ISSN: 1692-7257 - Volumen 2 - Número 16 - Año 2010

Revista Colombiana de Tecnologías de Avanzada

Recibido: 29 de mayo de 2010
Aceptado: 03 de agosto de 2010

ENCRYPTION ALGORITHM IMPLEMENTATION: SAFETY STRATEGY FOR THE PROTECTION OF INFORMATION

IMPLEMENTACIÓN DE ALGORITMOS DE ENCRIPTACIÓN: UNA ESTRATEGIA DE SEGURIDAD PARA LA PROTECCIÓN DE LA INFORMACIÓN

MSc. Luis Carlos Revelo Tovar

Institución Universitaria CESMAG, Facultad de Ingeniería
Carrera 20A, No 14-54, San Juan de Pasto, Nariño, Colombia.
Tel.: 7216535, Fax: 7212314, Ext. 232-240.
E-mail: lcrevelo@gmail.com

Abstract: It will study the following algorithms: Blowfish, Twofish, Serpent, AES or RC6 RIJNDAEL, the study intends to implement an encryption algorithm designed as network services from a server such as email, chat, ftp, remote access, inter alia, as well as information given by a user of a computer; for this outcome to submit BLOWFISH implanting in the code of a mail server and finally e-mail can be sent encrypted between the server bills protecting the privacy of the both personal and corporate information.

Keywords: Algorithm, encryption, Computer security, free software.

Resumen: Se estudiarán los siguientes algoritmos: Blowfish, Twofish, Serpent, RIJNDAEL ó AES y RC6; el estudio pretende implantar un algoritmo de encriptación ya diseñado en los servicios de red de un servidor como correo electrónico, chat, ftp, accesos remotos, entre otros, así como también en la información administrada por un usuario de un computador ; para este resultado a presentar se implantara BLOWFISH en el código de un servidor de correo y al final se podrá enviar correo electrónico encriptado entre las cuentas de dicho servidor salvaguardando la privacidad de la información tanto personal como empresarial.

Palabras clave: Algoritmo, criptografía, Seguridad informática, software libre.

1. INTRODUCCIÓN

Una rama de la seguridad informática es la criptografía, esta permite codificar o convertir un texto completamente legible a uno ilegible en los que los caracteres pueden ser otras letras o símbolos y sin relación lingüística el uno con el otro, entonces, al final se tiene el criptograma resultado de un proceso de cifrado utilizando una clave. Este criptograma se puede leer después de un proceso de descifrado utilizando la clave provista para el proceso de cifrado. Con la globalización y la era de la información, todos necesitan de privacidad y protección de los intrusos que intentan invadir la información personal y empresarial de grandes sistemas digitales y redes corporativas como la Internet , generando gran importancia para la criptografía que también se ha encaminado a ofrecer seguridad a los usuarios del correo electrónico mediante el uso de algoritmos de encriptación, unos libres, otros no, unos más rápidos y otros más seguros pero que al final cumplen con la misión de cifrar texto, ocultar su contenido real de extraños y garantizar en gran medida que sólo aquel que tenga la clave sea quien puede descifrar y conocer el significado verdadero.

Desde esta perspectiva se estudian los algoritmos con sus características particulares y los servidores de correo electrónico con sus cualidades infinitas las cuales se describen a continuación:

- BLOWFISH: Codificador de bloques simétricos, usa bloques de 64 bits y claves de 32 hasta 448 bits.
- TWOFISH: Codificador de bloques simétricos, usa bloques de 128 bits y claves hasta 256 bits.
- SERPENT: Codificador de bloques simétricos, usa bloques de 128 bits y claves de 128 bits, 192 y 256 bits.
- RIJNDAEL ó AES:Codificador de bloques simétricos, usa bloques de 128 bits, claves de 128 bits a 256 bits.
- RC6: Codificador de bloques simétricos, usa bloques de 128 bits, claves de 128 bits, 192 y 256 bits.

Los algoritmos anteriormente mencionados fueron objeto de estudio por las siguientes razones:

- Los últimos 4 algoritmos fueron finalistas en la convocatoria realizada por el NIST (Instituto Nacional de Estándares y Tecnología de los Estados Unidos) para reemplazar en aquel entonces al algoritmo DES y que uno de ellos se convierta en el nuevo AES (Estándar de Encriptación Avanzada). Además, 3 de ellos Twofish, Serpent y Rijndael no tienen patente y se pueden utilizar libremente.
- Blowfish es gratuito y disponible para cualquiera que desee utilizarlo en sus aplicaciones y además es muy reconocido por su seguridad confirmada en su documento.

