesar
- Sistemas monoalfabéticos
- Playfair
- Cifrado Hill.
- Sistemas Polialfabéticos
- Sistemas de Permutación
- Técnicas Combinadas

A continuación solo se detallan los tres métodos utilizados en la tesis, los cuales fueron implementados para dotar de entropía al sistema de seguridad.

3.3.1 Sistemas polialfabéticos

Los sistemas polialfabéticos surgen como una variante de los sistemas monoalfabéticos con el fin de aumentar seguridad y anular el criptoanálisis estadístico.

Uno de los problemas de estos sistemas, es recordar todos los alfabetos que componen al sistema, para solucionar esto, se utiliza una palabra clave y una tabla de alfabetos. Un ejemplo de cómo funcionan estos sistemas, es el algoritmo de Vigenère (1586) [9].

La forma en como funcionan el método de Vigenère es simple: dada una palabra clave *x* y un texto en claro *y*, el criptograma *V*, es el resultado de la intersección de renglones *x* y columnas *y* coincidentes en la letra correspondiente.

El método de Vigenère fue muy utilizado pues resolvía el problema de cómo recordar tantos alfabetos, pero en la actualidad, con el uso de las computadoras, este problema ya está resuelto, por lo cuál, han surgido infinidad de algoritmos creados por programadores en donde se utilizan polialfabetos como base de sus algoritmos. En el actual proyecto, se utilizan polialfabetos en combinación con indicadores de origen de alfabeto (ver tema 4.3.5.1 Método de Sustitución Simple con Polialfabetos).

3.3.2 Sistemas de permutación

Consiste en cambiar el orden de lugar de cada uno de los símbolos de los que consta un texto. La permutación de los símbolos del texto que se desea encriptar, permite anular técnicas de criptoanálisis tales como el de diccionarios. Uno de los métodos más utilizados en los sistemas de permutación, es el método de las columnas [9].

3.3.3 Técnicas combinadas

En los algoritmos simétricos actuales, se implementan conjuntamente los métodos de sustitución y permutación logrando así, obtener dos aspectos fundamentales en la defensa de ataques criptográficos [14]:

1. Confusión. Para hacer más compleja la relación clave-criptograma realizar sustituciones.

2. Difusión. Para vencer los métodos estadísticos, se realizan permutaciones de los símbolos.

Una ejemplo de la utilización de técnicas combinadas es el método conocido como *Rotor machines,* el cual utiliza tanto el método de sustitución, como el de permutación [9].

Capítulo 4

Proyecto de Tesis

4.1 INTRODUCCIÓN

En este capítulo se trata todo lo concerniente al análisis, desarrollo, pruebas e implementación del presente proyecto de tesis.

Como primer paso, se analiza el problema y se dan a conocer resultados de muestras tomadas para definir la viabilidad del proyecto. Se mencionan también, puntos importantes a considerar de los controles ActiveX y la necesidad de implementar un módulo para el control de accesos a través de un asistente, así como las características con las que deberá de contar éste.

Después de haberse realizado el análisis del problema, prosigue el diseño, desarrollo, y pruebas internas de la solución. El diseño de la interfaz se realiza instantes previos al desarrollo, pues al utilizar el lenguaje de programación Visual Basic, el diseño se concretiza a bosquejar en papel, sólo el contenido de los objetos con los que contará un formulario determinado. Por otro lado, las pruebas internas se realizan con el fin de probar el funcionamiento de la parte del desarrollo que se esté realizando en ese momento.

El desarrollo se divide en dos partes: la primer parte es el desarrollo de los controles ActiveX y la segunda consta del desarrollo del asistente.

Posteriormente, se menciona la forma de implementación del actual sistema de tesis en un desarrollo de software comercial. Después, se muestran las políticas de seguridad que deben conocer y seguir los usuarios que vayan a utilizar un sistema computacional con niveles de seguridad y se culmina con las pruebas externas del sistema implementado.

El paradigma de desarrollo utilizado en el presente proyecto de tesis es una combinación del paradigma de desarrollo clásico del software y herramientas 4GL, pues al utilizar el lenguaje de programación Visual Basic, el cual consta de herramientas tipo RAD, permiten al momento de la programación, una rápida creación de interfaz. Los pasos utilizados en el presente proyecto fueron: análisis, diseño (solo bosquejos de la interfaz y modelado de bases de datos, este último no colocado en la tesis pues no es el objetivo de la misma), programación, pruebas internas, implementación y por último, pruebas externas.

4.2 ANÁLISIS

Después de haber estudiado todo lo concerniente a seguridad computacional, criptografía y control de accesos, entre otros temas, se ve la necesidad de desarrollar un sistema de seguridad que cuente con las siguientes características:

- Algoritmo de encriptación Twofish.
- Control de accesos con opción a RBAC (*Role-Based Access Controls*).
- Poder administrar de 1 a n usuarios, cada uno de ellos con claves de acceso y privilegios diferentes para cada proyecto que se desarrolle.
- Control de accesos con autenticación de usuarios.
- Generador de claves de acceso al activar la opción OTP (One Time Password).
- Registro de todos los accesos al sistema ya sean estos efectivos o rechazados.

- Bloqueo automático de contraseñas en caso de accesos fallidos al sistema, previniendo intrusos en éste.
- Un nivel de entropía (cantidad de desorden en la información) utilizando el método criptográfico de sustitución.
- Un nivel de entropía utilizando el método criptográfico de permutación.
- Poder combinar los dos métodos de entropía antes mencionados, para lograr un nivel de seguridad mayor.

Por todo lo anterior, al desarrollar una solución de seguridad computacional con las características anteriores, las aplicaciones en las que se implemente el desarrollo de tesis podrán contar con hasta 3 niveles de seguridad global, los cuales se muestran a continuación:

- *Primer nivel: Encriptación*.- Consta del algoritmo Twofish y los dos niveles de entropía.
- *Segundo nivel: Autenticación*.- Consiste en pedir, después de haber introducido su clave de acceso exitosamente, información personal para corroborar si efectivamente corresponde la clave de acceso con la persona que está accediendo al sistema físicamente.
- *Tercer nivel: Control de accesos*.- Consta de las opciones OTP, RBAC, historial de accesos, bloqueo y denegación de claves de acceso.

4.2.1 Estudio de Necesidades

El actual proyecto surgió con la intención de proveer a los desarrolladores de software, herramientas para la programación, que hagan fácil el desarrollo e implementación de seguridad en sistemas computacionales.

Con base en la convivencia con programadores, líderes de proyecto y gerentes de área, y en la experiencia propia en el área de programación, se han observado los siguientes aspectos:

- Los programadores tienen conocimientos básicos, sino es que nulos en cuanto a seguridad computacional.
- Los programadores para poder desarrollar sistemas con un nivel de seguridad aceptable, requieren como mínimo, de dos a tres semanas de investigación sobre el tema.
- Después de haber realizado la investigación sobre seguridad computacional, el desarrollo e implementación de éste les llevaría de tres a cinco semanas, y juntando el tiempo de investigación y desarrollo, nos da como resultado que el tiempo completo que se llevaría un programador en desarrollar un sistema de seguridad básico, sería de dos a tres meses.
- Los alcances del sistema de seguridad que por lo general se desarrollan son los siguientes: control de accesos por medio de contraseñas utilizando un algoritmo de encriptación para guardar encriptada, la información más importante en bases de datos.
- Ya que el desarrollar soluciones para seguridad computacional implica tiempo y conocimiento, generalmente el programador que se hace cargo del desarrollo del módulo de seguridad, es el que se seguirá haciendo cargo de este mismo módulo en otros desarrollos, de lo contrario, si se asigna esta tarea a otro programador, se volvería a repetir todo el proceso de investigación y desarrollo, lo cual equivale a una inversión costosa de tiempo y dinero.
- De esta forma, las soluciones que se pueden visualizar para el desarrollo de sistemas computacionales con niveles de seguridad son dos: la primera, es asignar la tarea de la seguridad computacional a una persona en todos los proyectos, y la segunda, es la creación de nuevas herramientas de fácil implementación y administración, que permitan a cualquier programador utilizarlas sin tener que realizar una investigación exhaustiva.

- Del punto anterior, surgen las siguientes observaciones:
  - Cuando se cuenta únicamente con una persona dedicada a la seguridad computacional, se corre el riesgo de sobrecargar de trabajo a esta persona, si es que se desarrollan dos o más proyectos a la vez.
  - Y peor aún, si por algún motivo se llevaran dos o más proyectos en lugares distantes entre sí, sería desgastante para el programador encargado del módulo de seguridad computacional, trasladarse a cada uno de estos lugares, además de ser costoso y tardado.
- Por lo anterior, la solución óptima es crear herramientas de desarrollo para programadores que permitan implementar seguridad computacional.
- Ahora bien, la solución debe ser pensada para implementarse en sistemas computacionales de la micro, pequeña y hasta mediana empresa, ya que las grandes empresas que desarrollan su propio software, por lo general, trabajan con sistemas operativos tales como AS400, Solaris de Sun Microsystems, bases de datos de Oracle o de Informix, y sistemas RPG.
- Por otro lado, el sistema operativo que mas prevalece en este tipo de empresas pequeñas es Windows, por lo tanto, la plataforma de desarrollo Visual Basic es idónea para el desarrollo de la solución de seguridad, y Access ó SQL Server como bases de datos.

Como se había mencionado anteriormente, todo lo anterior es información obtenida de la observación y experiencia en el problema observado, y con el objeto de dar solidez a la hipótesis planteada, se realizó una encuesta a través de Internet para definir óptimamente las características con las que deberá de contar el presente proyecto de tesis.

El cuestionario que se diseñó, fue pensado para obtener información relevante sobre las necesidades de seguridad a implementarse en un sistema (ver Figura 4.1):

Herramienta criptográfica RAD para el desarrollo de sistemas multiusuarios.
Proyecto de tesis.
Encuesta de Viabilidad

1. ¿Trabajas por tu cuenta o para alguna empresa de desarrollo?. Especifica.

_____

2. Si trabajas para una empresa, ¿cuál es tu puesto?
_____

3. ¿Bajo qué lenguajes de programación desarrollas? (puedes seleccionar más de una respuesta)

☐ C, C++ ☐ Java ☐ Delphi ☐ Perl ☐ Visual Fox ☐ Visual Basic Otro:____

4. ¿Qué lenguaje de programación es el que más utilizas en proyectos de bases de datos? (puedes seleccionar más de una respuesta).

☐ C, C++ ☐ Java ☐ Delphi ☐ Perl ☐ Visual Fox ☐ Visual Basic Otro:___

5. Por lo general, ¿qué bases de datos utilizas en tus desarrollos?

☐ Dbase ☐ Access ☐ SQL Server ☐ Oracle ☐ Informix Otra:_____

6. De todos los desarrollos que haz realizado, ¿cuál es el porcentaje de éstos en los que haz necesitado o te han pedido implementar seguridad? (si la respuesta es 0% saltar a pregunta 13)

☐ 0% ☐ 10% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

7. De los desarrollos en los que haz necesitado implementar seguridad computacional, ¿cuál es tipo de seguridad por el que haz optado? (puedes seleccionar más de una respuesta, si la respuesta no es por contraseña, saltar a la pregunta 13)

☐ Contraseñas ☐ Firewall ☐ Sistema Biométrico ☐ Tocken

8. De los desarrollos que haz hecho implementando seguridad, ¿cuál es el porcentaje de éstos en los que haz implementado seguridad a través de contraseñas? (Si la respuesta es 0%, saltar a pregunta 13)

☐ 0% ☐ 10% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

9. Al implementar seguridad a través de claves de contraseñas, ¿utilizaste algún algoritmo, método ó técnica para ocultar la información?. Define.

_____

10. En caso de haber utilizado algún algoritmo de encriptación, ¿cuál ha sido y por qué?

_____

Figura 4.1 – Encuesta de Viabilidad.

11. Describe como fue que implementaste el algoritmo de encriptación:

☐ No tenía conocimientos de criptografía, investigué, desarrollé e implementé toda la solución.

☐ No tenía conocimientos de criptografía, investigué, adquirí una solución comercial e implementé la solución.

☐ Tengo personas que me asesoran, implementé una solución comercial o una solución dada por las personas que me asesoran.

12. ¿Cuanto tiempo te tomó desarrollar toda la solución de seguridad, desde la investigación hasta la implementación?

_____

13. ¿En un sistema con seguridad, qué características de seguridad crees que deba tener para considerarlo realmente seguro? (puedes elegir más de 1 respuesta)

☐ Contraseñas ☐ Criptografía ☐ OTP ☐ RBAC ☐ Hash ☐ Tocken

☐ Sistema Biométrico ☐ Firewall Otro: _____

14. ¿Cuentas con herramientas se seguridad computacional que sean fácil de implementar?

☐ Sí ☐ No ☐ Sí, pero no es completa

Figura 4.1 – Encuesta de Viabilidad (continuación).

La encuesta fue aplicada tanto vía Internet a través de la página http://www13.brinkster.com/manuelarias/encuesta como por vía encuesta directa con algunos grupos de desarrollo. Para aplicar las encuestas vía Internet, se enviaron correos electrónicos invitando a las personas a visitar la página de encuesta, y de esta forma, contribuir con el actual proyecto. Los correos electrónicos se enviaron tanto a personas conocidas, como a listas de correos de empresas que se dedican al desarrollo de software.

4.2.2 Resultados Obtenidos

Los resultados de la encuesta se muestran a continuación:

Cantidad de encuestas: 150

- A la pregunta: ¿Trabajas por tu cuenta o para alguna empresa de desarrollo?, los resultados fueron: el 68% trabaja para empresas o con grupos de desarrollo de

software, el 22% trabaja por su cuenta y el 10% trabaja por su cuenta y al mismo tiempo para una empresa.

- A la pregunta número dos, el resultado fue: el 53% es programador, el 29% es líder de proyecto y el 18% es jefe de departamento o gerente de sistemas.
- Para la tercer pregunta, las respuestas se dieron así: el 48% programan en C/C++, el 14% programa en Java, el 16% en Delphi, el 3% en Perl, el 14% en Visual Fox Pro, el 92% en Visual Basic, el 2% en Smaltalk, 78% en Java Script, 87% en Visual Basic Script, y el 4% en Cobol.
- En la cuarta pregunta, las respuestas fueron: 33% en Delphi, el 27% en Visual Fox Pro, el 84% Visual Basic, el 2% en Oracle.
- A la pregunta número cinco, los resultados se presentaron así: Dbase 8%, Access 66%, SQL Server 24%, 2% Oracle.
- Para la pregunta número seis, el resultado fue: 10% el 2% de los encuestados, el 75% el 52% de los encuestados, 100% el 46% de los encuestados.
- A la pregunta número siete, el resultado fue: contraseñas 99%, tockens 2%, firewall 5%.
- En la pregunta número ocho, los resultados son: 100% el 98% de los encuestados, el 75% el 2% de los encuestados.
- A la pregunta número nueve, los resultados concretos se presentaron así: el 59% utilizaron un método propio de encriptamiento por medio de sustitución simple monoalfabético, el 18% utilizó método propio de ocultamiento de base de datos y encriptamiento con sustitución simple monoalfabético, el 4% método Hash, el 19% un algoritmo de encriptación.
- A la pregunta número diez los resultados son así: el 28% Blowfish por ser gratuito y rápido, el 17% Triple DES por popular, el 12% Twofish por gratuito, rápido y rápida migración, el 14% RC6 por su rapidez, el 4% Mars por probar algún algoritmo, el 16% Serpent, el 9% AEA por haber ganado el concurso convocado por el NIST.
- A la pregunta número once, los resultados son: el 39% “No tenía conocimientos de criptografía, investigué, desarrollé e implementé toda la solución”, el 42%

"No tenía conocimientos de criptografía, investigué, adquirí una solución comercial e implementé la solución", el 19% "Tengo personas que me asesoran, implementé una solución comercial o una solución dada por las personas que me asesoran".

- A la pregunta número doce, las respuestas fueron: Se quitó la cantidad más alta y las más baja, y se promedió entre todos los encuestados descubriendo que en promedio, se tardan en desarrollar un sistema con seguridad alrededor de 18.4 días.
- A la pregunta número trece los resultados son: el 99% contraseñas, el 89% criptografía, el 42% OTP, el 32% RBAC, el 23% tockens, el 5% sistemas biométricos, el 11% firewall.
- A la pregunta número catorce los resultados son: el 11% respondieron "*Sí*", el 66% respondieron "*No*", el 23% respondieron "*Si, pero no es completa*".

De esta forma, analizando los resultados obtenidos en el estudio de viabilidad, podemos concluir lo siguiente:

- La mayoría de los programadores utilizan Windows y Visual Basic para desarrollar sistemas con bases de datos.
- De los desarrolladores que trabajan con Visual Basic, casi la totalidad desarrolla en Access y SQL Server como bases de datos.
- La gran mayoría de los desarrolladores, requieren implementar seguridad computacional.
- Al implementar seguridad computacional, la mayoría opta por utilizar sistema de contraseñas.
- Una gran cantidad de programadores, al no tener conocimientos sobre criptografía, crea sus propios algoritmos y métodos de "Encriptación", pero éstos son fáciles de descifrar hasta en un ataque de fuerza bruta.
- El desarrollo e implementación de sistemas de seguridad les toma entre 15 y 45 días a lo menos, por lo cual, darle seguridad a un sistema se traduce en un costo considerable.

- Los programadores que trabajan por su cuenta y aún algunas empresas dedicadas al desarrollo de sistemas, no cuentan con herramientas de criptografía fáciles de implementar.
- La gran mayoría de los programadores coinciden en que el contar con características de OTP, RBAC y Tockens en un sistema de seguridad, conlleva a tener un sistema realmente seguro de ataques.

Por todo lo anterior, el actual proyecto de tesis resulta viable, partiendo de la hipótesis anteriormente planteada.

4.3 CONTROLES ACTIVEX

4.3.1. Definiciones

Los controles ActiveX, anteriormente llamados controles OLE, son una extensión del cuadro de herramientas de Visual Basic entre algunos otros lenguajes de programación que utilizan estos controles. Los controles ActiveX se usan como cualquiera de los controles estándar incorporados. Cuando se agrega un control ActiveX a un programa, éste pasa a formar parte del entorno de desarrollo y por consiguiente, del programa en tiempo de ejecución, y proporciona nueva funcionalidad a la aplicación [15].

El desarrollo de componentes de software disminuye el tiempo de programación y produce aplicaciones más robustas ya que permite a los programadores, ensamblar aplicaciones a partir de componentes probados y estandarizados. Éste es el fin que se busca al utilizar los controles ActiveX.

Por otro lado, un componente ActiveX puede verse como una unidad de código ejecutable, tales como un archivo .exe, .dll u .ocx, siempre y cuando, éstos sigan la especificación ActiveX para proporcionar objetos. Por esto, se pueden adquirir componentes ActiveX que proporcionan servicios genéricos como análisis numérico o elementos de interfaz de usuario.

El desarrollo de software mediante la tecnología ActiveX, no se debe confundir con la programación orientada a objetos (OOP). La programación orientada a objetos es una forma de crear componentes de software basados en objetos; ActiveX es una tecnología que le permite combinar componentes basados en objetos creados con muchas herramientas diferentes. Por decirlo de otra forma, la programación orientada a objetos se ocupa de crear objetos, mientras que ActiveX se ocupa de que los objetos funcionen juntos [15].

La Figura 4.2 muestra una aplicación de Visual Basic que utiliza varios componentes, todos ellos creados con Visual Basic.

