Grado en Matemática Computacional

Estancia en Prácticas y Proyecto Final de Grado

Cifrado de clave privada: AES

Autor:
Javier Martínez de la Torre

Supervisor:
Arturo Beltran Fonollosa
Tutor académico:
Fernando Javier Hernando Carrillo

Fecha de lectura: 03 de Octubre de 2016
Curso académico 2015/2016

Agradecimientos

A Fernando, por las horas invertidas en ayudarme con este trabajo y preparar conmigo la exposición. Y al resto de personal docente de la Universidad Jaume I, que siempre han mostrado su disposición por ayudarme y enseñarme.

A las personas que han compartido mis alegrías y frustraciones estos años trabajando junto a mi en la pecera. De forma especial, a Cristian que, como amigo y compañero, me ha ayudado y entrenado.

A mi familia, son ellos quienes me apoyan y me mantienen despierto.

Resumen

Este trabajo detalla el proyecto realizado en la empresa Sofistic Telematic Security SL, este proyecto consiste en el desarrollo de un programa para cifrar archivos escritos en el lenguaje de programación PHP y una manera de desencriptar y ejecutar estos archivos. Aprovechando este cifrado se realiza una explicación de la criptografía, se desarrolla la historia y evolución de la criptografía y una explicación sobre criptografía de llave pública y llave privada, donde más adelante se profundizará en AES no sin antes realizar una explicación sobre los cuerpos finitos ya que son necesarios para entender el funcionamiento de AES.

Palabras clave

Criptografía, clave privada, AES , Cuerpos finitos, Encriptación.

Keywords

Cryptography, private key, AES, Galois Fields, Encryption

Índice general

1. Introducción **11**

1.1. Contexto y motivación del proyecto 11

2. Estancia en prácticas **13**

2.1. Introducción 13

2.1.1. Servidor Apache 14

2.1.2. Protocolo HTTP 15

2.1.3. HTML 15

2.1.4. PHP 16

2.1.5. CGI 16

2.1.6. Localhost 16

2.1.7. VirtualHost 17

2.2. Objetivos del proyecto formativo 17

2.2.1. Idea primaria del proyecto 17

2.2.2. Módulo de Apache 18

2.2.3. Extensión de PHP 19

2.3. Explicación detallada del proyecto realizado en la empresa 21
2.3.1. Funciones 21
2.3.2. Metodología y definición de tareas 22
2.3.3. Planificación temporal de las tareas 23
2.3.4. Estimación de recursos del proyecto 24
2.3.5. Grado de consecución de los objetivos propuestos 24

3. Criptología **25**

3.1. Introducción a la criptología 25
3.2. Criptosistema 30
3.2.1. Clave asimétrica o clave pública 31
3.2.2. Clave simétrica o clave privada 32

4. Requisitos matemáticos **33**

4.1. Grupo 33
4.2. Anillo 34
4.3. Cuerpo 34
4.4. Teorema de Bézout 38
4.5. Algoritmo de Euclides 38
4.6. $GF(2^8)$ Cuerpo finito utilizado en AES 39
4.6.1. Suma 39
4.6.2. Multiplicación 39
4.6.3. Inversa 40

5. AES **41**

5.1. Historia y precedentes de AES 41

5.2. Funcionamiento de AES 45

5.2.1. Matriz de Estado 45

5.2.2. Estructura de AES para encriptar 46

5.2.3. Estructura de AES para desencriptar 56

5.3. Seguridad de AES 59

5.3.1. Criptoanálisis diferencial 59

5.3.2. Criptoanálisis lineal 60

5.4. Ejemplo de AES 60

5.4.1. Key Addition layer Ronda 0 62

5.4.2. Ronda 1 62

5.4.3. Ronda 2 63

5.4.4. Ronda 3 64

5.4.5. Ronda 4 65

5.4.6. Ronda 5 66

5.4.7. Ronda 6 67

5.4.8. Ronda 7 68

5.4.9. Ronda 8 69

5.4.10. Ronda 9 70

5.4.11. Ronda 10 71

A. Anexo I **75**

A.0.12. Query string 75

A.0.13. URL 75

A.1. Blowfish 76

A.2. Script 76

A.3. Hexadecimal 77

A.4. XOR 78

A.5. Código ASCII 78

Capítulo 1

Introducción

A lo largo de la historia la necesidad de ocultar la información ha ido en aumento, desde el método de la escítala usado en Esparta pasando por los métodos de cifrado utilizados en las diferentes guerras, que tienen como culmen la máquina enigma utilizada por los alemanes, hasta llegar a la época actual en la cual los métodos de cifrado son imprescindibles en los distintos sistemas informáticos. En la actualidad todos los dispositivos informáticos que contienen información privada cuentan con sistemas de encriptación tanto para el almacenamiento de información como para el intercambio de información entre estos sistemas informáticos.

1.1. Contexto y motivación del proyecto

Con la aparición de los primeros computadores surgió la necesidad de cifrar la información en los distintos sistemas informáticos, y las organizaciones de seguridad de los distintos países buscaron algoritmos de cifrado lo suficientemente fuertes y rápidos para poder usarlos. Estos primeros algoritmos se trataban de algoritmos de clave privada en los cuales existe una clave con la cual se cifra de manera que para poder acceder a la información se debe conocer dicha clave. Es en este contexto aparece AES como método de cifrado para sustituir a DES, un algoritmo que quedó obsoleto.

Este algoritmo fue implantado como estándar por la NSA y desde ese momento ha sido el algoritmo más utilizado en gran diversidad de ámbitos, tanto en el sector privado como en el público y en los gobiernos. Tanto es así que el gobierno de Estados Unidos permite que se cifre información considerada TOP SECRET utilizando este algoritmo.

A lo largo de este trabajo hablaremos sobre criptografía profundizando en la criptografía de clave privada y más concretamente en el algoritmo de cifrado AES. El trabajo se divide en 5 capítulos contando con esta introducción. En el segundo capítulo,"Estancia en prácticas", se realiza una explicación sobre el proyecto desarrollado en las horas destinadas a las prácticas. En el tercer capítulo damos una introducción la criptología, la definición de criptosistema y la introducción de los conceptos de algoritmo de clave pública y clave privada. El cuarto capítulo, "Requisitos matemáticos", explica los conceptos matemáticos utilizados por AES y que por tanto son necesarios para comprender su funcionamiento. En el quinto capítulo se desarrolla toda la explicación sobre AES, un apartado sobre la historia de AES y su "predecesor" DES, un subapartado donde se explica el funcionamiento de AES tanto para cifrar como para descifrar y otro apartado con un ejemplo de cifrado. Por último hay un anexo en el que se desarrollan brevemente algunos conceptos nombrados a lo largo del trabajo.

Capítulo 2

Estancia en prácticas

En este apartado se detalla la labor de la empresa, mi estancia en las prácticas, el objetivo del proyecto y el trabajo realizado en la empresa.

2.1. Introducción

Sofistic Telematic Security, S.L es una empresa dedicada a la seguridad informática, se encuentra en el Espaitec (Parque científico, tecnológico y empresarial de la Universidad Jaume I de Castellón) dentro del campus de la Universidad Jaume I. Esta empresa realiza preparación y mantenimiento de servidores en entornos seguros, auditorías de seguridad externas e internas, Test de intrusión interna o externa, auditorías de código fuente, etc. Además, ofrece cursos para formar a personal de otras empresas. Dentro del servicio de mantenimiento y preparación de servidores se utiliza servidores Apache.

2.1.1. Servidor Apache

Se trata de un servidor web, es decir, es un programa especialmente diseñado para transferir datos de hipertexto. El hipertexto es una herramienta que permite crear, agregar, enlazar y compartir información de diversas fuentes por medio de enlaces asociativos, los datos que transfieren los servidores son páginas web con todos sus elementos (textos, widgets, banners, etc). Estos servidores web utilizan el protocolo http.

Estos servidores consisten en un programa ejecutándose sobre un ordenador que se encuentra a la espera de que algún navegador le haga alguna petición, como por ejemplo acceder a una página web. Una vez se realiza la petición responde a la misma enviando código HTML.

HISTORIA:

Apache 0.2 vería la luz el 18 de marzo de 1995 basado en el servidor NCSA httpd 1.3 (National Center for Supercomputing Applications) realizado por Rob McCool.[12] El nombre del proyecto Apache se debe a la filosofía de desarrollo y de organización. Al igual que la tribu de los apaches, los desarrolladores de Apache decidieron que su forma organizativa debía fundamentarse en los méritos personales de los desarrolladores para con el resto de la comunidad Apache. Habría que esperar a enero de 1996 para poder disfrutar de la primera versión estable de Apache, la Apache 1.0, que incluía la carga de módulos en tiempo de ejecución a modo de pruebas además de otras funcionalidades interesantes. Los primeros meses de ese año fueron especialmente fructíferos para el proyecto, ya que la versión 1.1 se publicó apenas dos meses después contando con módulos de autentificación contra bases de datos (como MySQL).

Desde entonces hasta la actualidad, los hitos más grandes del proyecto han sido la total conformidad con el estándar HTTP 1.1 (incluido en abril de 1997 en Apache 1.2), la inclusión de la plataforma Windows NT (que comenzó ya en julio de 1997 con las versiones en pruebas de Apache 1.3), la unificación de los archivos de configuración en uno solo (habría que esperar a octubre de 1998 y a Apache 1.3.3 para ello) y mas tarde el lanzamiento de Apache 2.

Estos servidores contienen modulos de interpretación de PHP de forma que se pueden insertar códigos en PHP directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante, el módulo que se encarga de esto se llama *mod_php*.

Ejemplo de como se integra el código php en un html:

```
// hello.php
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <title> Ejemplo básico PHP</title>
    </head>
    <body>
        <?php
            echo 'Hola mundo';
        ?>
    </body>
</html>
```

La parte de código escrito dentro del "< *body* >" es el código en lenguaje php que lo que hace es mostrar por pantalla el mensaje "Hola mundo".

2.1.2. Protocolo HTTP

HTTP (en español protocolo de transferencia de hipertexto o Hypertext Transfer Protocol en inglés) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. Se basa en sencillas operaciones de solicitud/respuesta entre el cliente, quien establece una conexión con un servidor, y dicho servidor que envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado.

2.1.3. HTML

HTML (HyperText Markup Language, en español lenguaje de marcas de hipertexto) es el lenguaje de elaboración de páginas web. Es un estándar de elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web.

2.1.4. PHP

PHP es un lenguaje abierto especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML. El lenguaje fue desarrollado en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El lenguaje fue denominado Personal Home Page Tools abreviándolo como PHP Tools.

Debido a la aceptación del PHP Rasmus Lerdorf diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas: PHP/FI.

En 97 se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP versión 3. Actualmente se encuentra en su versión 7 y utiliza el motor de Zend (es un motor de procesamiento para la interpretación y cifrado del código php, desarrollado para brindar un equipo de soporte y acelerar la carga de aplicaciones realizadas con php.)

2.1.5. CGI

Interfaz de entrada común (en inglés Common Gateway Interface, abreviado CGI) es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa.

2.1.6. Localhost

Localhost es un nombre reservado que tienen los dispositivos, tengan o no una tarjeta de red ethernet. El nombre localhost es traducido como la dirección IP de loopback 127.0.0.1 en IPv4, o como la dirección ::1 en IPv6.1 Es el lugar donde se añaden los sitios web en tu servidor y desde http://localhost/ son accesibles. En el caso de Apache los archivos de los sitios web se añaden en la carpeta /www/ del servidor. En el caso de querer acceder al sitio web sin tener que referenciar el localhost entonces debería de configurarse un VirtualHost de forma que podamos acceder al sitio web con la dirección que establezcamos en el virtualhost.

2.1.7. VirtualHost

El VirtualHost sirve para poder alojar varias webs en un mismo servidor, lo que hace es referenciar dichas webs o páginas mediante nombre o mediante IP, el primero implica que tienes varios nombres de páginas sobre una misma dirección IP y la segunda quiere decir que cada página o web tiene su propia dirección IP. Lo que es un VirtualHost es un archivo que relaciona nombres con IP para poder acceder a las páginas web.

2.2. Objetivos del proyecto formativo

2.2.1. Idea primaria del proyecto

PHP es un lenguaje interpretado y no compilado. Un lenguaje interpretado es aquel en el cual sus instrucciones o más bien el código fuente, escrito por el programador en un lenguaje de alto nivel, es traducido instrucción a instrucción por el intérprete a un lenguaje entendible para la máquina. Por esta razón en los servidores el código php es "visible", es decir, no existe un archivo compilado de ese programa, sino que el programa de PHP contiene el código en texto plano. El proyecto desarrollado en la estancia en prácticas lo hemos dedicado a este proposito, es decir, el código es visible y la empresa no quiere que se pueda acceder a él, bien para que no pueda verse el código o bien para que no pueda ser modificado.

Debido a esto era necesaria una forma de encriptar el código y desencriptarlo solo para su ejecución. Así pues, el código se encuentra cifrado en el servidor en los respectivos archivos PHP y a los servidores se les asigna una clave para poder desencriptar y ejecutar dicho código.

Además, se buscaba una manera de encriptar proyectos enteros de PHP de forma que se copiara la estructura de los proyectos y se creara un proyecto con la misma estructura, nombre y nombre de ficheros pero que dichos ficheros estuvieran encriptados. De esta manera la empresa podía añadir un servicio para que los clientes encriptaran sus proyectos y que los pudieran ejecutar en su servidor.

A petición de la empresa y por motivos de rendimiento el cifrado de los archivos debía de hacerse mediante un algoritmo de clave privada, o clave simétrica, debido a que es rápida y utiliza menos recursos informáticos que otras formas de cifrado. Para desarrollar esta idea se buscaron varias maneras de llevarla a cabo, una primera opción era desarrollar un módulo de Apache y la otra opción era desarrollar una extensión de PHP. Durante las primeras semanas de la estancia en prácticas se decidió llevar a cabo un módulo de Apache que introdujera estas funciones en el servidor.

2.2.2. Módulo de Apache

Un Módulo de Apache es un archivo compilado que contiene nuevas funciones que se pueden añadir al servidor. Es una forma de agrupar y modularizar ciertos funcionamientos para el Servidor, existen una gran cantidad de Módulos para utilizarse con Apache, por ejemplo: "Virtual Hosting","Mod_JK(Java)" y "Rewrite".

Un motivo para emplear módulos en Apache es que no todos los servidores o no todas las instalaciones requieren todas las funcionalidades e incluir todos los módulos en un único Apache "completo" haría que los requerimientos de Memoria RAM fueran muy pesados, además ocuparían mucho más espacio en el disco duro debido a que existen muchos módulos.