Se tendrá en cuenta los siguientes servidores de correo:

- hMailServer: servidor de correo gratuito y libre para Windows, compatible con IMAP/POP3/SMTP, dominios virtuales, listas de distribución, antivirus, anti-spam, alias, dominios distribuidos, backup.
- XMail: servidor de correo gratuito y libre para Windows y Unix, compatible con SMTP/POP3, dominios múltiples, anti-spam, alias, listas de distribución, administración remota, no hay necesidad de que el usuario tenga una cuenta en el sistema, en lugar de ello, utiliza bases de datos de archivo plano para almacenar información de usuario.
- Squirrelmail: servidor de correo electrónico que funciona bajo Linux, compatible con SMTP/POP3/IMAP, libros de direcciones, manipulación de carpetas

Estos servidores de correo fueron seleccionados porque:

- Todos son libres
- Todos ofrecen su código fuente para modificaciones
- 2 de ellos, hMailServer y Squirrelmail, son fáciles de instalar y administrar sobre Windows.

Los algoritmos se implementaron utilizando estos protocolos: SMTP, POP3 y la investigación se desarrolló sobre el sistema operativo Windows XP SP2.

2. FUNCIONAMIENTO DEL ALGORITMO DE ENCRIPTACIÓN SELECCIONADO

Los análisis y prácticas respectivas se realizaron por cada uno de los cinco algoritmos de encriptación propuestos, para este escrito se omitieron y solo se tuvo en cuenta el algoritmo y servidor de correo seleccionado en la investigación los cuales fueron Blowfish y squirrelmail respectivamente.

Blowfish: está originalmente codificado en C, sin embargo se ha encontrado una librería en PHP mucho más fácil de utilizar e implementar que el código en C, se ha escogido aquella librería por su facilidad para mostrar el funcionamiento del algoritmo, además, la librería en PHP y el código en C brindan la misma seguridad criptográfica.

La figura 1 muestra el formulario codificado por el equipo de investigación utilizado para la demostración.

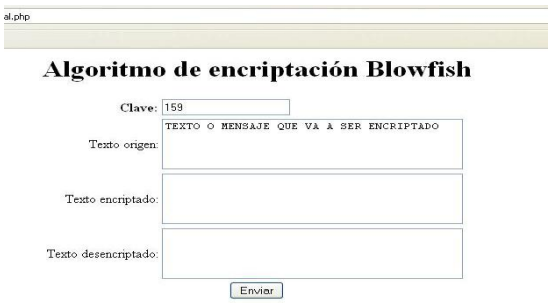

Fig. 1: Formulario de Blowfish
Fuente: El autor

El campo Clave servirá para la contraseña usada en la encriptación del mensaje, para este ejemplo se ingresó “159”, claro que la longitud de la clave debería ser más extensa para mayor seguridad y Blowfish permite una clave de hasta 448 bits, es decir 56 letras, números o combinaciones de los

mismos; el campo "Texto de Origen" tendrá el texto o mensaje que va a ser encriptado, en este caso es "TEXTO O MENSAJE QUE VA A SER ENCRIPTADO". Al presionar en el botón enviar, el texto de origen pasa a ser encriptado y se muestra en forma de símbolos en el campo "Texto Encriptado" tal y como se enseña en la figura 2.

Fig. 2: Texto encriptado con Blowfish
Fuente: El autor

Para su posterior descifrado se dispondrá de la clave y el texto encriptado y pulsado el botón enviar los símbolos en el campo "Texto encriptado" se convierten a texto plano y se muestran en el campo "Texto desencriptado", esto se aprecia a continuación en la figura 3.