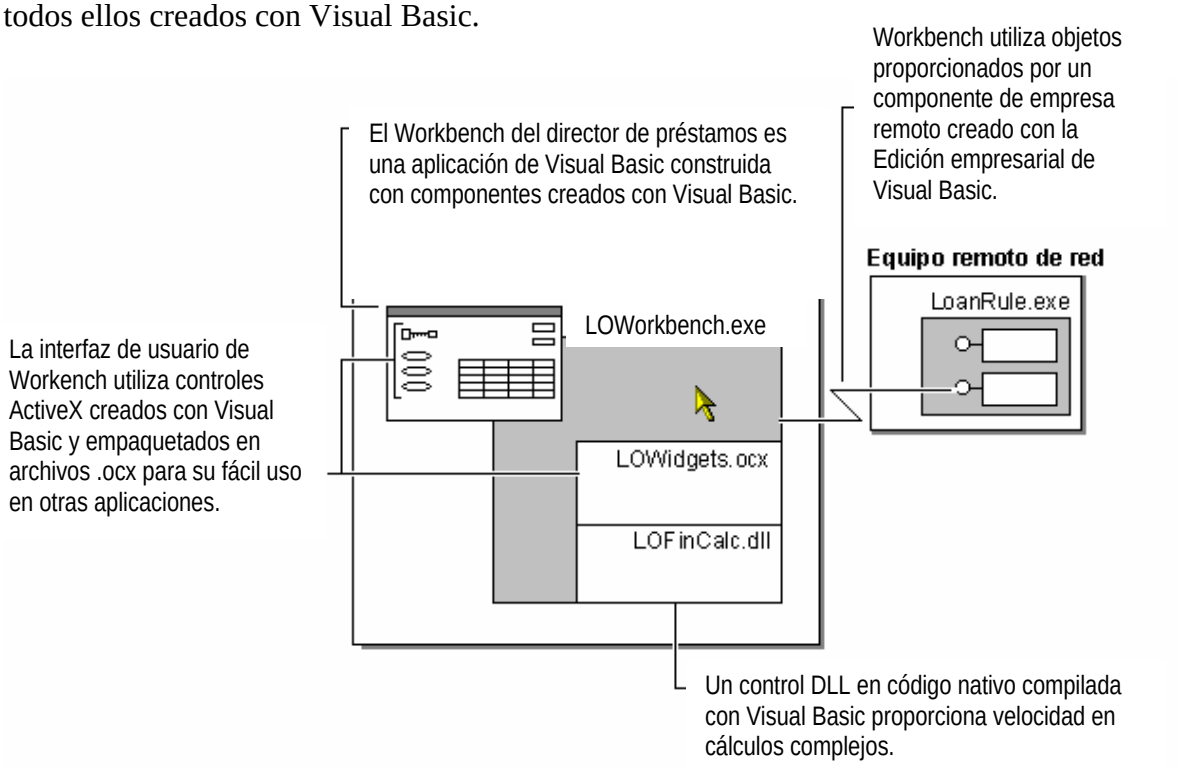


Figura 4.2 Aplicación de Visual Basic que utiliza componentes [15]

4.3.2 Diferencias entre DLL y OCX

Existen distintos tipos de componentes ActiveX y la mejor forma de elegir el tipo de control que se necesita, es tomando en cuenta la función que realizará del control ActiveX y el nivel de seguridad de la que puede dotar el tipo de control.

- Si se necesita un componente invisible que proporcione un servicio, es conveniente utilizar un componente de código (ya sea éste un control ActiveX Exe o un ActiveX DLL)[15].

    - Si se necesita un componente que se pueda ejecutar en el mismo proceso que su aplicación, es conveniente utilizar un control ActiveX DLL. Un ejemplo sería un componente que realizase cálculos complejos.

    - Si se necesita un componente que pueda servir a múltiples aplicaciones y se pueda ejecutar en un equipo remoto, lo que conviene utilizar es un ActiveX Exe. Por ejemplo, el mejor modo de implementar un servidor para un negocio que incluya de forma obligada un capítulo para impuestos, será utilizando un ActiveX Exe.

- Si necesita un componente visible que se pueda colocar en una aplicación en tiempo de diseño, le interesará generar un control ActiveX OCX. Un ejemplo de esto sería un control para números de teléfono que diese formato y validase los números de teléfono correctamente; un control de este tipo sería útil, sin duda, en muchas aplicaciones.

- Si necesita un componente visible que pueda ocupar una ventana de aplicación en tiempo de ejecución, es conveniente elegir un documento ActiveX.

Para el actual proyecto de tesis, se desarrollaron dos controles .ocx, debido a que permiten mayor seguridad que un control DLL o EXE. Es de mi conocimiento que, respecto a la funcionalidad de los controles ActiveX, la mejor opción para el presente proyecto de tesis debería ser el implementar la encriptación de la información en un control DLL, pero en contraparte con el aspecto de seguridad, se optó por implementar la criptografía en controles OCX. Lo anterior debido a que éstos últimos permiten identificar la rutina desde la cual se les invoca y no trabajan con múltiples procesos al mismo tiempo, cubriendo con esto dos aspectos importantes respecto a posibles ataques al sistema.

4.3.3 Firma de Controles ActiveX

Las licencias de los controles son un asunto delicado. Puede ocurrir que, después de haber dedicado cientos de horas a desarrollar un control, alguien colocara una instancia del mismo en un objeto determinado, expusiera todas las propiedades, métodos y eventos, agregara una o dos propiedades triviales y lo compilara y vendiera como un control nuevo[19].

El soporte para licencias de Visual Basic protege los derechos de autor de los controles ActiveX entre otros componentes. Al agregarlo a un componente de control, se compila con él una clave de licencia que cubre todos los controles que lo forman. Al ejecutar el programa de instalación se transfiere la clave de licencia al registro de otro equipo, permitiendo el uso de los controles en entornos de desarrollo. Con sólo copiar el archivo .ocx a otro equipo y registrarlo, no se transfiere la clave de licencia, por lo que no se puede usar los controles.

Al crear el archivo .ocx, Visual Basic creará un archivo .vbl con la clave de registro correspondiente a la licencia del componente de control. Cuando se utiliza el Asistente para instalar con el cual se crea un programa de instalación del archivo .ocx, el archivo .vbl se incluirá automáticamente en el procedimiento de instalación.

Funcionamiento de la licencia

Cuando un programador adquiere el componente de control y ejecuta el programa de instalación, la clave de la licencia se agrega al registro del equipo. A partir de ese momento, siempre que el programador coloque una instancia del control en un formulario, Visual Basic (o cualquier otra herramienta que emplee) indicará al control que se cree mediante la clave del registro. Si un programador obtiene una copia del componente de control, pero no la clave del registro, el control no podrá crear instancias de sí mismo en el entorno de desarrollo [19].

Cuando un programador compila un programa que utiliza uno de los controles que previamente se ha firmado por el autor del mismo, la clave de licencia del componente se incluye en la compilación. Al crear un programa de instalación para la aplicación, se incluye el .ocx correspondiente. Los usuarios pueden entonces adquirir el programa compilado y ejecutar la instalación. El control se instala en el equipo de cada usuario, pero la clave de licencia no se agrega al registro.

Las licencias y las aplicaciones de usuario de propósito general

Los programadores de empresas que creen controles ActiveX para que los utilicen usuarios finales de su propia compañía, pueden encontrar más cómodo omitir el soporte para licencia. Con ello será más fácil para los usuarios finales distribuir los documentos que contengan controles.

Las licencias y el autor del control

Supongamos ahora que alguien que ha adquirido el componente de control que se ha creado decide usar uno de sus controles para crear su propio control. Como ocurre con cualquier otro programa, cuando se compile el componente de control, la clave de licencia se incluirá en la compilación. El Asistente para instalar, creará una nueva clave de licencia para el componente nuevo, pero no agregará la del control creado al programa de instalación.

Cuando un programador instale este nuevo componente de código, la clave de licencia del control creado se colocará en el registro. Entonces, el programador ejecutará Visual Basic e intentará colocar una instancia del control en un formulario. En este momento se pide al control que se cree a sí mismo con la clave del registro. El control, a su vez, pide a sus *controles constituyentes* que se creen a sí mismos con sus claves de registro. Como en el registro no se encuentra la clave del control que se ha creado, el proceso fracasa.

Distribución de controles que utilizan controles con licencia

Si un segundo autor desea distribuir un nuevo control que incluye un control que se ha creado, deberá informar a los clientes de que para usar el control deberán tener instalado en su equipo el control que se ha creado.

Como alternativa, el autor del control puede negociar el derecho a distribuir la clave de la licencia del control creado junto con la de él en el programa de instalación.

En ambos casos se instalarán las dos claves de licencia en el equipo del programador, con lo que éste podrá crear instancias en tiempo de diseño de los controles del otro autor. Cuando se compilen los controles para formar un programa ejecutable, se incluirán en él las dos claves de licencia.

Cuando posteriormente un usuario instale y ejecute el programa, se pedirá al control del segundo autor que se cree a sí mismo. Con ello también se pide a sus controles constituyentes que se creen a sí mismos, y se les facilita sus claves de licencia (y así sucesivamente, si el control que se ha creado utiliza controles constituyentes con claves de licencia).

4.3.4 Twofish.ocx

El nombre del control ActiveX que permite encriptar y desencriptar la información por medio del algoritmo Twofish es twofish.ocx, buscando de esta forma poder tener flexibilidad y transoportabilidad a la hora de su implementación en otros proyectos, ya sean éstos grandes o pequeños.

4.3.4.1 Modo de operación del algoritmo

El diseño del algoritmo Twofish se basó en el del algoritmo "Blowfish". Twofish usa tablas internas de datos variables que dependen de la clave. La intención de B. Schneier fue la de crear un algoritmo rápido y robusto. Para ello tuvo en cuenta [16]:

1.- Forma de encriptamiento por bloques. Los bloques son de 128 bits.

2.- La longitud de las claves, de 128 bits, 192 bits y 512 bits.

3.- Hacer uso de operaciones eficientes en procesadores de 32 bits, tal es el caso del operador XOR.

4.- Encriptación de los datos en menos de 500 ciclos de reloj por bloque.

El algoritmo consiste en dos partes: una primera parte donde se prepara la clave (un precálculo inicial que se puede almacenar en memoria para una mayor velocidad de ejecución) y una segunda parte de encriptación de los datos.

El precálculo de la clave se encarga de dividir la clave (de un máximo de 512 bits) en varias subclaves que se almacenan en tablas hasta un total de 4168 bytes.

La encriptación de los datos transcurre en un bucle de 16 iteraciones. Cada iteración consiste en la permutación de una clave y en la sustitución clave-datos. Todas las operaciones son XOR y sumas de palabras de 64 bits. La única operación extra es la lectura de datos de cuatro tablas por cada iteración.

Twofish utiliza subclaves para la encriptación de los bloques. Estas subclaves deben ser precalculadas antes que se encripten o se desencripten los datos.

1.- La tabla P consiste en 34 subclaves de 64 bits.

2.- Hay cuatro S-Box (cajas de seguridad) de 64 bits por entrada y 256 entradas cada una (S1,...,S4).

Realización de la encriptación.

Como ya se ha mencionado antes, se tienen 16 iteraciones por palabra a encriptar. La entrada es una palabra de 128 bits, denominada como x. De ésta forma, los pasos a seguir para la encriptación son:

Dividir x en dos partes de 64 bits: xL, xR

Desde i = 1 hasta 16

xL = xL XOR Pi

xR = F(xL) XOR xR

Intercambia xL y XR

Intercambia xL y XR (deshace el último Intercambio)

xR = xR XOR P17

xL = xL XOR P18

Recombina xL y xR

Función F

Dividir xL en cuatro partes de 16 bits: a, b, c, d

F(xL) = ((S1(a)+S2(b)) mod 232 XOR S3(c)) + S4(d) mod 232

Para realizar la desencriptación se haría lo mismo que para la encriptación pero P1, ..., P34 se usarían en orden inverso.

Para calcular las subclaves se realiza lo siguiente:

1.- Primero se inicializa la tabla P y después las cuatro S-Boxes, en orden, con una cadena fija.

Esta cadena consiste en los dígitos hexadecimales de PI (sin el 3 inicial). Por ejemplo:

P1 = 0x243F6A88

P2 = 0x85A308D3

P3 = 0x13198A2E

P4 = 0x03707344

2.- Se hace la XOR de P1 con los primeros 64 bits de la clave, luego la XOR de P2 con los segundos 64 bits de la clave y así con todos los bits de la clave (posiblemente hasta P14). Este procedimiento se va repitiendo circularmente hasta que toda la tabla P ha sido XOR con los bits de la clave.

3.- Encriptar con la clave 0 con el algoritmo TwoFish usando las subclaves calculadas anteriormente (pasos 1 y 2).

4.- Cambiar P1 y P2 con la salida del paso 3.

5.- Encriptar la salida del paso 3 usando el algoritmo TwoFish usando las subclaves modificadas.

6.- Cambiar P3 y P4 con la salida del paso 5.

7.- Continuar el proceso, cambiando todas las entradas de la tabla P y después los cuatro S-Boxes en orden con la salida que va dando el algoritmo twofish.

En total se realizan 512 iteraciones para generar todas las subclaves. Las aplicaciones pueden almacenar las subclaves antes que ejecutar este proceso siempre.

4.3.4.2 Implementación del algoritmo

La implementación del algoritmo Twofish se realizó en dos etapas, la primera consistió en transcribir el código fuente del algoritmo Twofish del lenguaje ANSI C a Visual Basic. Inicialmente se comenzó a trabajar con ésta versión, pero presentaba algunos problemas al momento de encriptar los datos pues cuando la longitud del texto a encriptar era larga (más de 30 palabras aproximadamente), surgía un desbordamiento de pila al momento de generar las subclaves; y aunado al hecho de que han surgido mejoras en el algoritmo Twofish, se ha decidido actualizar la implementación del algoritmo Twofish con las mejoras actuales.

Como se había mencionado antes, cuando el NIST convocó al concurso para seleccionar al algoritmo AES, pidió que se entregara el código en lenguaje ANSI C, el motivo es la portabilidad de éste. Por lo cual, la totalidad de los algoritmos que concursaron se encuentran en ANSI C. De aquí se toman los códigos fuentes para transcribirse al lenguaje deseado.

El algoritmo que Bruce Schneier mostró a la opinión pública antes de la convocatoria del NIST, comparado con el algoritmo que hoy en día existe, es muy diferente, tiene cambios importantes, aunque todos han sido para mejorar al algoritmo.

Por lo anterior, la segunda etapa consistió en implementar la versión del algoritmo Twofish realizada por Jesper Soederberg [17] en el presente proyecto de tesis por los siguientes motivos:

- Es una versión actualizada del algoritmo Twofish.
- Es una versión probada y aprobada por Counterpane Internet Security, Inc[10].
- Los cambios más importantes que ha presentado el algoritmo Twofish desde sus inicios hasta el día de hoy, son dos: primero, Twofish acepta claves de diferente longitud (128, 192, 512 bytes), y segundo, al aceptar claves de longitud diferente, permite mayor rapidez al generar las subclaves de las cajas de seguridad (SBoxes), y por consiguiente, mayor velocidad de encriptación [16].
- Ya que el objetivo del proyecto de tesis no es la traducción del algoritmo nativo en ANSI C al Visual Basic, es correcto utilizar la versión ofrecida en el sitio http://www.counterpane.com/twofish/, pues ésta es libre de uso.

Como se había mencionado anteriormente, una de las metas que busca el actual proyecto de tesis es facilitar la implementación de seguridad e implementación con el mínimo de código y un ejemplo de ellos es la invocación de la función *Encriptar*, la cual realiza la encriptación de los datos que pasa por referencia a través de la variable *bInput* como se ve en la Figura 4.3 que muestra parte del código fuente que se utilizó en la implementación del algoritmo Twofish.

```
Public Function Encriptar(ByRef bInput() As Byte) As Byte()
    Dim bBlockInput() As Byte
    Dim bTemp() As Byte
    On Error GoTo ErrorHandle
      . . .
End Function
```

Figura 4.3.- Función de Encriptación

[10] Counterpane Internet Security, Inc. Compañía fundada por Bruce Schneier, creador del algoritmo Twofish.

Ahora, se ejemplificará la implementación del algoritmo en un caso concreto de programación bajo la plataforma de programación Visual Basic. Supongamos que tenemos la interfaz que se muestra en la Figura 4.4, ésta interfaz se realizó para mostrar la forma de encriptar y desencriptar utilizando el algoritmo Twofish, implementado en el control ActiveX twofish.ocx. En esta pequeña aplicación, lo que se hace es encriptar los datos que se introdujeron en el cuadro de texto "Texto a cifrar", lo único que se necesitó para realizar esta operación, además de la programación habitual de interfaz, fue introducir el código que se muestra en la Figura 4.5. Con una sola línea de código se pudo encriptar el texto deseado.

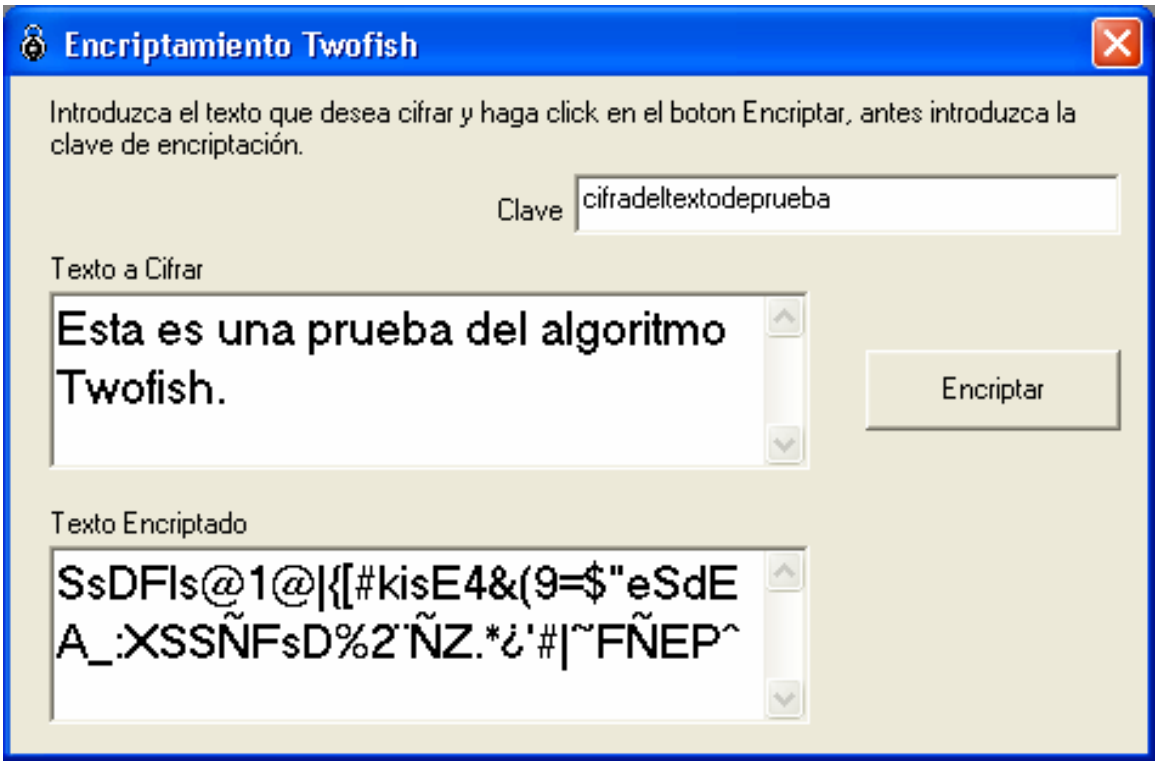


Figura 4.4.- Ejemplo de encriptación utilizando el control Twofish.ocx

```
TextoEncriptado.Text = Twofish.Encriptar(Texto_a_cifrar.Text)
```

Figura 4.5.- Código utilizado para la encriptación

```
TextoDesencriptado.Text = Twofish.Desencriptar(Texto_a_cifrar.Text)
```

Figura 4.6.- Código utilizado para la Desencriptación

De igual forma se realiza la desencriptación de la información como se muestra en la Figura 4.6, en ella se puede apreciar que se invoca la función *Desencriptar* la cual está también declarada en el control Twofish.ocx. Para ver el código completo, consulte el Apéndice A.

Como se ha apreciado, la implementación del algoritmo cumple con el cometido de que la programación sea fácil y rápida. En este tema, sólo se muestra la manera de implementar la encriptación mediante el control twofish.ocx, pero la solución de seguridad planteada en el presente proyecto permite implementar un nivel mayor de seguridad, en caso de utilizarse todas sus características. En el tema 4.4 *Asistente para la administración de usuarios,* se detallan todas las características de seguridad que se pueden implementar con el actual proyecto de tesis.

Respecto a la velocidad del algoritmo Twofish, éste es considerado como uno de los más veloces entre los candidatos finalistas del AES, puede encriptar un bit de información en dos ciclos del reloj de un Pentium. Eso significa que una PC trabajando a 400 MHz puede encriptar 200 Mbits o unos 20 Mbytes por segundo. Para tener una idea de la magnitud de esa velocidad, la misma PC puede encriptar o desencriptar a la vez a unas 20,000 de conversaciones telefónicas comprimidas [18].

A continuación se detalla una simple prueba de velocidad de cifrado de un archivo de tan sólo 7 Mb, en una computadora Pentium de 200 Mhz.

BlowFish = 4.4 Segundos
RC5 = 4.7 Segundos
Twofish = 5.8 Segundos
Idea = 5.9 Segundos

RC6 = 6.0 Segundos

Cast128 = 6.1 Segundos

Misty1 = 6.4 Segundos

Rijndael = 6.9 Segundos

Mars = 7.5 Segundos

Cast256 = 7.7 Segundos

Ghost = 8.0 Segundos

Lo anterior se probó con las herramientas de encriptación DozeCrypt [18], una de las más completas y confiables que se encuentra en Internet, y el algoritmo Twofish que se implementó en el actual proyecto de tesis, el cual cumplió con los tiempos mostrados en un reporte oficial del algoritmo publicado en http://www .counterpane.com/twofish.html.