Los módulos de Apache se pueden dividir en cuatro categorías diferentes:

- Módulos base

  Los módulos base están compilados en Apache por defecto. Todos los demás están disponibles como objetos compartidos: en lugar de estar incluidos en el binario del servidor en sí, se pueden incluir durante la ejecución.

- Módulos de extensión

  Por regla general, los módulos etiquetados como "extensiones" se incluyen en el paquete de software de Apache, aunque normalmente no se compilan en el servidor de manera estática.

- Módulos externos

  Los módulos etiquetados como externos no se incluyen en la distribución oficial de Apache.

- Módulos de multiprocesamiento

  Los MPM son los responsables de aceptar y gestionar peticiones al servidor Web, por lo que representan el núcleo del software del servidor Web.

Sin embargo, tras unas semanas de prácticas se decidió cambiar el módulo de Apache por una extensión de PHP para poder llevar a cabo la tarea, el cambio fue debido a una decisión tomada por la empresa tras comprobar cómo funcionaba un sistema similar.

2.2.3. Extensión de PHP

Las extensiones son archivos, librerías externas, que permiten añadir nuevas funciones al lenguaje PHP y utilizarlas de la misma manera que las que vienen por defecto en PHP.

Bajo el capó, las funciones y clases internas del PHP están implementadas en C. Los desarrolladores pueden escribir funciones y clases en C o C++ también. Hay dos razones principales para querer escribir tu propia extensión de PHP:

- Tener una librería que quieras acceder en PHP, la única solución es escribir una extensión de envoltura para ello, de forma que después puedas usarla desde PHP
- Rendimiento, debido a que las funciones en c no ejecutan la máquina virtual Zend entonces tienen significativamente menos sobrecarga, para funciones que no estén relacionadas con recursos externos es razonable realizar una extensión de PHP ya que son entre 10 y 100 veces más rápidas.

Una de las fortalezas de PHP es su curva de aprendizaje superficial. Una de las principales ventajas de usar un lenguaje de alto nivel (como PHP o Perl y diferencia de C o C ++) es que le protege de tener que realizar su propia gestión de memoria y de cometer errores que pueden hacer que el propio intérprete de PHP se bloquee. Escribir una extensión en C hace que pierdas esos beneficios.

Extensión básica

La manera más sencilla de crear una extensión de PHP es usar la estructura ("Skeleton") por defecto de una extensión [5]. Para hacer esto usar el ext_skel script que se encuentra en la carpeta ext dentro del directorio principal de PHP. Para crear una extensión de nombre example, debes usar lo siguiente desde la carpeta raíz del directorio PHP:

```
> cd ext
> ./ext_skel -extname=example // Esto lo que hace es usar el script ''ext_skel'' para
    crear una estructura de extensión, crea los ficheros :
    ''config.m4'',''php_example.h'' y ''example.c''
```

Para usar la nueva extensión se debe ejecutar lo siguiente:

```
$ cd ..
$ vi ext/example/config.m4
$ ./buildconf
```

```
$ ./configure -[with|enable]-example
$ make
$ ./php -f ext/example/example.php
$ vi ext/example/example.c
$ make
```

Nos encontramos con que se crea un fichero llamado "example.c" que es básicamente el que contiene todo el código en c que formará o creará la nueva extensión. Este fichero tiene esta estructura:

```
// include PHP API
#include <php.h>
#include "php_example.h"

// Define las funciones que queremos añadir
zend_function_entry example_functions[] = {
  PHP_FE(examplefunction, NULL)
  { NULL, NULL, NULL }
};

// "example_functions" se refiere a la estructura de arriba
// Esta estructura tiene dentro cada una de las funciones
zend_module_entry example_module_entry = {
  STANDARD_MODULE_HEADER,
  PHP_EXAMPLE_EXTNAME,
  example_functions,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  PHP_EXAMPLE_VERSION,
  STANDARD_MODULE_PROPERTIES
};

// instala el módulo
ZEND_GET_MODULE(example)

// actual non-template code!
PHP_FUNCTION(examplefunction) {
    php_printf("Esta función escribe este mensaje.\n");
}
```

2.3. Explicación detallada del proyecto realizado en la empresa

Tras unos días explorando las opciones con los miembros de la empresa se llegó a la conclusión de que la idea buscada con este proyecto era crear unas funciones que se añadirían a las funciones que ya contiene el lenguaje PHP, estas funciones serían añadidas creando una extensión de PHP que se instalaría en los servidores que fueran a usar estas funciones. Esta extensión consiste en un programa en lenguaje C que contiene la definición de dos funciones nuevas para PHP, "encripta_quasarcode" y "desencripta_quasarcode".

2.3.1. Funciones

encripta_quasarcode

Esta función cifra un fichero usando una clave y creando un archivo PHP en la ruta dada por la entrada de la función, tanto el nombre del fichero a encriptar, la clave y la ruta son dados como parámetros a la función. El archivo creado contiene una cadena de caracteres que es la información del archivo primario encriptada, además el archivo tiene una llamada a la función "desencripta_quasarcode" de forma que este fichero cuando se ejecuta llama a esta función con el fin de que el código sea desencriptado y ejecutado en el servidor.

Ejemplo de uso de la función

Sea el fichero "ejemplo.php" que se quiere encriptar, la clave "ktMWoEgLSs57mlk8dzGO9HQAY pf79UNG" y sea la ruta "/home" donde se quiere que el fichero encriptado se guarde entonces la llamada a la función es:

```
encripta\_quasarcode(''ejemplo.php'',''ktMWoEgLSs57mlk8dzGO9HQAYpf79UNGtFm5V8YRvkAu
knvjZCW5lZiJ'', ''/home'');
```

desencripta_quasarcode

Esta función recibe por parámetro el código encriptado y la clave para desencriptar. La variable con la clave debe existir en el localhost del servidor con el nombre "CLAVE". El programa encriptado busca dicha variable y una vez la tiene se la pasa a la función en la llamada. Una vez tiene la clave la utiliza para desencriptarlo y obtener el código desencriptado en un string, este código desencriptado pasa a una función que ejecuta este código y lo muestra en el navegador. Además, como estos archivos están pensados para que sean ejecutados desde un navegador entonces la función también tiene un parámetro que contiene el "query string" (Anexo A.0.1) dado por el navegador, este "query string" lo utiliza para obtener los posibles parámetros que necesite el código que hemos desencriptado para su ejecución.

Además en la llamada se añade dos números que sirven para que se la operación se realice correctamente, los añade la función "encripta_quasarcode" de forma automática al fichero encriptado y el usuario no debe preocuparse de eso.

Ejemplo de archivo encriptado con la llamada a la función:

```
<?php
$mensaje="¨Ëà?-TrgOSoixH6HiyYcilyCahPIzyg6yuCcDaZd1zAvzKNNVgDQT1v076sK8zIWDohJUxgwufdd
2SuB+PplGe/OO8+kBLEO38dEUe5jQLHVE3jHuAnm3Cish5HkaIgV4T83rOOOCUs/VpdA4rzyE23BxFLfqkm1oj
y7XAxN/7Tsl5OH2RLOE9yQFD2ie8jTLhWp3ntLlOo8rBgtmBSqUFObzjjdQnK+sfdkk/ynwQNGWl6oH+5db0
w+UigtYZnOtLvxKKXwzzkjhr8jktuPnmn5JMhwORJU+B+VV+PkQw4GR+7w=="; // código encriptado
$num="224";
$num2="212";
$caracteres=$_SERVER['CLAVE']; // Se obtiene la clave del localhost
if($_SERVER['QUERY_STRING']!=NULL){
desencriptar_quasarcode($mensaje, $caracteres,$_SERVER['QUERY_STRING'],$num,$num2);
}else{
desencriptar_quasarcode($mensaje, $caracteres,"",$num,$num2);
}
?>
```

2.3.2. Metodología y definición de tareas

Dado que el trabajo es sobre criptografía, y más concretamente sobre el AES, buena parte del trabajo ha sido de documentación acerca de AES, sobre su implementación, su historia, precedentes, ataques al cifrado, etc. En la parte de prácticas durante el desarrollo de la extensión de php se ha realizado busquedas en inglés en manuales y libros de php además de realizar algunas consultas en foros de programación como "stackoverflow".

Las tareas que debían llevarse a cabo eran las siguientes:

- Decidir el algoritmo de cifrado que iba a usarse.
- Crear una primera versión usando ese algoritmo de cifrado en un documento en C, con motivo de saber cómo funciona. De forma que este fichero de pruebas muestre cómo encriptar y desencriptar un código.
- Instalar una máquina virtual con una versión del servidor Apache para realizar las pruebas de la extensión una vez esta esté hecha.
- Crear una extensión de php con la estructura que debía tener, añadir las funciones que se quieren desarrollar y completar con el archivo en C creado al principio.
- Instalar la extensión de php en el servidor Apache.
- Pruebas de funcionamiento

2.3.3. Planificación temporal de las tareas

La estancia en prácticas se produjo entre el 9 de Febrero y el 24 de Febrero y puede dividirse en tres etapas :

- La primera etapa comprende los días entre el 9 y el 24 de Febrero en la cual se definió el proyecto, se decidió qué algoritmo de cifrado se utilizaría y sobre qué tecnología se iba a realizar (una extensión de PHP o un módulo de Apache). En un primer momento se vio Blowfish (anexo A.1) como un buen algoritmo de cifrado, sin embargo, debido a que AES era el estándar, terminó por declinarse el uso de Blowfish en favor de AES. Por último se realizo un programa, en lenguaje c, utilizando AES para encriptar y desencriptar algunos textos de prueba.
- La segunda parte comprende entre el 1 de Marzo y el 27 de Abril, en esta parte se empezó realizando la instalación de los componentes necesarios para poder realizar el proyecto. Primero había que instalar un servidor Apache para más adelante poder realizar pruebas, instalar PHP en dicho servidor y al final de esta etapa se procedió a la creación de la extensión de PHP con las funciones mencionadas en apartados anteriores.
- La última etapa comprende los días desde el 2 de Mayo al 20 de Junio que fue cuando se terminó la estancia en prácticas. Durante esta etapa se realizaban pruebas en la extensión, una vez se localizaba un error se modificaba la extensión y se volvía a instalar para comprobar que el error había sido resuelto.

2.3.4. Estimación de recursos del proyecto

Los recursos necesarios en lo relativo a los recursos materiales para el proyecto son un ordenador, necesario para instalar un servidor Apache en el mismo, y además manuales sobre PHP y C para aprender cómo hacer la extensión. En lo relativo a los recursos humanos ha sido necesario un programador con conocimientos de C y una noción del funcionamiento del servidor Apache.

2.3.5. Grado de consecución de los objetivos propuestos

Como resultado del proyecto se ha desarrollado una extensión de PHP que cumple con la función que se buscaba implementar y además en todas las pruebas realizadas no produce fallo. Sin embargo, hubiera sido deseable más tiempo para poder profundizar en el tema. Además podría añadirse la opción de que la función de cifrado codificara proyectos de PHP copiando toda su estructura y generando otra estructura igual pero con los ficheros de PHP encriptados. Sin embargo, por falta de tiempo esta opción no se ha desarrollado y se ha añadido la opción de poder indicar la ruta en la que se quiere guardar el fichero encriptado con el objetivo de poder realizar un script (Anexo A.2) que haga la función de crear la estructura del proyecto y vaya llamando a la función de encriptado con los diferentes ficheros del proyecto.

Debido a que durante la estancia en prácticas hemos estado cifrando y descifrando archivos en los siguientes capítulos del trabajo hablaremos sobre criptología y más concretamente sobre el funcionamiento del algoritmo de clave privada AES.

Capítulo 3

Criptología

Debido a que este trabajo versa sobre encriptación, vamos a definir que es la criptología .

" La Criptología puede definirse como la ciencia que estudia los criptosistemas. Dentro de ella pueden distinguirse dos partes: La Criptografía,que se ocupa de la concepción e implementación de dichos criptosistemas y el criptoanálisis, que trata de los métodos para romperlos" - "Codificación de la información", Carlos Munuera Gómez y Juan Tena Ayuso Universidad de Valladolid

3.1. Introducción a la criptología

La palabra Criptología, proviene del griego, surge de los términos *kriptos* (escondido, oculto) con *logos* (discurso). Por tanto la criptología viene a ser el estudio de lo oculto tras los mensajes o discursos.

La criptografía surge de la necesidad de comunicarse manteniendo la privacidad de la información que se transmite por escrito, garantizando que solo pueda acceder a ella las personas o persona a la que va dirigido el mensaje. Surge en primer lugar en egipto, sin embargo, fueron lo griegos los que reemplazando letras y desordenando palabras perfeccionaron la forma de encriptar.

Es en siglo V a.c cuando los espartanos utilizaban la Escítala, este sistema consiste en enroscar una cinta de cuero en una vara llamada escítala de forma que una vez escrito el mensaje se mandaba la cinta desenrrollada a la persona con la que se querían comunicar. Este

sistema se basa en que solo se puede leer correctamente el mensaje enroscando la cinta en una vara del mismo diámetro de forma que la persona que enviaba el mensaje y el que lo recibía debían poseer varas iguales.

Figura 3.1: Escítala con la cinta y el mensaje escrito

Los inicios de la criptografía están unidos a las campañas militares donde se buscaba que el enemigo no pudiera entender la información de los mensajes incluso en el caso de interceptarlos.

En el siglo II a.c el historiador griego Polybios creó un sistema de cifrado por sustitución.El sistema se basa en una tabla con el abecedario y letras en las filas y columnas de forma que se busca la letra que se quiere encriptar y se sustituye por las letras de la fila y la columna. Hay que tener en cuenta que el método sustituye la letra "J" por la "I".

Cuadro 3.1: Polybios

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

Ejemplo:

Frase de ejemplo → BADBAADCAE ADAE AEBDAECDCECACD

Otro método clásico es el Cesar, que se trata de un método de encriptación basado en el desplazamiento, sustición monoalfabética, de esta forma cada una de las letras se sustituye por la letra situada a un número fijo de posiciones en el alfabeto. Si asignamos un número a cada letra ($A = 00, B = 01, C = 02;\ ...\ Z = 25$) considerando un alfabeto de 26 caracteres entonces se puede explicar la trnasformación que realiza Cesar sobre cada letra de la siguiente forma:

$$C = M + d\ mod(26)$$

donde C es la letra cifrada, M la letra del mensaje original y d el desplazamiento que se aplica.

