Sistema de cifrado embebido sobre la placa MPS432.

Carlos López Benito
Máster Universitario en Ingeniería Informática
Sistemas Encastados

Jordi Bécares Ferrés
Pere Tuset Peiró

22 de Enero de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Sistema de cifrado embebido sobre la placa MPS432.*
Nombre del autor:	*Carlos López Benito*
Nombre del consultor:	*Jordi Bécares Ferrés*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	01/2018
Titulación:	Máster Universitario en Ingeniería Informática
Área del Trabajo Final:	*Sistemas Encastados*
Idioma del trabajo:	*Castellano*
Palabras clave	*Criptografía, MSP432, FreeRTOS*

Resumen del Trabajo (máximo 250 palabras*.*

Esta memoria documenta la realización, desarrollo y resultados del trabajo final del Máster de Ingeniería Informática en la Universidad Abierta de Cataluña en el área de Sistemas empotrados.

En el presente trabajo se ha diseñado y llevado a cabo un proyecto consistente en la creación de un sistema de cifrado sobre la placa MSP432. Dicho sistema de cifrado incluye los algoritmos de cifrado más utilizados actualmente: DES, 3DES, AES y RSA.

Además, se ha realizado un estudio computacional en el cual se ha estudiado el tiempo, memoria RAM y memoria flash necesaria para cada algoritmo sobre dicha placa, ya que este tipo de placas tienen unos recursos muy limitados.

Abstract (in English, 250 words or less):

This report documents the realisation, development and results of the final work of the Master's Degree in Computer Engineering at the Open University of Catalonia in the area of Embedded Systems.

In the present work, a project consisting in the creation of an encryption system on the MSP432 board has been designed and carried out. This encryption system includes the most commonly used encryption algorithms: DES, 3DES, AES and RSA.

In addition, a computational study has been carried out in which the time, RAM and flash memory necessary for each algorithm on said board has been studied, since this type of boards have very limited resources.

Índice

1 INTRODUCCIÓN 1

1.1 Contexto y justificación del trabajo 1
1.2 Descripción del trabajo 1
1.3 Objetivos del TFC. 2
1.4 Enfoque y método seguido 2
1.5 Planificación del trabajo. 3
1.6 Recursos empleados 7
1.7 Productos obtenidos 7
1.8 Breve descripción de los otros capítulos de la memoria 7

2 ANTECEDENTES 8

2.1 Estado del arte 8
2.2 Estudio de mercado 10
2.3 Algoritmos de Cifrado 12
2.3.1 DES 12
2.3.2 Triple DES 13
2.3.3 AES 13
2.3.4 RSA 13
2.4 Modos de cifrado por bloques 15
2.4.1 Electronic Code Book (ECB) 15
2.4.2 Cipher Block Chaining (CBC) 15
2.4.3 Cipher Feedback (CFB) 16
2.4.4 Ouput feedback (OFB) 17
2.4.5 Propagating Cipher Block Chaining (PCBC) 18

3 DESCRIPCIÓN FUNCIONAL 19

3.1 Sistema cifrado embebido 19
3.1.1 Diagrama de bloques del sistema 19
3.1.2 Descripción de la red 19
3.1.3 Interacción de los distintos objetos del sistema 20
3.2 Aplicación de Usuario 20
3.2.1 Diagrama de bloques 20
3.2.2 Interacción de los distintos objetos de la aplicación de usuario 21
3.3 Sistema empotrado 22
3.3.1 Diagrama de bloques 22
3.3.2 Diseño y funcionamiento de la aplicación del sistema empotrado 22

4 DESCRIPCIÓN DETALLADA 23

4.1 Algoritmo DES 23
4.1.1 Permutación Inicial 24
4.1.2 División en bloques de 32 bits. 24
4.1.3 La función F 24
4.1.4 Permutación inicial inversa 27
4.1.5 Generación de Claves 28
4.2 TDES 29
4.3 AES. 30
4.3.1 Generación de claves. 31
4.3.2 Función SubByte: 33
4.3.3 Función ShiftRow 33
4.3.4 Función MixColumns 34
4.3.5 Función AddRoundKey 34
4.3.6 Descifrado AES 35
4.4 RSA 35

4.5 MD5 36
5 RESULTADOS ESTUDIO COMPARATIVO DE LOS ALGORITMOS. 38
5.1 TIEMPO. 38
5.1.1 Tiempos DES 38
5.1.2 Tiempos TDES 39
5.1.3 Tiempos AES Software 40
5.1.4 Tiempos de AES Hardware 42
5.2 TIEMPOS RSA 44
5.3 TAMAÑO RAM 45
5.4 TAMAÑO EN LA MEMORIA FLASH 46
5.5 CONCLUSIONES: 47
6 VIABILIDAD TÉCNICA 48
7 VALORACIÓN ECONÓMICA 48
8 CONCLUSIONES 49
8.1 LECCIONES APRENDIDAS 49
8.2 AUTOEVALUACIÓN. 49
8.2.1 Una reflexión crítica sobre el logro de los objetivos 49
8.2.2 Un análisis crítico del seguimiento de la planificación 50
8.3 LAS LÍNEAS DE TRABAJO FUTURO 50
9 GLOSARIO 51
10 BIBLIOGRAFÍA 52
11 ANEXOS 54
11.1 CONSUMO DE RAM DE LOS ALGORITMOS DE CIFRADO 54

Lista de figuras

Figura 1 Cronograma Inicial 6
Figura 2 Cronograma Final..... 6
Figura 3 Cifrado Modo ECB 15
Figura 4 Descifrado Modo ECB..... 15
Figura 5 Cifrado Modo CBC 16
Figura 6 Descifrado Modo CBC..... 16
Figura 7 Cifrado Modo CFB..... 16
Figura 8 Descifrado Modo CFB 17
Figura 9 Cifrado Modo OFB 17
Figura 10 Descifrado Modo OFB..... 17
Figura 11 Cifrado Modo PCBC..... 18
Figura 12 Descifrado Modo PCBC 18
Figura 13 Diagrama funcional del sistema 19
Figura 14 Diagrama de conexión 20
Figura 15 Diagrama de bloques aplicación de usuario..... 21
Figura 16 Diagrama de capas sistema empotrado..... 22
Figura 17 Esquema de cifrado DES..... 23
Figura 18 Esquema creación subclaves DES 28
Figura 19 Fases algoritmo TDES 29
Figura 20 Fases algoritmo AES..... 30
Figura 21 Ejemplo clave AES 128bits 31
Figura 22 Función Rotword 31
Figura 23 Función SubBytes 32
Figura 24 Función XOR sobre la tabla RCON..... 32
Figura 25 Calculo últimas columnas de la subclave..... 33
Figura 26 Función SubByte..... 33
Figura 27 Función ShiftRow 34
Figura 28 Función MixColumns..... 34
Figura 29 Función AddRoundKey 35
Figura 30 Tiempos de ejecución algoritmo DES en segundos..... 39
Figura 31 Tiempos de ejecución algoritmo DES en segundos..... 40
Figura 32 Tiempos de descifrado AES Software..... 41
Figura 33 Tiempos de cifrado AES Software 41
Figura 35 Tiempos de descifrado de AES (Hardware) 43
Figura 34 Tiempos de cifrado de AES (Hardware)..... 43
Figura 36 Tamaño RAM de los diferentes algoritmos de cifrado..... 45
Figura 37 Tamaño en la Flash..... 46
Figura 38 Valoración económica..... 48

Lista de Tablas

Tabla 1 Tareas del proyecto 4
Tabla 2 Plan de riesgos 6
Tabla 3 Permutación inicial 24
Tabla 4 Estado inicial L 24
Tabla 5 Estado Inicial R 24
Tabla 6 Tabla de Expansión 25
Tabla 7 Matriz de sustitución S1 25
Tabla 8 Matriz de sustitución S2 26
Tabla 9 Matriz de sustitución S3 26
Tabla 10 Matriz de sustitución S4 26
Tabla 11 Matriz de sustitución S5 26
Tabla 12 Matriz de sustitución S6 26
Tabla 13 Matriz de sustitución S7 26
Tabla 14 Matriz de sustitución S8 27
Tabla 15 Matriz de Permutación P 27
Tabla 16 Matriz de permutación inversa DES 27
Tabla 17 Elección Permutada 1 28
Tabla 18 Matriz Li 28
Tabla 19 Matriz Ri 28
Tabla 20 Elección permutada (PC-2) 29
Tabla 21 Tiempos de ejecución algoritmo DES en segundos 38
Tabla 22 Tiempos de ejecución algoritmo TDES en segundos 39
Tabla 23 Tiempos de cifrado AES software en segundos 40
Tabla 24 Tiempos de descifrado de AES Software en segundos. 41
Tabla 25 Tiempos de Cifrado AES Hardware 42
Tabla 26 Tiempos Descifrado AES Hardware 42
Tabla 27 Tiempos RSA 44
Tabla 28 Tiempos RSA 44
Tabla 29 Consumos de RAM (en bits) 45
Tabla 30 Tamaño en la Flash 46
Tabla 31 Memoria RAM para cifrar con DES 54
Tabla 32 Memoria RAM necesaria cifrar con TDES 55
Tabla 33 Memoria RAM para cifrar un texto de 256B con AES software 56
Tabla 34Memoria RAM para cifrar un texto de 1024B con AES software 56
Tabla 35 Memoria RAM para cifrar un texto de 10240B con AES software 57
Tabla 36 Memoria RAM para cifrar un texto de 256B con AES hardware 57
Tabla 37 Memoria RAM para cifrar un texto de 1024B con AES Hardware 58
Tabla 38 Memoria RAM para cifrar un texto de 10240B con AES Hardware 58
Tabla 39 Memoria RAM para cifrar con RSA 59
Tabla 40 Memoria RAM para descifrar con RSA 59

1 Introducción

1.1 Contexto y justificación del trabajo

La seguridad y el procesamiento de datos en tiempo real son dos requisitos básicos en la transmisión de información y en las aplicaciones de los futuros sistemas de comunicaciones. En lo referente a seguridad, las aplicaciones que hacen uso de algoritmos de cifrado deben cumplir un conjunto de requerimientos y retos. Es necesario que el rendimiento de los procesos de cifrado sea compatible con la transmisión de datos por las líneas de comunicación. Los algoritmos de cifrado más empleados son DES, 3DES y AES; en estos nos basaremos para llevar a cabo el proyecto.

Para dar soporte a los distintos sistemas de cifrado estos deben ser sistemas flexibles; una solución posible sería algoritmos de cifrado sobre microprocesadores de propósito general, por ejemplo, en un PC. Sin embargo, puede resultar una solución cara; además cuenta con las vulnerabilidades de los sistemas operativos y de la propia seguridad física.

Otro tipo de aproximación sería el hardware a medida, más robusto, aunque también más caro de desarrollar, tanto en tiempo como económicamente.