De acuerdo con los tiempos obtenidos en las pruebas de encriptación, se puede decir que el algoritmo y la implementación de éste, superan las expectativas pues como sólo se desea encriptar con él las claves de acceso y algunos datos importantes de usuarios e información financiera de la organización o empresa, el tiempo para realizar estos procesos se llevan aproximadamente 0.5 segundos o menos

4.3.5 Entropia.ocx

Este es el nombre del archivo ActiveX que realiza las funciones de dar desorden a la información. Es criptografía con algoritmos propios, creados con la finalidad de complementar la seguridad que ofrece el algoritmo Twofish. Esta idea surge a raíz de comentarios hechos por criptologos tales como Bruce Schneier [17] y Dianelos Georgoudis [4] de implementar, además de un algoritmo altamente probado y analizado como es en este caso Twofish, otra forma de ocultar la información que no sea de conocimiento público, y que ésta, no pierda sencillez en la solución.

El control ActiveX entropia.ocx, permite utilizar el método criptográfico de sustitución simple con polialfabetos o el método de permutación ó la combinación de

ambos. La utilización de alguno de estos métodos o ambos, se indica a través de paso de parámetros. En la Figura 4.7, se muestra la función de la que consta el control:

```
Public Function Cifrar(cadena As Text, tipo As integer)
```

Figura 4.7- Declaración de la función Cifrar

Mediante la función cifrar se encriptan los datos contenidos en la variable de paso "cadena" y el método que se utilizará para la encriptación se indica por medio de la variable de paso "tipo".

Las opciones que reconoce la función "cifrar" para utilizar el tipo de encriptación, son tres:

1.- Cuando la variable de paso "tipo" tiene el valor de 1. En este caso, el método que se le está indicando utilizar a la función encriptar es el de Sustitución Simple con Polialfabetos.

2.- Cuando la variable de paso "tipo" tiene el valor de 2. Por medio de este valor, se indica la utilización del método de Permutación a la función "cifrar".

3.- Cuando la variable de paso "tipo" tiene el valor de 0. Este valor es el que tiene por default la variable, e indica a la función "cifrar" que encripte los datos utilizando los dos métodos consecutivamente, primero se encriptan los datos utilizando el método de sustitución simple con polialfabetos y después se encripta utilizando el método de permutación.

4.3.5.1 Método de Sustitución Simple con Polialfabetos

El método de sustitución simple con polialfabetos consiste en cambiar cada letra del texto a cifrar por otra diferente a ésta. El cambio o sustitución de las letras se realiza a través de una matriz de alfabetos (de ahí el nombre de polialfabetos), en total se tienen 16 alfabetos diferentes (aunque se pueden ir aumentado gradualmente), en donde cada alfabeto tiene un orden diferente de símbolos que lo conforman.

A cada letra del texto a encriptar o cifrar, se le asigna aleatoriamente un alfabeto diferente, por medio del cual, se sustituirá la letra en turno.

Cada alfabeto tiene una asignación de sustitución semejante a la mostrada en la Figura 4.8., en donde la letra A se sustituye por la letra B, la letra B se sustituye por la letra C y así consecutivamente.

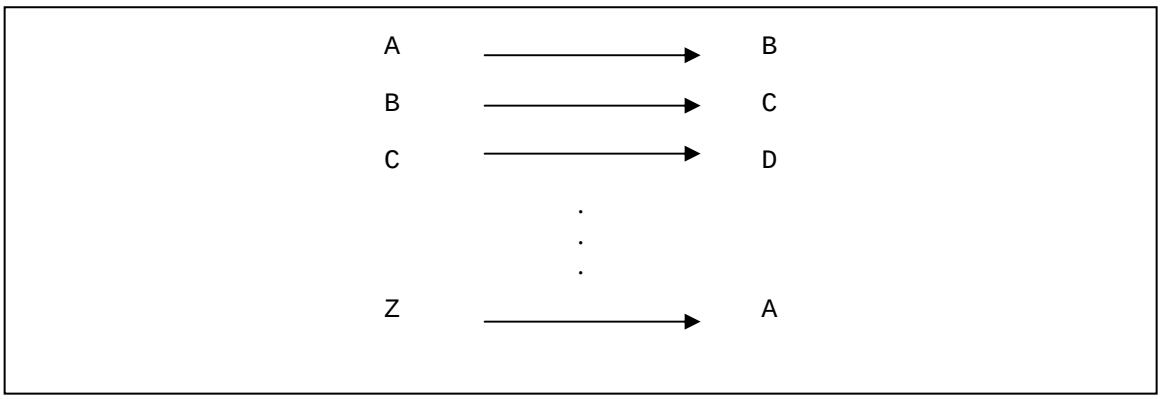


Figura 4.8.- Asignación de valores en un alfabeto.

Para este caso, la sustitución en todos los alfabetos no llevan un orden o continuidad, esto con el fin de crear mayor entropía en la encriptación y tener mayor seguridad en caso de ataques.

Por ejemplo, si se desea encriptar el texto “HOLA”, se tomará una a una cada letra del texto y se les asignará aleatoriamente un alfabeto de los 16 existentes para sustituir esa letra.

Por lo anterior, podría resultar que:

La letra “H” se sustituye por la letra “R” tomada del alfabeto 01
La letra “O” se sustituye por la letra “M” tomada del alfabeto 09
La letra “L” se sustituye por la letra “A” tomada del alfabeto 03
La letra “A” se sustituye por la letra “R” tomada del alfabeto 10

De esta forma, la cadena cifrada hasta el momento sería “RMAR”, es importante tomar en cuenta que la letra “H” y la letra “A” han coincidido en la letra sustituida en el momento de cifrarse, por lo que algún criptoanalista podría pensar que las dos letras

“R” del texto cifrado son las mismas, pero en la realidad no lo son, pues provienen de alfabetos diferentes, esto ayuda a nulificar ataques estadísticos.

Ahora surge la necesidad de poder identificar de qué alfabeto se cifró un símbolo determinado. Para esto, se convierte el número que ocupa el alfabeto en turno por una letra, ésta tomada de un alfabeto de asignaciones para cifrar el número asignado a los 16 alfabetos de la matriz, lo anterior se aprecia en la Figura 4.9.

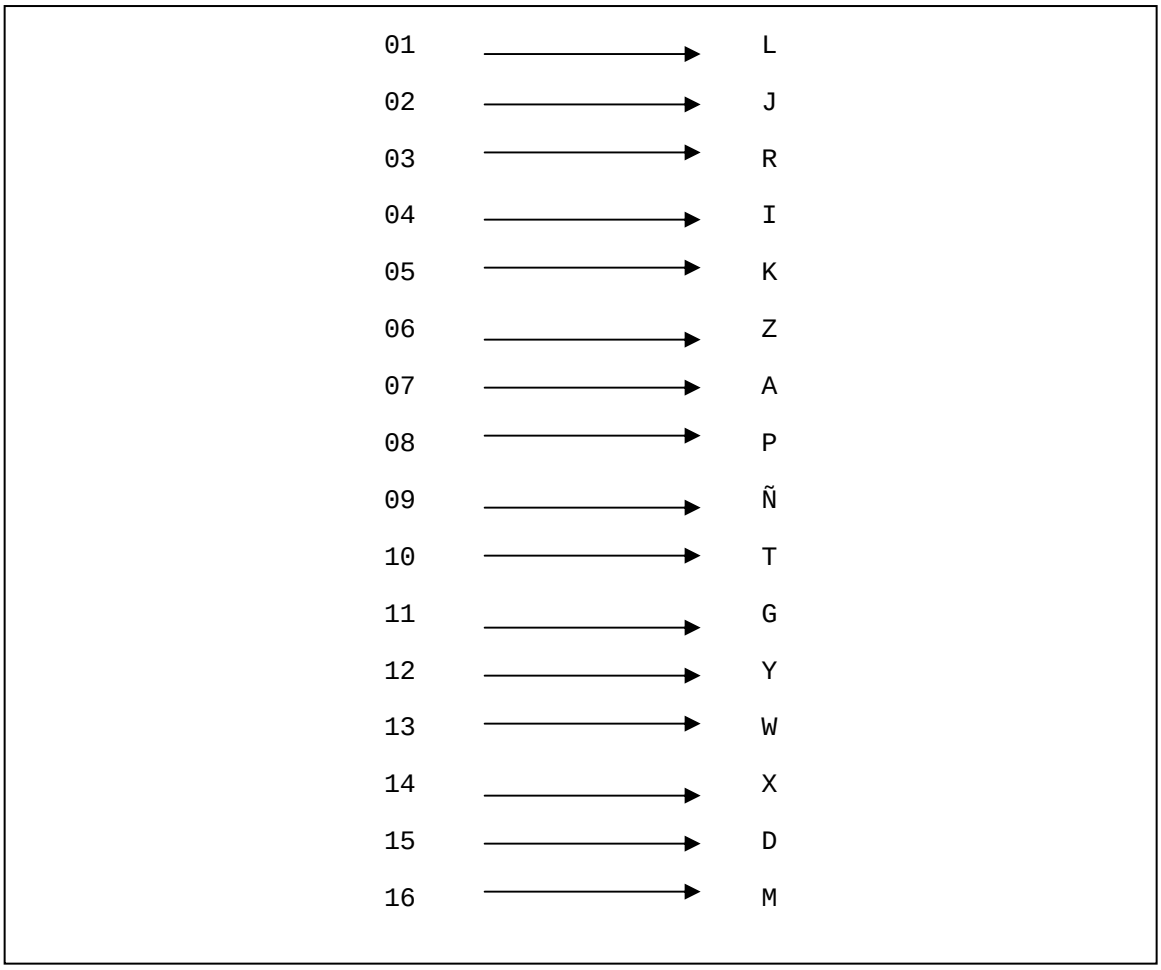


Figura 4.9.- Asignación de valores para cifrar el alfabeto utilizado.

Por lo tanto, el texto cifrado resultante será: **RLMÑARRT**, donde los símbolos **RMAR** representan a la palabra **HOLA** ya cifrada y los símbolos **LMRT**, representan al alfabeto utilizado para cifrar al símbolo que les precede.

Existen dos razones de peso para tener que cifrar el número de alfabeto utilizado en la matriz de los 16 alfabetos y son las siguientes:

1.- Al tener 10 alfabetos o más, como es el caso de este proyecto que cuenta con 16 alfabetos para el cifrado principal, a la hora de unir la cifra con el número de alfabeto utilizado para la cifra, el texto resultante es muy largo.

2.- Si se colocaran los números del alfabeto utilizado para la cifra de cada símbolo, se podría identificar inmediatamente de qué alfabeto se cifró un símbolo dado.

4.3.5.2 Método de Permutación

El método de la permutación consiste en cambiar de lugar los símbolos de un texto a cifrar, para el actual proyecto se divide en dos la cadena a cifrar, en caso de que la longitud de la cadena sea impar, se redondea a inmediato superior.

Una vez dividida la cadena, se realiza la permutación a una distancia de tres espacios de su lugar inicial para cada símbolo.

Continuando con el ejemplo de la sección 4.3.5.1, el texto cifrado quedará como se muestra en la Figura 4.10.

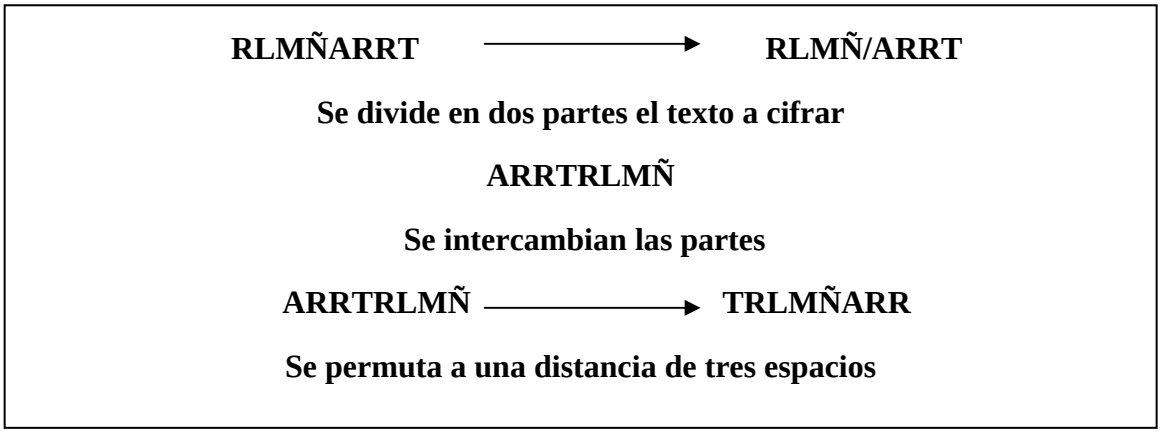


Figura 4.10.- Permutación a una distancia de tres espacios.

4.3.5.3 Métodos combinados

Este método viene por default en el actual proyecto, es decir, en caso de que no se indique valor en la variable de paso “tipo”, de la función, “cifrar”, se realizarán los dos métodos de cifrado, el de sustitución simple poli-alfabético y el de permutación. Es recomendable utilizar esta opción si se desea implementar mayor seguridad a la encriptación de datos.

4.4 ASISTENTE PARA LA ADMINISTRACIÓN DE USUARIOS

Ya que un administrador de usuarios necesita de muchos controles ActiveX, se optó por desarrollar un módulo tipo asistente, el cual facilita y automatiza tanto la instalación de formularios y bases de datos, como la inicialización de bases de datos para la definición del administrador del sistema y características RBAC. El asistente, así como los controles ActiveX .ocx, vienen incluidos en el programa de instalación el cual se puede acceder ejecutando el archivo setup.exe. Ver Figura 4.11

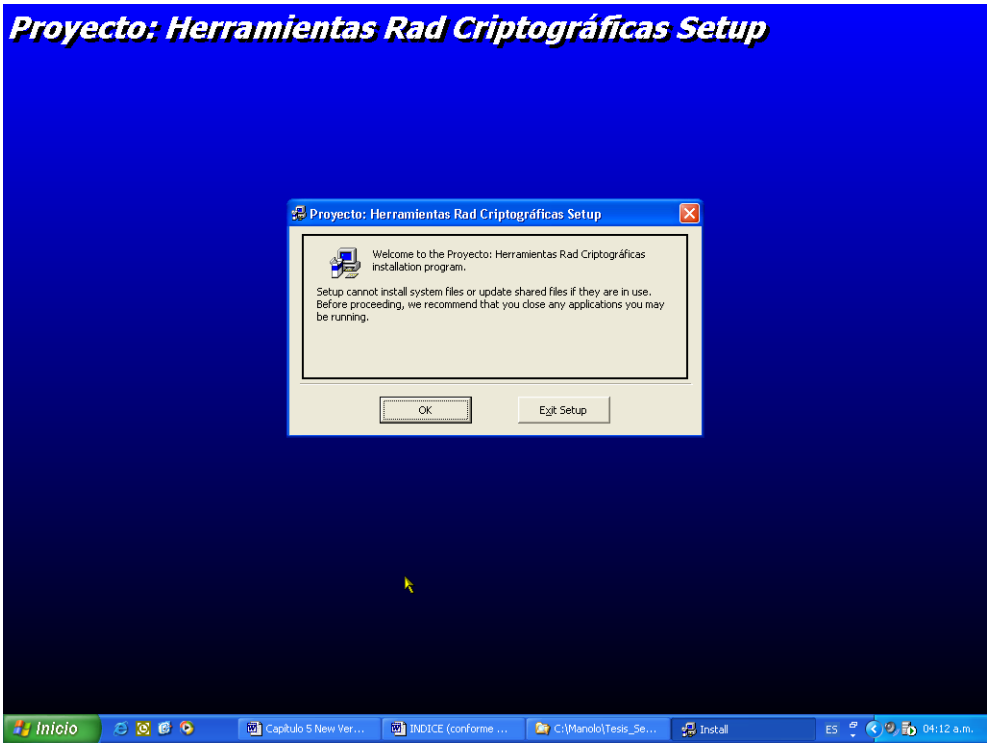


Figura 4.11 – Programa de instalación del proyecto de tesis.

Al instalarse el programa, se instalan los controles ActiveX .ocx en el sistema, al igual que las bases de datos y el asistente completo, del cual se describe su funcionamiento en los siguientes temas.

Después de esto, todo lo que resta por hacer es ejecutar el asistente para inicializar bases de datos, registros y variables necesarias para poder utilizar al conjunto de herramientas criptográficas RAD, que ayudarán a implementar seguridad criptográfica y entropía, y a administrar y controlar los accesos al sistema.

4.4.1 Inicialización del administrador de usuarios

El objetivo del asistente, es guiar paso a paso en el proceso de inicialización del sistema de seguridad que deseamos implementar en un desarrollo de software.

Como primer paso, se deberá ejecutar el archivo asistente.exe, al hacerlo, aparecerá una interfaz como se muestra en la Figura 4.12.

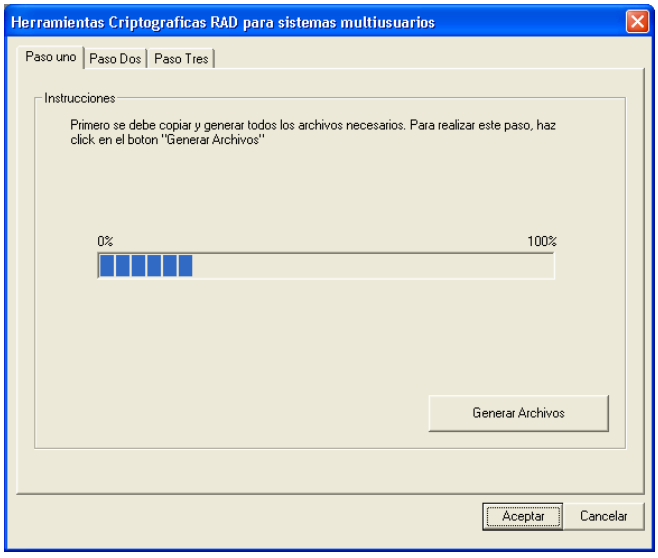


Figura 4.12 – Asistente, creación de archivos.

Al ejecutar el archivo asistente.exe, aparece una interfaz donde se pide, como primer paso, generar los archivos necesarios para su implementación en el sistema que se desea desarrollar.

Los archivos que se generan en este primer paso son:

1.- Formulario de claves de acceso.

2.- Formulario de autenticación de usuarios.

3.- Formularios para la administración de usuarios.

4.- Módulo de variables globales.

Cada uno de los formularios creados están listos para utilizarse, los archivos están ligados a las bases de datos (la ruta de acceso que tienen está dada por default, en la mayoría de los casos se deberá cambiar ésta por la que se necesite, dependiendo de la ubicación del proyecto), y cuentan con los controles necesarios para realizar la función esperada, así como también con el código necesario.

Es claro que los formularios que se generan no tienen diseño o funciones específicas, se podría decir que se entregan de forma austera para que cada programador le añada lo que necesite y le de un diseño de acuerdo al proyecto en mente. Por ejemplo, si se está desarrollando un sistema hotelero, es muy probable que se inserte el logotipo del hotel y se cambien los colores de fondo del formulario y del texto utilizado. De igual forma puede suceder si se desea colocar botones adicionales para poder acceder a otros sistemas utilizados para la administración del hotel.

En el segundo paso, el asistente pide se introduzcan los datos del administrador de usuarios, los cuales se guardan en las bases de datos utilizando encriptación Twofish dada por el control Twofish.ocx y dos niveles de entropía del control entropia.ocx, como se muestra en la Figura 4.13.

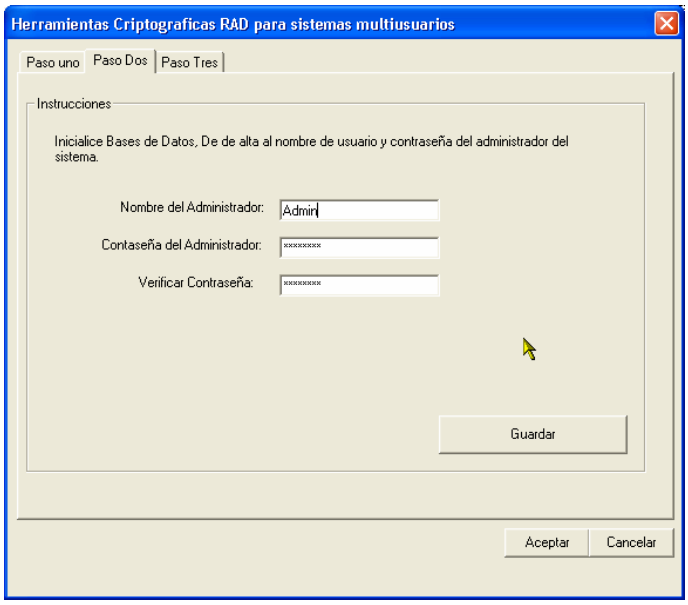


Figura 4.13 – Asistente, Inicialización de Bases de Datos

Por último, se debe indicar en qué directorio se desean colocar todos los formularios y demás archivos generados. Se recomienda colocarlos en un directorio independiente del desarrollo que se esté realizando, con la finalidad de aumentar seguridad al directorio a nivel de sistema operativo. Esto se muestra en la Figura 4.14.