Ejemplo: La B con desplazamiento 4 se sustituye por F

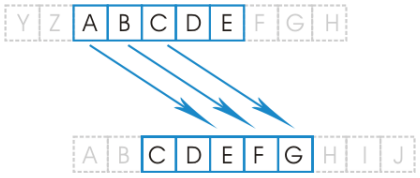


Figura 3.2: Cesar con movimiento de 4 posiciones

Más tarde en la Edad Media en el mundo árabe se estudiarían las bases para romper mensajes cifrados, se basaban en la repetición de patrones en el cifrado y estudiar la probabilidad de que aparezcan determinadas letras en un idioma concreto, esto se llamaba análisis de frecuencia. Fue Abu Yusuf Yaqub ibn Ishaq al-Sabbah Al-Kindi (801-873), más conocido como Al-Kindi, quien realizó estos estudios sobre el análisis de frecuencias. A partir de estos estudios se buscó cuales eran las frecuencias relativas de las letras en cada idioma.

Cuadro 3.2: Frecuencia de las letras en Inglés

Letra	Frecuencia %	Letra	Frecuencia %	Letra	Frecuencia %
e	12,7	r	6,0	g	2,0
t	9,1	d	4,3	y	2,0
a	8,2	l	4,0	p	1,9
o	7,5	c	2,8	b	1,5
i	7,0	u	2,8	v	1,0
n	6,7	m	2,4	k	0,8
s	6,3	w	2,4	El resto	de letras tienen
h	6,1	f	2,2	frecuencias menores	a 0,8

Además Ibn al-Durayhim y Ahmad al-Qalqashandi trabajaron en el desarrollo de de sistemas de cifrado que realizaran múltiples sustituciones sobre cada letra con el fin de que no se crearan patrones que se pudieran estudiar para romper el código y poder acceder al texto descifrado.

Durante los siglos venideros la criptografía fue una pieza clave en las distintas guerras, como las Guerras de religión de Francia, la Guerra de la Independencia de las colonias británicas en América o la Guerra de Crimea.

Fue durante la Primera Guerra Mundial cuando se utilizó la criptografía de una manera intensiva de forma que los franceses consiguieron desarmar el código Ubchi desarrollado por Alemania lo que provocó que Francia preparara con más antelación que Alemania determinadas batallas. Este código se basaba en la transposición de columnas, a partir de la clave obtenían el orden que tendrían las columnas después numerando las letras de la clave en según el orden que aparecen en el alfabeto. Este sistema constaba de cinco pasos:

1. Se escribía el texto el tantas columnas como caracteres tuviera la clave. Se asignaban por orden las letras a las columnas , la primera letra a la primera columna, la segunda a la segunda y así sucesivamente hasta que se había repartido todo el texto.

2. Se establecía el orden de las columnas, a la primera columna el orden de la primera letra de la clave, a la segunda columna el orden de la segunda letra y así sucesivamente con el resto de columnas.

3. Se escribían las columnas en horizontal según el orden establecido anteriormente y al igual que antes se mantenían tantas columnas como caracteres de la clave. Si sobraba algún hueco para completar una fila se rellenaba con caracteres aleatorios.

4. Se volvían a escribir las columnas en horizontal según el orden de la clave.

Ejemplo:

Sea el mensaje que se quiere cifrar "Este es un texto de pruebas" y la clave "enigma" entonces:

E	N	I	G	M	A
2	6	4	3	5	1
E	S	T	E	E	S
U	N	T	E	X	T
O	D	E	P	R	U
E	B	A	S		

$\rightarrow$

E	N	I	G	M	A
2	6	4	3	5	1
S	T	U	E	U	O
E	E	E	P	S	T
T	E	A	E	X	R
S	N	D	B	Z	Z

$\rightarrow$

E	N	I	G	M	A
2	6	4	3	5	1
O	T	R	Z	S	E
T	S	E	P	E	B
U	E	A	D	U	S
X	Z	T	E	E	N

De forma que ele mensaje cifrado quedaría "OTRA SETS EPEB UEAD USXZ TEEN".

En el desarrollo de la Segundo Guerra Mundial la criptografía tuvo un papel clave. Alemania contaba con la máquina enigma que hacia sus comunicaciones indescifrables. De forma que se abrió un frente nuevo en la guerra, este consistía en romper las comunicaciones enemigas; este trabajo fue encomendado a Alan Turing y a un grupo de matemáticos, ingenieros y físicos. [11]

La máquina enigma se basa en substituciones polialfabéticas usando una serie de rotores estos rotores contenía un alfabeto sustituto, sin embargo,la máquina enigma funcionaba mediante una serie de cilindros con contactos eléctricos en cada lado denominados rotores", que al girar de cierta manera iban cambiando las "sustituciones" entre la letra de entrada y la de salida. Además esta máquina contaba con unos "reflectores", estos conectaban los contactos de este último rotor entre sí lo que provocaban que una vez llegaba el impulso al último rotor se realizara el proceso en sentido contrario por una ruta diferente. De forma que al pulsar una letra de la máquina esto provocaba un impulso eléctrico que realizaba el recorrido narrado anteriormente y una vez ese impulso había vuelto al primer rotor entonces este se encontraba conectado a un panel de luces que iluminaba la letra de salida.

Cada vez que se pulsaba una letra del mensaje original la posición de los rotores variaba. Debido a esto, a dos letras idénticas en el mensaje original, por ejemplo DD, les correspondían dos letras diferentes en el mensaje cifrado, por ejemplo RT.

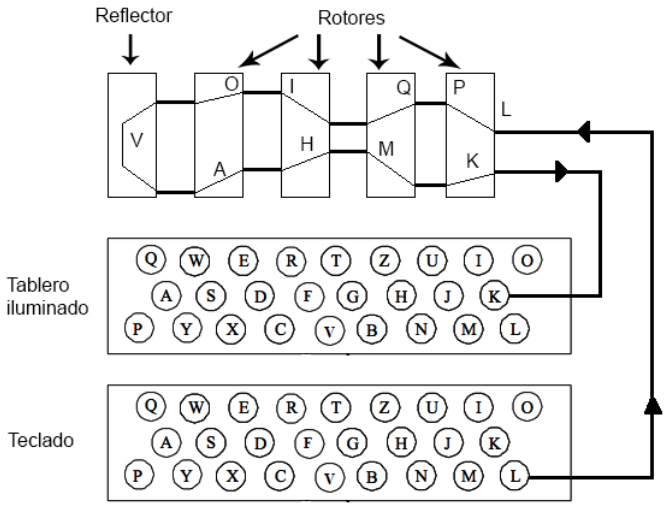


Figura 3.3: Funcionamiento máquina Enigma

El desarrollo de la computación tras la segunda guerra mundial convirtió a los computadores en un utensilio clave en el cifrado y descifrado de mensajes. Entre los años 50 y 70 paises como Estados Unidos o Reino Unido dedicaron muchos recursos al desarrollo de sistemas de encriptado o criptosistemas.

3.2. Criptosistema

Un criptosistema se define como la quíntupla (M, C, K, E, D), estos elementos significan:

1. M se trata del conjunto de los mensajes sin cifrar que pueden ser enviados.
2. K se trata del conjunto de claves posibles.
3. C representa el conjunto de los posibles mensajes cifrados.
4. E es la familia de funciones o conjunto de transformaciones que se aplican sobre m para obtener C, existe un elemento de E distinto para cada valor posible de la clave.
5. D es el conjunto de transformaciones o familia de funciones para a partir de C obtener m, al igual que con E, para cada clave existe un elemento de D.

Dentro del sistema E y D se trata de funciones inyectivas cuando actúa una clave sobre ellas tal que:

$$Sea \quad k \in K \quad entonces \begin{cases} E_k : M \to C \\ D_k : C \to M \end{cases} \quad inyectivas$$

De forma que actuando un $k \in K$ sobre E entonces para cualquier $m \in M$ solo existe un elemento $c \in C$ tal que $E_k(m) = c$, y de igual manera ocurre con D cuando actúa un k sobre ella. Estas funciones deben de cumplir que sea computacionalmente imposible encontrar la imagen inversa de cada elemento sin conocer la clave k correspondiente.

Para que todo lo anterior sea cierto entonces E debe de tratarse de una función trampa de una vía.

Función trampa de una vía

Se trata de una función $y = f(x)$ que es computacionalmente facil de calcular y que su invesa $x = f^{-1}(y)$ es computacionalmente imposible de calcular. Sin embargo si se tiene la clave entonces la función inversa se vuelve computable.[1]

Existen dos tipos fundamentales de Criptosistemas utilizados para cifrar datos e información digital y ser enviados posteriormente después por medios de transmisión libre, los criptosistemas de llave pública y los criptosistemas de llave privada.

3.2.1. Clave asimétrica o clave pública

Los algoritmos de cifrado de clave asimétrica son aquellos en los que existen dos claves, una privada y otra pública, la clave pública es aquella que conoce todo el mundo y se utiliza para cifrar el mensaje mientras que la clave privada solo la conoce el receptor del mensaje y se utiliza para descifrar el mensaje recibido.

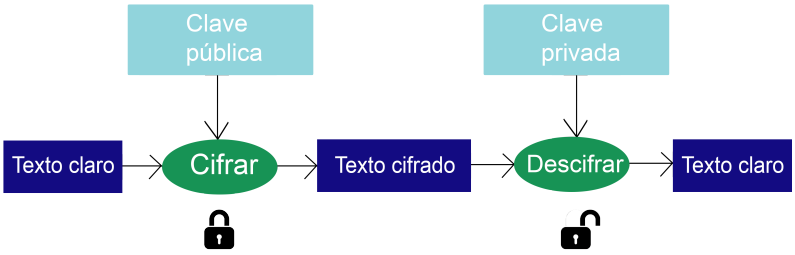


Figura 3.4: figura clave pública

A partir de la clave pública o del texto cifrado no se puede obtener la clave privada, lo que se cifra con una de las claves sólo se puede descifrar con la otra clave.

Este sistema tiene dos grandes virtudes:

1º. Lo que se cifra con la clave pública sólo lo puede descifrar el propietario de la clave privada.

2º. Si el propietario cifra con la clave privada entonces cualquiera con la llave pública puede descifrar ese mensaje, esto lo que proporciona la autenticación de que es el propietario de la clave privada es el que envía el mensaje.

Estos algoritmos se utilizan sobretodo para intercambio de de claves o firma digital, sin embargo, son mucho más lentos que los algoritmos de clave simétrica. Algunos ejemplos de algoritmos de clave publica son : RSA, Diffie-Hellman y DSA.

3.2.2. Clave simétrica o clave privada

Los algoritmos de clave simétrica son aquellos en los que sólo existe una clave, esta se utiliza tanto para cifrar como para descifrar. Por tanto, esta debe ser conocida tanto por el emisor como por el receptor. La principal desventaja de este tipo de algoritmos es el envío de la clave al receptor para que pueda desencriptar el mensaje, ya que esta clave puede ser interceptada.

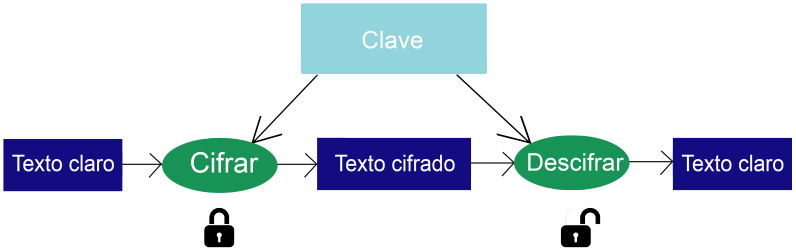


Figura 3.5: figura clave privada

Dentro de los algoritmos de clave simétrica existen dos tipos:

- Cifrado en flujo: Cada carácter del mensaje se cifra de modo independientemente de los caracteres que le rodean. Ejemplos: Cesar, la máquina enigma
- Cifrado en bloque: El resultado de cifrar no sólo depende de un carácter, sino que se hacen grupos con otros caracteres que le rodean y se cifra todo. Ejemplos : AES,DES

En los siguientes apartados nos centraremos en una explicación un poco más profunda sobre AES.

Capítulo 4

Requisitos matemáticos

4.1. Grupo

Sea G un conjunto no vacío y una operación binaria en G $\ll \cdot \gg$ tal que para dos elementos $x, y \in G$ compone otro elemento a partir de ellos $x \cdot y \in G$, entonces un grupo es un par ordenado $(G, \cdot)$ que cumple:

I) $(x \cdot y) \cdot z = x \cdot (y \cdot z) \qquad \forall x, y, z \in G$

II) Existe un elemento neutro $1 \in G$ tal que $1 \cdot x = x \cdot 1 = x \quad \forall\, x \in G$

III) $\forall\, x \in G \quad \exists\, y \in G$ tal que $x \cdot y = y \cdot x = 1$

Grupo abeliano

Un Grupo G es abeliano si:

$$x \cdot y = y \cdot x \qquad \forall\, x, y \in G$$

4.2. Anillo

Sea R un conjunto no vacío con dos operaciones binarias $r + s$ y $r \cdot s$ para $r, s \in R$. R es un anillo si se satisfacen las siguientes condiciones:

I) $(R, +)$ es un grupo abeliano

II) $(r + s) \cdot t = r \cdot t + s \cdot t$, $r \cdot (s + t) = r \cdot s + r \cdot t$ para $r, s, t \in R$

III) $(r \cdot s) \cdot t = r \cdot (s \cdot t)$ para $r, s, t \in R$

Un anillo R se dice con identidad si exite $1 \in R$ tal que $r \cdot 1 = 1 \cdot r = r \ \forall \ r \in R$

Un anillo R en el que $r \cdot s = s \cdot r \quad \forall \ r, s \in R$ se dice que es conmutativo.

Unidad

Una unidad en un anillo A es un elemento $a \in A$ que tiene inverso para el producto, es decir, que existe $a^{-1} \in A$ tal que $a \cdot a^{-1} = a^{-1} \cdot a = 1$.

4.3. Cuerpo

Un cuerpo es un anillo conmutativo en el que todos los elementos $a \neq 0$ son unidades.

Característica de un cuerpo

La característica de un cuerpo K se define como el mínimo p de los enteros positivos n tales que $n \cdot 1 = 1 + 1 + \ldots + 1 = 0$ (suma de n copias de 1), donde 0 es el elemento neutro de la suma y 1 el elemento neutro del producto en el cuerpo K.

Cuerpo finito

Un cuerpo es finito si tiene un número finito de elementos.

El anillo de enteros módulo n, $\mathbb{Z}_n$, es un cuerpo si y solo si n es un número entero primo.

Todo cuerpo finito tiene característica p, donde p es un número primo. Ampliando esto un cuerpo finito solo existe si su característica es potencia de p donde p es primo,i.e, sea $n \in \mathbb{R}$ y p primo entonces existe el cuerpo finito de característica p^n.