Por último y el que vamos a utilizar en este proyecto, los sistemas embebidos reconfigurables, que permiten mantener la flexibilidad y el rendimiento.

Actualmente, en el tiempo del Internet de las cosas, cada vez son más los dispositivos que están conectados; dichos dispositivos están basados en sistemas embebidos reconfigurables. Estos sistemas embebidos tienen muchas limitaciones en cuanto a computación o memoria disponibles, de ahí la motivación de estudiar el rendimiento y aplicabilidad de los algoritmos AES/DES/3Des/RSA sobre una plataforma embebida.

En este proyecto se va a realizar un estudio de los posibles sistemas de autentificación y diferentes algoritmos de cifrado, como DES, 3DES, AES y RSA sobre la placa MSP-EXP432P401R y el estudio del rendimiento de cada uno de ellos sobre dicha placa.

El sistema de cifrado embebido estará compuesto por una serie de aplicaciones para MSP432, las cuales serán capaces de recibir, a través del puerto serie, una serie de datos y una clave para poder cifrar y descifrar dichos datos en los algoritmos de cifrado anteriormente citados.

1.2 Descripción del trabajo

El sistema de cifrado embebido se trata de un sistema con el cual podremos encriptar y desencriptar con los siguientes algoritmos: DES, TDES, AES y RSA. Dicho sistema estará compuesto por una placa MSP432, gestionada mediante el sistema operativo FreeRtos, a la

cual nos podremos conectar mediante USB para proporcionarle la clave y el texto que deseemos cifrar o descifrar.

Para facilitar las tareas de conexión con la placa se ha creado una aplicación Java que permite el envío y recepción de los datos que se desean cifrar y descifrar usando un protocolo simple, lo que facilita la interacción con la placa y permite el envío y recepción de los datos de una manera cómoda, segura y transparente para el usuario. Esta aplicación también permite la comprobación del cifrado, ya que también cifra y descifra en los diferentes algoritmos utilizados. Además, dicha aplicación permite guardar y recuperar los datos recibidos para facilitar el uso de la misma.

Como el cifrado y descifrado de los diferentes mensajes puede afectar al rendimiento, se realizará un estudio del gasto computacional y de memoria de los distintos sistemas de cifrado utilizados para así dilucidar cuál es el mejor algoritmo que podemos aplicar, así como comprobar el uso del hardware del sistema embebido a la hora de realizar las tareas de cifrado y descifrado.

1.3 Objetivos del TFC.

Los objetivos principales del proyecto son:

- Desarrollo del algoritmo de cifrado DES para la placa MSP432P401R.
- Desarrollo del algoritmo de cifrado 3DES para la placa MSP432P401R.
- Desarrollo del algoritmo de cifrado AES para la placa MSP432P401R, tanto por software como utilizando los recursos hardware de la misma.
- Desarrollo de una herramienta de escritorio que permita la comunicación con la placa.
- Valorar el gasto computacional y el rendimiento de los distintos algoritmos sobre la placa.

Objetivos secundarios:

- Desarrollo del algoritmo de cifrado RSA para la placa MSP432P401R.

1.4 Enfoque y método seguido.

Para llevar a cabo la realización de este proyecto, se ha requerido inicialmente un proceso de formación, documentación y aprendizaje sobre la placa MSP432, que es la placa seleccionada para la realización de dicho proyecto.

También se ha requerido la formación en FreeRTOS, sistema en tiempo real que gestionará los recursos de la placa.

Y, por último, también ha sido necesaria la documentación en los diversos algoritmos de cifrado con los que va a poder cifrar el sistema, así como los modos de cada uno de ellos.

Adicionalmente, se ha realizado una división y planificación de tareas del proyecto para llevar un control y supervisión con el objetivo de cumplir con los plazos principales de entrega establecidos.

1.5 Planificación del trabajo.

Como principal pauta para la realización de todo el proyecto como conjunto, se ha dividido en diferentes fases y tareas el trabajo a realizar coincidiendo con las fechas de entrega parciales durante el semestre. Con la ayuda de los objetivos básicos y secundarios expuestos en la propuesta inicial del proyecto, se han definido una serie de tareas y subtareas a completar para alcanzar dichos objetivos.

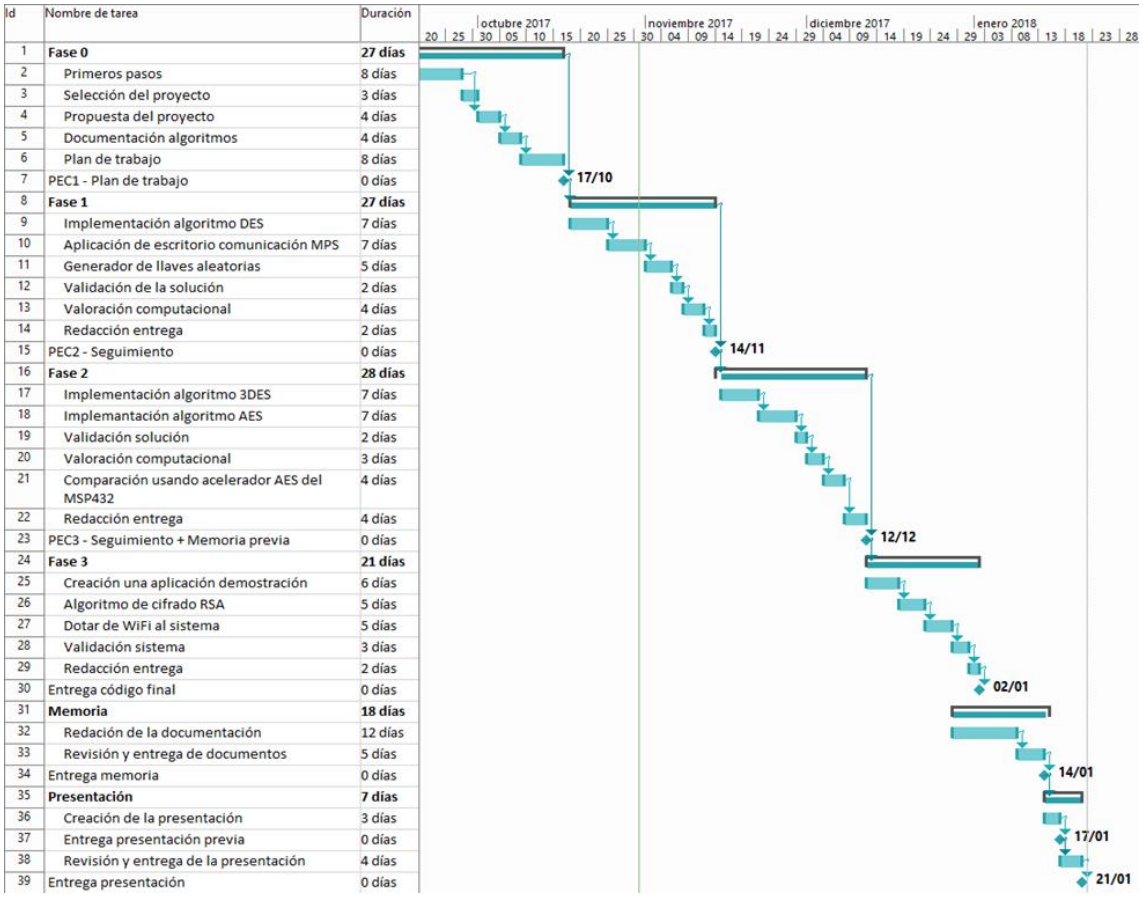
Las diversas tareas que se han llevado a cabo son las siguientes:

Tarea	**Nombre**	**Descripción**
Fase 0: Documentación y preparación del proyecto.		
1	Primeros Pasos	Durante esta tarea se aprenderá el uso de la placa MSP432 y la programación con FreeRTOS.
2	Selección del Proyecto	Nos documentaremos sobre los posibles proyectos con el fin de elegir el que más se adapte a nosotros.
3	Propuesta de Proyecto	Se redactará y enviará una propuesta con el proyecto elegido.
4	Documentación sobre los algoritmos de cifrado	Se adquirirán los conocimientos necesarios sobre los diferentes sistemas de cifrado y de autentificación para poder llevar a cabo correctamente el proyecto.
5	Plan de trabajo	Se realizará el plan de trabajo y la planificación.
Fase 1: Algoritmo de cifrado DES		
6	Implementación del algoritmo DES	Se programará el algoritmo de cifrado DES para la placa MSP432P401R.
7	Aplicación de escritorio	Se creará una aplicación de escritorio para comunicarse con la placa a fin de poder mandar los datos para el cifrado o descifrado.
8	Generador de llaves aleatorias	Se creará un generador de llaves aleatorias para cada tipo de cifrado.
9	Validación de la solución	Se harán pruebas suficientes para ver que el algoritmo funciona correctamente.
10	Valoración computacional	Se llevará a cabo un estudio del gasto computacional

		y de memoria del algoritmo de cifrado.
11	Redacción entrega PEC	Se realizará la documentación necesaria del seguimiento.
Fase 2: Ampliación de los algoritmos		
12	Implementación 3DES	Se programará el algoritmo de cifrado 3DES para la placa MSP432P401R.
13	Implementación AES	Se programará el algoritmo de cifrado AES para la placa MSP432P401R.
14	Validación de la solución	Se llevarán a cabo las pruebas necesarias para el correcto funcionamiento de los algoritmos desarrollados.
15	Valoración computacional	Se llevará a cabo un estudio del gasto computacional y de memoria del algoritmo de cifrado 3DES y AES.
16	Redacción entrega PEC	Se realizará la documentación necesaria del seguimiento y la entrega previa de la memoria.
Fase 3: Objetivos extras		
17	Creación de una aplicación demostración	Se creará una aplicación de demostración del cifrado y descifrado de los diferentes algoritmos.
18	Ampliación del algoritmo con cifrado RSA	Se ampliará el algoritmo para que sea capaz de encriptar y desencriptar mediante el algoritmo de RSA.
19	Dotar de WiFi al sistema	Se dotará al sistema de conectividad WiFi.
20	Validación del sistema	Se llevarán a cabo las pruebas necesarias para comprobar que todo el sistema funciona correctamente en su conjunto.
21	Redacción entrega PEC	Se realizará la documentación necesaria del seguimiento y el código final.
Fase 4: Documentación		
23	Memoria	Se redactará la memoria final.
24	Presentación	Se creará la presentación del proyecto.
25	Entrega del proyecto	Se entregará el proyecto.