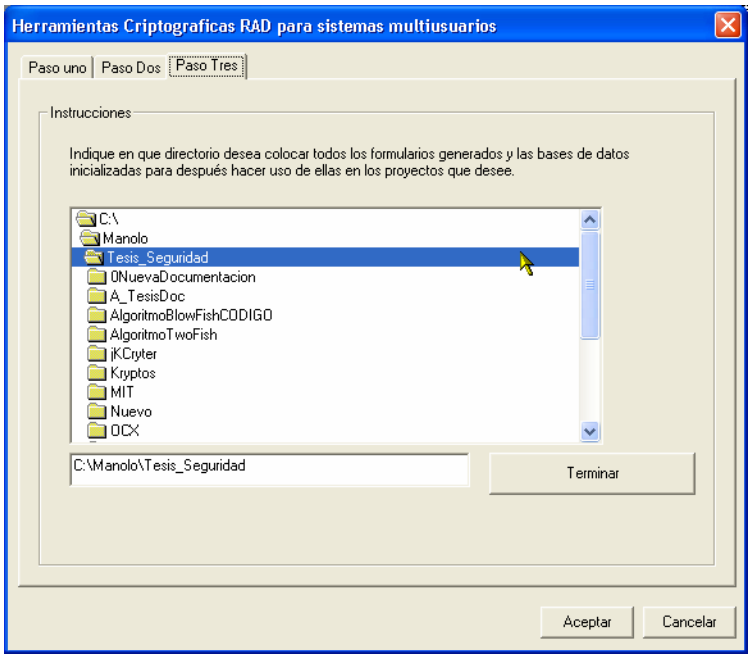


Figura 4.14 – Asistente, Indicar directorio destino.

Una vez generados todos los archivos en la carpeta deseada, lo único que se necesitará hacer, es agregar los formularios al proyecto que se desea dotar de seguridad computacional y control de accesos.

4.4.2 Altas, bajas y modificaciones de usuarios

Después de haber agregado los formularios al proyecto deseado, se puede comenzar a trabajar con ellos, lo primero es comenzar a dar de alta a usuarios y para poder acceder a este formulario, se tendrá que introducir el nombre y clave de acceso del administrador como se muestra en la Figura 4.15. En caso contrario, no se podrá acceder al formulario de alta de usuarios, pues éste detecta el inicio de sesión del administrador del sistema de seguridad. Es recomendable asignar la mayor seguridad al administrador de usuarios haciendo que el administrador pase por las etapas de identificación, autenticación y OTP. Esto lo decide el programador del proyecto, en tiempo de análisis y programación.

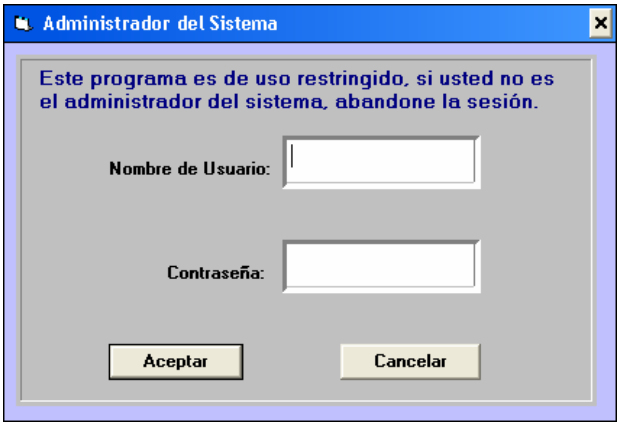


Figura 4.15 – Formulario de claves de acceso.

Después de haber introducido los datos del administrador del sistema de seguridad, se podrá administrar a los usuarios.

En la Figura 4.16 se muestra el formulario que se utiliza para dar de alta a los usuarios.

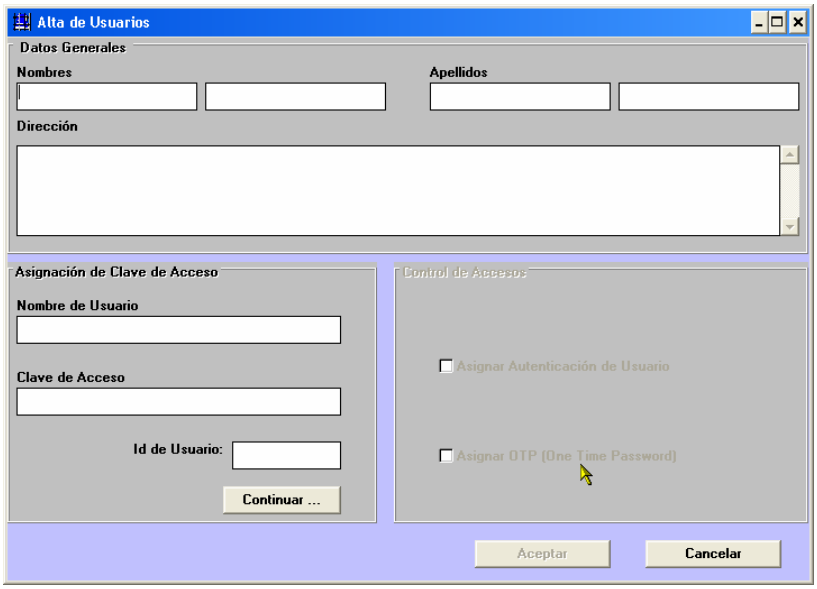


Figura 4.16 – Formulario de altas de usuarios.

Por medio de este formulario se guarda la información encriptada en bases de datos, utilizando el algoritmo Twofish para todos los campos, excepto en la clave de acceso, ya que para este caso, se encriptan los datos utilizando Twofish y los dos niveles de entropía, lo cual da mucho mayor seguridad a la información.

Entre las características especiales al dar de alta a usuarios, es la clave de acceso, la cual lleva este formato: *texto guión número.* Por ejemplo: puerta7correr-132, en donde *puerta7correr* es la parte de texto y *132* es la parte del número, ambos separados por un guión. La clave de acceso se guarda en bases de datos de forma separada y se interpreta conjuntamente en tiempo de ejecución.

La razón de dar este tipo de sintaxis a la clave de acceso, es que al ser encriptada, no se puede hacer una búsqueda para comparar lo que se tiene en base de datos con lo que se está introduciendo, por lo tanto, se utiliza el número después del guión para indicar el número de registro donde está alojada la clave de acceso del usuario que intenta entrar al sistema. Con lo anterior, se logra que la información viaje

encriptada a través de la red, pues lo que se hace, es acceder a la base de datos, extraer la información encriptada, llevarla a la terminal que solicito la información y allí, desencriptarla y compararla con la información que se esta introduciendo. Todo el proceso se muestra en la Figura 4.17.

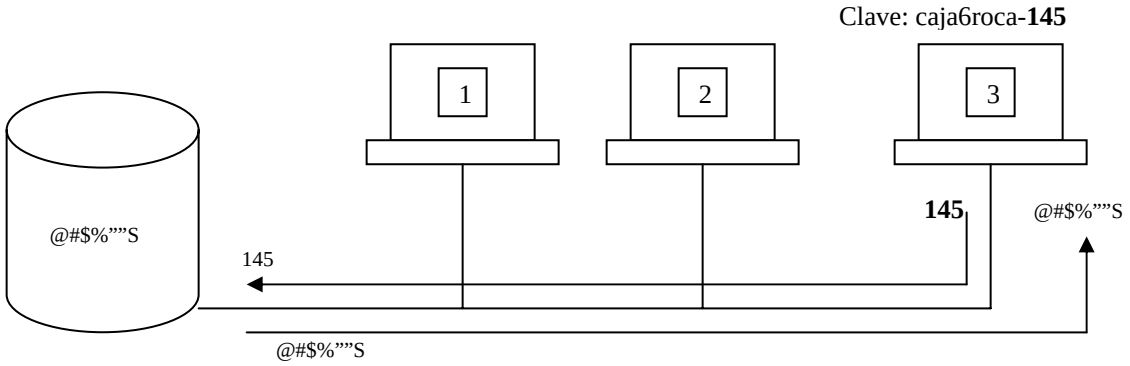


Figura 4.17 – Los datos viajan encriptados en la red.

Como se muestra en la figura anterior, en la terminal número 3 un usuario trata de acceder al sistema con la clave Avion6roca-145, al hacer clic en el botón *Aceptar*, el formulario de Clave de Acceso, guarda en memoria la clave de acceso introducida y el nombre de usuario, acto seguido, va a consultar la base de datos en el registro número 145, de allí extrae la información contenida en los campos *Nombre_usuario* y *Password*, y lleva los datos extraídos (aún encriptados) a la terminal que realizó la petición, una vez que se tienen los datos en la terminal de trabajo, se desencriptan para después compararse con los datos que se acaban de introducir.

La clave de acceso de los usuarios se guarda en la tabla “Usuarios”, la cual tiene la estructura mostrada en la Tabla 4.1. (sólo se muestran los tres campos necesarios para la clave de acceso)

Tabla 4.1 – Estructura de la tabla usuarios.

Usuarios			
Id_Usuario	….	**Login**	**Password**
144	....	*Manuel Arias*	*Tierra8salto*
Entero	….	***Memo***	***Memo***

Aspectos a considerar:

- La clave de acceso completa tiene la siguiente sintaxis: texto – número (texto guión número), ejemplo: *tierra8salto-144*
- Los campos *Login* y *Password* están encriptados a diferencia del campo *Id_Usuario*.
- En el campo *password* solo se guarda la parte de *texto* de la clave de acceso completa.
- La clave de acceso se reconstruye en tiempo de ejecución.
- La clave de acceso es de 128 bits como mínimo

4.4.3 Autenticación de Usuarios

La autenticación de usuarios consiste en verificar si realmente la persona que introdujo el nombre de usuario y la respectiva clave de acceso, es realmente ella y no una tercer persona que sabe ambos datos y desea sabotear el sistema.

Para implementar esto, al momento de dar de alta a un usuario se le realizan preguntas personales que entre el administrador de seguridad y el usuario definen.

Es recomendable dar de alta por lo menos de 5 preguntas para que éstas no se repitan constantemente a la hora de autenticar al usuario (ver Figura 4.18).

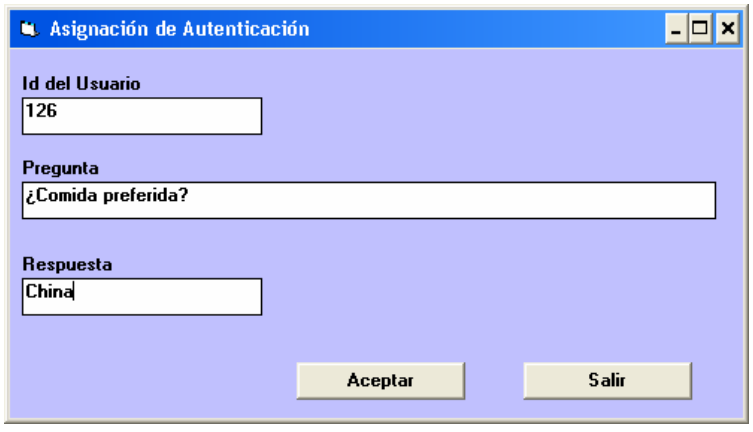


Figura 4.18 – Alta de preguntas para autenticación de usuarios

Después de haber dado de alta las preguntas para la autenticación, se podrá hacer uso del formulario utilizado para la autenticación de usuarios como se muestra en la Figura 4.19.

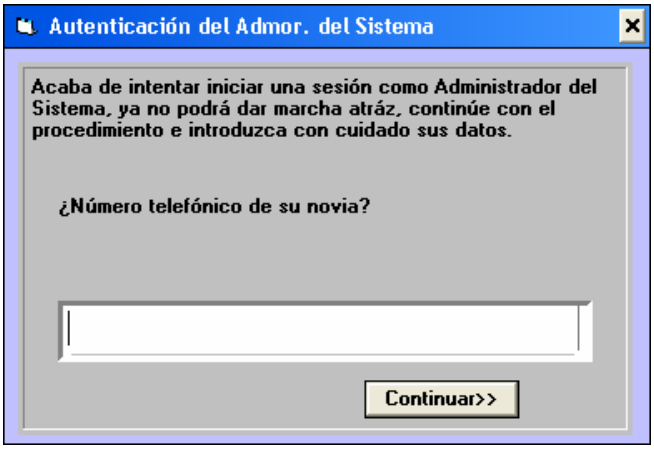


Figura 4.19 - Formulario para la autenticación de usuarios.

Al dar de alta a un usuario, se asigna o no esta opción marcándola como se ve en la parte inferior izquierda de la Figura 4.16.

Inicialmente las opciones de control de accesos al dar de alta a los usuarios, permanecen deshabilitadas. Se habilitan después de haber introducido correctamente los datos generales del usuario y asignar clave de acceso. Esta opción se puede activar y desactivar para cada usuario mediante el administrador de usuarios (Ver apéndice B).

La función que realiza el formulario de autenticación, es consultar la base de datos de autenticaciones en el registro que indicó la clave de acceso en el paso anterior, la desencripta y compara con la respuesta que se está introduciendo, todo esto después de haber presionado el botón *Continuar*. En caso de que los datos introducidos sean los correctos, se guardará en bases de datos el acceso al sistema y se mostrará el formulario correspondiente al programado, de lo contrario, en el primer intento fallido, se cerrará la aplicación y se guardará en bases de datos el intento de acceso como fallido.

4.4.4 RBAC

La opción de RBAC (*Role-Based Access Controls*), consiste en asignar alcance y limitaciones a cada usuario del sistema, con base en la función del rol que desempeñan en la empresa u organización.

Para que sea más comprensible esta opción, pongamos un ejemplo, supongamos que el usuario Pedro Rojas entra al sistema, que en este caso es un sistema de inventarios, punto de venta y administración para un supermercado, por mencionar alguno, con su nombre de usuario y clave de acceso correctamente. Pedro Rojas es un empleado que opera en una caja ó punto de venta del supermercado y él solo debe utilizar el sistema para cobrar a los clientes y algunas veces para buscar el precio de algún producto.

Por lo anterior, el ámbito de trabajo de Pedro Rojas se reduce a trabajar con dos o tres ventanas de todo el sistema. De esta forma, la opción RBAC ayudará a limitar el acceso a Pedro en caso de que él desee acceder a ventanas ó formularios a los que no está calificado o permitido hacerlo.

Para poder trabajar con esta opción, es necesario dar de alta primero las ventanas con las que cuenta el proyecto que se esta desarrollando (ver Figura 4.20) y después dar de alta el ámbito de trabajo de cada usuario por medio de la interfaz que se muestra en la Figura 4.21 de la siguiente página.

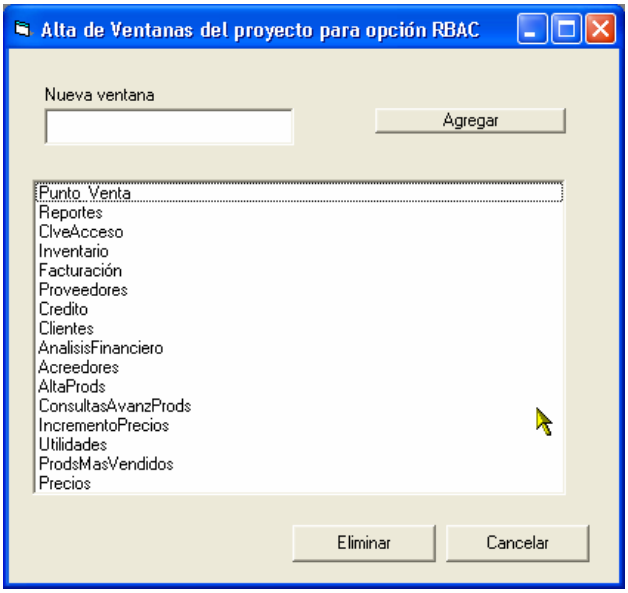


Figura 4.20 – Alta de ventanas del proyecto para inicializar opción RBAC

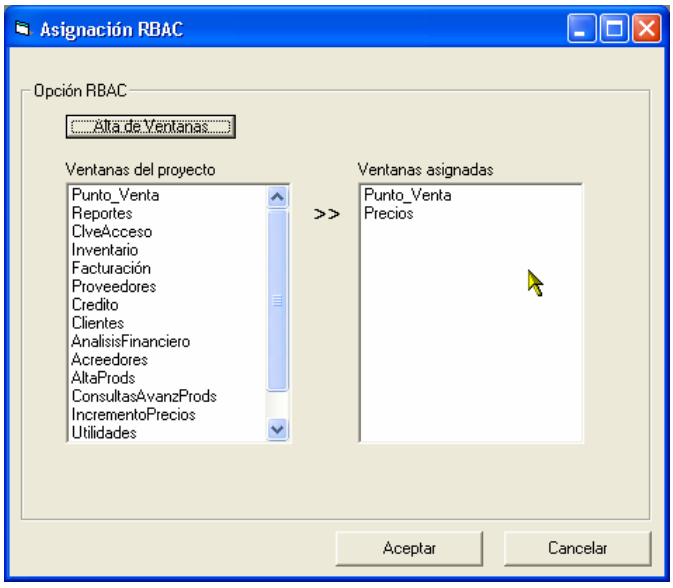


Figura 4.21 – Asignación RBAC a usuarios

4.4.5 OTP

La opción OTP (*One Time Password*), consiste en asignar a un usuario determinado, la opción de que su clave de acceso se cambiará cada vez que acceda al sistema. Estas medidas son tomadas cuando se ha detectado que han tratado de acceder

al sistema con el nombre de usuario y contraseña, pero ha sido fallido el intento por no haber introducido correctamente la clave de acceso varias veces consecutivas, o bien, por haber detectado un ataque al sistema y la vulnerabilidad consistió en que un usuario dio a conocer su clave de acceso.

La opción OTP se activa marcando la misma a la hora de dar de alta a usuarios o modificando sus datos (Figura 4.22).

Figura 4.22 – Asignación de Opción OTP

La opción OTP se puede activar o desactivar para cada usuario por medio del administrador de usuarios (ver apéndice B). Por lo general, se recomienda tener esta opción desactivada para evitar pérdidas de tiempo en el momento de acceder al sistema y solo activarla cuando se desee cambiar de clave de acceso por alguna circunstancia.

Una vez activada esta opción, cada vez que el usuario entre al sistema, primero le pedirá su nombre de usuario y clave de acceso (ver Figura 4.15), después pasará a la autenticación de este (ver Figura 4.19), y en caso de que se haya introducido correctamente todos los datos, el sistema le asignará una nueva clave de acceso como se muestra en la Figura 4.23.

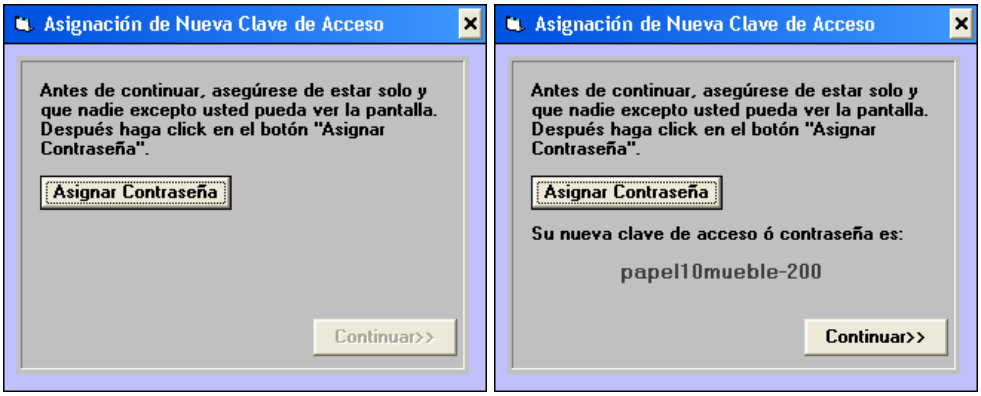


Figura 4.23 – Asignación de claves de acceso mediante OTP.