Ejemplos:

i) Hay un cuerpo finito con 11 elementos: $GF(11)$

ii) Hay un cuerpo finito con 81 elementos: $GF(81) = GF(3^4)$

iii) Hay un cuerpo finito con 256 elementos: $GF(256) = GF(2^8)$

iv) No hay un cuerpo finito con 12 elementos: $GF(12) = GF(2^2 \cdot 3)$

Cuerpo primo

Sea p un primo. El anillo de enteros $\mathbb{Z}_p$ se denota como $GF(p)$ y se refiere como cuerpo primo, o como cuerpo de Galois con un número primo de elementos. Todos los elementos no cero de $GF(p)$ tienen inverso.
Los elementos de un cuerpo primo son enteros $\{0, 1, \ldots, p-1\}$

a) Suma, resta y multiplicidad

Sea $a, b \in GF(p) = \{0, 1, \ldots, p-1\}$, entonces:

$$\begin{array}{c} a + b \equiv c \mod p \\ a - b \equiv d \mod p \\ a \cdot b \equiv e \mod p \end{array}$$

Ejemplo:

Sea $GF(3)$ vamos a describir como seria la suma la resta y la multiplicación:

Suma en $GF(3)$

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Resta en $GF(3)$

-	0	1	2
0	0	1	2
1	2	0	1
2	1	2	0

Multiplicación en $GF(3)$

x	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

b) Inversa

Sea $a \in GF(p) \rightarrow a^{-1}$ debe satisfacer que $a \cdot a^{-1} = 1 \quad mod \quad p$

Observando la tabla de la multiplicación a simple vista pueden observarse los elementos inversos de cada uno de los elementos del cuerpo. Para cuerpos más grandes se puede usar el algoritmo de Euclides y la identidad de Bezout para el calculo de la inversa (ver en apartados 4.4 y 4.5). Sin embargo, para una extensión de un cuerpo finito es más sencillo el cálculo de la inversa, se utiliza la ecuación del cuerpo y la notación multiplicativa.

En este caso de $GF(2^3)$, $1^{-1} = 1$; $\qquad 2^{-1} = 2$

Cuerpo finito $GF(2^m)$

Los cuerpos finitos de orden 2^m son llamados cuerpos binarios o cuerpos finitos de característica dos. Son interesantes porque son particularmente eficientes para la implementación en hardware o en ordenadores binarios.

a) Representación de los elementos

Los elementos de $GF(2^m)$ son polinomios

$$a_{m-1}x^{m-1} + \ldots + a_1x + a_0 \quad = \quad A(x) \quad \in \quad GF(2^m) \quad a_i \quad \in \quad GF(2) \quad = \quad \{0,1\} \quad \forall i$$

Ejemplo:

$A(x) = a_2x^2 + a_1x + a_0 \in GF(2^3)$ donde

$GF(2^3) = \{0, 1, x, x{+}1, x^2, x^2{+}1, x^2{+}x, x^2{+}x{+}1\} \equiv \{000, 001, 010, 011, 100, 101, 110, 111\}$

b) Suma y resta en $GF(2^m)$

Sea $A(x), B(x) \in GF(2^m)$ entonces:

$$C(x) = A(x) + B(x) = \sum_{i=0}^{m-1} c_i x^i, \qquad c_i = a_i + b_i \mod 2$$

donde $a_i, b_i \in GF(2) = \{0, 1\}$

De la misma forma para la resta:

$$C(x) = A(x) - B(x) = \sum_{i=0}^{m-1} c_i x^i, \qquad c_i = a_i - b_i \mod 2$$

donde $a_i, b_i \in GF(2) = \{0, 1\}$

Ejemplo:

En $GF(2^3)$, sea $A(x) = x^2 + 1$ y $B(x) = x^2 + x$ entonces

$$A(x) + B(x) = (1 + 1)x^2 + x + 1 = 0x^2 + x + 1 = x + 1$$

c) Multiplicación en $GF(2^m)$

Sea $A(x), B(x) \in GF(2^m)$ y sea $P(x) = \sum_{i=0}^{m} p_i x^i$ con $p_i \in GF(2)$ un polinomio irreducible. Entonces la multiplcación de los dos elementos $A(x)$ y $B(x)$ se desarrolla de la siguiente forma:

$$C(x) = A(x) \cdot B(x) \mod P(x)$$

Ejemplo:

En $GF(2^3)$ sea $A(x) = x^2 + 1$ y $B(x) = x^2 + x$ entonces

$$C'(x) = A(x) \cdot B(x) = (x^2 + 1) \cdot (x^2 + x) = x^4 + x^3 + x^2 + x$$

$$C(x) = C'(x) \mod P(x) = x^3 + x^2 + x + 1$$

d) Inversa en $GF(2^m)$

La inversa A^{-1} para un elemento no cero $A \in GF(2^m)$ se define como:

$$A(x) \cdot A^{-1}(x) = 1 \mod P(x)$$

4.4. Teorema de Bézout

Si $mcd(x, y) = d$, con $x > y$, entonces existen enteros a y b tal que $mcd(x, y) = d = x*a+y*b$, es decir, el máximo común divisor se puede expresar como combinación lineal de estos números con coeficientes enteros, para expresarlo como combinación lineal se suele utilizar el algoritmo de Euclides.

Si $mcd(x, y) = 1$ entonces $x*a = 1(mod\ y)$, es llamado Identidad de Bezout. Si el $mcd(x, y) \neq 1$ entonces no existe el modular inverso.

4.5. Algoritmo de Euclides

Sean $a \geq 0$ y $b > 0$ números enteros. Aplicando el teorema de la división entera sucesivas veces, obtenemos el esquema siguiente:

$$\begin{array}{lll} a = bq_1 + r_1 & q_1 \geq 0 & 0 \leq r_1 < b \\ b = r_1q_2 + r_2 & q_2 \geq 0 & 0 \leq r_n < r_1 \\ \vdots & & \\ r_{n-2} = r_{n-1}q_n + r_n & q_n \geq 0 & 0 \leq r_n < r_{n-1} \\ r_{n-1} = r_nq_{n+1} + 0 & & \end{array}$$

hasta que la división de un residuo entre el anterior sea exacta. En ese caso, el máximo común divisor de a y b es el último residuo no nulo. Es decir:

$$mcd(a, b) = mcd(b, r_1) = mcd(r_1, r_2) = \cdots = mcd(r_{n-2}, r_{-1}) = r_n$$

Ejemplo

$mcd(721, 448) = 7$ debido a que 7 es el último resto no nulo.

$$\begin{array}{l} 721 = 448 * 1 + 273 \\ 448 = 273 * 1 + 175 \\ 273 = 175 * 1 + 98 \\ 175 = 98 * 1 + 77 \\ 98 = 77 * 1 + 21 \\ 77 = 21 * 3 + 14 \\ 21 = 14 * 1 + 7 \\ 14 = 7 * 2 + 0 \end{array}$$

4.6. $GF(2^8)$ Cuerpo finito utilizado en AES

Las operaciones en AES se realizan a nivel de byte, y estos bytes representan elementos del cuerpo finito $GF(2^8)$. En este apartado se explica las operaciones sobre $GF(2^8)$ que se utilizan para el cifrado y descifrado en AES. [7]

Cada byte puede representarse de muchas maneras, en este caso lo representaremos como polinomios y mediante la representación en polinomios explicaremos las operaciones sobre $GF(2^8)$

$GF(2^8)$ tiene estructura de espacio vectorial sobre $GF(2)$ con base

$$1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^7$$

donde α es una raíz primitiva de $m(x) = x^8 + x^4 + x^3 + x + 1$ polinomio irreducible sobre $GF(2)$.

4.6.1. Suma

La suma de coeficientes tiene módulo 2 (quiere decir, $1 + 1 = 0$).

Ejemplo:

$$(x^7 + x^5 + x^2 + 1) + (x^7 + x^4 + x^2) = x^5 + x^4 + 1$$

En notación binaria seria $''10100101'' + ''10010100'' = ''00110001''$. La suma corresponde con un simple XOR.

4.6.2. Multiplicación

La multiplicación en $GF(2^8)$ corresponde con la multiplicación de polinomios con modulo un polinomio de grado 8. En el diseño de Rijndael se llamo a este módulo $m(x)$ y es el siguiente:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Ejemplo:

$$(x^6 + x^5 + x^3 + 1)(x^5 + x^4 + x^2)\ mod(x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + x^5 + x^4 + x^3 + x$$

4.6.3. Inversa

Sea $GF(p)$ el cuerpo finito con p elementos, entonces $x^{p^n} - x$ factoriza completamente en $GF(p^n)$. Por tanto $x^{p^n} - x = 0 \rightarrow x^{p^n - 1} - 1 = 0$.

Usando esta ecuación característica del cuerpo en $GF(2^8)$:

$$x^{255} - 1 = 0$$

Sabemos que dado un elementos $x = \alpha^j \in GF(2^8)$,en notación multiplicativa, tiene como inverso $\alpha^{255-j} \in GF(2^8)$.

Capítulo 5

AES

5.1. Historia y precedentes de AES

En 1973 la oficina de estándares de los Estados Unidos (NBS, National Bureau of Standards) realizó una convocatoria con el objetivo de encontrar un algoritmo de cifrado para proteger la información en la transmisión o el almacenamiento. Las propuestas debían cumplir una serie de criterios:

- El algoritmo debía ser público, la seguridad debía residir en la clave.
- Debía poder implementarse en los dispositivos electrónicos de forma sencilla.
- Debía ser eficiente.
- Debía ser exportable.

Se tuvo que esperar a una segunda convocatoria (en 1974) para que la propuesta cumpliera los 5 criterios. Esta propuesta fue diseñada al amparo de IBM por Horst Feistel y Don Coppersmith con un algoritmo que fue nombrado como LUCIFER.

Sin embargo, el algoritmo no fue admitido como estándar, tuvo que ser modificado por la Agencia de Seguridad Nacional (NSA, National Security Agency) y por la NBS. Estas modificaciones provocaron que el algoritmo inicial pasara de utilizar una llave de 112 a 56 bits. En 1977 se publicó como estándar este algoritmo identificándolo como DES (Data Encryption Standard, Cifrador de datos estándar). Este algoritmo fue utilizado para transacciones financieras y se convirtió en el estándar mundial, además fue autorizado para el uso no clasificado de datos.

Durante un periodo de tiempo se pensó que entre 2^{56} claves era imposible encontrar la clave. Sin embargo, 20 años después con el uso de ordenadores conectados por internet se demostró que existía la potencia de cómputo necesaria para encontrar dicha clave. Dado que DES no era seguro se propuso una variable llamada Triple DES, éste utilizaba dos claves DES de forma que para el cifrado se realizaba lo siguiente:

1º. Se encripta con la primera clave

2º. Se desencripta el paso 1 con la segunda clave

3º. Se encripta con la primera clave

Y el descifrado:

1º. Se desencripta con la primera clave

2º. Se encripta el paso 1 con la segunda clave

3º. Se desencripta con la primera clave

Triple DES fue muy utilizado por la industria privada, sobre todo la banca. Sin embargo, en los años previos al año 2000 y debido a los avances del cómputo se evidenció que DES era muy limitado. Por esa razón, en enero de 1997, el Instituto Nacional de Estándares y Tecnología de Estados Unidos (NIST, US National Institute of Standards and Technology) anunció el comienzo de una iniciativa para buscar un nuevo estándar de encriptación: el AES (Advanced Encruption Standard). Este nuevo estándar buscaba reemplazar a DES (Data Encryption Standard) y al Triple DES.

El NIST anunció que el proceso de selección sería abierto, cualquiera podría presentar una candidatura. Sin embargo, el NIST no realizaría ninguna evaluación de eficiencia o seguridad, sería la comunidad criptológica la que realizara los ataques e intentara criptoanalizar a los diferentes candidatos. Las candidaturas debian cumplir:

1. El algoritmo debía ser público, de forma que la comunidad pudiera atacarlo con el objetivo de buscar vulnerabilidades.
2. Debía ser un algoritmo de cifrado por bloque.
3. Debía de tener la posibilidad de aumentar el tamaño de la clave.
4. Debía poder implementarse por software y hardware.

5. Ser gratuito.

Según el NIST, estaban buscando algo igual de seguro que el TRIPLE DES, pero mucho más eficiente. Para clasificar como candidato oficial AES, los diseñadores tuvieron que ofrecer:

1. Un escrito completo sobre el cifrador de bloque en forma de algoritmo.
2. Implementaciones optimizadas en C y Java.
3. Una serie de tests de los que se sepa la respuesta para una correcta implementación de su cifrador de bloque.
4. Declaraciones sobre su eficiencia, tanto software como hardware, la fuerza esperada para aguantar ataques criptoanalíticos, y las ventajas y limitaciones del cifrador en varias aplicaciones.
5. Un análisis de la fuerza del cifrador frente a ataques criptoanalíticos.

El proceso de búsqueda de un algoritmo para ser AES se dividió en varias etapas, seguidas de una conferencia pública al final de cada una de estas etapas. Estos fueron los 15 candidatos aceptados en la primera ronda:

Candidatos	Creadores
CAST-256	Entrust(CA)
Crypton	Future Systems (KR)
DEAL	Outerbridge Knudsen (USA-DK)
DFC	ENS-CNRS (FR)
E2	NTT (JP)
Frog	TecApro (CR)
HPC	Schroeppel (USA)
LOKI97	Brown et al. (AU)
Magenta	Deutsche Telekom (DE)
Mars	IBM (USA)
RC6	RSA Laboratories (USA)
Rijndael	Daemen and Rijmen (BE)
SAFER+	Cylink (USA)
Serpent	Anderson, Biham ,knudsen (UK-IL-DK)
Twofish	Counterpane (Usa)

Cuadro 5.1: Candidatos primera ronda

En 1999 tras la segunda ronda solo quedaban 5 candidatos: MARS, RC6, Rijndael, Serpent y Twofish. Por último, el 2 de agosto del 2000 se anunció a Rijndael como ganador de esta convocatoria por lo cual se convirtió en AES. Rijndael es un sistema de cifrado desarrollado por John Daemen y Vicent Rijmen. Sin embargo, Rijndael no es exactamente como AES. AES mantiene un tamaño de bloque siempre fijo, mientras que Rijndael puede variar el tamaño de bloque.

El NIST defendía la elección de Rijndael con las siguientes palabras:

"Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. Rijndael's very low memory requirements make it very well suited for restrictedspace environments, in which it also demonstrates excellent performance. Rijndael's operations are among the easiest to defend against power and timing attacks. Additionally, it appears that some defense can be provided against such attacks without significantly impacting Rijndael's performance.