Tabla 1 Tareas del proyecto

Inicialmente se planificaron las tareas según el siguiente cronograma:



Id	Nombre de tarea	Duración
1	**Fase 0**	**27 días**
2	Primeros pasos	8 días
3	Selección del proyecto	3 días
4	Propuesta del proyecto	4 días
5	Documentación algoritmos	4 días
6	Plan de trabajo	8 días
7	PEC1 - Plan de trabajo	0 días
8	**Fase 1**	**27 días**
9	Implementación algoritmo DES	7 días
10	Aplicación de escritorio comunicación MPS	7 días
11	Generador de llaves aleatorias	5 días
12	Validación de la solución	2 días
13	Valoración computacional	4 días
14	Redacción entrega	2 días
15	PEC2 - Seguimiento	0 días
16	**Fase 2**	**28 días**
17	Implementación algoritmo 3DES	7 días
18	Implemantación algoritmo AES	7 días
19	Validación solución	2 días
20	Valoración computacional	3 días
21	Comparación usando acelerador AES del MSP432	4 días
22	Redacción entrega	4 días
23	PEC3 - Seguimiento + Memoria previa	0 días
24	**Fase 3**	**21 días**
25	Creación una aplicación demostración	6 días
26	Algoritmo de cifrado RSA	5 días
27	Dotar de WiFi al sistema	5 días
28	Validación sistema	3 días
29	Redacción entrega	2 días
30	Entrega código final	0 días
31	**Memoria**	**18 días**
32	Redación de la documentación	12 días
33	Revisión y entrega de documentos	5 días
34	Entrega memoria	0 días
35	**Presentación**	**7 días**
36	Creación de la presentación	3 días
37	Entrega presentación previa	0 días
38	Revisión y entrega de la presentación	4 días
39	Entrega presentación	0 días

Figura 1 Cronograma Inicial

Aunque como veremos a continuación el cronograma se ha ido a adaptando. Esto ha sido debido a la inexperiencia a la hora de realizar la planificación de un proyecto como este, ha habido tareas que inicialmente no se han tenido en cuenta que una vez puestos en faena se han tenido que introducir, tareas que se han realizado en un menor tiempo de lo que en un principio se preveía y tareas tareas que ha sucedido todo lo contrario. El cronograma final queda como el siguiente:

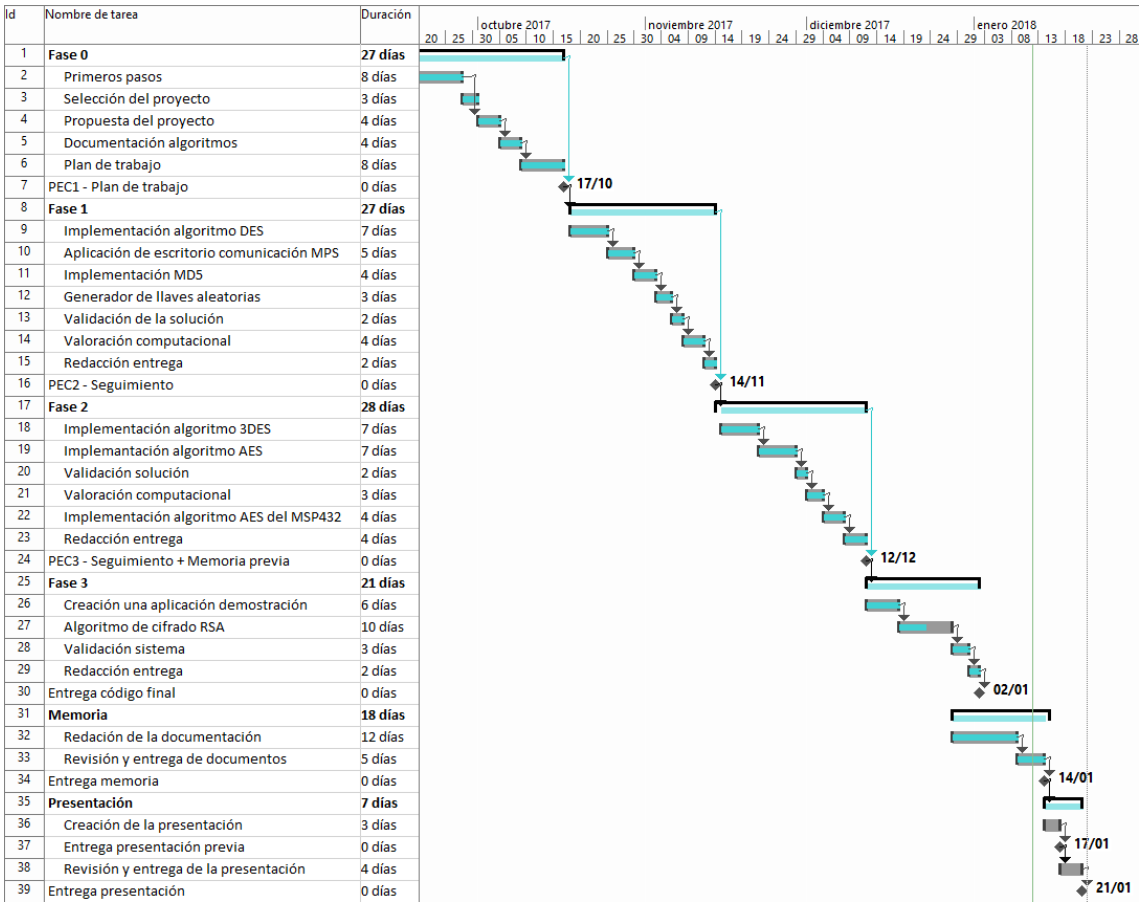


Figura 2 Cronograma Final

Por otro lado, se ha diseñado un plan de riesgos y medidas correctoras para disponer de un plan de respuesta ante la posible ocurrencia de problemas previsibles en un proyecto de estas características.

Riesgos	Probabilidad	Impacto	Solución
Utilización de tecnología poco conocida	MEDIA	MEDIO	Se dedicará tiempo a estudiar la tecnología que se usará durante el proyecto.
Cambio de situación laboral	MEDIA	MEDIO	Se recortará el alcance de las tareas secundarias.
Enfermedad	BAJA	BAJO	Se recortará el alcance de las tareas secundarias.
Fallo del material	BAJA	MEDIO	Se repondrá lo antes posible y se recortará el alcance de las tareas secundarias.
Pérdida de datos fortuita	BAJA	MEDIO	Se realizará una copia de seguridad diaria en un dispositivo secundario.

Tabla 2 Plan de riesgos

1.6 Recursos empleados

Para llevar a cabo este proyecto usaremos los módulos anteriormente expuestos:

- MSP432P401R LaunchPad™ Development Kit

Además de esto se utilizará un ordenador personal.

Y el siguiente software de desarrollo:

- Microsoft Project 2016, utilizado para la planificación del proyecto y su seguimiento.
- Suite LibreOffice, para la redacción de documentos y elaboración de presentaciones.
- Code Composer Studio 7.3.0, compilador para el microprocesador.
- FreeRTOS, sistema operativo en tiempo real para el microprocesador.
- Eclipse Oxigen 1, conjunto de herramientas de programación de código abierto.

1.7 Productos obtenidos

Un sistema de cifrado empotrado, capaz de cifrar en DES, 3DES y AES, así como la verificación de que el cifrado se realiza correctamente.

Un programa de escritorio para la comunicación con el sistema empotrado.

Estudio de valoración computacional y de memoria requeridos para cada tipo de cifrado.

1.8 Breve descripción de los otros capítulos de la memoria.

En el siguiente capítulo se indican los antecedentes y se analizan los diferentes tipos de tecnologías disponibles en el mercado para la realización del proyecto diseñado.
Capítulo 3: Describe desde una perspectiva funcional el sistema de cifrado y la interacción con la aplicación de escritorio y su descripción.
Capítulo 4: Explicación detallada y técnica del diseño de la aplicación, sus librerías y sus módulos.
Capítulo 5: Se realizará una comparativa entre los distintos algoritmos de cifrado en cuanto a tiempo de procesado, memoria necesitada, etc.
Capítulo 6: Se analiza la viabilidad técnica del producto.
Capítulo 7: Realiza un análisis de las conclusiones producto de este proyecto.

2 Antecedentes

2.1 Estado del arte

Los sistemas encastrados o embebidos son sistemas de computación pensados para realizar una o unas pocas funciones dedicadas, frecuentemente en tiempo real. La mayoría de estos sistemas esta muy limitado tanto en Memoria RAM, como en flash y velocidad del procesador.
El desarrollo de los sistemas empotrados ha hecho que cada vez sean más populares y podamos encontrarlos en más sitios. Además, el bajo precio y la cada vez mayor facilidad de programación han ayudado a su popularización.
En la actualidad existe una enorme cantidad de sistemas embebidos y plataformas para el desarrollo de test y pruebas. Veamos unos ejemplos:

MSP432P401R [8]

El equipo de desarrollo de LaunchPad MSP-EXP432P401R de Texas Instruments permite desarrollar aplicaciones de alto rendimiento que aprovechan el funcionamiento de bajo consumo. El equipo incorpora el MSP432P401R, que incluye un núcleo ARM Cortex M4F de 48 MHz, con un consumo energético en modo activo de 95 uA/MHz y funcionamiento RTC de 850 nA, ADC SAR de 12 bits diferencial y acelerador AES256.

Características

- MCU MSP432P401R de baja potencia, alto rendimiento
- M4F ARM Cortex de 48 MHz y 32 bits con unidad de punto flotante y aceleración DSP
- Consumo de energía: 9 5uA/MHz activo, y 850 nA RTC en modo en espera
- Analógico: ADC SAR de 12 bits, 24 canales y 1 MSPS diferencial, dos comparadores
- Digital: Acelerador estándar de cifrado avanzado (AES256), CRC, DMA, HW MPY32
- Memoria: 256 KB de memoria Flash, 64 KB de memoria RAM
- Temporizadores: 4x16 bits, y 2x32 bits
- Comunicación: Hasta 4 I2C, 8 SPI, 4 UART
- Conector de BoosterPack de 40 pines y compatibilidad para BoosterPacks de 20 pines
- Emulador XDS-110ET en la placa que incorpora tecnología EnergyTrace+
- 2 botones y 2 LEDs para interacción con el usuario
- UART de canal retroactivo a través de USB con PC
- Precio: 12,64 €

Arduino Zero

En vez del microcontrolador Atmel ATmega basado en arquitectura AVR de 8 bits, el Zero contiene un potente Atmel SAMD21 MCU de 48Mhz con un core ARM Cortex M0 de 32 bits. Con 256 KB de memoria flash, 32 KB de SRAM y una EEPROM de más de 16KB por emulación. El voltaje en el que opera es de 3,3v/5v (7mA) y contiene 14 pines E/S digitales, de los cuales 12 son PWM y UART. En el terreno analógico se dispone de 6 entradas para un canal ADC de 12 bits y una salida analógica para DAC de 10 bits. En definitiva, esta placa va destinada para los que Arduino UNO se les quede corto y necesitan algo más de potencia de procesamiento.[19] Tiene un precio de unos 23 euros.