4.4.6 Bloqueo automático de contraseñas

En un sistema de seguridad es muy necesario contar con un bloqueo automático de contraseñas, ya que para un ataque al sistema por medio de la interfaz de usuario, el bloqueo automático de contraseñas representa un obstáculo importante para frenar los ataques. El bloqueo de contraseñas ó claves de acceso se puede dar por las siguientes razones:

1.- Cuando se introduce mal la sintaxis de la clave de acceso por tres veces consecutivas. En este caso se bloqueará el sistema en su totalidad.
2.- Cuando se introduce la clave de acceso y el nombre de usuario, ambos erróneamente por tres veces consecutivas. En este caso se bloquea automáticamente el Id de Usuario correspondiente.

4.4.7 Registros de accesos al sistema

Cada vez que un usuario accede al sistema, se guarda en bases de datos, encriptado con Twofish, el día, la hora de inicio de sesión, nombre de usuario y hora de fin de sesión.

También el sistema guarda los accesos fallidos al sistema con el objeto de identificar a los usuarios que constantemente introducen sus datos erróneamente, o bien, para poder identificar a través de qué usuarios y claves de acceso están intentando realizar un ataque.

Por último, el sistema guarda un record de los bloqueos realizados a los usuarios, esto con el fin de sancionar, cambiar o tomar medidas enérgicas con los usuarios que no estén haciendo un conciente y buen uso del sistema.

4.5 PRUEBAS INTERNAS

Como se mencionó en la introducción de este capítulo, las pruebas internas consisten en probar cada uno de los módulos de programación por separado. Tienen como objetivo probar la parte programada para identificar errores de programación y solucionarlos inmediatamente, por lo tanto, los resultados se pueden apreciar al ver el funcionamiento del actual proyecto en conjunto. Las pruebas internas son realizadas únicamente por el programador.

Los errores más comunes detectados en las pruebas internas fueron:

1.- Mala asignación de variables.

2.- Error al utilizar las propiedades de los objetos incluidos en un formulario.

3.- Error al momento de invocar algún método de un objeto en particular.

4.6 IMPLEMENTACIÓN DEL PROYECTO

Para poder saber si efectivamente el actual proyecto de tesis es eficiente en las metas que se trazaron, la mejor forma de comprobarlo es implementando éste en un proyecto real. El proyecto que se eligió, es un sistema de inventarios y punto de venta, denominado con el nombre de “Ventari Ver. 9.0”, en el cual se utilizan todas las herramientas de los que consta el actual proyecto de tesis.

En la Figura 4.24 se muestra el proceso que se realizó para llegar a implementar el actual proyecto de tesis en el sistema Ventari Ver. 9.0, pasando por la instalación de las herramientas criptográficas y de seguridad, que hasta ese punto el ámbito de trabajo es totalmente para el programador, hasta llegar a la instalación del software final, ya implementado en éste, las herramientas de criptografía y seguridad del actual proyecto de tesis.

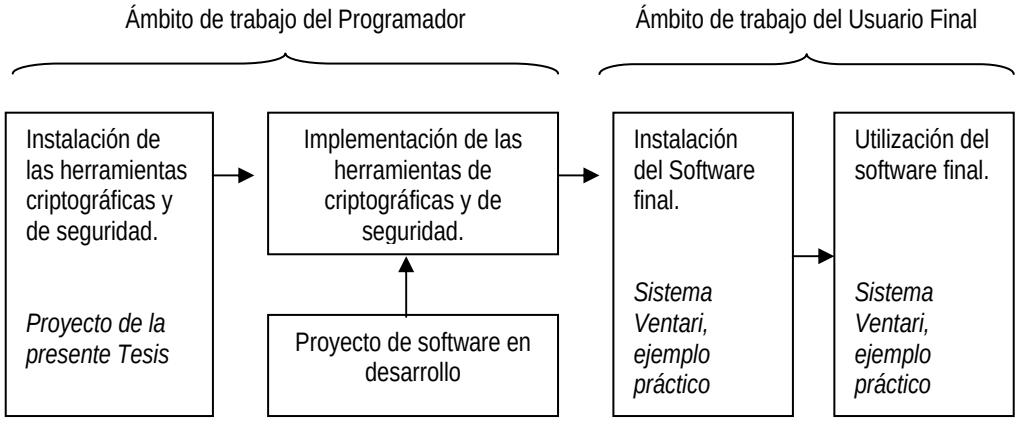


Figura 4.24- Ámbitos de trabajo del programador y usuario final.

En este tema solo se muestran las ventanas en donde se utiliza la implementación del proyecto de tesis y se toma en consideración que están implementadas y activadas todas las posibles opciones de seguridad de las que puede dotar el actual proyecto de tesis.

Una vez instalado el sistema de seguridad, visto anteriormente en el tema 4.4, se procede a implementar las herramientas en el sistema en desarrollo. La implementación corre por cuenta del programador, el cual se debe basar en el manual del programador para saber cómo utilizar cada herramienta y módulo del presente proyecto de tesis (ver apéndice B).

Para esta implementación, se creó un archivo ejecutable con el nombre de ventari.exe, a través del cual, al ejecutarse, se podrá acceder al sistema.

En la implementación del proyecto, se programó el sistema Ventari Ver. 9.0 para que mostrará una ventana de presentación (habitualmente llamada “Splash”) al inicio del programa (ver Figura 4.25).

Figura 4.25 – Ventana de presentación

Posteriormente muestra una ventana donde pide la identificación del usuario, en donde se le pide introduzca su nombre de usuario y clave de acceso.

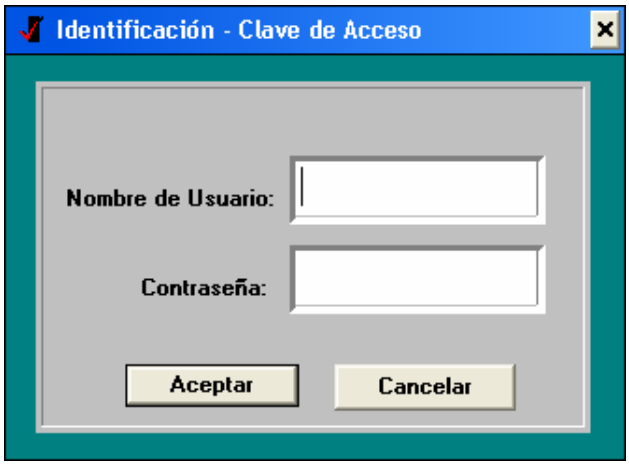


Figura 4.26 – Identificación por nombre y contraseña.

En caso de no haber tecleado correctamente, ya sea, el nombre de usuario y/o contraseña, el usuario tendrá hasta dos intentos más para poder introducir los datos correctamente. De no ser así, el sistema se cerrará (ver Figura 4.27).

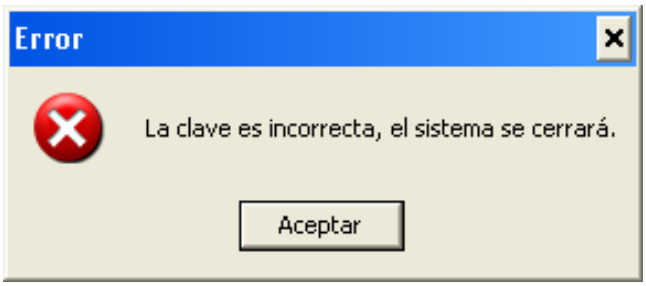


Figura 4.27 – Mensaje de error en la identificación de Usuario

Si se introdujeron datos correctamente, el sistema mostrará una ventana donde se autenticará la identidad del usuario (ver Figura 4.28). Para poder acceder al sistema Ventari, se debe de responder correctamente la pregunta que se hace en la etapa de autenticación. En caso de no saber la respuesta ó haber errado en ella, se cerrará inmediatamente el sistema Ventari, de ser así, se tendrá que volver a intentar acceder al sistema, ejecutando nuevamente el archivo ventari.exe.

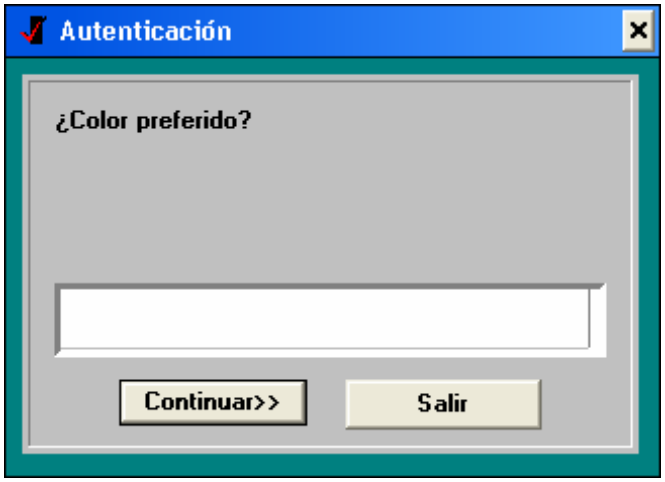


Figura 4.28 – Autenticación de usuario

En este caso, se activó la opción OTP (One Time Password) en la cuenta de todos los usuarios con el objeto de mostrar todas las características del sistema de seguridad. La opción OTP cambia automáticamente la clave de acceso a los usuarios cada vez que acceden al sistema, por lo tanto, el sistema Ventari le mostrará una ventana donde le asigna una nueva clave de acceso, la cual utilizará la próxima vez que intente acceder al sistema. La opción OTP muestra primero un texto de advertencia y

hasta no haber presionado el botón “Asignar Contraseña”, no permitirá continuar, ni mostrar la nueva clave de acceso (ver Figura 4.29).

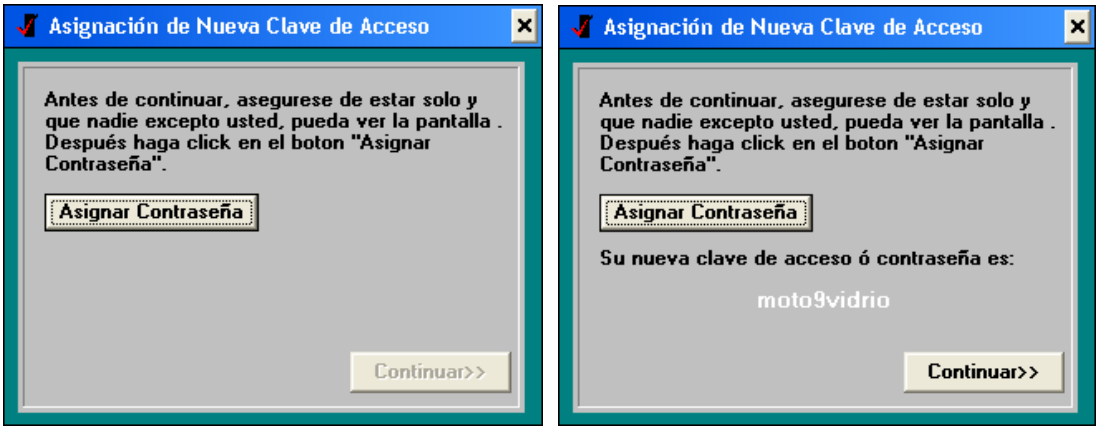


Figura 4.29 – Asignación de nueva clave de acceso.

Una vez introducido correctamente los datos pedidos por el sistema en las etapas de identificación y autenticación, el sistema mostrará el menú principal ilustrado en la Figura 4.30.

Figura 4.30 – Menú principal del sistema Ventari

4.7 PRUEBAS EXTERNAS

Las pruebas externas consisten en evaluar el funcionamiento del sistema de seguridad en su totalidad. A diferencia de las pruebas internas, las externas son aplicadas en circunstancias reales, en diferentes equipos de cómputo y utilizando el sistema desarrollado por usuarios finales.

Para ello, se implementó el proyecto de tesis en el sistema Ventari Ver. 9.0 y se evaluaron los siguientes aspectos:

Funcionamiento óptimo del sistema final

- Al instalar el sistema en diferentes computadoras, presentó problemas en su instalación ya que en la misma computadora estaban instalados controladores del programa MS Visual Basic Ver. 6.0 y 7.0 (MS Visual Studio .NET). Investigando a fondo la compatibilidad de versiones, se encontró con que no pueden estar instaladas en la misma computadora, versiones 3.0 y 4.0 del lenguaje de programación MS Visual Basic. De igual forma, no pueden estar instaladas al mismo tiempo las versiones 5.0 y 7.0. Por lo anterior, se instalaron solo los controladores necesarios y de esa forma se resolvió el problema
- Otro problema presentado fue al instalarse el sistema Ventari Ver. 9.0 en computadoras funcionando bajo el sistema operativo Windows XP. El problema radica en haber dejado inhabilitadas propiedades de algunos objetos. El problema se solucionó activando estas propiedades, pues el API de Windows XP, así lo requiere.
- Aparte de los problemas antes mencionados, no se encontró algún otro y considerando que estuvieron trabajando tres equipos de cómputo diferentes en la evaluación durante 10 días, se considera que la etapa de evaluación mínima requerida se ha pasado con éxito.

Amigabilidad del sistema.

Al instalarse el sistema, se activaron todas las características de seguridad de las que puede dotar el actual proyecto de tesis (identificación, autenticación, OTP y RBAC), obteniendo los siguientes resultados:

- Los usuarios necesitaron una capacitación de 3 a 4 días como mínimo para poder habituarse al sistema.
- Durante los primeros días, los usuarios tardaron en promedio de 30 a 40 segundos en entrar al sistema para poder operarlo. Las causas primordiales fueron la identificación de usuario y la característica OTP. En ellas se observó que los usuarios al introducir el nombre de usuario y su clave de acceso, constantemente erraban a raíz de la nueva sintaxis, pero a partir del cuarto día, ya estaban habituados a ella. Respecto a la característica OTP, los usuarios tardaban en escribir en papel su nueva clave de acceso, ninguno hizo el intento de memorizarla, lo cuál representa mayor peligro pues al perderla y caer ésta en conocimiento de otra persona, la única defensa del sistema era la autenticación.
- Retomando el punto anterior, la autenticación resultó muy buena solución, pues frenaba perfectamente los intentos de ataques al sistema cuando una tercer persona encontraba una clave de acceso escrita en papel de un usuario determinado.
- Por otra parte, el utilizar claves de acceso compuestas en tiempo de ejecución (ver tema 4.4.2), permitió evitar que ataques al sistema tuvieran éxito, pues la mayoría de los usuarios, cuando escribían su clave de acceso en un papel, solo escribían la primer parte (la parte de texto dada por la característica OTP) ya que la segunda parte (número de usuario) siempre permanece igual.

Analizando de esta forma el funcionamiento del sistema, se tuvieron los siguientes resultados en concreto:

- Se realizaron 186 intentos de acceso al sistema, lográndose exitosamente 142, lo cual representa el 76.34% de los accesos.

- Se registraron 3 bloqueos a las claves de acceso a usuarios y solo un bloqueo total del sistema.
- Se observó que los usuarios tardaban mucho tiempo en entrar al sistema, por lo que se optó por sólo activar las opciones de identificación y autenticación para todos los usuarios. La opción de OTP, sólo se utilizó para cambiar claves de acceso a usuarios y en casos donde se sospechaba que una clave de acceso de un usuario ya era conocida por una tercer persona.
- Al desactivarse la opción OTP, la clave de acceso fue la misma hasta nuevo cambio. Por lo anterior, se observó que los usuarios ahora recordaban su clave de acceso y no la anotaban en papel como sucedía anteriormente. El memorizar la clave de acceso fue fácil debido a la sintaxis que se utiliza, ejemplo: carro3piedra.

4.8 POLÍTICAS DE SEGURIDAD

En este tema se mencionan recomendaciones que son muy necesarias de tomar en cuenta al momento de implementar y estar trabajando con la solución de seguridad del actual proyecto de tesis.

Las políticas de seguridad se indican a continuación y tienen como objetivo poder minimizar ataques al sistema, ya sean estos de forma directa (a través de la interfaz de usuario o bases de datos) o indirecta (por medio de espías en la red o accesos remotos a nivel sistema operativo). Las políticas de seguridad se dividen en dos: políticas a cargo de la gerencia (administrador del sistema, administrador de red, directores de la empresa, departamento de sistemas) y políticas a cargo del usuario final.

Políticas a cargo de la gerencia.

1. Después de haber implementado el sistema de seguridad, es indispensable capacitar al personal que va a estar trabajando con dicho sistema, solo así se podrán obtener los resultados deseados.

2. La capacitación deberá de ser considerada como una actividad obligatoria y de suma importancia para que se le tome la seriedad requerida.
3. En la capacitación se pueden afinar detalles de las características de seguridad con las que deberá estar operando el sistema de seguridad. Lo anterior se determina con base en las necesidades, infraestructura y requerimientos de la empresa u organización.
4. Siempre es conveniente tener dos partes encargadas de la seguridad de la organización, donde una parte cuide a la otra parte y viceversa (entendiéndose por parte, como la persona o grupo de personas encargadas de la seguridad o, inclusive, puede ser una empresa externa). Esto debido a que la responsabilidad de la seguridad informática, siempre debe de involucrar a los directivos de la organización y no solo a la persona encargada de la administración de seguridad.
5. Respecto al administrador de usuarios, se recomienda que el nombre de usuario y clave de acceso, nunca se escriban en papel, por el contrario, memorizar los datos es la mejor opción. Para esto, se debe tener en cuenta que la característica de OTP en el administrador de usuarios, no deberá estar activada, salvo en caso de ser necesario.
6. Las claves de acceso tienen un tiempo de caducidad, hablando en el sentido de seguridad; es decir, se recomienda que las claves de acceso se cambien cada determinado tiempo.
7. El administrador del sistema de seguridad deberá de informarle al usuario final hasta cuántas veces se le es permitido equivocarse al introducir los datos, esto ayuda a prevenir que se bloquee el sistema completo o su cuenta de acceso.
8. Es recomendable colocar el desarrollo y bases de datos en un directorio al cual solo tenga acceso el administrador del sistema de seguridad. Esto se debe de coordinar con la persona encargada de la administración de red y del sistema operativo (en caso de haberlos), permitiendo así, integridad física de la información, aunque ésta la dé el sistema operativo y/o protocolos de red.
9. Ya que el actual proyecto está pensado para su implementación en sistemas operativos Windows; es del conocimiento público que éste sistema operativo tiene muchos problemas de seguridad (lo cual es una de las causas por las que se

realizó el presente proyecto de tesis), por lo que se recomienda investigar cómo cubrir esas deficiencias para prevenir ataques directos a bases de datos y archivos.

10. Ya que el sistema de seguridad del actual proyecto de tesis no detecta inactividad en el sistema, se podría dar que un usuario por descuido, deja abierta su sesión en el sistema y por algún motivo, se ausenta de su lugar de trabajo. De esta forma se presentan muchos ataques al sistema, por lo cual, se recomienda que el programador implemente esta parte de desarrollo, y como complemento al problema plantado, se advierta a los usuarios no dejar sesiones abiertas. En caso de necesitar ausentarse de su lugar de trabajo, deberán por lo menos, cerrar la sesión iniciada en el sistema.
11. Se sabe que la mayor cantidad de ataques provienen de personas internas a la empresa u organización [12]. Por lo anterior se recomienda que los usuarios no den a conocer el nombre y clave de acceso a los compañeros de trabajo.
12. Al dar de alta a un usuario e introducir las preguntas que le realizará el sistema se seguridad en el momento de autenticarse, éstas deberán ser lo más personales que se pueda, de otra forma, este nivel de seguridad será nulo y se estaría desperdiciando una buena herramienta de seguridad.

Políticas a cargo del usuario final.

1. Respecto a la seguridad a nivel interfaz, la primera etapa de seguridad del actual proyecto es la identificación. En ella se deben de introducir los datos cuidando mayúsculas y minúsculas, tanto en el nombre de usuario, como en la clave de acceso.
2. Estando en la etapa de autenticación, se debe tener cuidado al introducir los datos, puesto que el sistema, en esta etapa, no permite equivocarse y volver a intentar introducirlos. Esto se debe a que las preguntas de autenticación son personales y por lo tanto, el usuario, mejor que nadie, sabe la respuesta a las preguntas.

3. No dejar abiertas sesiones del sistema que se esté utilizando y ausentarse de su lugar de trabajo. En caso de necesitar ausentarse de su lugar de trabajo, deberán por lo menos, cerrar la sesión iniciada en el sistema.

4. Es muy importante no dar a conocer, por ningún motivo, su nombre y clave de acceso a los compañeros de trabajo.
5. Las preguntas y respuestas dadas al administrador del sistema para su autenticación, deberán de tomarse como confidenciales entre la empresa y usted. Tratar de no hablar con sus compañeros de trabajo sobre los temas que involucran a las preguntas de autenticación, es lo más correcto.

Capítulo 5

Conclusiones y trabajo a futuro

5.1 CONCLUSIONES

El actual proyecto de tesis ha sido un trabajo extenso tanto en la investigación del tema como en el desarrollo e implementación del software. Hablar de seguridad computacional no solamente es hablar de un lenguaje de programación y un problema concreto a solucionar, sino es hablar de todo un proceso a seguir el cual involucra, tecnología, conocimiento y capacitación.