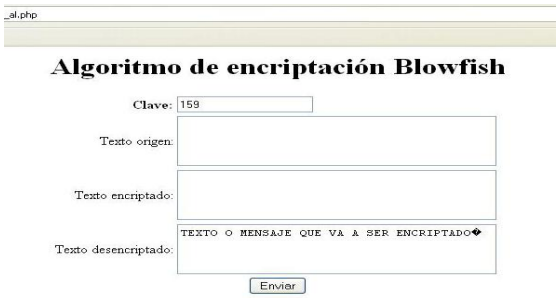

Fig. 3: Desencriptado con Blowfish
Fuente: El autor

Si la Clave es correcta, en el campo "Texto Desencriptado", debe aparecer el mensaje original; en este caso "TEXTO O MENSAJE QUE VA A SER ENCRIPTADO".

Estas son las funciones utilizadas por el algoritmo para encriptar o desencriptar, pero primero hay que declarar un objeto tipo *Crypt_Blowfish* que es la clase del algoritmo, así:

- *$bf = new Crypt_Blowfish($clave);*
- *$bf*: variable de tipo *Crypt_Blowfish*
- *$clave*: clave que se utiliza para el proceso de encriptado o desencriptado.
- *$bf->setKey($clave);* función que asigna al objeto *$bf* la clave.
  - *$clave*: clave que se utiliza para el proceso de encriptado o desencriptado.
- *$bf->encrypt($texto_plano);* función que encripta el texto plano.
  - *$texto_plano*: variable que contiene el texto plano.
- *$bf->decrypt($criptograma);* función que desencripta un texto encriptado.
  - *$criptograma*: variable que contiene el texto encriptado.

El funcionamiento es así: hay 18 vectores llamados vectores P de 32 bits cada uno usados como subclaves para encriptar y desencriptar, éstos deben ser pre-calculados antes de cualquier proceso de encriptación o desencriptación. Hay 4 cajas-S de 32 bits cada una con 256 entradas por caja.

Encriptación Blowfish es un codificador Feistel de 16 rondas, su entrada es un dato de 64 bits.Cada línea representa 32 bits de los 64 de entrada, es decir se divide la entrada en 2 mitades de 32 bits cada una xL y xR; luego, en un ciclo i que va desde 1 hasta 16 (16 rondas) se asigna a xL el resultado de la operación xL XOR vector Pi, donde Pi corresponde a la ronda i (ciclo); seguidamente, xR guarda el resultado de la función Feistel F(xL) XOR xR; se intercambia las mitades; se incrementa i. Luego hay un nuevo intercambio de mitades; xR almacena el resultado de xR XOR P17; xL guarda el resultado de xL XOR P18 y finalmente se combinan las mitades obteniéndose el criptograma.

3. FUNCIONAMIENTO DEL SERVIDOR DE CORREO SELECCIONADO

Squirrelmail: esta aplicación está escrita en PHP y en HTML 4.0 para que sea soportado por la mayoría de los navegadores web, también está diseñado para trabajar con plugins (nuevas características) para hacer más llevaderas las tareas de agregar nuevas características a la aplicación. Squirrelmail está bajo la licencia GNU lo cual le convierte en software libre.

Funcionamiento: Squirrelmail es intuitivo y muy fácil de manejar, primero hay que iniciar sesión, luego una bandeja de entrada que tiene una forma básica como la mayoría de los servidores de correo con webmail (correo administrado vía Web) en la que hay una lista de mensajes en una parte de la pantalla y en la otra están las carpetas de bandeja de entrada, correo basura, papelera y demás, véase en la figura 4 un ejemplo de esto.

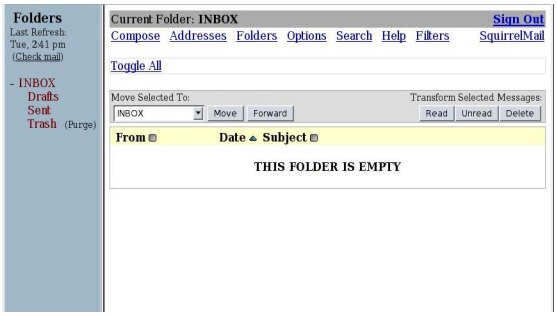

Fig. 4: Bandeja de entrada de Squirrelmail
Fuente: El autor

Para enviar mensajes basta con dar clic en *Compose* y se tiene acceso a un formulario típico para escribir un mensaje. Mírese en la figura 5 el formulario para componer mensajes de Squirrelmail.