Arduino Due

Es una placa con un microcontrolador Atmel SAM3X8E ARM Cortex-M3 de 32 bits. Este chips trabaja a 84Mhz (3,3v). Al tener un core a 32 bits permite realizar operaciones con datos de 4 bytes en un solo ciclo de reloj. Además, la memoria SRAM es de 96KB e incorpora un controlador DMA para acceso directo a memoria que intensifica el acceso a memoria que puede hacer la CPU. Para el almacenamiento se dispone de 512KB de flash, una cantidad muy grande de memoria para cualquier código de programación. El sistema dispone de 54 pines de E/S digitales, 12 de ellos pueden ser usados como PWM. También tiene 12 analógicos, 4 UARTs, capacidades de conexión USB OTG, dos conexiones DAC (conversión digital a analógico), 2 TWI, un power jack, SPI y JTAG. [19] Tiene un precio estimado de unos 26 euros.

Parallax Propeller

Contiene un chip Parallax P8X32A Propeller, con arquitectura multicore con CPUs RISC de 32 bits. Su programación se realiza en lenguaje ensamblador o en lenguaje Spin (diseñado por Chip Gracey y el ingeniero Jeff Martin de Parallax). Esta placa, con sus 32KB de RAM y 32KB de ROM, junto con el resto de características la hace idónea para los más profesionales. El mayor problema es que no es una placa open-source. [19]

Conclusión:

Los microcontroladores vistos aquí son solo una mínima parte de los microcontroladores que se pueden encontrar mercado, de los cuales se ha seleccionado para nuestro estudio la placa **MSP432P401R.** Esta placa tiene un precio menor al resto y sus características como el acelerador estándar de cifrado, que nos facilitara el estudio del cifrado AES, son las que más se adaptan a nuestro proyecto.

2.2 Estudio de mercado

En la actualidad hay bastantes alternativas para la encriptación sobre sistemas embebidos, entre los cuales podemos destacar:

Wolfssl (https://www.wolfssl.com): (anteriormente CyaSSL o yaSSL) es una biblioteca SSL/TLS ligera, portátil y embebida, diseñada para ser utilizada por los desarrolladores de sistemas encastrados. Es una aplicación Open Source de TLS (SSL 3.0, TLS 1.0, 1.1, 1.2, 1.3, y DTLS 1.0 y 1.2) escrito en el lenguaje C [22].

Un predecesor de wolfSSL, yaSSL, es una biblioteca SSL basada en C++ para entornos integrados y sistemas operativos en tiempo real con recursos limitados.

Tiene las siguientes características:

- Hasta 20 veces más pequeño que OpenSSL
- Solo requiere 20-100KB Flash
- Solo requiere 1-36KB RAM
- Admite TLS 1, 1.1 y 1.2 (cliente y servidor)
- Admite DTLS 1 y 1.2 (cliente y servidor)
- Funciones hash: MD2, MD4, MD5, SHA-1, SHA-2, SHA-256, SHA-384, SHA-512, BLAKE2b y RIPEMD-160
- Cifrado de bloques, flujos y AEAD: Cifrados AES (CBC, CTR, GCM, CCM), Camellia, DES, 3DES, ARC4, RABBIT, HC-128
- Opciones de clave pública: RSA, DSS, DH, EDH, NTRU
- Encriptación de clave privada: PKCS #8, #5, #12
- Admite certificados PEM y DER
- Generación de claves y soporte ECC
- Generación de certificados
- Capa de compatibilidad con FreeRTOS
- Capa de compatibilidad OpenSSL

OpenSSL (https://www.openssl.org) es un proyecto Open Source, que provee una herramienta comercial y robusta con un conjunto de herramientas para la capa de transporte seguro (TLS,

Transport Layer Security) y SSL (Secure Sockets Layer) siendo también una librería de criptografía de propósito general [21].
Tiene las siguientes características:

- Cifrado de bloques, flujos y AEAD: AES, Blowfish, Camellia, SEED, CAST-128, DES, IDEA, RC2, RC4, RC5, Triple DES, GOST 28147-893
- Funciones HASH: MD5, MD4, MD2, SHA-1, SHA-2, RIPEMD-160, MDC-2, GOST R 34.11-944
- Opciones de clave pública: RSA, DSA, Intercambio de claves Diffie–Hellman, curvas elípticas
- Admite certificados PEM y DER
- Encriptación de clave privada: PKCS #8, #5, #12

mbed TLS (anteriormente PolarSSL) (https://tls.mbed.org) Es una implementación de los protocolos TLS y SSL y de los respectivos algoritmos criptográficos. Está especialmente enfocado para el mundo del Internet de las cosas (Internet of Things, IoT), tiene doble licencia con la versión 2.0 de Apache License (con GPLv2 también disponible) [22]. Está escrito en C.

Sus principales características son las siguientes:

- Requiere menos de 60 KB de espacio de programa y menos de 64 KB de RAM
- Soporte API para el lado del cliente y del lado del servidor, todos los estándares SSL y TLS actuales: SSL versión 3, TLS versión 1.0, TLS versión 1.1 y TLS versión 1.2
- Algoritmos de encriptación simétrica: AES, Blowfish, Triple-DES (3DES), DES, ARC4, Camellia, XTEA
- Algoritmos hash: MD2, MD4, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, RIPEMD-160
- Criptografía de clave pública: RSA, intercambio de claves Diffie-Hellman, criptografía de curva elíptica (ECC) , curva elíptica Diffie-Hellman (ECDH) , DSA de curva elíptica (ECDSA)
- Modos de operación: ECB, CBC, CFB, CTRGCM y CCM
- Admite certificados PEM y DER
- Encriptación de clave privada: PKCS #8, #5, #12
- Generación aleatoria de números

Como acabamos de ver, existen una gran variedad de bibliotecas que podríamos haber utilizado o adaptado para el fin de nuestro proyecto, pero hemos preferido desarrollar de cero las bibliotecas, ya que muchas de las herramientas que estas nos ofrecen son innecesarias en nuestro proyecto. Además, como ya hemos comentado, se va a realizar el estudio comparativo del tiempo, memoria y tamaño utilizado y resulta más fácil analizar estos factores creando nosotros mismos los algoritmos, ya que controlamos estos factores.

2.3 Algoritmos de Cifrado

Los algoritmos criptográficos son funciones matemáticas, que son utilizadas en los procesos de encriptación o desencriptación de datos que serán la entrada o parámetro de estas funciones.

Un algoritmo de cifrado se denomina simétrico cuando utiliza la misma clave para cifrar y descifrar. Los métodos más conocidos de este tipo de cifrado son el **DES**, el **Triple DES** y el **AES**.

Un algoritmo es asimétrico al usar claves diferentes: una pareja compuesta por una clave pública, que sirve para cifrar, y por una clave privada, que sirve para descifrar. El punto fundamental sobre el que se sostiene esta descomposición pública/privada es la imposibilidad práctica de deducir la clave privada a partir de la clave pública. El método más conocido de este tipo de cifrado es el **RSA**.

2.3.1 DES

DES es el algoritmo prototipo del cifrado por bloques, un algoritmo que toma un texto en claro de una longitud fija de bits y lo transforma mediante una serie de operaciones básicas en otro texto cifrado de la misma longitud. En el caso de DES el tamaño del bloque es de 64 bits. DES utiliza también una clave criptográfica para modificar la transformación, de modo que el descifrado sólo puede ser realizado por aquellos que conozcan la clave concreta utilizada en el cifrado. La clave mide 64 bits, aunque en realidad, sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad y después son descartados. Por tanto, la longitud de clave efectiva en DES es de 56 bits y así es como se suele especificar.

El algoritmo se basa fundamentalmente en una combinación de técnicas de confusión y difusión. Se realiza una sustitución seguida de una permutación del texto de entrada en función de la llave. Esta operación se conoce como round o ronda. Este algoritmo realiza 16 rondas, lo que quiere decir que se aplica este proceso 16 veces sobre el texto plano.

2.3.2 Triple DES

Es un algoritmo formado a partir del algoritmo DES. Básicamente realiza el proceso de DES tres veces. La longitud de la clave usada en TDES será de 168 bits (3x56 bits), aunque su eficacia sólo sea de 112 bits. Se continúa cifrando bloques de 64 bits.

2.3.3 AES

Advanced Encryption Standard (AES), también conocido como Rijndael (pronunciado "Rain Doll" en inglés), es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el Gobierno de los Estados Unidos.

El algoritmo se basa en varias sustituciones, permutaciones y transformaciones lineales, cada una ejecutada en bloques de datos de 16 byte. Esas operaciones se repiten varias veces, llamadas rondas. Durante cada ronda, una clave circular única se calcula a partir de la clave de cifrado y se incorpora en los cálculos. Basado en la estructura de bloques de AES, el cambio de un solo bit, ya sea en la clave o en el bloque de texto sin cifrado, da como resultado un bloque de texto cifrado completamente diferente; una ventaja clara sobre los cifrados de flujo tradicionales. La diferencia entre AES-128, AES-192 y AES-256 finalmente es la longitud de la clave: 128, 192 o 256 bits.

2.3.4 RSA

RSA es el algoritmo de cifrado asimétrico más popular en la actualidad. Creado por Ron Rivest, Adi Shamir y Leonard Adleman en 1977.

A diferencia de los sistemas tradicionales de cifrado simétrico, RSA trabaja con dos claves diferentes: una pública y una privada. Ambos trabajan complementarios entre sí, lo que significa que un mensaje cifrado con uno de ellos sólo puede ser descifrado por su contraparte. Dado que la clave privada no puede calcularse a partir de la clave pública, esta está generalmente disponible para el público.

Estas propiedades permiten que los criptosistemas asimétricos se utilicen en una amplia gama de funciones, como las firmas digitales. En el proceso de firma de un documento, una huella digital cifrada con RSA se adjunta al archivo y permite al receptor verificar tanto el remitente como la integridad del documento. La seguridad de RSA se basa principalmente en el problema matemático de la factorización entera. Un mensaje que está a punto de ser cifrado se trata como un gran número. Al cifrar el mensaje, se eleva a la fuerza de la llave y se divide con el resto por un producto fijo de dos números primos. Repitiendo el proceso con la otra clave, el texto sin formato se puede recuperar de nuevo. El mejor método actualmente conocido para romper el cifrado requiere factorizar el producto utilizado en la división. Actualmente, no es posible calcular estos factores para números mayores de 768 bits. Es por eso que los criptosistemas modernos usan una longitud de clave mínima de 3072 bits.

Actualmente el algoritmo es considerado seguro siempre que las llaves utilizadas sean de longitud suficientemente larga (aunque se siguen utilizando de 1024, se recomienda una longitud de 2048).

2.4 Modos de cifrado por bloques

2.4.1 Electronic Code Book (ECB)

Es el más sencillo; en este modo los mensajes se dividen en bloques y cada uno de ellos es cifrado por separado utilizando la misma clave K, como podemos ver en la figura. La desventaja de este método es que a bloques de texto plano o claro idénticos les corresponden bloques idénticos de texto cifrado, de manera que se pueden reconocer estos patrones como guía para descubrir el texto en claro a partir del texto cifrado. De ahí que no sea recomendable para protocolos cifrados.