Con base en lo desarrollado en esta tesis, se concluye lo siguiente:

- La tesis resuelve el problema de dotar de seguridad en la información, teniendo en consideración que las características de seguridad de las que provee son: identificación, autenticación, roles, transacciones y control de accesos.
- La flexibilidad en la implementación del sistema es aceptable pues puede implementarse tanto en proyectos pequeños donde solo se necesite de encriptación, como en proyectos grandes, donde además de encriptación, sea necesario implementar control de accesos, identificación, autenticación y registros al sistema.
- El sistema de seguridad presenta puntos débiles, los cuales enumero a continuación.

1.- Mejorar la asignación de roles con la opción RBAC. La solución planteada presenta una forma limitada de asignación, lo óptimo es que también cambiará la interfaz y menú para cada usuario.

2.- Poder hacer uso de otros algoritmos de encriptación tales como AES y RC5. De hecho esto es posible pues el algoritmo Twofish es en si un archivo ActiveX con formato .OCX, por lo cual, al implementar más algoritmos, estos se compilarían en archivos de la misma índole.

4.- Migrar la interfaz de la solución tipo asistente a Visual Basic versión Visual Studio .NET.

5.2 TRABAJO A FUTURO

Este tema contiene información acerca de tecnologías y formas de implementación que ayuden al control de accesos e implementación del proyecto en otros ámbitos de la computación y que podrían implementarse en un futuro, es por eso que no se profundiza en los temas, sino que se proponen como temas de investigación y solo se aportan ideas de posibles formas y herramientas de desarrollo.

El actual proyecto se podría implementar en Internet si se cuenta con lo siguiente:

- Los controles ActiveX se pueden instalar en servidores Windows NT y Windows 2000, permitiendo así, poder invocar sus funciones y poder encriptar y desencriptar datos.
- Para poder implementar la interfaz de administración en Internet, se necesitará contar con la plataforma de desarrollo Visual Studio Ver. 6.0 ó mejor aún, Visual Studio .NET, con ella se podrán generar, casi directamente, páginas DHTML con contenido de formularios Visual Basic.
    - Para complementar la programación de subrutinas y conexiones a base de datos ya sea esta MS Access ó MS SQL Server, es preferible utilizar MS

ASP.NET Web Matrix, esta herramienta es gratuita y se puede obtener del sitio: http://www.microsoft.com/

Todo lo anterior podría ser posible de implementarse haciendo mínimos cambios en la programación, pero se obtendría una solución bastante lenta y de difícil mantenimiento, lo mejor sería reutilizar solo los controles ActiveX, y desarrollar aparte la interfaz administradora de usuarios. Ésta podría ser desarrollada en el lenguaje ASP, ya que se está trabajando bajo el sistema operativo Windows. Se recomienda utilizar como plataformas de desarrollo a Dreamweaver MX de Macromedia y ASP.Net Web Matrix de Microsoft. Con Dreamweaver se puede realizar fácilmente la interfaz, conexión a BD's y altas, bajas, modificaciones y búsquedas de datos. Por su parte, con Web Matrix, se podrán generar formularios muy semejantes a Visual Basic y, con esto, poder reutilizar parte del código del proyecto, lo anterior debido a que Web Matrix permite código nativo de Visual Basic y traduce las instrucciones a Visual Basic Script.

Referencias

[1] Manunta, Giovanni. Presentación del Libro: Seguridad: Una introducción. Revista virtual seguridad corporativa. http://www.seguridadcorporativa.org

[2] Aldegani, Gustavo Miguel. "Seguridad Informática". MP ediciones. Argentina 1997.

[3] Schneier, Bruce. "Applied Cryptography". Editorial John Wiley & Sons. 2ª Edición. 1995.

[4] Dianelos Georgoudis, "AES, el algoritmo de encriptación del próximo siglo". Junio 1999. http://www.tecapro.com

[5] Enciclopedia Médica Familiar, Your Family Doctor. No. 6. Editorial Televisión, S.A. de C.V. 1992.

[6] MSDN Library Visual Studio, Componentes ActiveX, Microsoft 2001.

[7] NIST (National Institute of Standards and Technology), http://www.nist.gov.

[8] MSDN Library Visual Studio, Características de los componentes ActiveX de Visual Basic, Microsoft 2001.

[9] Stallings, William. "Cryptography and Network Security: principles and practice". Editorial Prentice Hall. 2ª Edición. 1998.

[10] Borghello, Cristian. Tesis "Seguridad Informática: sus implicancias e implementación". 2001. http://www.cfbsoft.com.ar

[11] Pons, Martorell Manuel. Control de Accesos. Departamento de Telecomunicaciones, Escuela Politécnica Universitaria de Mataró. 2000.

[12] Angel Angel, José de Jesús. "Criptografía para principiantes". Seguridata. jesus@seguridata.com. 2000.

[13] Pardo Seco, Fernando. Códigos y Criptografía. Apuntes sobre criptografía. 2001.

[14] Pons, Martorell Manuel. Criptografía. Departamento de Telecomunicaciones, Escuela Politécnica Universitaria de Mataró. 2000.

[15] MSDN Library Visual Studio, ActiveX, Microsoft 2001.

[16] Schneier, Bruce, Kelsey John, Doug Whitingz, David Wagner. Twofish: A 128-Bit Block Cipher. 1998. http://www.counterpane.com/twofish

[17] Counterpane Internet Security, Inc., Labs. 2002. http://www.counterpane.com/twofish/sourcecode.htm

[18] Kehner, Javier. Candados, cortafríos y otras herramientas. kehner_javier@bigfoot.com. http://get.to/neuralabyss.software

[19] Microsoft Corporation 2002. Authenticode, Microsoft Security Advisor, http://www.microsoft.com/security