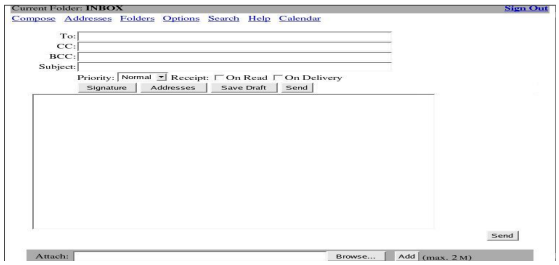

Fig. 5: Formulario de mensajes Squirrelmail
Fuente: El autor

4. IMPLEMENTACIÓN DEL ALGORITMO DE ENCRIPTACIÓN BLOWFISH EN EL SERVIDOR DE CORREO SQUIRRELMAIL

Debido a que el algoritmo es simétrico, se necesita de un campo de texto en el que el usuario ingresa la contraseña o bien para encriptar o desencriptar un correo, es decir: que el archivo ***compose.php*** tendría un campo de texto adicional además de los ya existentes y que se llamaría *Key* o similar para ingresar la clave que encriptará el mensaje, pero si en el caso de que el usuario no desee enviar un correo encriptado entonces a un lado del campo de texto *Key* estaría una caja de chequeo (checkbox) llamada *Encrypt* que marcada habilitaría el campo *Key* y sino sucedería lo contrario, no obstante, el campo *Key* siempre estaría deshabilitado desde el momento de cargar el archivo ***compose.php***; eso por una parte, por otra, el fichero ***read_body.php*** debería poseer también un caja de texto para entrar la clave y desencriptar el mensaje y que tendría el mismo nombre *Key* como en el archivo ***compose.php*** y conjuntamente con un botón nombrado *Decrypt* que permitiría decodificar el correo.

Recuérdese que una de las razones por las que se seleccionó el algoritmo es su fácil implementación y que ello no signifique que la seguridad es baja o mínima, en el análisis de los algoritmos se señaló que todos ofrecen seguridad de muy alto nivel así como su velocidad pero que la implementación era más compleja en unos que en otros.

La implementación de Blowfish, el algoritmo seleccionado, se puede hacer prácticamente en 3 líneas de código PHP que seguirían este trazo:

1. Incluir la librería que aloja el código de Blowfish.
2. Crear una nueva instancia de la clase de Blowfish y asignarle la clave.
3. Encriptar o desencriptar una cadena utilizando la clase anteriormente declarada.

Eso fue exactamente lo que se hizo con el algoritmo en los dos archivos para enviar y recibir correo, ***compose.php*** y ***read_body.php*** respectivamente. Claro, pero además se tuvo que agregar otras líneas que modificaron en un tanto el aspecto del par de archivos, esto es, los campos de texto, la caja de chequeo y el botón de los que se hablaron al iniciar este apartado.

A continuación se mostrarán las líneas que se adicionaron al archivo ***compose.php*** y más adelante su nuevo aspecto y funcionamiento visual.

```
require_once(SM_PATH
'functions/Crypt/Crypt/Blowfish.php');
```

Esta pequeña porción de código es la función para habilitar o deshabilitar la caja de chequeo *Encrypt* y el campo de texto *key*.

```
echo"<script type='text/javascript'>";
echo"function enabling()".
"{".
        "var author = document.getElementById('enabler');".
        "var target = document.getElementById('pswd');".
        "if (author.checked == true)".
        "{".
                "target.disabled=false;".
        "}".
        "else".
        "{".
                "target.disabled=true;".
        "}".
"}";
echo"</script>";
echo"<tr><td align='right'><b>Encrypt</b>:</td>".
"<td><table><tr><td align='left'>".
"<input type='checkbox' value=" id='enabler'
onclick='enabling()' /><br/>".
"</td><td><b>Key:</ b></td>".
"<td align='left'><input type='password' name='clave' value="
size='45' maxlength='40' disabled='true' id='pswd'></td>".
"</tr></table></td></tr>";
```

Primero está la función *enabling()* en Java script que utiliza *getElementById()* para obtener el estado de la caja de chequeo y preguntar si está o no marcada, así modifica la propiedad *disabled* del campo de texto a falso y habilitarlo o a verdadero y deshabilitar el campo de texto. Luego, hay código HTML en medio de un "*Echo"* para agregar la caja de chequeo *Encrypt* y el campo de texto *Key*, como se ve éste último está deshabilitado por defecto.