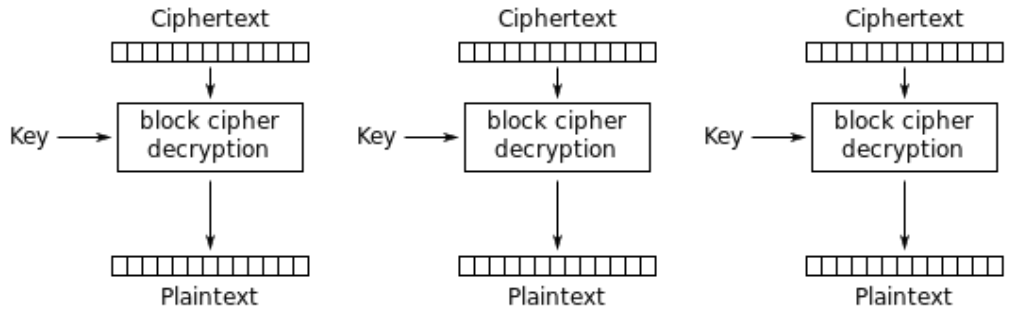


Figura 3 Cifrado Modo ECB [9]

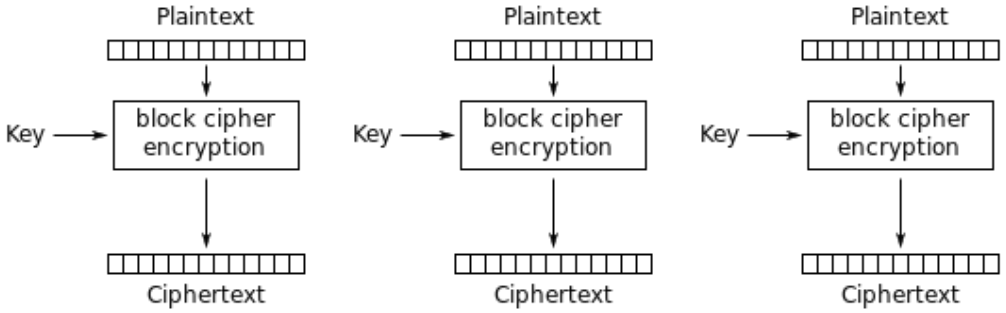


Figura 4 Descifrado Modo ECB [9]

2.4.2 Cipher Block Chaining (CBC)

Cifrado con encadenamiento de bloque, en el cual el resultado o salida de la encriptación de un bloque será utilizado en el proceso de cifrado del siguiente. Es decir, a cada bloque de texto plano se le aplica la operación XOR con el bloque cifrado anterior antes de ser cifrado (como podemos ver en la figura 5), excepto al primero, que utilizamos el contenido del vector de iniciación o IV proporcionado. De esta manera se consigue que dos mensajes iguales no generen el mismo texto cifrado. El descifrado sigue el esquema de la figura 6.

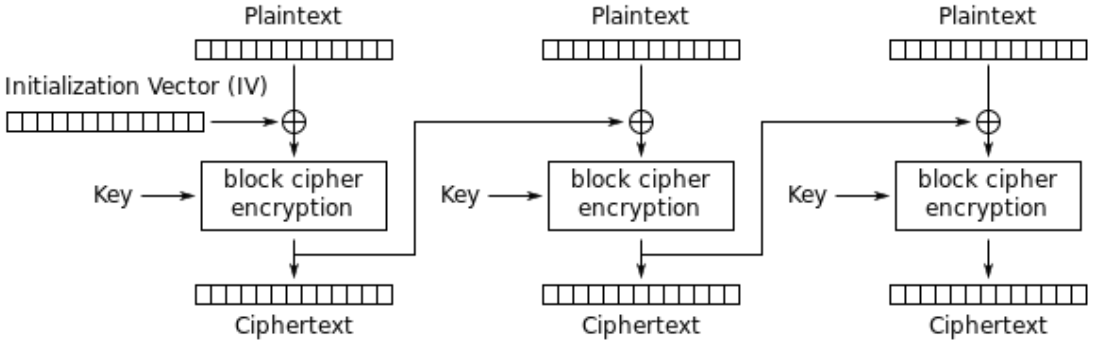


Figura 5 Cifrado Modo CBC [9]

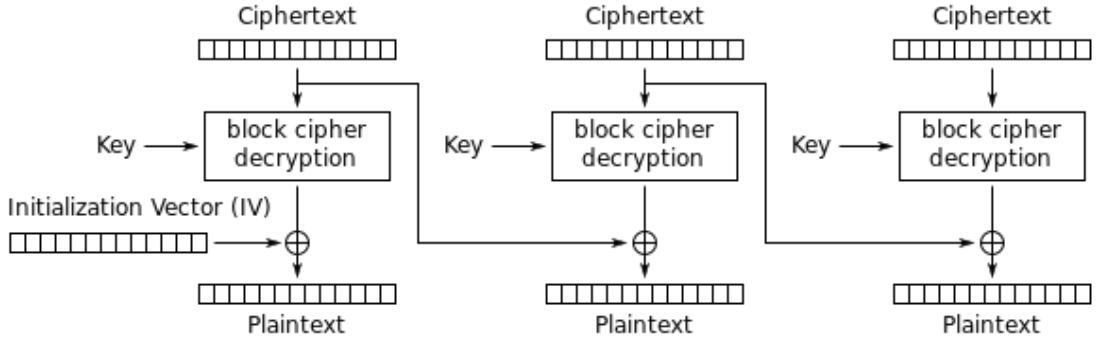


Figura 6 Descifrado Modo CBC [9]

2.4.3 Cipher Feedback (CFB)

También llamado cifrado retroalimentado. Es un método que permite utilizar cifradores por bloques como cifradores por flujo. De esta forma, la información puede ser encriptada en tamaños menores a la longitud del bloque. En las figuras 7 y 8 se puede ver los pasos para el cifrado y descifrado en este modo.

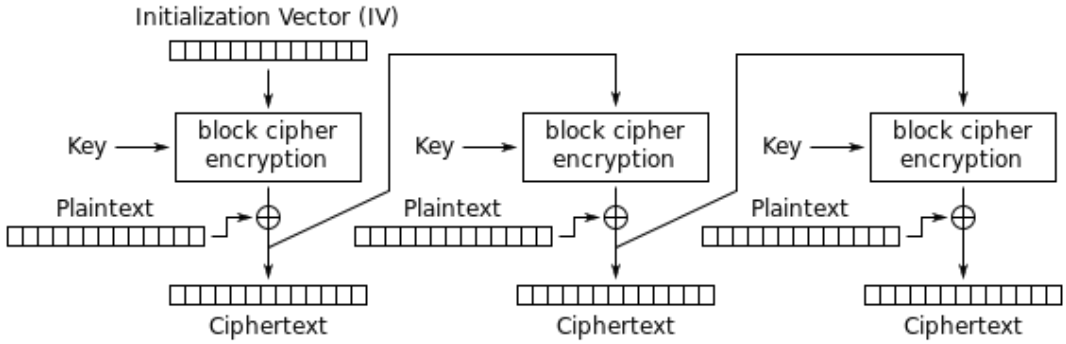


Figura 7 Cifrado Modo CFB [9]

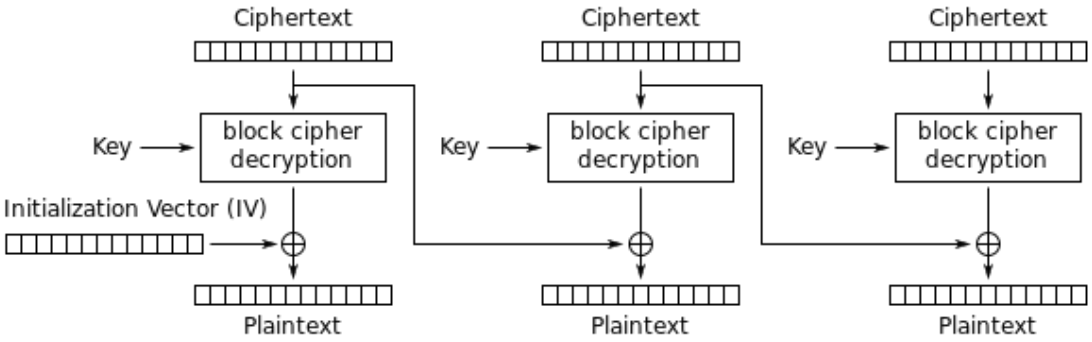


Figura 8 Descifrado Modo CFB [9]

2.4.4 Ouput feedback (OFB)

También llamado retroalimentación de salida. Se trata de otro método para utilizar un cifrador por bloques como un cifrador por flujo. En las figuras 8 y 9 se muestran los pasos de la encriptación y desencriptación respectivamente.

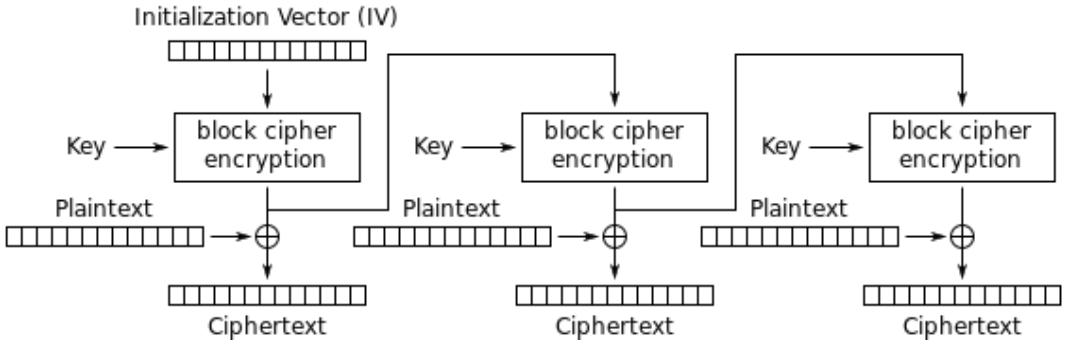


Figura 9 Cifrado Modo OFB [9]

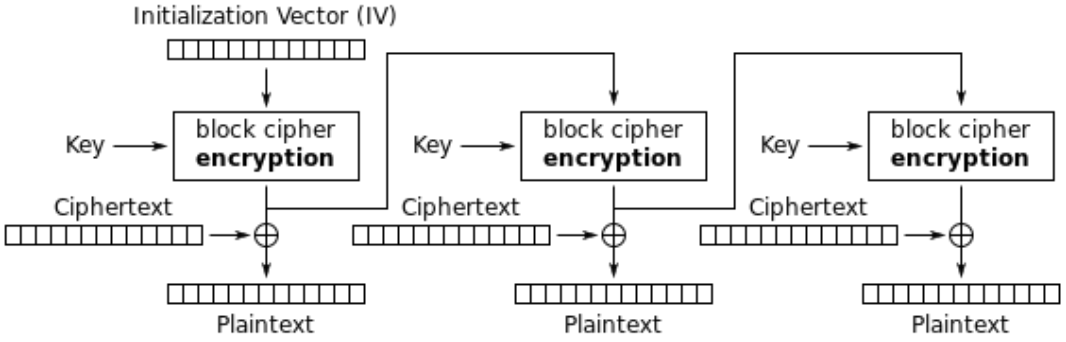


Figura 10 Descifrado Modo OFB [9]

2.4.5 Propagating Cipher Block Chaining (PCBC)

Se diseñó para provocar pequeños cambios en el texto cifrado para propagarse indefinidamente al descifrar, así como al encriptar. A cada bloque de texto, antes de cifrarse, se le aplica la operación xor con el texto plano anterior y el bloque cifrado anterior, como puede verse en la figura 11. Al igual que con el modo CBC, se usa un vector de inicialización en el primer bloque. La forma de desencriptar se puede ver en la figura 12.