Apéndice A

Código del Algoritmo Twofish

```
‘ --------   Definición de la longitud de llaves, ya que Twofish permite longitudes
‘ --------  desde los 64 bits, hasta los 256

Public Enum KeyLengths
   b256 = 256
   b196 = 196
   b128 = 128
   b64 = 64
End Enum

‘ -----  Declaración de variables   ----------
‘La encriptación de los datos transcurre en un bucle de 16 iteraciones
Private Const BLOCK_SIZE = 16
‘ --------  Aquí se define el número de rounds, que en las versiones anteriores del algoritmo,
‘ --------  esta variable no ha cambiado.
Private Const ROUNDS = 16
Private Const MAX_ROUNDS = 16

Private Const INPUT_WHITEN = 0
Private Const OUTPUT_WHITEN = INPUT_WHITEN + BLOCK_SIZE / 4
Private Const ROUND_SUBKEYS = OUTPUT_WHITEN + BLOCK_SIZE / 4
Private Const TOTAL_SUBKEYS = ROUND_SUBKEYS + 2 * MAX_ROUNDS

Private Const SK_STEP = &H2020202
Private Const SK_BUMP = &H1010101
Private Const SK_ROTL = 9
‘ ---------- Fin de declaración de variables generales ------
‘ ---------- Tabla P la cual consiste de 34 subclaves de 64 bits
Private Const P_00 = 1
Private Const P_01 = 0
Private Const P_02 = 0
Private Const P_03 = P_01 Xor 1
Private Const P_04 = 1

Private Const P_10 = 0
Private Const P_11 = 0
Private Const P_12 = 1
Private Const P_13 = P_11 Xor 1
Private Const P_14 = 0

Private Const P_20 = 1
Private Const P_21 = 1
```

```
Private Const P_22 = 0
Private Const P_23 = P_21 Xor 1
Private Const P_24 = 0

Private Const P_30 = 0
Private Const P_31 = 1
Private Const P_32 = 1
Private Const P_33 = P_31 Xor 1
Private Const P_34 = 1

Private Const GF256_FDBK = &H169
Private Const GF256_FDBK_2 = &H169 / 2
Private Const GF256_FDBK_4 = &H169 / 4

Private Const RS_GF_FDBK = &H14D

Private MDS(3, 255) As Long
Private P(1, 255) As Byte
‘ ------------  Define las cajas de seguridad y prepara las llaves con las cuales
‘ ------------   se realizarán las interaciones.
Private Type SessionKeyDef
   sBox() As Long
   subKeys() As Long
End Type

Private tSessionKey As SessionKeyDef

‘ ------------- Las funciones LFSR1 y LFSR2 se utilizan para agrupar
‘-------------- las operaciones e iteracciones del algoritmo al realizar la función de encriptación F
Private Function LFSR1(ByRef x As Long) As Long

   On Error GoTo ErrorHandle

   LFSR1 = lBSR(x, 1) Xor ((x And &H1) * GF256_FDBK_2)
Exit Function

ErrorHandle:
End Function


Private Function LFSR2(ByRef x As Long) As Long

   On Error GoTo ErrorHandle

   LFSR2 = lBSR(x, 2) Xor ((x And &H2) / &H2 * GF256_FDBK_2) Xor ((x And &H1) * GF256_FDBK_4)
Exit Function

ErrorHandle:
End Function

‘  ------  Las funciones MX_1, MX_X  y MX_Y realizan la preparación
‘ -------  de la palabra x en cada una de las 16 interacciones.
Private Function Mx_1(ByRef x As Long) As Long

   On Error GoTo ErrorHandle
```

```
    Mx_1 = x
Exit Function

ErrorHandle:
End Function


Private Function Mx_X(ByRef x As Long) As Long

    On Error GoTo ErrorHandle

    Mx_X = x Xor LFSR2(x)
Exit Function

ErrorHandle:
End Function


Private Function Mx_Y(ByRef x As Long) As Long

    On Error GoTo ErrorHandle

    Mx_Y = x Xor LFSR1(x) Xor LFSR2(x)
Exit Function

ErrorHandle:
End Function


Private Function b0(ByRef x As Long) As Byte

    On Error GoTo ErrorHandle

    b0 = x And &HFF
Exit Function

ErrorHandle:
End Function

‘  -----------   Las funciones b1…b3 realizan parte de la preparación de la tabla P
‘  -----------   y forman parte de las cajas de seguridad
Private Function b1(ByRef x As Long) As Long

    On Error GoTo ErrorHandle

    b1 = ((x And &HFFFFFF00) \ &H100) And &HFF
Exit Function

ErrorHandle:
End Function


Private Function b2(ByRef x As Long) As Long

    On Error GoTo ErrorHandle
```

```
    b2 = ((x And &HFFFF0000) \ &H10000) And &HFF
Exit Function

ErrorHandle:
End Function


Private Function b3(ByRef x As Long) As Long

    On Error GoTo ErrorHandle

    b3 = ((x And &HFF000000) \ &H1000000) And &HFF
Exit Function

ErrorHandle:
End Function


Private Function RS_MDS_Encode(ByRef k0 As Long, ByRef k1 As Long) As Long
    Dim i As Long

    On Error GoTo ErrorHandle

    RS_MDS_Encode = k1
    For i = 0 To 3
        RS_MDS_Encode = RS_Rem(RS_MDS_Encode)
    Next
    RS_MDS_Encode = (RS_MDS_Encode Xor k0)
    For i = 0 To 3
        RS_MDS_Encode = RS_Rem(RS_MDS_Encode)
    Next
Exit Function

ErrorHandle:
End Function


Private Function RS_Rem(ByRef x As Long) As Long
    Dim b As Long
    Dim g2 As Long
    Dim g3 As Long

    On Error GoTo ErrorHandle

    b = (lBSRU(x, 24) And &HFF)
    g2 = ((lBSL(b, 1) Xor (b And &H80) / &H80 * RS_GF_FDBK) And &HFF)
    g3 = (lBSRU(b, 1) Xor ((b And &H1) * lBSRU(RS_GF_FDBK, 1)) Xor g2)
    RS_Rem = lBSL(x, 8) Xor lBSL(g3, 24) Xor lBSL(g2, 16) Xor lBSL(g3, 8) Xor b
Exit Function

ErrorHandle:
End Function

‘   ----------  En esta función es donde se realiza la mayor cantidad del encriptamiento
‘  ----------  pues aqui es donde se divide a x en dos partes para poder mandar a llamar
‘ ----------  a la función F, en este caso F32 por tratarse de bloques de 32 bits en cada palabra preparada.
```

```
‘ ------------ No confundir con la longirud de bloques de encriptamiento,
‘ ------------ pues en esta función ya se prepararon las 4 cajas de seguridad.
Private Function F32(ByRef k64Cnt As Long, ByRef x As Long, ByRef k32() As Long) As Long
   Dim b(3) As Byte
   Dim k0 As Long
   Dim k1 As Long
   Dim k2 As Long
   Dim k3 As Long
   Dim exception As Boolean

   On Error GoTo ErrorHandle

   RtlMoveMemory VarPtr(b(0)), VarPtr(x), 4
   k0 = k32(0)
   k1 = k32(1)
   k2 = k32(2)
   k3 = k32(3)

   F32 = 0
   exception = False
   If (k64Cnt And 3) = 1 Or exception = True Then
      F32 = MDS(0, (P(P_01, b(0)) And &HFF) Xor b0(k0)) Xor _
          MDS(1, (P(P_11, b(1)) And &HFF) Xor b1(k0)) Xor _
          MDS(2, (P(P_21, b(2)) And &HFF) Xor b2(k0)) Xor _
          MDS(3, (P(P_31, b(3)) And &HFF) Xor b3(k0))
      exception = False
   End If
   If (k64Cnt And 3) = 0 Or exception = True Then
      b(0) = (P(P_04, b(0)) And &HFF) Xor b0(k3)
      b(1) = (P(P_14, b(1)) And &HFF) Xor b1(k3)
      b(2) = (P(P_24, b(2)) And &HFF) Xor b2(k3)
      b(3) = (P(P_34, b(3)) And &HFF) Xor b3(k3)
      exception = True
   End If
   If (k64Cnt And 3) = 3 Or exception = True Then
      b(0) = (P(P_03, b(0)) And &HFF) Xor b0(k2)
      b(1) = (P(P_13, b(1)) And &HFF) Xor b1(k2)
      b(2) = (P(P_23, b(2)) And &HFF) Xor b2(k2)
      b(3) = (P(P_33, b(3)) And &HFF) Xor b3(k2)
      exception = True
   End If
   If (k64Cnt And 3) = 2 Or exception = True Then
      F32 = MDS(0, (P(P_01, (P(P_02, b(0)) And &HFF) Xor b0(k1)) And &HFF) Xor b0(k0)) Xor _
          MDS(1, (P(P_11, (P(P_12, b(1)) And &HFF) Xor b1(k1)) And &HFF) Xor b1(k0)) Xor _
          MDS(2, (P(P_21, (P(P_22, b(2)) And &HFF) Xor b2(k1)) And &HFF) Xor b2(k0)) Xor _
          MDS(3, (P(P_31, (P(P_32, b(3)) And &HFF) Xor b3(k1)) And &HFF) Xor b3(k0))
      exception = False
   End If
Exit Function

ErrorHandle:
End Function


Private Function Fe32(ByRef sBox() As Long, ByRef x As Long, ByRef R As Long) As Long
```

```
    On Error GoTo ErrorHandle

    Fe32 = sBox(2 * x_b(x, R)) Xor _
          sBox(2 * x_b(x, R + 1) + 1) Xor _
          sBox(&H200 + 2 * x_b(x, R + 2)) Xor _
          sBox(&H200 + 2 * x_b(x, R + 3) + 1)
Exit Function

ErrorHandle:
End Function


Private Function x_b(ByRef x As Long, ByRef N As Long) As Long

    On Error GoTo ErrorHandle

    x_b = 0
    Select Case (N Mod 4)
       Case 0
          x_b = b0(x)
       Case 1
          x_b = b1(x)
       Case 2
          x_b = b2(x)
       Case 3
          x_b = b3(x)
    End Select
Exit Function

ErrorHandle:
End Function

‘---------  Por medio de esta función se preparan las llaves que se utilizarán en las interacciones. ----------
Private Function makeKey(ByRef k() As Byte) As SessionKeyDef
    Dim k64Cnt As Long
    Dim subkeyCnt As Long
    Dim k32e(3) As Long
    Dim k32o(3) As Long
    Dim sBoxKey(3) As Long
    Dim length As Long
    Dim i As Long
    Dim j As Long
    Dim offset As Long
    Dim lA As Long
    Dim lB As Long
    Dim subKeys() As Long
    Dim b(3) As Byte
    Dim k0 As Long
    Dim k1 As Long
    Dim k2 As Long
    Dim k3 As Long
    Dim sBox(1023) As Long
    Dim exception As Boolean

    On Error GoTo ErrorHandle
```

```
length = UBound(k) + 1
k64Cnt = length / 8
subkeyCnt = ROUND_SUBKEYS + 2 * ROUNDS

i = 0
j = k64Cnt - 1
offset = 0
While i < 4 And offset < length
   RtlMoveMemory VarPtr(k32e(i)), VarPtr(k(offset)), 4
   offset = offset + 4
   RtlMoveMemory VarPtr(k32o(i)), VarPtr(k(offset)), 4
   offset = offset + 4
   sBoxKey(j) = RS_MDS_Encode(k32e(i), k32o(i))
   i = i + 1
   j = j - 1
Wend

ReDim subKeys(subkeyCnt)
For i = 0 To (subkeyCnt / 2) - 1
   lA = F32(k64Cnt, i * SK_STEP, k32e)
   lB = F32(k64Cnt, i * SK_STEP + SK_BUMP, k32o)
   lB = lBSL(lB, 8) Or lBSRU(lB, 24)
   lA = (lA + lB)
   subKeys(2 * i) = lA
   lA = (lA + lB)
   subKeys(2 * i + 1) = lBSL(lA, SK_ROTL) Or lBSRU(lA, 32 - SK_ROTL)
Next

k0 = sBoxKey(0)
k1 = sBoxKey(1)
k2 = sBoxKey(2)
k3 = sBoxKey(3)
For i = 0 To 255
   b(0) = i
   b(1) = i
   b(2) = i
   b(3) = i
   exception = False
   If (k64Cnt And 3) = 1 Or exception = True Then
      sBox(2 * i) = MDS(0, (P(P_01, b(0)) And &HFF) Xor b0(k0))
      sBox(2 * i + 1) = MDS(1, (P(P_11, b(1)) And &HFF) Xor b1(k0))
      sBox(&H200 + 2 * i) = MDS(2, (P(P_21, b(2)) And &HFF) Xor b2(k0))
      sBox(&H200 + 2 * i + 1) = MDS(3, (P(P_31, b(3)) And &HFF) Xor b3(k0))
      exception = False
   End If
   If (k64Cnt And 3) = 0 Or exception = True Then
      b(0) = (P(P_04, b(0)) And &HFF) Xor b0(k3)
      b(1) = (P(P_14, b(1)) And &HFF) Xor b1(k3)
      b(2) = (P(P_24, b(2)) And &HFF) Xor b2(k3)
      b(3) = (P(P_34, b(3)) And &HFF) Xor b3(k3)
      exception = True
   End If
   If (k64Cnt And 3) = 3 Or exception = True Then
      b(0) = (P(P_03, b(0)) And &HFF) Xor b0(k2)
      b(1) = (P(P_13, b(1)) And &HFF) Xor b1(k2)
      b(2) = (P(P_23, b(2)) And &HFF) Xor b2(k2)
```

```
            b(3) = (P(P_33, b(3)) And &HFF) Xor b3(k2)
            exception = True
        End If
        If (k64Cnt And 3) = 2 Or exception = True Then
            sBox(2 * i) = MDS(0, (P(P_01, (P(P_02, b(0)) And &HFF) Xor b0(k1)) And &HFF) Xor b0(k0))
            sBox(2 * i + 1) = MDS(1, (P(P_11, (P(P_12, b(1)) And &HFF) Xor b1(k1)) And &HFF) Xor b1(k0))
            sBox(&H200 + 2 * i) = MDS(2, (P(P_21, (P(P_22, b(2)) And &HFF) Xor b2(k1)) And &HFF) Xor
b2(k0))
            sBox(&H200 + 2 * i + 1) = MDS(3, (P(P_31, (P(P_32, b(3)) And &HFF) Xor b3(k1)) And &HFF)
Xor b3(k0))
            exception = False
        End If
    Next

    makeKey.sBox = sBox
    makeKey.subKeys = subKeys
Exit Function

ErrorHandle:
End Function

‘ ----------  Función donde se realiza la encriptación de los datos, esta función es la que se invoca.
Private Function blockEncrypt(ByRef bInput() As Byte, ByRef inOffset As Long, ByRef sessionKey As
SessionKeyDef) As Byte()
    Dim bOutput() As Byte
    Dim sBox() As Long
    Dim sKey() As Long
    Dim x(3) As Long
    Dim t0 As Long
    Dim t1 As Long
    Dim k As Long
    Dim R As Long

    On Error GoTo ErrorHandle

    sBox = sessionKey.sBox
    sKey = sessionKey.subKeys

    RtlMoveMemory VarPtr(x(0)), VarPtr(bInput(inOffset)), 16

    x(0) = x(0) Xor sKey(INPUT_WHITEN)
    x(1) = x(1) Xor sKey(INPUT_WHITEN + 1)
    x(2) = x(2) Xor sKey(INPUT_WHITEN + 2)
    x(3) = x(3) Xor sKey(INPUT_WHITEN + 3)

    k = ROUND_SUBKEYS
    For R = 0 To ROUNDS - 1 Step 2
        t0 = Fe32(sBox, x(0), 0)
        t1 = Fe32(sBox, x(1), 3)
        x(2) = lBSRRot(x(2) Xor (t0 + t1 + sKey(k)), 1)
        k = k + 1
        x(3) = (lBSRRot(x(3), 31)) Xor (t0 + t1 + t1 + sKey(k))
        k = k + 1

        t0 = Fe32(sBox, x(2), 0)
        t1 = Fe32(sBox, x(3), 3)
```

```
        x(0) = lBSRRot(x(0) Xor (t0 + t1 + sKey(k)), 1)
        k = k + 1
        x(1) = (lBSRRot(x(1), 31)) Xor (t0 + t1 + t1 + sKey(k))
        k = k + 1
    Next

    x(2) = x(2) Xor sKey(OUTPUT_WHITEN)
    x(3) = x(3) Xor sKey(OUTPUT_WHITEN + 1)
    x(0) = x(0) Xor sKey(OUTPUT_WHITEN + 2)
    x(1) = x(1) Xor sKey(OUTPUT_WHITEN + 3)

    ReDim bOutput(15)
    RtlMoveMemory VarPtr(bOutput(0)), VarPtr(x(2)), 8
    RtlMoveMemory VarPtr(bOutput(8)), VarPtr(x(0)), 8
    blockEncrypt = bOutput
Exit Function

ErrorHandle:
End Function

‘ ------  Función que realiza la desencriptación por bloques dado por la variable de paso por referencia
‘ ------  bInput. A esta función es a la que se invoca finalmente
Private Function blockDecrypt(ByRef bInput() As Byte, ByRef inOffset As Long, ByRef sessionKey As
SessionKeyDef) As Byte()
    Dim bOutput() As Byte
    Dim sBox() As Long
    Dim sKey() As Long
    Dim x(3) As Long
    Dim k As Long
    Dim t0 As Long
    Dim t1 As Long
    Dim R As Long

    On Error GoTo ErrorHandle

    sBox = sessionKey.sBox
    sKey = sessionKey.subKeys

    RtlMoveMemory VarPtr(x(2)), VarPtr(bInput(inOffset)), 8
    RtlMoveMemory VarPtr(x(0)), VarPtr(bInput(inOffset + 8)), 8

    x(2) = x(2) Xor sKey(OUTPUT_WHITEN)
    x(3) = x(3) Xor sKey(OUTPUT_WHITEN + 1)
    x(0) = x(0) Xor sKey(OUTPUT_WHITEN + 2)
    x(1) = x(1) Xor sKey(OUTPUT_WHITEN + 3)

    k = ROUND_SUBKEYS + 2 * ROUNDS - 1
    For R = 0 To ROUNDS - 1 Step 2
        t0 = Fe32(sBox, x(2), 0)
        t1 = Fe32(sBox, x(3), 3)
        x(1) = lBSRRot(x(1) Xor (t0 + t1 + t1 + sKey(k)), 1)
        k = k - 1
        x(0) = lBSRRot(x(0), 31) Xor (t0 + t1 + sKey(k))
        k = k - 1
        t0 = Fe32(sBox, x(0), 0)
        t1 = Fe32(sBox, x(1), 3)
```

```
        x(3) = lBSRRot(x(3) Xor (t0 + t1 + t1 + sKey(k)), 1)
        k = k - 1
        x(2) = lBSRRot(x(2), 31) Xor (t0 + t1 + sKey(k))
        k = k - 1
    Next

    x(0) = x(0) Xor sKey(INPUT_WHITEN)
    x(1) = x(1) Xor sKey(INPUT_WHITEN + 1)
    x(2) = x(2) Xor sKey(INPUT_WHITEN + 2)
    x(3) = x(3) Xor sKey(INPUT_WHITEN + 3)

    ReDim bOutput(15)
    RtlMoveMemory VarPtr(bOutput(0)), VarPtr(x(0)), 16
    blockDecrypt = bOutput
Exit Function

ErrorHandle:
End Function


Private Sub Class_Initialize()
    Dim m1(1) As Long
    Dim mX(1) As Long
    Dim mY(1) As Long
    Dim i As Long
    Dim j As Long
    Dim bP0() As Byte
    Dim bP1() As Byte

    On Error GoTo ErrorHandle
 'Cajas de seguridad de  32 x 8 = 256 bits, normalmente denominadas con la letra S. S1…S4.
    bP0 = CStr(ChrB(&HA9) & ChrB(&H67) & ChrB(&HB3) & ChrB(&HE8) & ChrB(&H4) &
ChrB(&HFD) & ChrB(&HA3) & ChrB(&H76))
    bP0 = CStr(bP0) & CStr(ChrB(&H9A) & ChrB(&H92) & ChrB(&H80) & ChrB(&H78) & ChrB(&HE4)
& ChrB(&HDD) & ChrB(&HD1) & ChrB(&H38))
    bP0 = CStr(bP0) & CStr(ChrB(&HD) & ChrB(&HC6) & ChrB(&H35) & ChrB(&H98) & ChrB(&H18) &
ChrB(&HF7) & ChrB(&HEC) & ChrB(&H6C))
    bP0 = CStr(bP0) & CStr(ChrB(&H43) & ChrB(&H75) & ChrB(&H37) & ChrB(&H26) & ChrB(&HFA)
& ChrB(&H13) & ChrB(&H94) & ChrB(&H48))
    bP0 = CStr(bP0) & CStr(ChrB(&HF2) & ChrB(&HD0) & ChrB(&H8B) & ChrB(&H30) & ChrB(&H84)
& ChrB(&H54) & ChrB(&HDF) & ChrB(&H23))
    bP0 = CStr(bP0) & CStr(ChrB(&H19) & ChrB(&H5B) & ChrB(&H3D) & ChrB(&H59) & ChrB(&HF3)
& ChrB(&HAE) & ChrB(&HA2) & ChrB(&H82))
    bP0 = CStr(bP0) & CStr(ChrB(&H63) & ChrB(&H1) & ChrB(&H83) & ChrB(&H2E) & ChrB(&HD9) &
ChrB(&H51) & ChrB(&H9B) & ChrB(&H7C))
    bP0 = CStr(bP0) & CStr(ChrB(&HA6) & ChrB(&HEB) & ChrB(&HA5) & ChrB(&HBE) & ChrB(&H16)
& ChrB(&HC) & ChrB(&HE3) & ChrB(&H61))
    bP0 = CStr(bP0) & CStr(ChrB(&HC0) & ChrB(&H8C) & ChrB(&H3A) & ChrB(&HF5) & ChrB(&H73)
& ChrB(&H2C) & ChrB(&H25) & ChrB(&HB))
    bP0 = CStr(bP0) & CStr(ChrB(&HBB) & ChrB(&H4E) & ChrB(&H89) & ChrB(&H6B) & ChrB(&H53)
& ChrB(&H6A) & ChrB(&HB4) & ChrB(&HF1))
    bP0 = CStr(bP0) & CStr(ChrB(&HE1) & ChrB(&HE6) & ChrB(&HBD) & ChrB(&H45) & ChrB(&HE2)
& ChrB(&HF4) & ChrB(&HB6) & ChrB(&H66))
    bP0 = CStr(bP0) & CStr(ChrB(&HCC) & ChrB(&H95) & ChrB(&H3) & ChrB(&H56) & ChrB(&HD4) &
ChrB(&H1C) & ChrB(&H1E) & ChrB(&HD7))
    bP0 = CStr(bP0) & CStr(ChrB(&HFB) & ChrB(&HC3) & ChrB(&H8E) & ChrB(&HB5) & ChrB(&HE9)
```

```
& ChrB(&HCF) & ChrB(&HBF) & ChrB(&HBA))
   bP0 = CStr(bP0) & CStr(ChrB(&HEA) & ChrB(&H77) & ChrB(&H39) & ChrB(&HAF) & ChrB(&H33)
& ChrB(&HC9) & ChrB(&H62) & ChrB(&H71))
   bP0 = CStr(bP0) & CStr(ChrB(&H81) & ChrB(&H79) & ChrB(&H9) & ChrB(&HAD) & ChrB(&H24) &
ChrB(&HCD) & ChrB(&HF9) & ChrB(&HD8))
   bP0 = CStr(bP0) & CStr(ChrB(&HE5) & ChrB(&HC5) & ChrB(&HB9) & ChrB(&H4D) & ChrB(&H44)
& ChrB(&H8) & ChrB(&H86) & ChrB(&HE7))
   bP0 = CStr(bP0) & CStr(ChrB(&HA1) & ChrB(&H1D) & ChrB(&HAA) & ChrB(&HED) & ChrB(&H6)
& ChrB(&H70) & ChrB(&HB2) & ChrB(&HD2))
   bP0 = CStr(bP0) & CStr(ChrB(&H41) & ChrB(&H7B) & ChrB(&HA0) & ChrB(&H11) & ChrB(&H31)
& ChrB(&HC2) & ChrB(&H27) & ChrB(&H90))
   bP0 = CStr(bP0) & CStr(ChrB(&H20) & ChrB(&HF6) & ChrB(&H60) & ChrB(&HFF) & ChrB(&H96) &
ChrB(&H5C) & ChrB(&HB1) & ChrB(&HAB))
   bP0 = CStr(bP0) & CStr(ChrB(&H9E) & ChrB(&H9C) & ChrB(&H52) & ChrB(&H1B) & ChrB(&H5F)
& ChrB(&H93) & ChrB(&HA) & ChrB(&HEF))
   bP0 = CStr(bP0) & CStr(ChrB(&H91) & ChrB(&H85) & ChrB(&H49) & ChrB(&HEE) & ChrB(&H2D)
& ChrB(&H4F) & ChrB(&H8F) & ChrB(&H3B))
   bP0 = CStr(bP0) & CStr(ChrB(&H47) & ChrB(&H87) & ChrB(&H6D) & ChrB(&H46) & ChrB(&HD6)
& ChrB(&H3E) & ChrB(&H69) & ChrB(&H64))
   bP0 = CStr(bP0) & CStr(ChrB(&H2A) & ChrB(&HCE) & ChrB(&HCB) & ChrB(&H2F) & ChrB(&HFC)
& ChrB(&H97) & ChrB(&H5) & ChrB(&H7A))
   bP0 = CStr(bP0) & CStr(ChrB(&HAC) & ChrB(&H7F) & ChrB(&HD5) & ChrB(&H1A) & ChrB(&H4B)
& ChrB(&HE) & ChrB(&HA7) & ChrB(&H5A))
   bP0 = CStr(bP0) & CStr(ChrB(&H28) & ChrB(&H14) & ChrB(&H3F) & ChrB(&H29) & ChrB(&H88) &
ChrB(&H3C) & ChrB(&H4C) & ChrB(&H2))
   bP0 = CStr(bP0) & CStr(ChrB(&HB8) & ChrB(&HDA) & ChrB(&HB0) & ChrB(&H17) & ChrB(&H55)
& ChrB(&H1F) & ChrB(&H8A) & ChrB(&H7D))
   bP0 = CStr(bP0) & CStr(ChrB(&H57) & ChrB(&HC7) & ChrB(&H8D) & ChrB(&H74) & ChrB(&HB7)
& ChrB(&HC4) & ChrB(&H9F) & ChrB(&H72))
   bP0 = CStr(bP0) & CStr(ChrB(&H7E) & ChrB(&H15) & ChrB(&H22) & ChrB(&H12) & ChrB(&H58) &
ChrB(&H7) & ChrB(&H99) & ChrB(&H34))
   bP0 = CStr(bP0) & CStr(ChrB(&H6E) & ChrB(&H50) & ChrB(&HDE) & ChrB(&H68) & ChrB(&H65)
& ChrB(&HBC) & ChrB(&HDB) & ChrB(&HF8))
   bP0 = CStr(bP0) & CStr(ChrB(&HC8) & ChrB(&HA8) & ChrB(&H2B) & ChrB(&H40) & ChrB(&HDC)
& ChrB(&HFE) & ChrB(&H32) & ChrB(&HA4))
   bP0 = CStr(bP0) & CStr(ChrB(&HCA) & ChrB(&H10) & ChrB(&H21) & ChrB(&HF0) & ChrB(&HD3)
& ChrB(&H5D) & ChrB(&HF) & ChrB(&H0))
   bP0 = CStr(bP0) & CStr(ChrB(&H6F) & ChrB(&H9D) & ChrB(&H36) & ChrB(&H42) & ChrB(&H4A)
& ChrB(&H5E) & ChrB(&HC1) & ChrB(&HE0))

   bP1 = CStr(ChrB(&H75) & ChrB(&HF3) & ChrB(&HC6) & ChrB(&HF4) & ChrB(&HDB) &
ChrB(&H7B) & ChrB(&HFB) & ChrB(&HC8))
   bP1 = CStr(bP1) & CStr(ChrB(&H4A) & ChrB(&HD3) & ChrB(&HE6) & ChrB(&H6B) & ChrB(&H45)
& ChrB(&H7D) & ChrB(&HE8) & ChrB(&H4B))
   bP1 = CStr(bP1) & CStr(ChrB(&HD6) & ChrB(&H32) & ChrB(&HD8) & ChrB(&HFD) & ChrB(&H37)
& ChrB(&H71) & ChrB(&HF1) & ChrB(&HE1))
   bP1 = CStr(bP1) & CStr(ChrB(&H30) & ChrB(&HF) & ChrB(&HF8) & ChrB(&H1B) & ChrB(&H87) &
ChrB(&HFA) & ChrB(&H6) & ChrB(&H3F))
   bP1 = CStr(bP1) & CStr(ChrB(&H5E) & ChrB(&HBA) & ChrB(&HAE) & ChrB(&H5B) & ChrB(&H8A)
& ChrB(&H0) & ChrB(&HBC) & ChrB(&H9D))
   bP1 = CStr(bP1) & CStr(ChrB(&H6D) & ChrB(&HC1) & ChrB(&HB1) & ChrB(&HE) & ChrB(&H80) &
ChrB(&H5D) & ChrB(&HD2) & ChrB(&HD5))
   bP1 = CStr(bP1) & CStr(ChrB(&HA0) & ChrB(&H84) & ChrB(&H7) & ChrB(&H14) & ChrB(&HB5) &
ChrB(&H90) & ChrB(&H2C) & ChrB(&HA3))
   bP1 = CStr(bP1) & CStr(ChrB(&HB2) & ChrB(&H73) & ChrB(&H4C) & ChrB(&H54) & ChrB(&H92)
& ChrB(&H74) & ChrB(&H36) & ChrB(&H51))
```

```
   bP1 = CStr(bP1) & CStr(ChrB(&H38) & ChrB(&HB0) & ChrB(&HBD) & ChrB(&H5A) & ChrB(&HFC)
& ChrB(&H60) & ChrB(&H62) & ChrB(&H96))
   bP1 = CStr(bP1) & CStr(ChrB(&H6C) & ChrB(&H42) & ChrB(&HF7) & ChrB(&H10) & ChrB(&H7C)
& ChrB(&H28) & ChrB(&H27) & ChrB(&H8C))
   bP1 = CStr(bP1) & CStr(ChrB(&H13) & ChrB(&H95) & ChrB(&H9C) & ChrB(&HC7) & ChrB(&H24)
& ChrB(&H46) & ChrB(&H3B) & ChrB(&H70))
   bP1 = CStr(bP1) & CStr(ChrB(&HCA) & ChrB(&HE3) & ChrB(&H85) & ChrB(&HCB) & ChrB(&H11)
& ChrB(&HD0) & ChrB(&H93) & ChrB(&HB8))
   bP1 = CStr(bP1) & CStr(ChrB(&HA6) & ChrB(&H83) & ChrB(&H20) & ChrB(&HFF) & ChrB(&H9F)
& ChrB(&H77) & ChrB(&HC3) & ChrB(&HCC))
   bP1 = CStr(bP1) & CStr(ChrB(&H3) & ChrB(&H6F) & ChrB(&H8) & ChrB(&HBF) & ChrB(&H40) &
ChrB(&HE7) & ChrB(&H2B) & ChrB(&HE2))
   bP1 = CStr(bP1) & CStr(ChrB(&H79) & ChrB(&HC) & ChrB(&HAA) & ChrB(&H82) & ChrB(&H41) &
ChrB(&H3A) & ChrB(&HEA) & ChrB(&HB9))
   bP1 = CStr(bP1) & CStr(ChrB(&HE4) & ChrB(&H9A) & ChrB(&HA4) & ChrB(&H97) & ChrB(&H7E)
& ChrB(&HDA) & ChrB(&H7A) & ChrB(&H17))
   bP1 = CStr(bP1) & CStr(ChrB(&H66) & ChrB(&H94) & ChrB(&HA1) & ChrB(&H1D) & ChrB(&H3D)
& ChrB(&HF0) & ChrB(&HDE) & ChrB(&HB3))
   bP1 = CStr(bP1) & CStr(ChrB(&HB) & ChrB(&H72) & ChrB(&HA7) & ChrB(&H1C) & ChrB(&HEF) &
ChrB(&HD1) & ChrB(&H53) & ChrB(&H3E))
   bP1 = CStr(bP1) & CStr(ChrB(&H8F) & ChrB(&H33) & ChrB(&H26) & ChrB(&H5F) & ChrB(&HEC)
& ChrB(&H76) & ChrB(&H2A) & ChrB(&H49))
   bP1 = CStr(bP1) & CStr(ChrB(&H81) & ChrB(&H88) & ChrB(&HEE) & ChrB(&H21) & ChrB(&HC4)
& ChrB(&H1A) & ChrB(&HEB) & ChrB(&HD9))
   bP1 = CStr(bP1) & CStr(ChrB(&HC5) & ChrB(&H39) & ChrB(&H99) & ChrB(&HCD) & ChrB(&HAD)
& ChrB(&H31) & ChrB(&H8B) & ChrB(&H1))
   bP1 = CStr(bP1) & CStr(ChrB(&H18) & ChrB(&H23) & ChrB(&HDD) & ChrB(&H1F) & ChrB(&H4E)
& ChrB(&H2D) & ChrB(&HF9) & ChrB(&H48))
   bP1 = CStr(bP1) & CStr(ChrB(&H4F) & ChrB(&HF2) & ChrB(&H65) & ChrB(&H8E) & ChrB(&H78) &
ChrB(&H5C) & ChrB(&H58) & ChrB(&H19))
   bP1 = CStr(bP1) & CStr(ChrB(&H8D) & ChrB(&HE5) & ChrB(&H98) & ChrB(&H57) & ChrB(&H67)
& ChrB(&H7F) & ChrB(&H5) & ChrB(&H64))
   bP1 = CStr(bP1) & CStr(ChrB(&HAF) & ChrB(&H63) & ChrB(&HB6) & ChrB(&HFE) & ChrB(&HF5)
& ChrB(&HB7) & ChrB(&H3C) & ChrB(&HA5))
   bP1 = CStr(bP1) & CStr(ChrB(&HCE) & ChrB(&HE9) & ChrB(&H68) & ChrB(&H44) & ChrB(&HE0)
& ChrB(&H4D) & ChrB(&H43) & ChrB(&H69))
   bP1 = CStr(bP1) & CStr(ChrB(&H29) & ChrB(&H2E) & ChrB(&HAC) & ChrB(&H15) & ChrB(&H59)
& ChrB(&HA8) & ChrB(&HA) & ChrB(&H9E))
   bP1 = CStr(bP1) & CStr(ChrB(&H6E) & ChrB(&H47) & ChrB(&HDF) & ChrB(&H34) & ChrB(&H35)
& ChrB(&H6A) & ChrB(&HCF) & ChrB(&HDC))
   bP1 = CStr(bP1) & CStr(ChrB(&H22) & ChrB(&HC9) & ChrB(&HC0) & ChrB(&H9B) & ChrB(&H89)
& ChrB(&HD4) & ChrB(&HED) & ChrB(&HAB))
   bP1 = CStr(bP1) & CStr(ChrB(&H12) & ChrB(&HA2) & ChrB(&HD) & ChrB(&H52) & ChrB(&HBB)
& ChrB(&H2) & ChrB(&H2F) & ChrB(&HA9))
   bP1 = CStr(bP1) & CStr(ChrB(&HD7) & ChrB(&H61) & ChrB(&H1E) & ChrB(&HB4) & ChrB(&H50)
& ChrB(&H4) & ChrB(&HF6) & ChrB(&HC2))
   bP1 = CStr(bP1) & CStr(ChrB(&H16) & ChrB(&H25) & ChrB(&H86) & ChrB(&H56) & ChrB(&H55) &
ChrB(&H9) & ChrB(&HBE) & ChrB(&H91))
   ‘ ---------- parte de la preparación de la función F en donde se recombina la palabra x.
   For i = 0 To 255
      P(0, i) = bP0(i)
      P(1, i) = bP1(i)
   Next

   For i = 0 To 255
      j = (P(0, i) And &HFF)
```

```
        m1(0) = j
        mX(0) = (Mx_X(j) And &HFF)
        mY(0) = (Mx_Y(j) And &HFF)

        j = (P(1, i) And &HFF)
        m1(1) = j
        mX(1) = (Mx_X(j) And &HFF)
        mY(1) = (Mx_Y(j) And &HFF)

        MDS(0, i) = (m1(P_00) Or lBSL(mX(P_00), 8) Or lBSL(mY(P_00), 16) Or lBSL(mY(P_00), 24))
        MDS(1, i) = (mY(P_10) Or lBSL(mY(P_10), 8) Or lBSL(mX(P_10), 16) Or lBSL(m1(P_10), 24))
        MDS(2, i) = (mX(P_20) Or lBSL(mY(P_20), 8) Or lBSL(m1(P_20), 16) Or lBSL(mY(P_20), 24))
        MDS(3, i) = (mX(P_30) Or lBSL(m1(P_30), 8) Or lBSL(mY(P_30), 16) Or lBSL(mX(P_30), 24))
    Next
Exit Sub

ErrorHandle:
End Sub


Public Property Let bKey(Optional ByVal lMinKeyLength As KeyLengths, ByRef bKey() As Byte)
    Dim lKeyLength As Long

    On Error GoTo ErrorHandle

    If boolIsArrayInit(bKey) = False Then Exit Property
    lKeyLength = (UBound(bKey) + 1) * 8
    If lKeyLength < lMinKeyLength Then ReDim Preserve bKey((lMinKeyLength \ 8) - 1)
    If lKeyLength > 256 Then ReDim Preserve bKey(31)
    If lKeyLength > 192 And lKeyLength < 256 Then ReDim Preserve bKey(23)
    If lKeyLength > 128 And lKeyLength < 192 Then ReDim Preserve bKey(15)
    If lKeyLength > 64 And lKeyLength < 128 Then ReDim Preserve bKey(7)
    tSessionKey = makeKey(bKey)
Exit Property

ErrorHandle:
End Property

‘ -------  Esta función permite la encriptación Byte a byte
Public Function bEncrypt(ByRef bInput() As Byte) As Byte()
    Dim bBlockInput() As Byte
    Dim bTemp() As Byte
    Dim lCount As Long
    Dim lEncodeLength As Long
    Dim lInputLength As Long
    Dim bOutput() As Byte

    On Error GoTo ErrorHandle

    If boolIsArrayInit(bInput) = False Then Exit Function
    If boolIsArrayInit(tSessionKey.sBox) = False Then
        ReDim bTemp(15)
        tSessionKey = makeKey(bTemp)
    End If
    lInputLength = UBound(bInput) + 1
    lEncodeLength = lInputLength + 4
```

```
    If lEncodeLength Mod 16 <> 0 Then lEncodeLength = lEncodeLength + 16 - (lEncodeLength Mod 16)
    ReDim bBlockInput(lEncodeLength - 1)
    ReDim bOutput(lEncodeLength - 1)
    RtlMoveMemory VarPtr(bBlockInput(0)), VarPtr(lInputLength), 4
    RtlMoveMemory VarPtr(bBlockInput(4)), VarPtr(bInput(0)), lInputLength
    For lCount = 0 To lEncodeLength - 1 Step 16
        bTemp = blockEncrypt(bBlockInput, lCount, tSessionKey)
        RtlMoveMemory VarPtr(bOutput(lCount)), VarPtr(bTemp(0)), 16
    Next
    bEncrypt = bOutput
Exit Function

ErrorHandle:
End Function

‘  ------------  Realiza la desencriptación byte a byte
Public Function bDecrypt(ByRef bInput() As Byte) As Byte()
    Dim lTemp As Long
    Dim bBlockOutput() As Byte
    Dim bOutput() As Byte
    Dim bTemp() As Byte
    Dim lCount As Long

    On Error GoTo ErrorHandle

    If boolIsArrayInit(bInput) = False Then Exit Function
    If (UBound(bInput) + 1) Mod 16 <> 0 And UBound(bInput) > 0 Then Exit Function
    If boolIsArrayInit(tSessionKey.sBox) = False Then
        ReDim bTemp(15)
        tSessionKey = makeKey(bTemp)
    End If
    ReDim bBlockOutput(UBound(bInput))
    For lCount = 0 To UBound(bInput) Step 16
        bTemp = blockDecrypt(bInput, lCount, tSessionKey)
        RtlMoveMemory VarPtr(bBlockOutput(lCount)), VarPtr(bTemp(0)), 16
    Next

    RtlMoveMemory VarPtr(lTemp), VarPtr(bBlockOutput(0)), 4
    If lTemp > UBound(bInput) - 3 Then Exit Function
    ReDim bOutput(lTemp - 1)
    RtlMoveMemory VarPtr(bOutput(0)), VarPtr(bBlockOutput(4)), lTemp
    bDecrypt = bOutput
Exit Function

ErrorHandle:
End Function


Private Function boolIsArrayInit(ByRef vArray As Variant) As Boolean

    On Error Resume Next

    boolIsArrayInit = IsNumeric(UBound(vArray))
End Function
```