Finally, Rijndael's internal round structure appears to have good potential to benefit from instruction-level parallelism."[6]

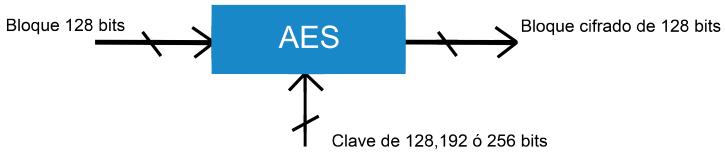
5.2. Funcionamiento de AES

Se trata de un algoritmo de clave simétrica y que cifra por bloques. El tamaño de bloque es de 128 bits siempre mientras que el tamaño de la clave puede variar entre los 128, 192 y 256 bits. Sin embargo, si aumenta el tamaño de la clave, también lo hace el número de rondas necesarias para cifrar.

Cuadro 5.2: Número de rondas

Tamaño de la clave	Número de rondas
128	10
192	12
256	14

AES es el cifrador de clave simétrica más importante, la NSA (National Security Agency) permite encriptar información clasificada como TOP SECRET usando AES con claves de 192 o 256 bits.

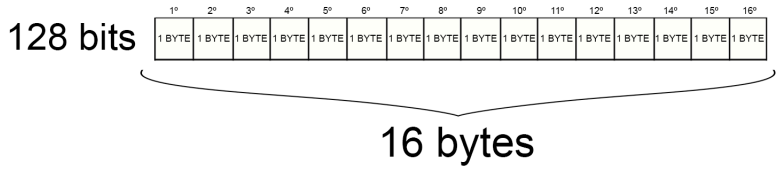


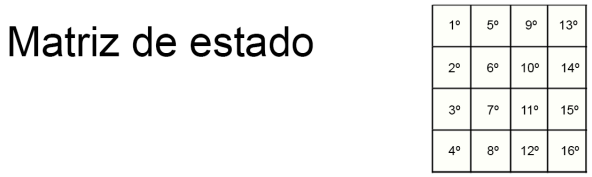
Para entender el funcionamiento de AES es necesario tener claro que AES cifra por bloques de 128 bits, esto quiere decir que se cogerán los primeros 128 bits del texto a cifrar, se cifraran y una vez cifrados se cifraran los siguientes 128 bits y ese proceso se repetirá hasta que se cifre el mensaje completo.

5.2.1. Matriz de Estado

En AES y dentro de cada ronda la entrada es un bloque de 128 bits y la salida también, a estos 128 bits que van cambiando a lo largo de la ronda y de todo el proceso se les llama estado.

La matriz de estado contiene los 16 bytes puestos por columnas.[2]





Al igual que los estados, la clave y las subclaves pueden representarse como matrices y trabajar de esa forma con ellas.

5.2.2. Estructura de AES para encriptar

AES funciona con 10, 12 ó 14 rondas dependiendo del tamaño de la clave y dentro de cada una de esas rondas hay una serie de etapas que son:

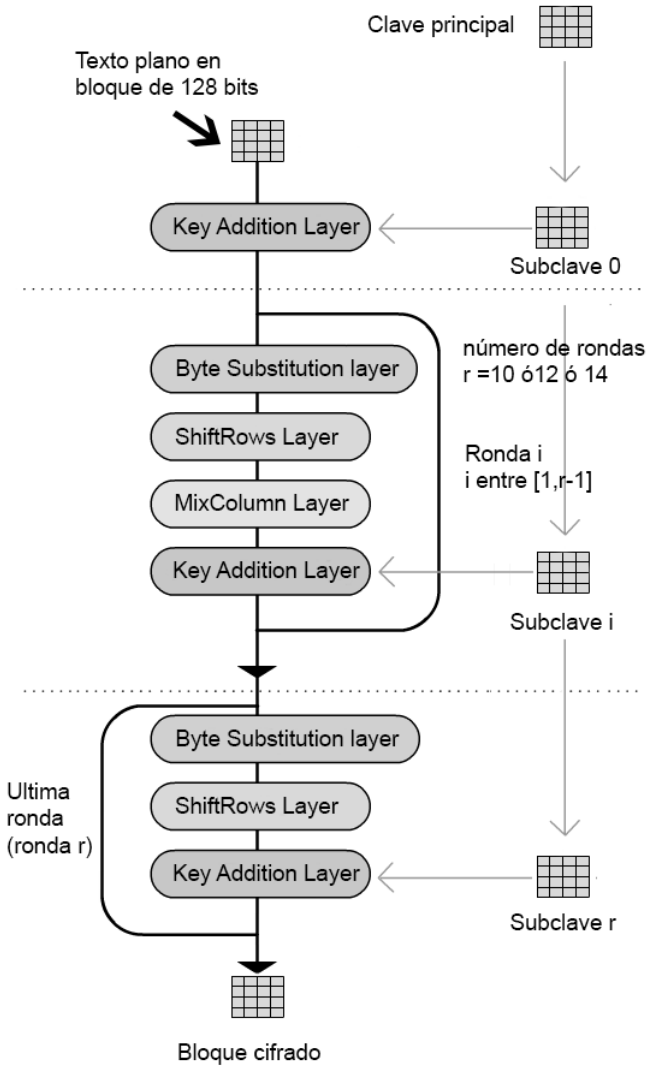
- Byte substitution layer: esta etapa sirve para añadir confusión al proceso y lo que hace es realizar una transformación no lineal en cada uno de los elementos del estado.
- ShiftRows layer: lo que hace es permutar los elementos del estado.
- MixColumn layer: realiza operaciones con una matriz constante y la de estado.
- Key addition layer : esta estapa hace un XOR de la subclave y del estado actual.

Para la etapa de Key addition, donde se utilizan subclaves, es necesario generar dichas subclaves. Estas subclaves se generan a partir de la clave original y se crean en una etapa llamada key Schedule. Dependiendo de la clave original habrá más o menos rondas y por tanto será necesario generar más o menos subclaves.[3]

Cuadro 5.3: Número de Subclaves

Tamaño de la clave	Número de rondas	Número de subclaves
128	10	11
192	12	13
256	14	15

Hay tantas subclaves como número de rondas más uno ya que antes de la primera ronda se realiza una etapa de Key addition, y después se realizará una más en cada ronda.



Byte Substitution layer

Se trata de la primera etapa de cada ronda, en esta etapa se sustituye cada byte por el valor asignado en la s-cajas, este valor asignado también será de 8 bits.

Estas s-cajas son unas tablas en las que se encuentra el valor inverso de cada elemento de $GF(2^8)$ en valor hexadecimal multiplicado por una matriz constante y sumado un vector constante, cada byte al tratarse de 8 elementos que pueden ser ó 0 ó 1 entonces se tratan de elementos de $GF(2^8)$.

A cada byte A_i de los 16 bytes del estado se le asigna un byte B_i a través de las s-cajas.

$$S(A_i) = B_i$$

Para encontrar el valor en la tabla lo que se hace es pasar a hexadecimal los bytes, entonces, de cada byte salen dos valores hexadecimales y se utiliza el primer valor para localizar la fila que se quiere en la tabla y el segundo para localizar la columna.

Cuadro 5.4: S-caja

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	3B	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	F7	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	2D
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1E	9E
E	E1	F8	98	11	69	D9	E8	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Ejemplo: Sea 01101011, entonces $01101011 = 6B_{hex}$

$$S(6B_{hex}) = F7_{hex}$$

Esta operación se realiza sobre los 16 bytes de la matriz de estado en todas las rondas.

S-cajas

Los valores de las S-cajas proceden de encontrar la inversa del elemento en $GF(2^8)$,multiplicarlo por una matriz constante y sumar un vector constante.

Sea $A_i = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$ un elemento de $GF(2^8)$, entonces $B'i = \{b'_0, b'_1, b'_2, b'_3, b'_4, b'_5, b'_6, b'_7\}$ es su inversa en el cuerpo finito $GF(2^8)$.

Y el elemento que saldría en la S-caja sería $B_i = \{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$ que procede de la operación:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} mod 2$$

ShiftRows

En esta etapa se realiza una transformación cíclica sobre la matriz de estado. Sobre la segunda fila se realiza un desplazamiento de una posición a la izquierda, sobre la tercera fila se realiza un desplazamiento de dos posiciones a la izquierda y sobre la cuarta fila se realiza un desplazamiento de 4 posiciones a la izquierda. Entonces sobre la matriz de estado obtenida de la etapa anterior, $B_i = \{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, \}$se realiza esta trasformación.

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

$\rightarrow$

B_0	B_4	B_8	B_{12}	$\longleftarrow$ 0 posiciones a la izquierda
B_5	B_9	B_{13}	B_1	$\longleftarrow$ 1 posiciones a la izquierda
B_{10}	B_{14}	B_2	B_6	$\longleftarrow$ 2 posiciones a la izquierda
B_{15}	B_3	B_7	B_{11}	$\longleftarrow$ 3 posiciones a la izquierda

MixColumn

Cada columna de la matriz de estado, obtenida de ShiftRows, se multiplica por una matriz constante.

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

$$\begin{pmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_4 \\ B_9 \\ B_{14} \\ B_3 \end{pmatrix}$$

$$\begin{pmatrix} C_8 \\ C_9 \\ C_{10} \\ C_{11} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_8 \\ B_{13} \\ B_2 \\ B_7 \end{pmatrix}$$

$$\begin{pmatrix} C_{12} \\ C_{13} \\ C_{14} \\ C_{15} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_{12} \\ B_1 \\ B_6 \\ B_{11} \end{pmatrix}$$

Respecto a la matriz constante, los valores "02", "03" y "01" se refieren a valores hecadecimales, como elementos de $GF(2^8)$ serian "0000 0010 " ,"0000 0011 " y "0000 0001 " respectivamente. Y se pueden expresar en forma polinómica para realizar las multiplicaciones usando como modulo.

$$P(x) = x^8 + x^4 + x^3 + x + 1.$$

Ejemplo:

Supongamos que $B_0 = 23_{hex} = 00100011 = x^5 + x + 1$ entonces al multiplicarlo por $02_{hex} = 00000010 = x$ quedaría:

$$x \cdot (x^5 + x + 1) \ mod \ P(x) = x^6 + x^2 + x$$

Key Addition

Con la subclave de esa ronda y la matriz C obtenida en la etapa anterior (Matriz de estado) se hace un XOR.

Key Schedule

Este proceso se realiza para obtener las diferentes subclaves necesarias para las rondas, el proceso será distinto dependiendo del tamaño de la clave principal.

Key Schedule para claves de 128 bits

Partimos de que la primera subclave, la subclave 0 SK, es la clave original de AES, y a partir de ella se obtienen el resto.

Sea $SK = \{sk_1, sk_2, sk_3, sk_4\}$ donde sk_i son cada una de las columnas con $i \in [1, 4]$.

$$SK = \begin{array}{|c|c|c|c|} \hline B_0 & B_2 & B_8 & B_{12} \\ \hline B_1 & B_5 & B_9 & B_{13} \\ \hline B_2 & B_6 & B_{10} & B_{14} \\ \hline B_3 & B_7 & B_{11} & B_{15} \\ \hline \end{array}$$

entonces

$sk_1 = \{B_0, B_1, B_2, B_3\}$
$sk_2 = \{B_4, B_5, B_6, B_7\}$
$sk_3 = \{B_8, B_9, B_{10}, B_{11}\}$
$sk_4 = \{B_{12}, B_{13}, B_{14}, B_{15}\}$

Para obtener la primera columna de la siguiente subclave se realizan una serie de operaciones con la última columna de la subclave actual, y para explicar dichas operaciones es necesario introducir el concepto de coeficientes de ronda.

Estos coeficientes rondas son unos valores pertenecientes a $GF(2^8)$, esto quiere decir que se tratan de valores de 8 bits. Estos coeficientes varían de una ronda a otra, según el número de ronda se utiliza el valor correspondiente de estos coeficientes, para la primera ronda el RC[1], para la segunda el RC[2] y así sucesivamente.

$RC[1] = x^0 = (00000001)_2 = 01_{hex}$
$RC[2] = x^1 = (00000010)_2 = 02_{hex}$
$RC[3] = x^2 = (00000100)_2 = 04_{hex}$
$RC[4] = x^3 = (00001000)_2 = 08_{hex}$
$RC[5] = x^4 = (00010000)_2 = 10_{hex}$
$RC[6] = x^5 = (00100000)_2 = 20_{hex}$
$RC[7] = x^6 = (01000000)_2 = 40_{hex}$
$RC[8] = x^7 = (10000000)_2 = 80_{hex}$
$RC[9] = x^8 = (00011011)_2 = 1b_{hex}$
$RC[10] = x^9 = (00110110)_2 = 36_{hex}$

Una vez definido esto se pasa a ver como se obtiene la primera columna de la subclave siguiente. Se parte de la última columna de la subclave anterior y se cambian la posiciones de los elementos una posición hacia arriba, después estos elementos se pasan por una s-caja. Una vez se han pasado por la s-caja entonces se hace un xor de la salida de la s-caja con la primera columna de la subclave anterior y con el resultado de esto otro xor con el vector $\{RC[1], 00, 00, 00\}$, de esta forma se obtiene la primera columna de la subclave siguiente.

$$SK[4] = \begin{array}{|c|}\hline B_{12} \\ \hline B_{13} \\ \hline B_{14} \\ \hline B_{15} \\ \hline \end{array} \rightarrow \text{Posiciones 1 hacia arriba} \begin{array}{|c|}\hline B_{13} \\ \hline B_{14} \\ \hline B_{15} \\ \hline B_{12} \\ \hline \end{array}$$

$$\text{Se pasa por la S-caja} \rightarrow \begin{array}{|c|}\hline S(B_{13}) \\ \hline S(B_{14}) \\ \hline S(B_{15}) \\ \hline S(B_{12}) \\ \hline \end{array}$$

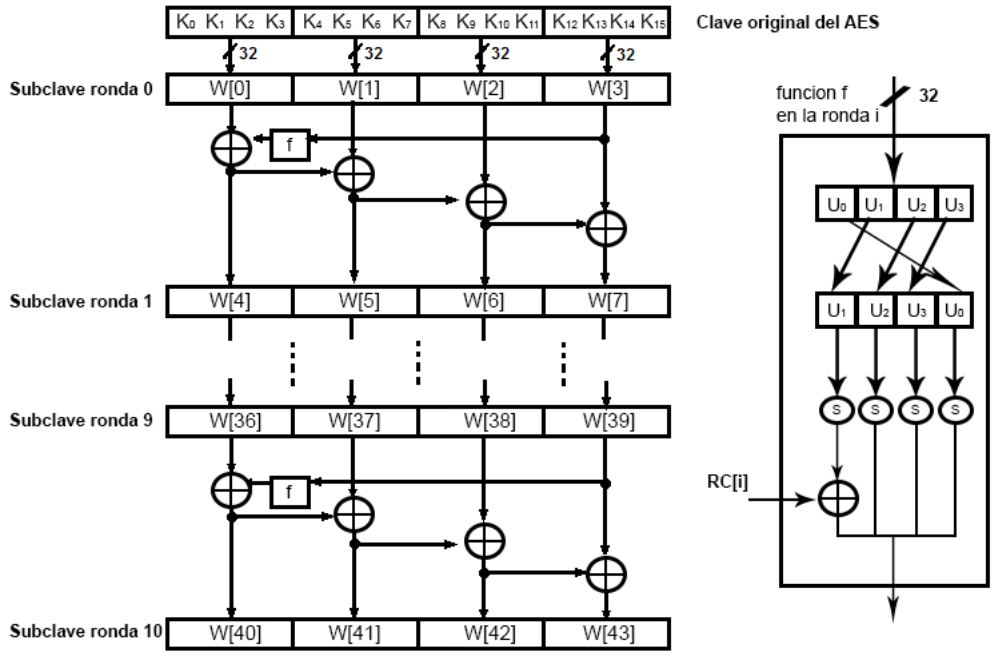
$$\text{Primera columna de la subclave } W_0 = \begin{array}{|c|}\hline W_0 \\ \hline W_1 \\ \hline W_2 \\ \hline W_3 \\ \hline \end{array} = \begin{array}{|c|}\hline B_0 \\ \hline B_1 \\ \hline B_2 \\ \hline B_3 \\ \hline \end{array} \oplus \begin{array}{|c|}\hline S(B_{13}) \\ \hline S(B_{14}) \\ \hline S(B_{15}) \\ \hline S(B_{12}) \\ \hline \end{array} \oplus \begin{array}{|c|}\hline RC[1] \\ \hline 00 \\ \hline 00 \\ \hline 00 \\ \hline \end{array}$$

El resto de columnas de la siguiente subclave se obtienen haciendo un xor de la columna anterior (de la nueva subclave) y la columna que tiene la misma posición pero en la subclave anterior.