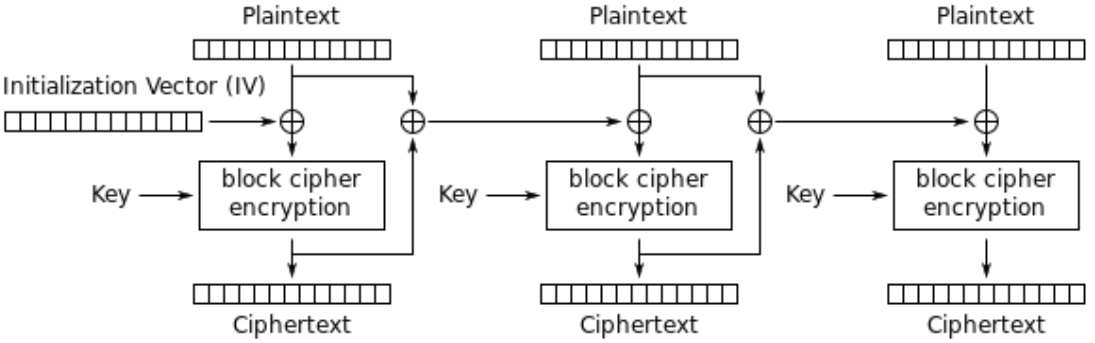


Figura 11 Cifrado Modo PCBC [9]

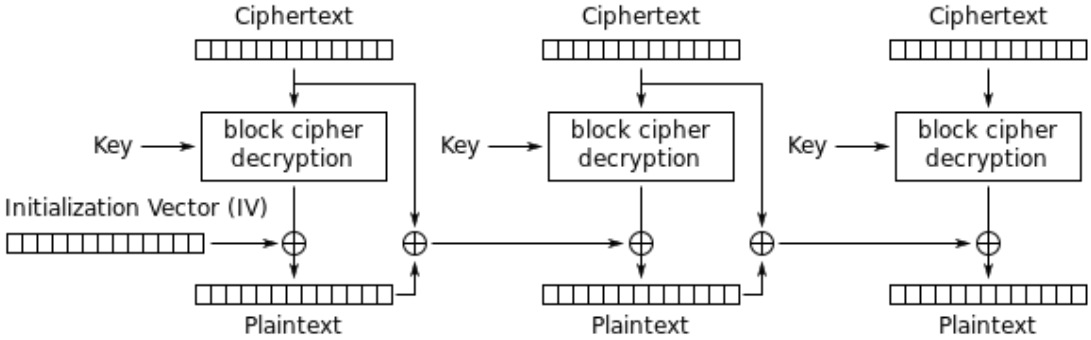


Figura 12 Descifrado Modo PCBC [9]

3 Descripción funcional

En este capítulo se describe funcionalmente el sistema. Este debe ser capaz de recibir, encriptar o desencriptar mensajes en los diversos algoritmos que se han comentado ya, DES, TDES, AES o RSA, y devolver la respuesta.

La interacción con el usuario se lleva a cabo con un ordenador conectado a la placa por un puerto USB y una aplicación Java que es capaz de conectarse a la placa para enviar y recibir mensajes. Además, es capaz de cifrar y descifrar mensajes para comprobar su correcto funcionamiento.

3.1 Sistema cifrado embebido

3.1.1 Diagrama de bloques del sistema

La siguiente Figura muestra el diagrama de funcional del sistema.

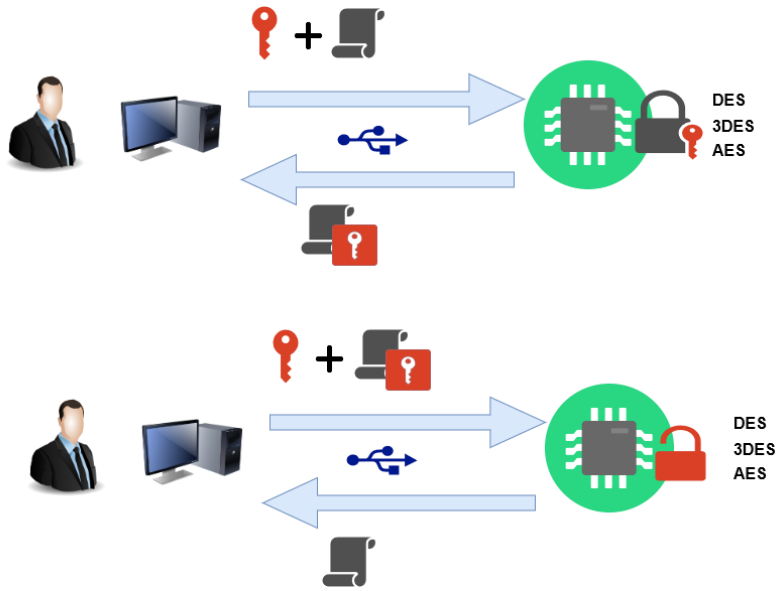


Figura 13 Diagrama funcional del sistema

3.1.2 Descripción de la red

La forma que tenemos de conectar nuestro sistema con el ordenador personal es mediante un cable USB, sobre el cual se creará una conexión UART con la que la aplicación podrá comunicarse con la placa y mandar de esta manera la información necesaria para la encriptación y desencriptación. La siguiente figura muestra el diagrama de conexión del sistema.

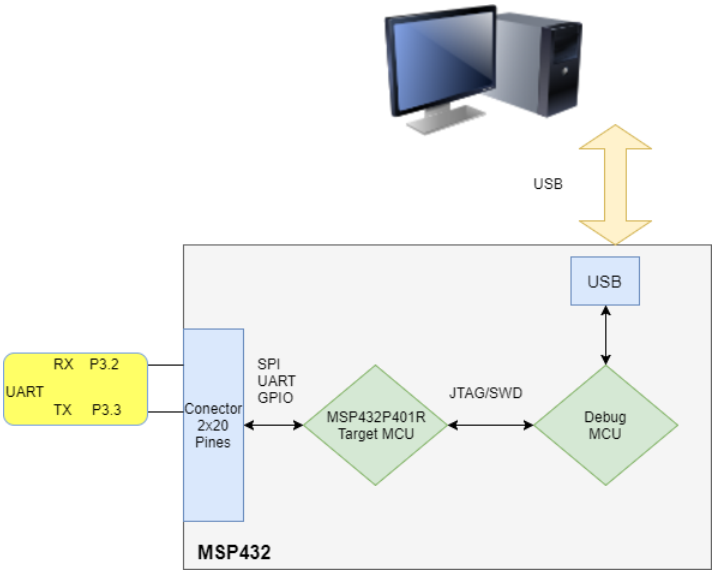


Figura 14 Diagrama de conexión

3.1.3 Interacción de los distintos objetos del sistema

Las interacciones que podemos realizar con nuestro sistema son:

- Envío del mensaje, clave y vector a la placa.
- Cifrado y descifrado con el algoritmo DES.
- Cifrado y descifrado con el algoritmo TDES.
- Cifrado y descifrado con el algoritmo AES tanto por software como por hardware.
- Cifrado y descifrado con el algoritmo RSA.
- Autentificación básica mediante RSA.
- Verificación de la información enviada y recibida mediante MD5

3.2 Aplicación de Usuario

3.2.1 Diagrama de bloques

En la siguiente figura podemos el diagrama de bloques de la aplicación del usuario.

Figura 15 Diagrama de bloques aplicación de usuario.

3.2.2 Interacción de los distintos objetos de la aplicación de usuario

La aplicación de usuario permite la conexión y comunicación con la placa de una manera fácil y transparente para el usuario. Una vez conectada la aplicación con la placa permite las siguientes funciones:

Envío de textos: Permite el envío, de una manera transparente para el usuario, de la información necesaria para el cifrado o descifrado de los mensajes. Una vez que se ha enviado dicha información, la aplicación verifica la correcta recepción de dicha información por parte de la placa mediante MD5.

Recepción de la información: Una vez que la información es tratada, ya sea para cifrar o descifrar los datos enviados, estos son devueltos a la aplicación, la cual se encarga de verificar que se han recibido correctamente mediante MD5 y de presentarlos por pantalla.

Verificación de la correcta codificación: Una vez que se ha recibido la información, la aplicación comprueba que se ha cifrado o descifrado la información correctamente por parte de la placa, procediendo a la encriptación o desencriptación de la información enviada a la placa.

3.3 Sistema empotrado

3.3.1 Diagrama de bloques

En la siguiente figura podemos ver el diagrama de capas del sistema empotrado.

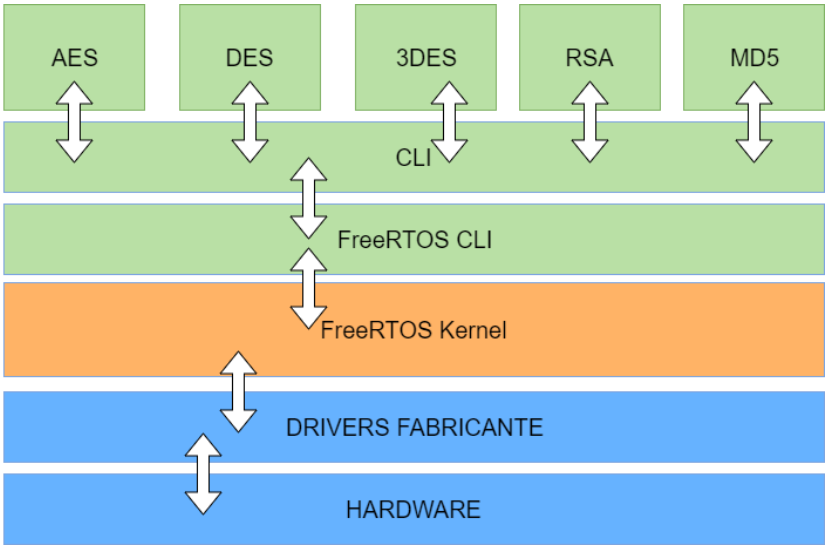


Figura 16 Diagrama de capas sistema empotrado

3.3.2 Diseño y funcionamiento de la aplicación del sistema empotrado

En el diseño seguido para el desarrollo de la aplicación del sistema empotrado destaca la modularidad. Esta se basa en dividir el sistema en distintas capas, desde los drivers a bajo nivel hasta la aplicación principal en el más alto.