Apéndice B

Manual del Programador

CONTENIDO

INTRODUCCIÓN

- Requisitos del sistema y conocimientos necesarios de programación.
- Instalación
- Cómo comenzar.

UTILIZANDO LOS CONTROLES ACTIVEX

- Incorporando los controles ActiveX al proyecto
- Cómo encriptar y desencriptar información.
- Dando mayor seguridad de encriptación.

UTILIZANDO LAS CARACTERÍSTICAS DE SEGURIDAD

- Implementando contraseñas en el desarrollo
- Combinando contraseñas y autenticación
- Implementando todas las características

INTRODUCCIÓN

El sistema de seguridad permite crear aplicaciones de forma rápida y con las características necesarias para resguardar la información de su empresa, negocio u organización.

Este sistema está creado para que usted como desarrollador, pueda implementar seguridad a través de contraseñas, encriptación y control de accesos, utilizando la plataforma de programación Visual Basic 5.0 en adelante.

REQUISITOS DEL SISTEMA Y CONOCIMIENTOS NECESARIOS DE PROGRAMACIÓN

Para poder utilizar esta solución, usted deberá tener conocimientos básicos de programación en MS Visual Basic. También deberá estar familiarizado con el manejo de bases de datos, insertar y eliminar formularios de un proyecto, inserción de controles ActiveX en formularios, creación de menús y creación de reportes.

Para poder utilizar el sistema de seguridad, su sistema deberá contar con:

· Computadora PC compatible con IBM. Microprocesador mínimo a 300 MHZ (recomendable de 1.2 Ghz. en adelante).

· 64 MB en RAM.

· Sistema Operativo Windows 95 o mayor.

· Visual Basic Ver. 5.0 en adelante.

· Espacio libre en disco duro de 30 MB.

· Unidad lectora de CD-ROM

INSTALACIÓN

Para instalar el sistema de seguridad, usted debe realizar los siguientes pasos:

1.- Introduzca el CD-ROM en su unidad lectora y espere unos segundos. El sistema de instalación se ejecutará automáticamente.

2.- En caso de que el sistema de instalación no se haya ejecutado, en la barra de inicio de Windows, pulse en *Inicio*, después en *Ejecutar*. Introduzca la instrucción

D:\SETUP.EXE en la línea de comando, donde *D* es la letra de tu unidad lectora de CD-ROM.

3.- Aparecerá una ventana semejante a la de abajo:

4.- Siga las instrucciones que le indicará el programa de instalación.

Al finalizar el proceso, se encontrará instalado en su computadora el programa de seguridad.

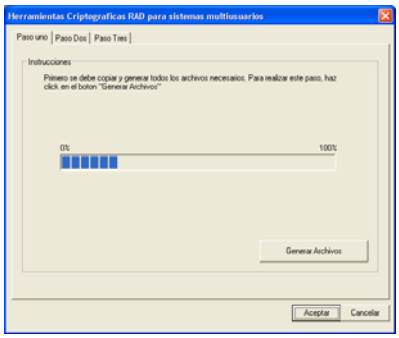
CÓMO COMENZAR.

Una vez instalado el sistema de seguridad, el siguiente paso es ejecutar el archivo: *Asistente.exe*. Este archivo lo podrá encontrar en la carpeta seleccionada al momento de ejecutar el programa de instalación. Otra forma de acceder a este archivo, es pulsando en *Inicio*, posteriormente en *Programas* y después en *Proyecto: Herramientas RAD Criptográficas*. Allí encontrará una opción con el nombre de *Asistente.exe*.

Después de haber ejecutado el Asistente, siga los pasos que se le indican en el programa.

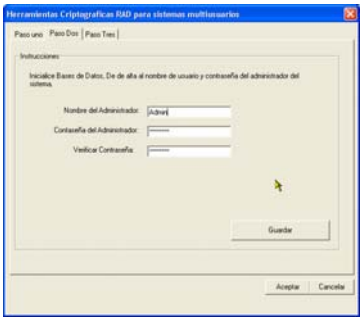
Primero se mostrará en la pantalla una ventana semejante a la siguiente:

Primer paso del asistente: generar todos los archivos necesarios.



Al presionar el botón *Generar Archivos*, se mostrará inmediatamente el segundo paso del Asistente:

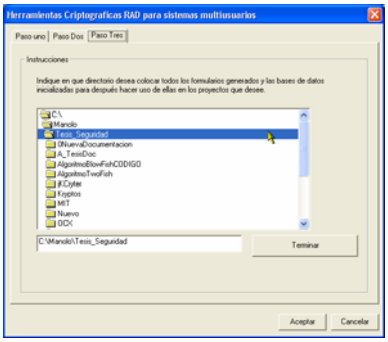
Segundo paso del asistente: inicialice bases de datos con la clave de acceso del administrador.



En este paso, se recomienda estar solo en la habitación, lo anterior con el fin de evitar que puedan estar viendo lo que escribe y de esta forma, saber la clave de acceso del administrador del sistema.

Al introducir el nombre del Administrador, elija un nombre fácil de recordar y que no sea obvio. Cuando introduzca la clave de acceso, cuide que ésta no sea menor a 10 caracteres por lo menos.

Tercer paso del asistente: seleccione el directorio destino de los archivos.



Por último, seleccione el directorio destino en donde quiera colocar los archivos que utilizará en sus desarrollos para implementar seguridad computacional.

UTILIZANDO CONTROLES ACTIVEX

Después de haber ejecutado el Asistente, ya puede utilizar todos los archivos generados por éste.

Entre estos archivos se encuentran los controles ActiveX, que específicamente se trata de dos archivos: Twofish.ocx y Entropía.ocx. Ambos le servirán para poder encriptar y desencriptar información.

INCORPORANDO LOS CONTROLES ACTIVEX AL PROYECTO

Los controles ActiveX los puede utilizar por separado o en conjunto, si es que desea dar mayor seguridad de encriptamiento al desarrollo en proceso.

Para utilizar cualquiera de los dos controles en su proyecto, realice los siguientes pasos:

Barra de Herramientas.

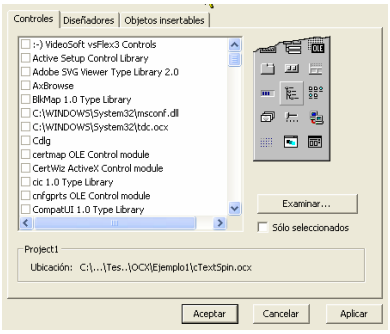
1.- Abra su proyecto de Visual Basic.

2.- Sitúese en la barra de herramientas

3.- Oprima el botón derecho del mouse.

4.- Seleccione la opción *componentes*

5.- En la carpeta de Componentes, seleccione a los dos controles .ocx: Twofish.ocx y Entropia.ocx.



6.- Oprima Aceptar.

7.- Aparecerán los controles en la barra de herramientas.

Ahora ya podrá arrastrar los controles para utilizarlos en su proyecto.

CÓMO ENCRIPTAR Y DESENCRIPTAR INFORMACIÓN

Cada control ActiveX tiene sus propias funciones. Si desea encriptar la información utilizando el algoritmo Twofish, deberá invocar las funciones del control Twofish.ocx. Las declaraciones y características de este control se muestran abajo:

```
       Control: Twofish.ocx
       Descripción: el control provee del algoritmo Twofish para la
encriptación de datos. Utiliza llave de 128 bits.
       Métodos:
Function Encriptar(EncrpCifra as String ) as String
Function Desencriptar(DescrpCifra as String ) as String

       Ejemplo:
       clveTwofish_b256 = "Mi_llave_de_256B" 'inicializa la llave de 256
bits
       Texto=Text1.text ' obtiene los datos del text1
       TextoCifrado = Twofish.Encriptar(Texto)       'Se guarda el texto
encriptado
       TextoDescifrado = Twofish.Desencriptar(TextoCifrado) 'Se desencripta
el texto
       Importante: Antes de encriptar el texto deseado, se debe de
inicializar la llave con la que se desee trabajar, dependiendo de la
longitud de ésta. Las variables a inicializar son:
           clveTwofish_b256 = 256
           clveTwofish_b196 = 196
           clveTwofish_b128 = 128
           clveTwofish_b64 = 64
```

Para encriptar la información todo lo que debe hacer es invocar las funciones adecuadas de cada control, dependiendo del tipo de encriptación que desee utilizar. Es importante tomar en cuenta que dependiendo de la longitud de la llave que utilizará, es la llave que inicializará antes de utilizar la función de encriptación.

Para el control Entropia.ocx, las declaraciones y métodos se muestran abajo:

```
       Control: entropia.ocx
       Descripción: el control provee de algoritmos propios de encriptación
utilizando el método de sustitución con polialfabetos y permutación.
       Métodos:

       Function Cifrar(cadena As Text, tipo As Integer)

       Function Descifrar(cadena As Text, tipo As Integer)

       Ejemplo:
       Texto= “Texto a encriptar”
       TextoCifrado = Entropia.Cifrar(Texto,1) ‘Se utilizará el método de
sustitución simple con polialfabetos.
       TextoCifrado = Entropia.Cifrar(Texto,2) ‘Se utilizará el método de
permutación.
       TextoCifrado = Entropia.Cifrar(Texto,0) ‘Se utilizarán los dos
métodos.
       TextoDescifrado = Desencriptar(TextoCifrado) ’Se desencripta el
texto
       Importante: Antes de encriptar el texto deseado, se debe de decidir
con qué método de encriptación se desea trabajar, teniendo las siguientes
opciones:
       tipo = 1 ‘ permite encriptación por el método de sustitución simple
con polialfabetos.
       tipo = 2 ‘ permite encriptación por el método de prmutación.
       tipo = 0 ‘ valor por default, permite encriptación por los dos
métodos anteriores.
```

DANDO MAYOR SEGURIDAD DE ENCRIPTACIÓN

Una forma de dar mayor seguridad a la información en el momento de encriptarse, es utilizando los dos controles ActiveX, cuidando siempre de utilizar el mismo orden al encriptar y desencriptar los datos, pero éste último en forma inversa.

Ejemplo:

```
       Texto= “Texto a encriptar”
       ‘ -- Inicia encriptación --
       clveTwofish_b256 = “Mi_llave_de_256B” ‘inicialización de la llave
para utilizar el algoritmo Twofish
       TxtCifradoTwfish = Twofish.Encriptar(Texto) ‘ Se encripta primero
con el algoritmo Twofish
       TextoCifradoEntropia = Entropia.Cifrar(TxtCifradoTwfish,0) ‘Se
utilizarán los dos métodos en el control entropia.ocx.
       ‘ -- Inicia desencriptación --
       TextoDescEntropia = Entropia.Descifrar(TextoCifradoEntropia,0)
‘Cuidar que se vuelva a utilizar el mismo método al desencriptarse .
       TxtDescifradoTwfish = Twofish.Descencriptar(TextoDescEntropia)
```

UTILIZANDO LAS CARACTERÍSTICAS DE SEGURIDAD

Después de saber cómo utilizar los controles ActiveX, ahora se pueden utilizar los formularios que contienen las características de seguridad por contraseñas, autenticación, OTP (One Time Password) y RBAC.

Cada formulario viene programado y prácticamente listo para que lo utilice e implemente. Los detalles que debe de cuidar a la hora de su implementación son:

1.- La ruta de acceso a las bases de datos.

2.- La vinculación entre los formularios al utilizar la instrucción “*show*”.

3.- Incluir los formularios necesarios.

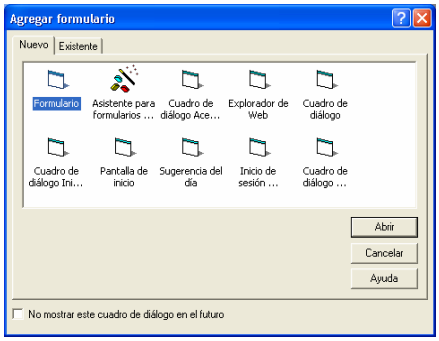
4.- Siempre incluir el archivo “*module1.bas*”, no importando si solo se utiliza un formulario. En este archivo se encuentran declaradas las variables globales.

IMPLEMENTANDO CONTRASEÑAS AL DESARROLLO

Para poder implementar el control de accesos mediante contraseñas, se debe incluir en el proyecto el formulario *clavesis.frm*, Este formulario ya está programado para encriptar y desencripar la información utilizando los dos controles ActiveX.

Para incrustar éste formulario en su proyecto, realice lo siguiente:

1.- Con su proyecto de Visual Basic abierto, en el menú principal presione en la opción *Proyecto* y después en *Formulario.*



2.- Seleccione la carpeta *Existentes* y diríjase a la carpeta que eligió para guardar los archivos del sistema de seguridad. El nombre de la carpeta lo seleccionó al momento de ejecutar el archivo *Asistente.exe*.

3.- Situado en la carpeta correcta, seleccione el archivo *clavesis.frm* y oprima en abrir.

4.- El formulario se aumentará en su proyecto.

5.- Realice los mismos pasos anteriores (1 y 2), pero ahora incorpore el archivo *module1.bas* a su proyecto.

6.- En el archivo *module1.bas* existe una función llamada *ruta,* la cual contiene la variable *rutasistema*, verifique que el valor de la variable sea el mismo que la ruta de acceso en donde se encuentra ubicado el sistema de seguridad. De no ser así, cambie su valor por el correcto.

7.- Ahora, en el formulario *clavesis.frm*, cambie el valor de la variable *Ir_a* , declarada en las *Declaraciones Generales* por el valor del formulario al que desee vincularlo, éste puede ser el menú principal o algún otro. Ejemplo: menu.show, en donde menú es el nombre del formulario en su proyecto, al que desea vincular el formulario *clavesis.frm*

Realizando lo anterior, podrá utilizar el formulario de contraseñas, y como pudo observar, realmente solo tiene que realizar tres pasos en general:

Uno: Aumente los archivos *clavesis.frm y module1.bas* a su proyecto.

Dos: Cambie la ruta del sistema.

Tres: Vincule el formulario al formulario que desee.

COMBINANDO CONTRASEÑAS Y AUTENTICACIÓN

Para implementar servicio de contraseñas y autenticación, debe de seguir los mismos pasos que realizó al implementar el servicio de contraseñas en su proyecto.

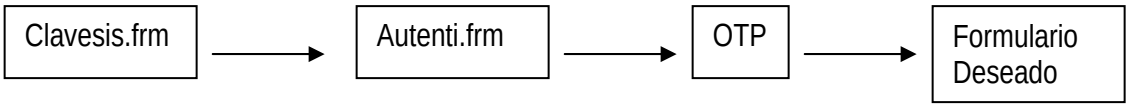
La variante es que ahora tendrá que incorporar a su proyecto, un nuevo formulario denominado con el nombre de *autenti.frm*. Para realizar esto, siga el mismo proceso que utilizó cuando incorporó el formulario *clavesis.frm*.

Las vinculaciones de los formularios, deberán de ser como se muestra abajo:

De igual forma que en el formulario *clavesis.frm,* cambie el valor de la variable Ir_a por la del formulario que desee.

IMPLEMENTANDO TODAS LAS CARACTERÍSTICAS DE SEGURIDAD

Para implementar todas las características de seguridad del que puede dotar esta solución, se deberá de cuidar la siguiente secuencia en los formularios.



La forma de implementar el formulario OTP.frm, es idéntica a los formularios anteriores, solo se debe cuidar de seguir el orden descrito arriba.