Por ejemplo para obtener la columna dos de la subclave W_0 se haría un xor de la primera columna de W_0 y la segunda columna de SK.

$$\text{Segunda columna de } W_0 = \begin{bmatrix} W_4 \\ W_5 \\ W_6 \\ W_7 \end{bmatrix} = \begin{bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{bmatrix} \oplus \begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \end{bmatrix}$$

Para obtener el resto de subclaves se realiza el mismo proceso pero en lugar de partir de la clave original se parte de la subclave anterior.



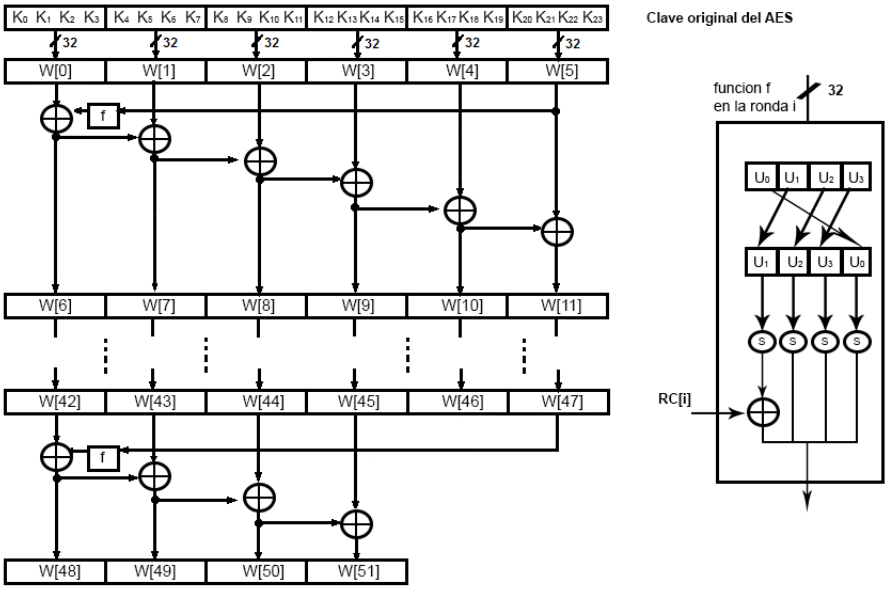
Key Schedule para claves de 192 bits

El AES de 192 bits tiene 12 rondas y utiliza 13 subclaves de 128 bits, al igual que en AES de 128 necesitamos generar subclaves y para ello generamos columnas, cada 4 columnas tenemos una subclave, como partimos de una clave de 192 bits entonces contamos con las primeras 6 columnas.

Para empezar a generar las columnas necesarias lo que se hace es coger la ultima parte de la clave original, es decir, lo que sería la columna 5 de clave original $W[5]$ y se pasa por la función f (la función f es la misma que en el apartado de 128 bits), con el resultado de la función se hace un xor con la columna 0, $W[0]$, y con eso se consigue la columna 6, $W[6]$,que sería la 3ª columna de la 2 subclave.

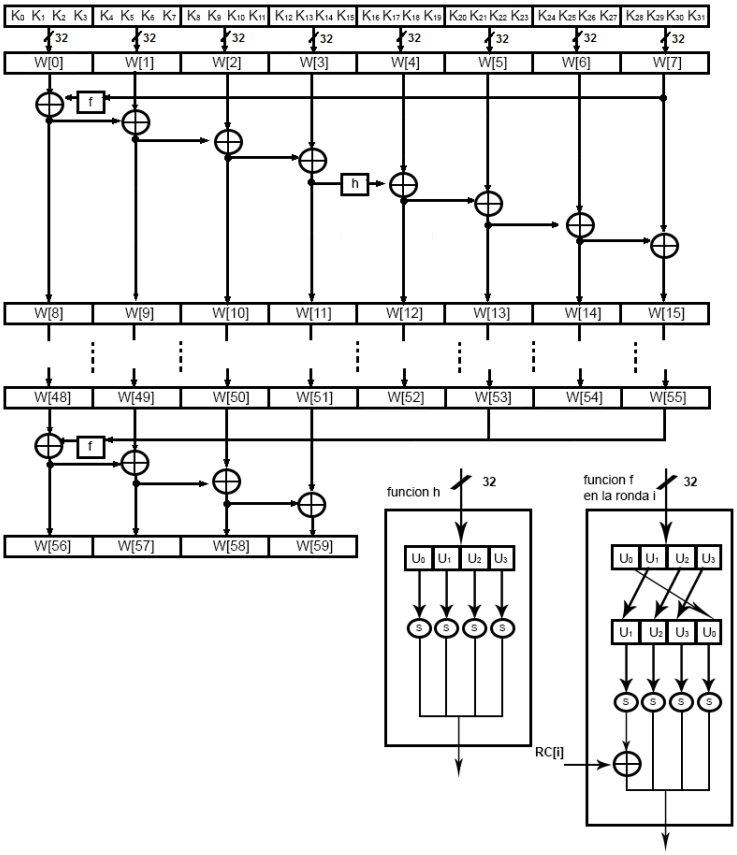
Para obtener la columnas $W[i]$ se realiza un xor con la columna $W[i-1]$ y la columna $W[i-6]$, salvo en las columnas que tienen indice multiplo de 6, en estas hay que pasar la columna $W[i-1]$ por la función f y entonces hacer un xor con la columna $W[i-6]$.

Una vez obtenidas las 52 columnas se agrupan en grupos de 4 para tener todas las subclaves necesarias.



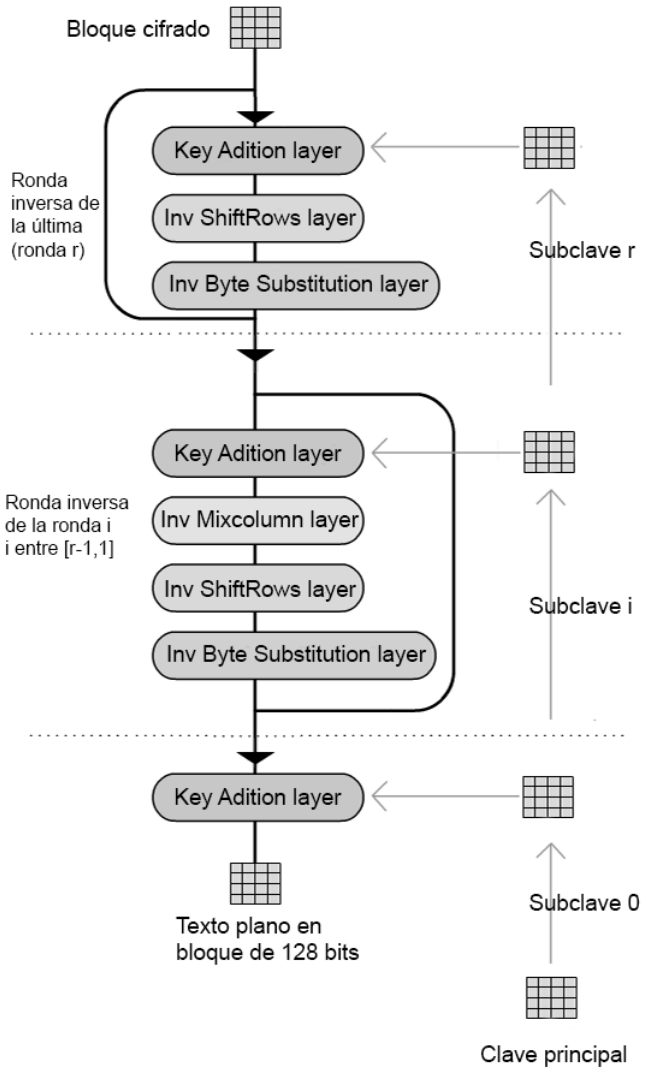
Key Schedule para claves de 256 bits

Para el AES con clave de 256 bits son necesarias 15 subclaves, para obtenerlas necesitamos 60 columnas de 4 bytes y para generarlas partimos de las 8 columnas de 4 bytes que nos proporciona la clave original y que serán las 8 primeras columnas de donde sacaremos las 2 primeras subclaves.
Este proceso se divide en 9 rondas en las cuales se obtienen 8 columnas nueva a partir de las 8 columnas anteriores. En cada ronda la primera columa se obtiene pasando la columna anterior por la función f y después haciendo un xor con la columna de 8 posiciones más atrás respecto a la que queremos sacar, el resto de columnas se obtiene haciendo un xor de la columna anterior y la columna 8 posiciones más atrás salvo en el caso de la quinta columna de cada ronda que se obtiene pasando la columna anterior por la función h y después haciendo un xor con la columna de 8 posiciones atrás. La función h lo que hace es pasar los bytes de la columna por s-cajas, mientras que la función f hace las mismas operaciones que en el caso de claves de tamaño 128 y 192.



Toda la estructura mencionada anteriormente forma parte del proceso de encriptado. Para el proceso de desencriptado toda las etapas anteriores cuentan con una etapa inversa que "deshace el camino". Entonces para desencriptar existe una estructura que coincide con la anterior pero en el sentido contrario donde las etapas deshacen las operaciones anteriores.

5.2.3. Estructura de AES para desencriptar



Como se observa en la figura se trata de la misma estructura al revés donde las operaciones son inversas a las del cifrado. En la primera ronda no aparece la inversa de MixColumn ya que en la última ronda del cifrado no aparecía, y en el resto de las $r-1$ rondas se observan las inversas de las operaciones de la parte del cifrado. La única operación que no varía es Key Addition debido a que este apartado solo realiza un xor y realizar otro xor deshace el del cifrado, la operación xor es su propia inversa.

Inverse MixColumn

Ya que en el cifrado esta etapa se trata de multiplicar una columna por una matriz, entonces en este caso se multiplica cada columna por la inversa de la matriz de la etapa de cifrado.

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} E2 & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

Inverse ShiftRows

Sea $B = \{B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}, B_{11}, B_{12}, B_{13}, B_{14}, B_{15}\}$ la matriz de estado que entra al Inverse ShiftRows, entonces el proceso será:

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

$\rightarrow$

B_0	B_4	B_8	B_{12}	$\longleftarrow$ 0 posiciones a la derecha
B_{13}	B_1	B_5	B_9	$\longleftarrow$ 1 posiciones a la derecha
B_{10}	B_{14}	B_2	B_6	$\longleftarrow$ 2 posiciones a la derecha
B_7	B_{11}	B_{15}	B_3	$\longleftarrow$ 3 posiciones a la derecha

Inverse Byte Substitution

La idea de esta etapa es que se construye una s-caja inversa.Para obtener los elementos de la caja se realiza una transformación inversa a la realizada en el cifrado y después de la transformación se busca el inverso de cada uno de los elementos del resultado de la transformación anterior.

Sea $B = \{B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}, B_{11}, B_{12}, B_{13}, B_{14}, B_{15}\}$ entonces para cada

$B_i = \{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$ se realiza la siguiente transformación:

$$\begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} mod 2$$

Con lo que se obtiene $B'_i = \{b'_0, b'_1, b'_2, b'_3, b'_4, b'_5, b'_6, b'_7\}$, y como segundo paso se obtiene su inversa sobre $GF(2^8)$

$$A_i = (B'_i)^{-1} \in GF(2^8)$$

Cuadro 5.5: S-caja inversa

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	OC	7D

Key Schedule del desencriptado

Esta etapa es igual que en el cifrado, lo que varía es el orden en el que se aplican las subclaves. Para desencriptar habrá que usar la última subclave en la primera ronda, la penúltima para la segunda ronda y así sucesivamente hasta usar la llave original en la ultima ronda.

5.3. Seguridad de AES

La seguridad de AES se basa en la longitud de la clave, cuanto mayor es el tamaño de clave mayor es el número de rondas que debe realizar el cifrador y por tanto mayor número de operaciones. La forma más efectiva de encontrar la clave es una búsqueda clave por clave lo que provoca que para los tamaños de 128 bits, 192 bits y 256 bits se tenga que aplicar AES 2^{127}, 2^{191} y 2^{255} veces. Y en cada una de esas aplicaciones de AES se realizaran 10, 12 o 14 rondas respectivamente para los distintos tamaños de clave. Por tanto, el número de operaciones que se deben realizar para encontrar la clave es inviable en cuestión de tiempo para los ordenadores actuales incluso con varios ordenadores trabajando en paralelo.

Según un grupo de investigadores de Microsoft y de la Dutch Katholieke Universiteit Leuven en 2011 existe un defecto en AES que permite "romper" el algoritmo. Sin embargo, en el mismo estudio aseguran que incluso con este defecto un billón de ordenadores que pudieran cada uno probar mil millones de claves por segundo, tardarían más de 2.000 millones de años en dar con una clave del sistema AES-128, y hay que tener en cuenta que las máquinas actuales sólo pueden probar 10 millones de claves por segundo.