Finalmente, los 2 últimos puntos del anterior trazo.

```
$clave = $_POST["clave"];
if($clave != '')
{
        $bf = new Crypt_Blowfish($clave);
        $body = $bf->encrypt($body);
}
```

No está demás en utilizar el *IF* para verificar si la variable *$clave* está vacía o no, se aclara que la variable *$clave* toma el valor del campo de texto *Key* al momento de pulsar el botón *Enviar* (*Send*). Luego se declara una nueva instancia de la clase *Crypt_Blowfish* con la variable *$clave*, esta servirá para codificar el mensaje, y por último se encripta el mensaje que está guardado en la variable *$body* y se reasigna a ella misma, se deja continuar con el resto del código original del servidor y lo que se envía es un mensaje encriptado.

El mismo procedimiento se empleó con ***read_body.php***, se incluyó el código de Blowfish en el fichero, se alteró un poco su aspecto y se sumaron las líneas necesarias para desencriptar un correo.

Seguidamente se indicarán las líneas nuevas en el archivo ***read_body.php*** y más adelante su nueva vista y funcionamiento también visual.

Como antes, se hace referencia al trazo para usar Blowfish, entonces esta primera línea incluye el código del algoritmo en el fichero.

```
require_once(SM_PATH
'functions/Crypt/Crypt/Blowfish.php');
```

El siguiente trozo de las líneas agrega el campo de texto *Key* y el botón *Decrypt*.

```
echo "<form name='form1' action='read_body.php'
method='POST'>";
echo "<tr><td align='right'><b>Key:  </b></td>".
"<td align='left'><table><tr>".
"<td><input type='password' name='clave' value="
maxlength='45' size='40'></td>".
"<td><input type='submit' name='decod' value='Decrypt'>".
"</td></tr></table></td></tr>";
echo "<center>".
"<input type='hidden' name='mailbox' value='INBOX'>".
"<input type='hidden' name='sM' value='1'>".
"<input type='hidden' name='passed_id'
value='".$passed_id2."'></center>";
echo "</form>";
```

Las líneas son un formulario HTML construido con "*Echos*", tiene como objetivo el mismo fichero puesto que cuando el usuario entra la clave y pulsa en el botón *Decrypt* el mensaje se muestra en la misma página, más abajo hay 3 objetos tipo *Hidden* que ayudan a que el archivo ***read_body.php*** abra el mismo mensaje que estaba leyendo.

La decodificación viene dada en estás líneas, así:

```
$clave = $_POST["clave"];
if($clave != '')
{
        $bf = new Crypt_Blowfish($clave);
        $bf->setKey($clave);
        $messagebody = $bf->decrypt($messagebody);
}
```

Se evalúa si la variable *$clave* no esté vacía, se declara una nueva instancia de *Crypt_Blowfish* asignándole como clave la variable *$clave*, luego una nueva línea para asegurarse de que la nueva instancia trabaje con la clave dada en el principio y se sabe que el mensaje se guarda en la variable $*message_body*, por lo tanto se vuelve a guardar en la misma el mensaje desencriptado utilizando la instancia anteriormente hecha.

De esa forma es como trabaja el algoritmo en el servidor y su ratificación estará disponible enseguida en la explicación del acondicionamiento del servidor.

5. ACONDICIONAMIENTO DEL SERVIDOR

Este apartado se ha llamado Acondicionamiento del servidor debido a que se tenía que modificar la parte visual del mismo, estas razones ya se explicaron en la sección inmediatamente anterior.

Las figuras 6 y 7 del archivo ***compose.php*** son el formulario común, sin cambios y el formulario acondicionado respectivamente.

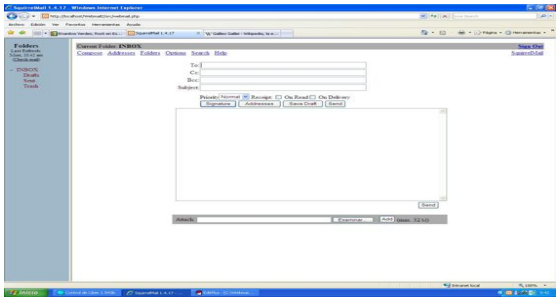

Fig. 6: ***Compose.php*** *original*
Fuente: El autor

Así lucía el formulario original para el envío de mensajes de *Squirrelmail*, fácil de utilizar, amigable y sin ninguna novedad; la próxima figura es el mismo formulario pero con la caja de chequeo (*checkbox*) *Encrypt* y el campo de texto *key* agregados.