A continuación, se describe la función de cada uno de los módulos:

- **Módulo CLI**: ofrece un interface al usuario, es la encargada de recibir la información desde la aplicación del usuario, así como de gestionar estos recursos y hacerlos llegar a los diferentes algoritmos de cifrado. También es el responsable de obtener el resultado de los algoritmos y de hacerlo llegar a la placa.
- **Módulo DES**: es el módulo encargado de cifrar y descifrar la información en el algoritmo DES.
- **Módulo TDES**: es el encargado de cifrar y descifrar la información en el algoritmo TDES, está basado en el módulo DES.
- **Módulo AES:** es el responsable de cifrar y descifrar la información con el algoritmo AES.
- **Módulo RSA**: es el responsable de cifrar y descifrar la información con el algoritmo AES.
- **Módulo MD5**: es el encargado de calcular el MD5.

4 Descripción detallada

En el presente capítulo se realiza la descripción detallada del sistema, la aplicación de usuario y el sistema empotrado. En el capítulo anterior se describían el diseño y las partes del sistema de manera funcional, ahora se profundizará en aspectos más técnicos, tanto en hardware como en software, que permitirá aclarar conceptos que han podido quedar difusos o faltos de explicación.

4.1 Algoritmo DES

Como podemos ver en la siguiente figura, hay 16 fases idénticas, denominadas rondas; estas rondas van precedidas de una permutación inicial y tras ellas se realiza una permutación final, que es inversa a la primera.

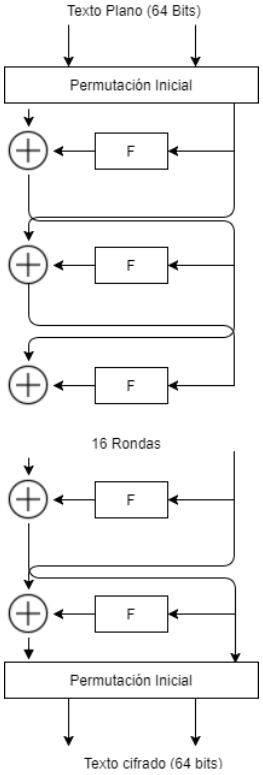


Figura 17 Esquema de cifrado DES

Antes de las rondas, el bloque es dividido en dos mitades de 32 bits y estas son procesadas alternativamente. A este cruzamiento se le conoce como esquema Feistel.

En la función F, que explicaremos más adelante, es donde se realizan las fases de permutación y de sustitución.

Tras esto se realiza la reconexión de las partes izquierda y derecha, seguida de la permutación inicial.

4.1.1 Permutación Inicial

En primer lugar, se realiza una permutación inicial de cada bloque de tal manera que los bits queden de la siguiente manera:

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

Tabla 3 Permutación inicial

4.1.2 División en bloques de 32 bits.

Después de realizar la permutación inicial, el bloque de 64 bits se divide en dos bloques de 32 denominados L y R; los primeros 32 bits serán parte izquierda y los siguientes serán parte derecha.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8

Tabla 4 Estado inicial L

57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tabla 5 Estado Inicial R

Como podemos ver la tabla L contiene sólo los bits pares, mientras que R contiene bits impares.

4.1.3 La función F

La función-F, que podemos ver en la figura, opera sobre medio bloque (32 bits) cada vez y consta de cuatro pasos:

Expansión: la mitad del bloque de 32 bits se expande a 48 bits mediante la permutación de expansión, denominada E en el diagrama, duplicando algunos de los bits, como podemos ver en la siguiente tabla.

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

Tabla 6 Tabla de Expansión

Mezcla: en el resultado se combina con una subclave utilizando una operación XOR. Dieciséis subclaves, una para cada ronda, se derivan de la clave inicial mediante la generación de subclaves que veremos en el siguiente apartado.

Sustitución: tras mezclarlo con la subclave, el bloque es dividido en ocho trozos de 6 bits antes de ser procesados por las S-cajas, o cajas de sustitución. Cada una de las ocho S-cajas reemplaza sus seis bits de entrada con cuatro bits de salida, de acuerdo con una trasformación no lineal, especificada por una tabla de búsqueda. Las S-cajas constituyen el núcleo de la seguridad de DES; sin ellas, el cifrado sería lineal, y fácil de romper.

Los primeros y últimos bits de cada D0i determinan (en valor binario) la línea de la función de selección; los otros bits (2, 3, 4 y 5, respectivamente) determinan la columna. Como la selección de la línea se basa en dos bits, existen 4 posibilidades (0,1,2,3). Como la selección de la columna se basa en 4 bits, existen 16 posibilidades (0 a 15). Gracias a esta información, la función de selección selecciona un valor cifrado de 4 bits.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6

Tabla 7 Matriz de sustitución S1

Cada uno de los 8 bloques de 6 bits pasa a través de la función de selección correspondiente, dando un resultado de 8 valores con 4 bits cada uno. A continuación, están las otras funciones de selección:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11

2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14

Tabla 8 Matriz de sustitución S2

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2

Tabla 9 Matriz de sustitución S3

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2

Tabla 10 Matriz de sustitución S4

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5

Tabla 11 Matriz de sustitución S5

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8

Tabla 12 Matriz de sustitución S6

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3

Tabla 13 Matriz de sustitución S7

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9
1	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5
1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6

Tabla 14 Matriz de sustitución S8

Permutación: finalmente, las 32 salidas de las S-cajas se reordenan de acuerdo a una permutación fija; la P-caja. La siguiente tabla muestra dicha permutación.

16	7	20	21	29	12	1	28
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	2	25

Tabla 15 Matriz de Permutación P

4.1.4 Permutación inicial inversa

Al final de las iteraciones los bloques se vuelven a conectar y se realiza una permutación final, que es inversa a la permutación inicial, tal como se muestra en la siguiente tabla.

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

Tabla 16 Matriz de permutación inversa DES

4.1.5 Generación de Claves

En la siguiente figura se pueden observar los pasos seguidos para la creación de las subclaves de DES a partir de la clave de 64 bits proporcionada por el usuario.

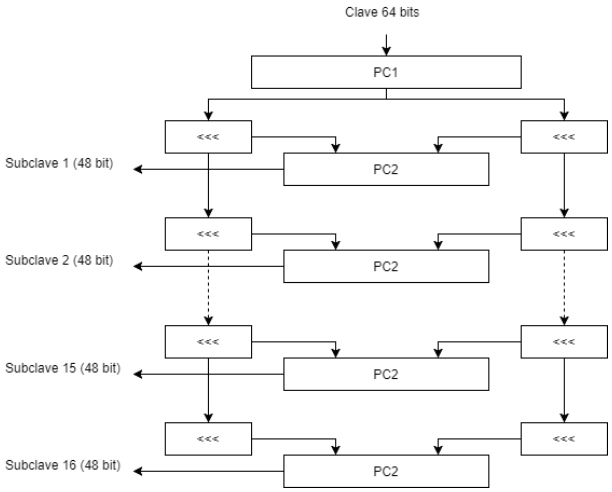


Figura 18 Esquema creación subclaves DES

Veamos ahora los pasos más detenidamente:

En primer lugar, para generar la clave se descartan los bits de paridad para obtener una clave de 56 bits, realizando una Elección Permutada 1 (PC-1) según la siguiente tabla:

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

Tabla 17 Elección Permutada 1

Seguidamente, esta tabla se divide en dos mitades de 28 bits, que se tratarán independientemente una de la otra.

63	55	47	39	31	23	15
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

Tabla 18 Matriz Li

57	49	41	33	25	17	9
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tabla 19 Matriz Ri

El resultado de esta primera permutación se denomina L0 y R0.

Luego, estos dos bloques se rotan hacia la izquierda, de manera que los bits que estaban en la segunda posición pasan a la primera, aquellos que estaban en tercera posición pasan a la segunda, etc.

Los bits que estaban en la primera posición se mueven hacia la última posición.

Los dos bloques de 28 bits se agrupan en un bloque de 56 bits. Este pasa por una permutación, denominada Elección Permutada (PC-2), dando como resultado un bloque de 48 bits que representa la clave Ki.

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

Tabla 20 Elección permutada (PC-2)

Realizando iteraciones del algoritmo es posible obtener las 16 claves K_1 a K_{16} utilizadas en un algoritmo DES.

La generación de claves para descifrado es similar, pero se deben generar las claves en orden inverso.

Estas operaciones para generar las subclaves se llevan a cabo en la función *generate_sub_keys.*

4.2 TDES

Como hemos dicho anteriormente, el algoritmo TDES se basa en la aplicación consecutiva de tres veces el algoritmo DES, como puede verse en la figura.

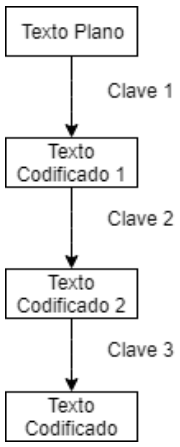


Figura 19 Fases algoritmo TDES

En nuestro algoritmo hemos implementado el tipo DES-EDE3, en el cual se usa una clave diferente para cada una de las operaciones de triple DES y se realiza cifrando con la primera clave, descifrando con la segunda y cifrando con la tercera. Este tipo de cifrado permite obtener el mismo resultado que se obtendría con DES si usamos las tres claves iguales.