En lo relativo a las claves ninguna clave es "débil" para este algoritmo y no existen restricciones en la selección de claves (ver en [2]).

Existen algoritmos que consiguen reducir el número de rondas lo que provoca por tanto una reducción en el tiempo. Sin embargo, continua siendo computacionalmente imposible obtener la clave.

Sin embargo, la mayor parte de los ataques se centran en los dispositivos sobre los que se implementa AES y buscan obtener datos que pueda dejar el algoritmo temporalmente en las memorias y por tanto puedan ser utilizados para revelar algunos datos que ayuden a romper el algoritmo.

Además AES previene ataques semejantes a los que realizaban sobre DES. El uso de la inversa sobre cuerpos finitos en las s-cajas hace que los ataques lineales y diferenciales se vuelvan muy complicados.

5.3.1. Criptoanálisis diferencial

Es un ataque que se centra en el estudio de cómo modificando las entradas se producen diferencias en las salidas del cifrador.

5.3.2. Criptoanálisis lineal

Este es un ataque de texto en claro conocido, que usa una aproximación lineal para describir el funcionamiento del algoritmo. Dados suficientes pares de texto en claro y cifrado se pueden obtener datos sobre la clave.

5.4. Ejemplo de AES

En este apartado se va a mostrar mediante un ejemplo el funcionamiento de AES.

Sea el texto "manda el mensaje" el mensaje que se quiere enviar, entonces se trata de 16 caracteres ASCII que ocupan 1 byte cada uno, se pasa el mensaje a hexadecimal.

ASCII	m	a	n	d	a		e	l		m	e	n	s	a	j	e
Hexadecimal	6D	61	6E	64	61	20	65	6C	20	6D	65	6E	73	61	6A	65

Entonces vamos a cifrar este mensaje con AES de 128 bits y la clave "clave principal "

ASCII	c	l	a	v	e		p	r	i	n	c	i	p	a	l	
Hexadecimal	63	6C	61	76	65	20	70	72	69	6E	63	69	70	61	6C	20

Lo primero será realizar el Key Schedule para obtener las 10 subclaves que necesitaremos para cada una de las rondas. Para ello se divide la clave original (que será la subclave 0) en 4 columnas.

$$w[0] = [63, 6C, 61, 76],\ w[1] = [65, 20, 70, 72]$$

$$w[2] = [69, 6E, 63, 69],\ w[3] = [70, 61, 6C, 20]$$

Ahora se aplica la función f sobre $w[3]$ que consiste en un cambio hacia la izquierda de las posiciones, pasar los bytes por una s-caja y añadir una constante de RC.

$$[61, 6C, 20, 70] \rightarrow [EF, 50, B7, 51] \rightarrow +[01, 00, 00, 00] \rightarrow [EE, 50, B7, 51]$$

y con lo obtenido de la función f se hace un XOR con $w[0]$

$$w[4] = w[0] \oplus [EE, 50, B7, 51] = [8D, 3C, D6, 27]$$

Y las siguientes 3 columnas se obtienen de hacer un XOR con la columna anterior y la de 4 posiciones atrás.

$$w[5] = w[4] \oplus w[1] = [E8, 1C, A6, 55]$$

$$w[6] = w[5] \oplus w[2] = [81, 72, C5, 3C]$$

$$w[7] = w[6] \oplus w[3] = [F1, 13, A9, 1C]$$

De forma que la subclave 1 será

$$[8D, 3C, D6, 27, E8, 1C, A6, 55, 81, 72, C5, 3C, F1, 13, A9, 1C]$$

Y se repite el proceso para las 9 subclaves

- Subclave 0: $[63, 6C, 61, 76, 65, 20, 70, 72, 69, 6E, 63, 69, 70, 61, 6C, 20]$
- Subclave 1: $[8D, 3C, D6, 27, E8, 1C, A6, 55, 81, 72, C5, 3C, F1, 13, A9, 1C]$
- Subclave 2: $[F2, EF, 4A, 86, 1A, F3, EC, D3, 9B, 81, 29, EF, 6A, 92, 80, F3]$
- Subclave 3: $[B9, 22, 47, 84, A3, D1, AB, 57, 38, 50, 82, B8, 52, C2, 02, 4B]$
- Subclave 4: $[94, 55, F4, 84, 37, 84, 5F, D3, 0F, D4, DD, 6B, 5D, 16, DF, 20]$
- Subclave 5: $[C3, CB, 43, C8, F4, 4F, 1C, 1B, FB, 9B, C1, 70, A6, 8D, 1E, 50]$
- Subclave 6: $[BE, B9, 10, EC, 4A, F6, 0C, F7, B1, 6D, CD, 87, 17, E0, D3, D7]$
- Subclave 7: $[1F, DF, 1E, 1C, 55, 29, 12, EB, E4, 44, DF, 6C, F3, A4, 0C, BB]$
- Subclave 8: $[D6, 21, F4, 11, 83, 08, E6, FA, 67, 4C, 39, 96, 94, E8, 35, 2D]$
- Subclave 9: $[56, B7, 2C, 33, D5, BF, CA, C9, B2, F3, F3, 5F, 26, 1B, C6, 72]$
- Subclave 10: $[CF, 03, 6C, C4, 1A, BC, A6, 0D, A8, 4F, 55, 52, 8E, 54, 93, 20]$

5.4.1. Key Addition layer Ronda 0

Se realiza un xor con la matriz de estado y la subclave 0.

$6D$	61	20	73
61	20	$6D$	61
$6E$	65	65	$6A$
64	$6C$	$6E$	65

$\oplus$

63	65	69	70
$6C$	20	$6E$	61
61	70	63	$6C$
76	72	69	20

$=$

$0E$	04	49	03
$0D$	00	03	00
$0F$	15	06	06
12	$1E$	07	45

5.4.2. Ronda 1

Partimos de la matriz estado

$0E$	04	49	03
$0D$	00	03	00
$0F$	15	06	06
12	$1E$	07	45

Byte Substitution layer

Se cambia cada entrada del estado por su valor correspondiente en la s-caja.

$0E$	04	49	03
$0D$	00	03	00
$0F$	15	06	06
12	$1E$	07	45

$\rightarrow$ s-caja $\rightarrow$

AB	$F2$	$3B$	$7B$
$D7$	63	$7B$	63
76	59	$6F$	$6F$
$C9$	72	$C5$	$6E$

ShitfRows layer

Se mueven las entradas de la fila 1 0 posiciones a la izquierda, las de la fila 2 1 posición a la izquierda, las de la fila 3 2 posiciones a la izquierda y las de la fila 4 3 posiciones a la izquierda.

AB	$F2$	$3B$	$7B$
$D7$	63	$7B$	63
76	59	$6F$	$6F$
$C9$	72	$C5$	$6E$

$\rightarrow$

AB	$F2$	$3B$	$7B$
63	$7B$	63	$D7$
$6F$	$6F$	76	59
$6E$	$C9$	72	$C5$

MixColumn layer

Se multiplica una matriz constante en todas las rondas por la matriz de estado.

$$\begin{array}{|c|c|c|c|} \hline 02 & 03 & 01 & 01 \\ \hline 01 & 02 & 03 & 01 \\ \hline 01 & 01 & 02 & 03 \\ \hline 03 & 01 & 01 & 02 \\ \hline \end{array} \cdot \begin{array}{|c|c|c|c|} \hline AB & F2 & 3B & 7B \\ \hline 63 & 7B & 63 & D7 \\ \hline 6F & 6F & 76 & 59 \\ \hline 6E & C9 & 72 & C5 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline E9 & D4 & D7 & 08 \\ \hline B2 & 7C & 15 & E0 \\ \hline A4 & 17 & 22 & 4A \\ \hline 36 & 90 & BC & 92 \\ \hline \end{array}$$

Por ejemplo la entrada $E9$ es resultado de realizar

$$(02 \cdot AB) \oplus (03 \cdot 63) \oplus (01 \cdot 6F) \oplus (01 \cdot 6E)$$

Key Addition layer

Aquí se realiza un xor entre la matriz de estado y la subclave de la ronda correspodiente, en este caso la 1.

$$\begin{array}{|c|c|c|c|} \hline 8D & E8 & 81 & F1 \\ \hline 3C & 1C & 72 & 13 \\ \hline D6 & A6 & C5 & A9 \\ \hline 27 & 55 & 3C & 1C \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline E9 & D4 & D7 & 08 \\ \hline B2 & 7C & 15 & E0 \\ \hline A4 & 17 & 22 & 4A \\ \hline 36 & 90 & BC & 92 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 64 & 3C & 56 & F9 \\ \hline 8E & 60 & 67 & F3 \\ \hline 72 & B1 & E7 & E3 \\ \hline 11 & C5 & 80 & 8E \\ \hline \end{array}$$

5.4.3. Ronda 2

Byte Substitution layer

$$\begin{array}{|c|c|c|c|} \hline 64 & 3C & 56 & F9 \\ \hline 8E & 60 & 67 & F3 \\ \hline 72 & B1 & E7 & E3 \\ \hline 11 & C5 & 80 & 8E \\ \hline \end{array} \rightarrow \text{s-caja} \rightarrow \begin{array}{|c|c|c|c|} \hline 43 & EB & B1 & 99 \\ \hline 19 & D0 & 85 & 0D \\ \hline 40 & C8 & 94 & 11 \\ \hline 82 & A6 & CD & 19 \\ \hline \end{array}$$

ShitfRows layer

43	EB	$B1$	99
$D0$	85	$0D$	19
94	11	40	$C8$
19	82	$A6$	CD

MixColumn layer

$$\begin{array}{|c|c|c|c|} \hline 02 & 03 & 01 & 01 \\ \hline 01 & 02 & 03 & 01 \\ \hline 01 & 01 & 02 & 03 \\ \hline 03 & 01 & 01 & 02 \\ \hline \end{array} \cdot \begin{array}{|c|c|c|c|} \hline 43 & EB & B1 & 99 \\ \hline D0 & 85 & 0D & 19 \\ \hline 94 & 11 & 40 & C8 \\ \hline 19 & 82 & A6 & CD \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 60 & CA & 88 & 07 \\ \hline 46 & 4B & CD & 25 \\ \hline 8B & D1 & CD & 47 \\ \hline B3 & AD & D2 & E0 \\ \hline \end{array}$$

Key Addition layer

$$\begin{array}{|c|c|c|c|} \hline F2 & 1A & 9B & 6A \\ \hline EF & F3 & 81 & 92 \\ \hline 4A & EC & 29 & 80 \\ \hline 86 & D3 & EF & F3 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline 60 & CA & 88 & 07 \\ \hline 46 & 4B & CD & 25 \\ \hline 8B & D1 & CD & 47 \\ \hline B3 & AD & D2 & E0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 92 & D0 & 12 & 6D \\ \hline A9 & B8 & 4C & B7 \\ \hline C1 & 3D & E4 & C7 \\ \hline 35 & 7E & 3D & 13 \\ \hline \end{array}$$

5.4.4. Ronda 3

Byte Substitution layer

$$\begin{array}{|c|c|c|c|} \hline 92 & D0 & 12 & 6D \\ \hline A9 & B8 & 4C & B7 \\ \hline C1 & 3D & E4 & C7 \\ \hline 35 & 7E & 3D & 13 \\ \hline \end{array} \rightarrow \text{s-caja} \rightarrow \begin{array}{|c|c|c|c|} \hline 4F & 70 & 7D & 3C \\ \hline D3 & 6C & 29 & A9 \\ \hline 78 & 27 & 69 & C6 \\ \hline 96 & F3 & 27 & 7D \\ \hline \end{array}$$