Se hizo esto para confrontar el antes y después de la modificación del servidor, de forma que se tenga idea de cómo funciona y actúa, que en pocas palabras las reformas no añaden dificultad alguna para su uso.

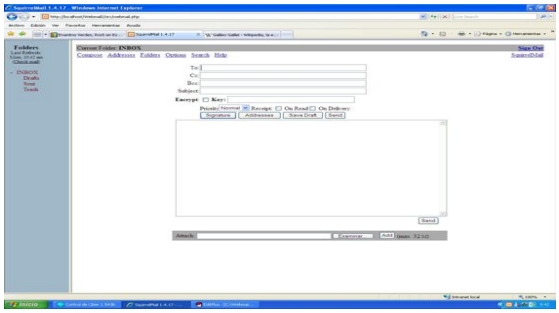

Figura 7. ***compose.php*** modificado
Fuente: El autor

Mírese los objetos agregados, bajo el campo de texto *Subject* están *Encrypt* y *Key* necesarios para todo el proceso de encriptación del mensaje; se había dicho anteriormente que el campo de texto *Key* se carga deshabilitado por defecto y así lo hace, sólo que con esta demostración hecha en el navegador para Internet IE (Internet Explorer) de Microsoft no se nota, pero en otros como Mozilla Firefox el campo de texto aparecerá de un color gris, así tal cual en la figura 8.

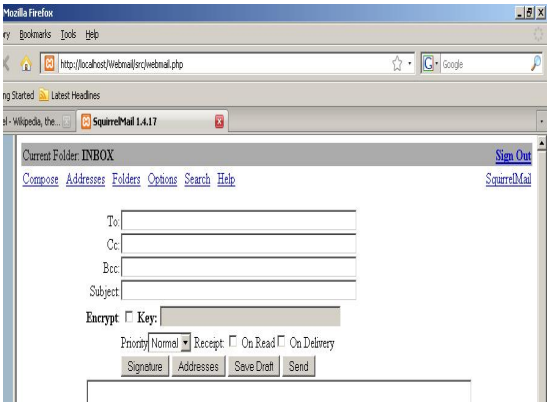

Fig. 8: ***compose.php*** *modificado y campo de texto Key*
Fuente: El autor

Efectivamente *Key* está deshabilitado desde la carga del formulario.

Las figuras 9 y 10 confrontan esta vez el archivo ***read_body.php***, la primera será la apariencia original del archivo, mientras que la segunda es los objetos agregados.

Fig. 9: ***read_body.php*** *original*
Fuente: El autor

Este aspecto de ***read_body.php*** es utilizado cuando se está leyendo un correo no encriptado, no hay campos de texto y tampoco un botón en el encabezado ni nada por el estilo y el mensaje aparece legible, comprensible.

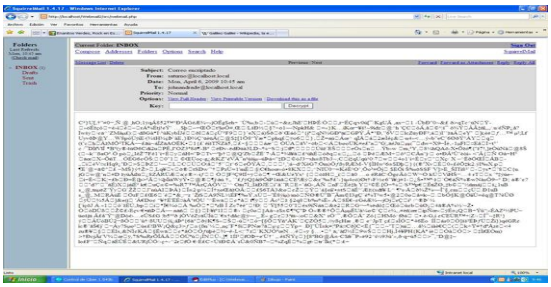

Fig. 10: ***read_body.php*** *modificado*
Fuente: El autor

Esta es la apariencia de ***read_body.php*** cuando se lee un correo encriptado, detállese que en el encabezado están el campo de texto *Key* y el botón

Decrypt útiles para decodificar el mensaje que en este caso parece relativamente extenso e ilegible, incomprensible, enigmático.

Pruebas: Las pruebas se realizaron de forma local en el mismo servidor, se crearon diferentes cuentas de correo y un par de nombres de dominio ficticios para experimentar si entre estos era posible el envío de mensajes encriptados pero utilizando el mismo tipo servidor. La figura 11 muestra el envío de un mensaje que va a ser encriptado pero con un solo receptor.