ShitfRows layer

$4F$	70	$7D$	$3C$
$6C$	29	$A9$	$D3$
69	$C6$	78	27
$7D$	96	$F3$	27

MixColumn layer

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\cdot$

$4F$	70	$7D$	$3C$
$6C$	29	$A9$	$D3$
69	$C6$	78	27
$7D$	96	$F3$	27

$=$

$3E$	CB	91	16
51	$E5$	$4F$	CF
76	$6F$	$2A$	$C8$
$2E$	48	AB	FE

Key Addition layer

$B9$	$A3$	38	52
22	$D1$	50	$C2$
47	AB	82	02
84	57	$B8$	$4B$

$\oplus$

$3E$	CB	91	16
51	$E5$	$4F$	CF
76	$6F$	$2A$	$C8$
$2E$	48	AB	FE

$=$

87	68	$A9$	44
73	34	$1F$	$0D$
31	$C4$	$A8$	CA
AA	$1F$	13	$B5$

5.4.5. Ronda 4

Byte Substitution layer

87	68	$A9$	44
73	34	$1F$	$0D$
31	$C4$	$A8$	CA
AA	$1F$	13	$B5$

$\rightarrow$ s-caja $\rightarrow$

17	45	$D3$	$1B$
$8F$	18	$C0$	$D7$
$C7$	$1C$	$C2$	74
AC	$C0$	$7D$	$D5$

ShitfRows layer

17	45	$D3$	$1B$
18	$C0$	$D7$	$8F$
$C2$	74	$C7$	$1C$
$D5$	AC	$C0$	$7D$

MixColumn layer

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\cdot$

17	45	$D3$	$1B$
18	$C0$	$D7$	$8F$
$C2$	74	$C7$	$1C$
$D5$	AC	$C0$	$7D$

$=$

11	09	$D8$	DD
AF	EE	$F4$	47
$F4$	82	CA	$2B$
52	38	$E5$	44

Key Addition layer

94	37	$0F$	$5D$
55	84	$D4$	16
$F4$	$5F$	DD	DF
84	$D3$	$6B$	20

$\oplus$

11	09	$D8$	DD
AF	EE	$F4$	47
$F4$	82	CA	$2B$
52	38	$E5$	44

$=$

85	$3E$	$D7$	80
FA	$6A$	20	51
00	DD	17	$F4$
$D6$	EB	$8E$	64

5.4.6. Ronda 5

Byte Substitution layer

85	$3E$	$D7$	80
FA	$6A$	20	51
00	DD	17	$F4$
$D6$	EB	$8E$	64

$\rightarrow$ s-caja $\rightarrow$

97	$B2$	$0E$	CD
$2D$	02	$B7$	$D1$
63	$C1$	$F0$	BF
$F6$	$E9$	19	43

ShitfRows layer

97	$B2$	$0E$	CD
02	$B7$	$D1$	$2D$
$F0$	BF	63	$C1$
43	$F6$	$E9$	19

MixColumn layer

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\cdot$

97	$B2$	$0E$	CD
02	$B7$	$D1$	$2D$
$F0$	BF	63	$C1$
43	$F6$	$E9$	19

$=$

80	$F4$	FE	$2E$
DB	EB	FB	$D6$
AB	61	39	52
$D6$	32	69	92

Key Addition layer

$C3$	$F4$	FB	$A6$
CB	$4F$	$9B$	$8D$
43	$1C$	$C1$	$1E$
$C8$	$1B$	70	50

$\oplus$

80	$F4$	FE	$2E$
DB	EB	FB	$D6$
AB	61	39	52
$D6$	32	69	92

$=$

43	00	05	88
10	$A4$	60	$5B$
$E8$	$7D$	$F8$	$4C$
$1E$	29	19	$C2$

5.4.7. Ronda 6

Byte Substitution layer

43	00	05	88
10	$A4$	60	$5B$
$E8$	$7D$	$F8$	$4C$
$1E$	29	19	$C2$

$\rightarrow$ s-caja $\rightarrow$

$1A$	63	$6B$	$C4$
CA	49	$D0$	39
$9B$	FF	41	29
72	$A5$	$D4$	25

ShitfRows layer

$1A$	63	$6B$	$C4$
49	$D0$	39	CA
41	29	$9B$	FF
25	72	$A5$	$D4$

MixColumn layer

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\cdot$

$1A$	63	$6B$	$C4$
49	$D0$	39	CA
41	29	$9B$	FF
25	72	$A5$	$D4$

$=$

$8B$	$F6$	$A3$	FD
$6E$	$D1$	$0A$	85
BE	77	$8B$	$8C$
$6C$	$B8$	$4E$	$D1$

Key Addition layer

BE	$4A$	$B1$	17
$B9$	$F6$	$6D$	$E0$
10	$0C$	CD	$D3$
EC	$F7$	87	$D7$

$\cdot$

$8B$	$F6$	$A3$	FD
$6E$	$D1$	$0A$	85
BE	77	$8B$	$8C$
$6C$	$B8$	$4E$	$D1$

$=$

35	BC	12	EA
$D7$	27	67	65
AE	$7B$	46	$5F$
80	$4F$	$C9$	06

5.4.8. Ronda 7

Byte Substitution layer

35	BC	12	EA
$D7$	27	67	65
AE	$7B$	46	$5F$
80	$4F$	$C9$	06

$\rightarrow$ s-caja $\rightarrow$

96	65	$C9$	87
$0E$	CC	85	$4D$
$E4$	21	$5A$	CF
CD	84	DD	$6F$

ShitfRows layer

96	65	$C9$	87
CC	85	$4D$	$0E$
$5A$	CF	$E4$	21
$6F$	CD	84	DD

MixColumn layer

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\cdot$

96	65	$C9$	87
CC	85	$4D$	$0E$
$5A$	CF	$E4$	21
$6F$	CD	84	DD

$=$

$4D$	$5C$	$3E$	FB
94	$F3$	$E0$	25
$5F$	29	$C0$	$B7$
$E9$	64	FA	$1C$

Key Addition layer

$1F$	55	$E4$	$F3$
DF	29	44	$A4$
$1E$	12	DF	$0C$
$1C$	EB	$6C$	BB

$\oplus$

$4D$	$5C$	$3E$	FB
94	$F3$	$E0$	25
$5F$	29	$C0$	$B7$
$E9$	64	FA	$1C$

$=$

52	09	DA	08
$4B$	DA	$A4$	81
41	$3B$	$1F$	BB
$F5$	$8F$	96	$A7$

5.4.9. Ronda 8

Byte Substitution layer

52	09	DA	08
$4B$	DA	$A4$	81
41	$3B$	$1F$	BB
$F5$	$8F$	96	$A7$

$\rightarrow$ s-caja $\rightarrow$

00	01	57	30
$B3$	57	49	$0C$
83	$E2$	$C0$	EA
$E6$	73	90	$5C$

ShitfRows layer

00	01	57	30
57	49	$0C$	$B3$
$C0$	EA	83	$E2$
$5C$	$E6$	73	90

MixColumn layer

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\cdot$

00	01	57	30
57	49	$0C$	$B3$
$C0$	EA	83	$E2$
$5C$	$E6$	73	90

$=$

65	$D5$	$4A$	DC
$A9$	50	$A2$	$E0$
28	$B6$	$D3$	$F7$
$2F$	77	90	$3A$

Key Addition layer

$D6$	83	67	94
21	08	$4C$	$E8$
$F4$	$E6$	39	35
11	FA	96	$2D$

$\oplus$

65	$D5$	$4A$	DC
$A9$	50	$A2$	$E0$
28	$B6$	$D3$	$F7$
$2F$	77	90	$3A$

$=$

$B3$	56	$2D$	48
88	58	EE	08
DC	50	EA	$C2$
$3E$	$8D$	06	17

5.4.10. Ronda 9

Byte Substitution layer

$B3$	56	$2D$	48
88	58	EE	08
DC	50	EA	$C2$
$3E$	$8D$	06	17

$\rightarrow$ s-caja $\rightarrow$

$6D$	$B1$	$D8$	52
$C4$	$6A$	28	30
86	53	87	25
$B2$	$5D$	$6F$	$F0$

ShitfRows layer

$6D$	$B1$	$D8$	52
$6A$	28	30	$C4$
87	25	86	53
$F0$	$B2$	$5D$	$6F$

MixColumn layer

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\cdot$

$6D$	$B1$	$D8$	52
$6A$	28	30	$C4$
87	25	86	53
$F0$	$B2$	$5D$	$6F$

$=$

13	96	20	CF
DB	$3C$	74	$5B$
19	$1E$	18	81
$A1$	BA	$7F$	BF

Key Addition layer

56	$D5$	$B2$	26
$B7$	BF	$F3$	$1B$
$2C$	CA	$F3$	$C6$
33	$C9$	$5F$	72

$\oplus$

13	96	20	CF
DB	$3C$	74	$5B$
19	$1E$	18	81
$A1$	BA	$7F$	BF

$=$

45	43	92	$E9$
$6C$	83	87	40
35	$D4$	EB	47
92	73	20	CD

5.4.11. Ronda 10

Byte Substitution layer

45	43	92	$E9$
$6C$	83	87	40
35	$D4$	EB	47
92	73	20	CD

$\rightarrow$s-caja$\rightarrow$

$6E$	$1A$	$4F$	$1E$
50	EC	17	09
96	48	$E9$	$A0$
$4F$	$8F$	$B7$	BD

ShitfRows layer

$6E$	$1A$	$4F$	$1E$
EC	17	09	50
$E9$	$A0$	96	48
BD	$4F$	$8F$	$B7$

Key Addition layer

CF	03	$6C$	$C4$
$1A$	BC	$A6$	$0D$
$A8$	$4F$	55	52
$8E$	54	93	20

$\oplus$

$6E$	$1A$	$4F$	$1E$
EC	17	09	50
$E9$	$A0$	96	48
BD	$4F$	$8F$	$B7$

$=$

$A1$	00	EF	90
EF	AB	46	04
85	06	$C3$	DB
79	42	DD	97

Con lo que el texto final cifrado es

$A1$ 00 EF 90 EF AB 46 04 85 06 $C3$ DB 79 42 DD 97

Bibliografía

[1] Carlos Munuera Gómez y Juan Tena Ayuso, *Codificación de la información* , Universidad de Valladolid, 1997. ISBN: 84-7762-764-9

[2] Federal Information Processing Standards Publication 197, *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*, 26 Noviembre de 2001.

[3] Christof Paar y Jan Pelzl, *Understanding Cryptography*, Springer. ISBN: 978-3-642-04100-6

[4] Gabriel Navarro, *Un curso de álgebra*, Universidad de Valencia, 2002. ISBN: 84-370-5419-2

[5] George Schlossnagle, *Advanced PHP Programming*, 2004. ISBN: 0-672-32561-6

[6] Joan Daemen y Vincent Rijmen, *The Design of Rijndael*, noviembre 2001.

[7] Joan Daemen y Vincent Rijmen, *AES Proposal: Rijndael*, 3 de septiembre de 1999.

[8] Caballero, P., *Introducción a la Criptografía*, Ed. Ra-Ma. Madrid, 2002.

[9] Galende, J.C.,*Criptografía: Historia de la escritura cifrada*, Ed. Complutense, Madrid, 1995.

[10] Llorenç Huguet Rotger, Josep Rifà Coma y Juan Gabriel Tena Ayuso, *Cuerpos finitos.*

[11] José Manuel Sánchez Muñoz, *Historia de Matemáticas: Criptología Nazi. Los Códigos Secretos de Hitler*, abril 2013.

[12] Mohammed J.Kabir, *La biblia de Servidor Apache 2*

Anexo A

Anexo I

A.0.12. Query string

Es parte de la URL donde se añaden una serie de datos o parámetros. Estos datos vienen definidos tras el nombre del archivo referenciado por la URL y tras un signo de interrogación "?". Suelen venir definidos por la siguiente estructura nombre=valor y para separar unos datos de otros se utiliza "&".

Ejemplo:

www.ejemplo.net/pagina.php?nombre1=valor1&nombre2=valor2

A.0.13. URL

URL en español significa localizador uniforme de recursos (son las siglas en inglés de uniform resource locator), se usa para referenciar recursos en Internet. Tiene un formato estándar y su propósito es asignar una dirección única a cada uno de los recursos disponibles en Internet, como páginas, imágenes, vídeos, etc.

Formato de URL → Esquema://máquina/directorio/archivo

Ejemplo: http://ejemplos.com/category/gramática

Tipos de esquema:

- http, es el esquema más frecuente en Internet.
- https, es el esquema usado para páginas seguras de Internet.
- mailto, es el esquema usado para direcciones de correo electrónico. Comúnmente encontrado en páginas con enlaces que permiten el envío de un correo electrónico.
- ftp, es el esquema usado para el protocolo de transferencia de archivos FTP.
- file es el esquema usado para obtener archivos, usualmente dentro de la computadora de uno mismo.

A.1. Blowfish

Se trata de un cifrador de bloques simétricos que usa bloques de 64 bits y claves entre 32 y 448 bits. Fue creado en 1993 por Bruce Schneier y fue presentado como candidato para ser nombrado AES y reemplazar a DES como algoritmo de cifrado estándar. La principal ventaja de este algoritmo es que puede funcionar con tan solo 5 KB de memoria libre. Sin embargo no es de los algoritmos más usados, AES o Twofish se usan más, Twofish fue creado también por Bruce Schneier y mantiene muchas similitudes con Blowfish.

A.2. Script

El interprete de comandos de UNIX (o "shell") es un lenguaje de programación completo,los programas escritos en lenguaje de shell se denominan "scripts" y son archivos con la extensión ".sh" y que suelen comenzar con uno de los siguientes encabezados:

#!/bin/bash ; #!/bin/ksh ; #!/bin/csh

A.3. Hexadecimal

El sistema Hexadecimal está en base 16, sus primeros 10 valores son números representados por los 10 primeros dígitos de la numeración decimal (del 0 al 9) , y los restantes 6 valores están representados por las letras del alfabeto de la A a la F.

Decimal	Hexadecimal	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

De esta forma un byte puede ser representado mediante 2 cifras en hexadecimal. De la misma forma un polinomio puede ser representado como en binario y desde ese punto pasarlo a hexadecimal.

Ejemplo:

$$x6 + x^4 + x + 1 = (01010011)_2 = (53)_{hex}$$

A.4. XOR

Se trata de un operador lógico, el operador lógico funciona de manera que es verdadero si cualquiera de las dos es verdadero sin serlo las dos a la vez.

Input 1	Input 2	Ouput
F	F	F
V	F	V
F	V	V
V	V	F

En el caso de los bytes es 1 si uno de los dos es 1 no siendo 1 los dos a la vez.

Input 1	Input 2	Ouput
0	0	0
1	0	1
0	1	1
1	1	0

A.5. Código ASCII

Se trata de un código para representación de caracteres alfanuméricos en las computadoras. Surgió en los años 60 como solución para unificar los diferentes criterios, hasta entonces cada computadora utilizaba un criterio propio. El código ASCII básico representaba caracteres utilizando 7 bits (para 128 caracteres posibles, enumerados del 0 al 127).

Los primeros 32 elementos se conocen como caracteres de control, y no estaban pensados para representar símbolos sino para controlar dispositivos. Los caracteres del 33 al 126 se denominan caracteres imprimibles ASCII y representan símbolos,letras y números.

Cuadro A.1: Tabla ASCII

Caracter	Código ASCII	Hexadecimal	Caracter	Código ASCII	Hecadecimal
NULL	0	00	’	39	27
SOH	1	01	(	40	28
STX	2	02	)	41	29
ETX	3	03	*	42	2A
EOT	4	04	+	43	2B
ENQ	5	05	,	44	2C
ACK	6	06	-	45	2D
BEL	7	07	.	46	2E
BS	8	08	/	47	2F
TAB	9	09	0	48	30
LF	10	0A	1	49	31
VT	11	0B	2	50	32
FF	12	0C	3	51	33
CR	13	0D	4	52	34
SO	14	0E	5	53	35
SI	15	0F	6	54	36
DLE	16	10	7	55	37
DC1	17	11	8	56	38
DC2	18	12	9	57	39
DC3	19	13	:	58	3A
DC4	20	14	;	59	3B
NAK	21	15	<	60	3C
SYN	22	16	=	61	3D
ETB	23	17	>	62	3E
CAN	24	18	?	63	3F
EM	25	19	@	64	40
SUB	26	1A	A	65	41
ESC	27	1B	B	66	42
FS	28	1C	C	67	43
GS	29	1D	D	68	44
RS	30	1E	E	69	45
US	31	1F	F	70	46
SP	32	20	G	71	47
!	33	21	H	72	48
”	34	22	I	73	49
#	35	23	J	74	4A
$	36	24	K	75	4B
%	37	25	L	76	4C
&	38	26	M	77	4D

Caracter	Código ASCII	Hexadecimal	Caracter	Código ASCII	Hexadecimal
N	78	4E	g	103	67
O	79	4F	h	104	68
P	80	50	i	105	69
Q	81	51	j	106	6A
R	82	52	k	107	6B
S	83	53	l	108	6C
T	84	54	m	109	6D
U	85	55	n	110	6E
V	86	56	o	111	6F
W	87	57	p	112	70
X	88	58	q	113	71
Y	89	59	r	114	72
Z	90	5A	s	115	73
[	91	5B	t	116	74
\	92	5C	u	117	75
]	93	5D	v	118	76
^	94	5E	w	119	77
_	95	5F	x	120	78
´	96	60	y	121	79
a	97	61	z	122	7A
b	98	62	{	123	7B
c	99	63	—	124	7C
d	100	64	}	125	7D
e	101	65	∼	126	7E
f	102	66			