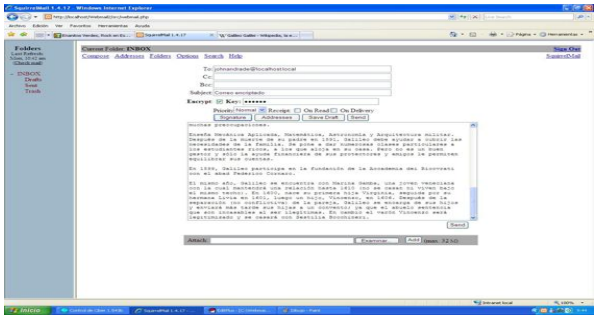

Fig. 11: Envío de mensaje encriptado con un sólo receptor
Fuente: El autor

El mensaje es enviado a *johnandrade@localhost.local* y activada la caja de chequeo (checkbox) *Encrypt* se habilita el campo de texto *Key* para el ingreso de la clave y sin más se pulsa el botón *Send* y listo, el correo se envía encriptado, en la figura 12 se le verá listado en la bandeja de entrada del receptor

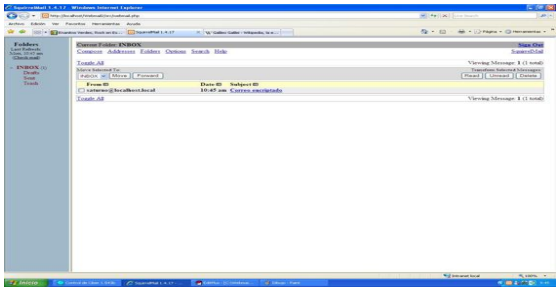

Fig. 12: Recepción de un mensaje encriptado
Fuente: Esta investigación

Aquí está el correo procedente de *saturno@localhost.local* y el asunto es indudablemente *Correo encriptado* el mismo que en la figura pasada, haciendo clic en él se abre utilizando el archivo ***read_body.php*** y debería mostrar caracteres sin sentido y de cualquier tipo no sólo letras y números, el mensaje será incoherente, caótico; esto que se ha dicho es visto en la próxima figura 13.

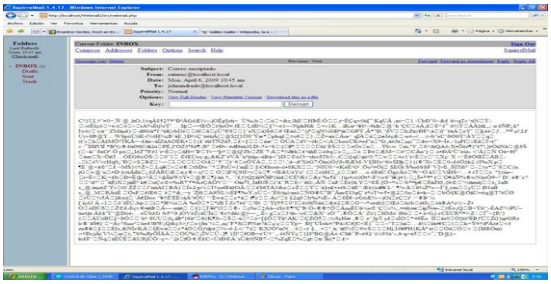

Fig. 13: Apertura de un mensaje encriptado
Fuente: El autor

Se observa el confuso conjunto de caracteres en el mensaje y en el encabezado está el asunto *Correo encriptado*, su autor *saturno@localhost.local* y saltándose unos cuantos se tiene al final el campo de texto *Key* y el botón *Decrypt* para la decodificación, lo único que tendría que hacer el receptor es ingresar la clave y pulsar el botón para obtener el mensaje original. La desencriptación de este mensaje se muestra en las figura 39 y 40.

6. CONCLUSIONES

Al finalizar el desarrollo de este proyecto, se deduce claramente que el termino de encriptación es el proceso para volver ilegible la información que se considere importante y de carácter privada; por lo tanto al implementar el algoritmo de encriptación dentro del servidor de correo electrónico, y al realizar el debido procedimiento de envió de mensajes, toda la información que haya sido encriptado; podrá leerse aplicándole la clave correcta, cabe destacar que este proyecto trata de una medida de seguridad de la información dentro de un correo electrónico, la cual no debería ser accesible a terceros.

REFERENCIAS

Cormen, Thomas H.; Leiserson, Charles E. And Rivest, Ronald L. Introduction to algorithms. Cambridge, The MIT Press, 2000, pp. 1.

Knuth, Donald. The art of computer programming. Massachusetts, Addison-Wesley, 1997, pp. 5.

Stallings, William. Cryptography and network security Principles and practices, fourth edition. Prentice Hall, 2005, p. 64.

Stallings, Op. Cit., pp. 67.

Stallings, William. Cryptography and network security Principles and practices, fourth edition. Prentice Hall, 2005, pp. 181.