4.3 AES.

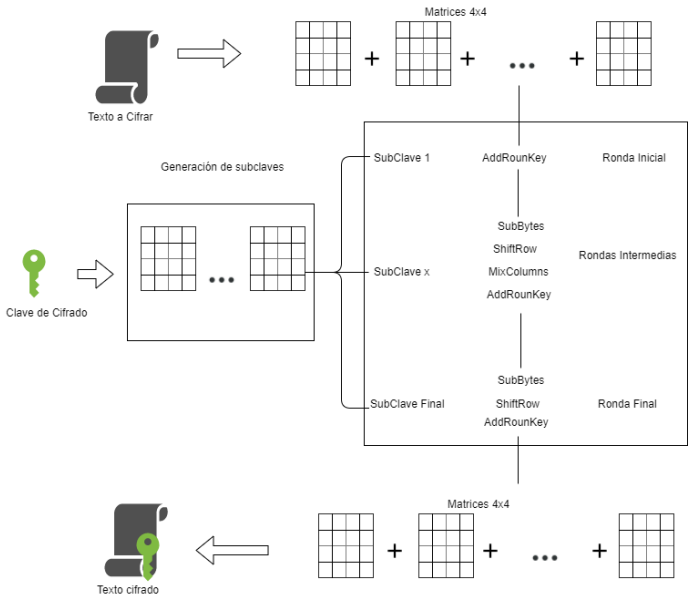


Figura 20 Fases algoritmo AES

Como podemos ver en la figura el cifrado, en AES consta de las siguientes etapas:

- **Expansión de la clave** usando el esquema de claves de Rijndael.
- Una **etapa inicial**, en la que se realiza la función AddRoudKey sobre la primera subclave.
- **Rondas**, que cada una de ellas se aplica sobre una subclave y está compuesta de las siguientes funciones:
    - SubBytes
    - ShiftRows
    - MixColumns
    - AddRoundKey
- Una **etapa final**, compuesta por:
    - SubBytes

- ShiftRows
- AddRoundKey

Ahora desarrollaremos más detenidamente cada una de las funciones por las que está formado dicho algoritmo.

4.3.1 Generación de claves.

Por ser un cifrado simétrico se utiliza la misma clave para encriptar y desencriptar; la longitud de la clave puede ser de 128, 192 o 256, esto permite la implementación de AES-128, AES-192 y AES-256. Aquí explicaremos la generación de claves de 128, aunque el proceso es similar con 192 o 256 bits.

El proceso de generación de subclaves parte de la clave inicial vista como una matriz de 4x4 bytes. En la siguiente figura vemos un ejemplo de una clave de 128 bits.

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Figura 21 Ejemplo clave AES 128bits

Para calcular la primera columna de la siguiente subclave, se toma la última columna de la subclave anterior (en este caso la clave inicial) y se aplica una operación llamada Rotword que consiste en realizar una rotación del primer byte hacia el último lugar en la columna.

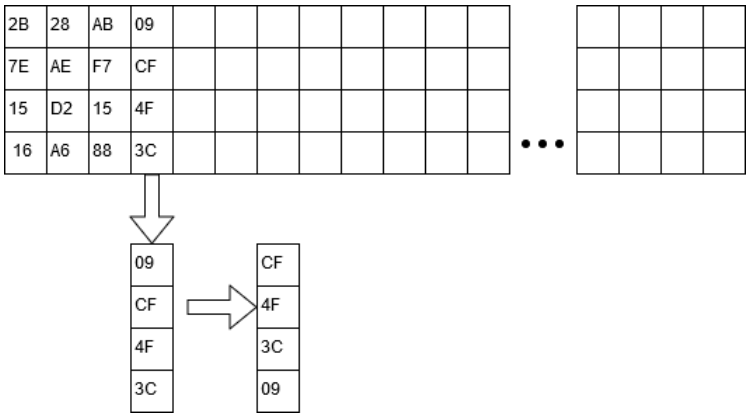


Figura 22 Función Rotword

Luego, a la columna resultante, se aplica una operación llamada SubBytes que consiste en reemplazar cada byte de la columna ya rotada por un byte almacenado en una tabla llamada S-Box; esta tabla contiene precalculados el resultado de aplicarle a cada byte la inversión en el campo GF y una transformación afín. La dimensión de la tabla es de 16x16 bytes, donde los índices tanto de las columnas como de las filas van de 0 a F, para obtener la transformación S-Box de un byte se toman los primeros 4 bits como el índice de la fila de la tabla y los segundos 4 como índice de la columna de la tabla:

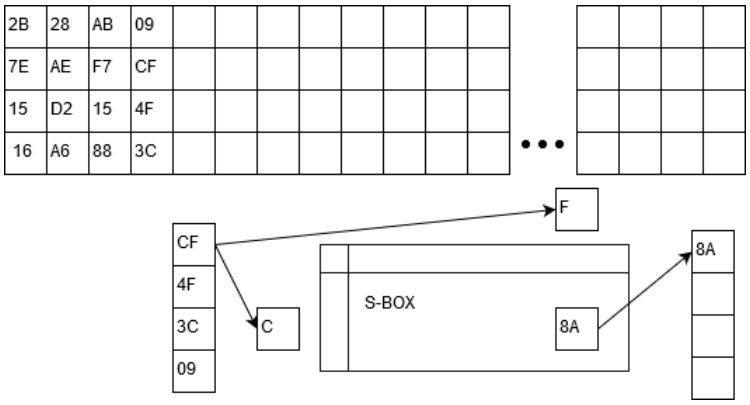


Figura 23 Función SubBytes

Luego, al resultado se le aplica un XOR byte a byte con la columna 4 posiciones atrás (en este caso la primer columna de la clave inicial) y un XOR byte a byte con una columna de una tabla llamada RCON que mantiene en la primera fila constantes 2^i en el campo GF y en las restantes filas 0. Por ser la primera subclave la que estamos calculando se toma para el cálculo la primera columna de la tabla RCON; para las siguientes subclaves se toma la próxima columna no utilizada de esta tabla:

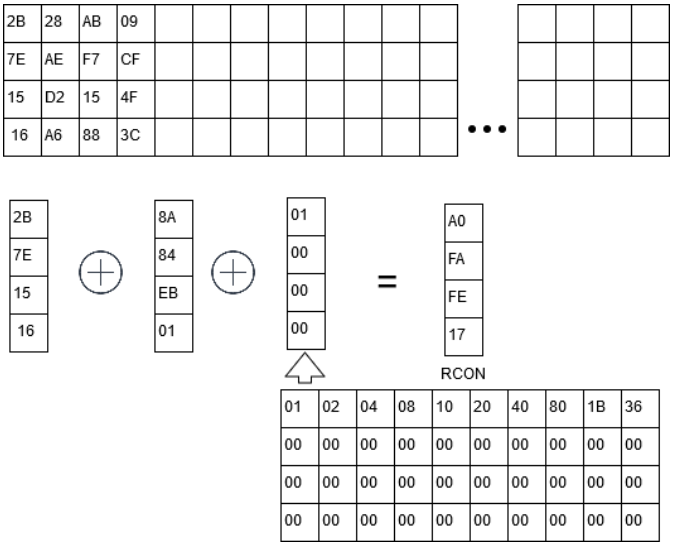


Figura 24 Función XOR sobre la tabla RCON

El resultado de esta última operación será la primera columna de la subclave calculada (en este caso la segunda subclave siguiente a la inicial). Para calcular las tres columnas siguientes se hace un XOR entre la columna anterior y la columna de cuatro posiciones atrás.

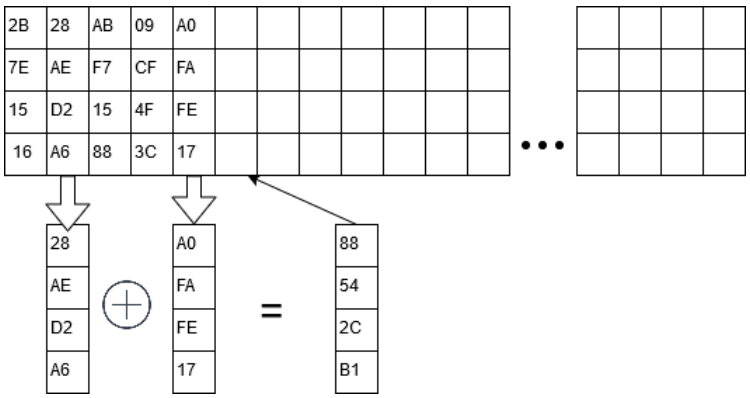


Figura 25 Calculo últimas columnas de la subclave

Para obtener el resto de subclaves se procederá de la misma manera.

4.3.2 Función SubByte:

Cada byte del estado se reemplaza por otro valor de acuerdo a la tabla de sustitución de bytes S-Box ya vista en el cálculo de las subclaves.

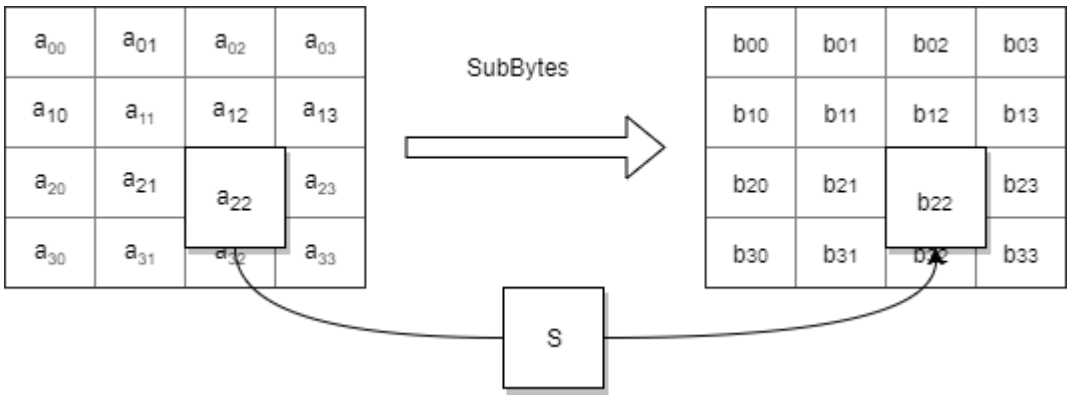


Figura 26 Función SubByte

4.3.3 Función ShiftRow

Consiste en rotar a la izquierda las filas que forman la matriz de Estado Intermedio en la que estamos. La fila 0 siempre permanece sin cambios y el resto de las filas se desplazan un número de posiciones que viene determinado por el tamaño del bloque de cifrado. La siguiente figura muestra cómo se realiza.

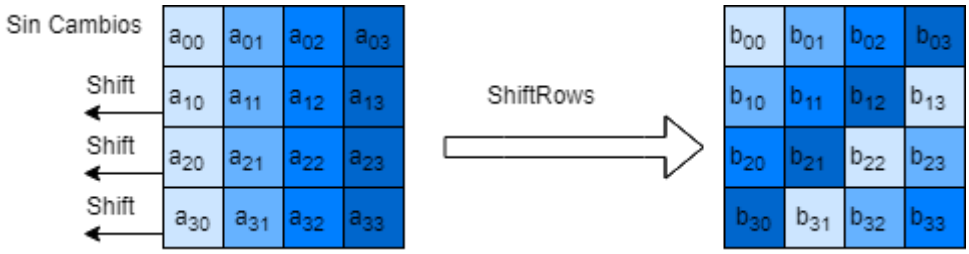


Figura 27 Función ShiftRow

4.3.4 Función MixColumns

Esta función realiza cambios en las columnas de la matriz de Estado Intermedio en que nos encontramos, multiplicando dichas columnas por un polinomio que es coprimo a cada columna de bytes. En esencia, realiza un cambio en las columnas. Podemos ver una representación gráfica en la figura.

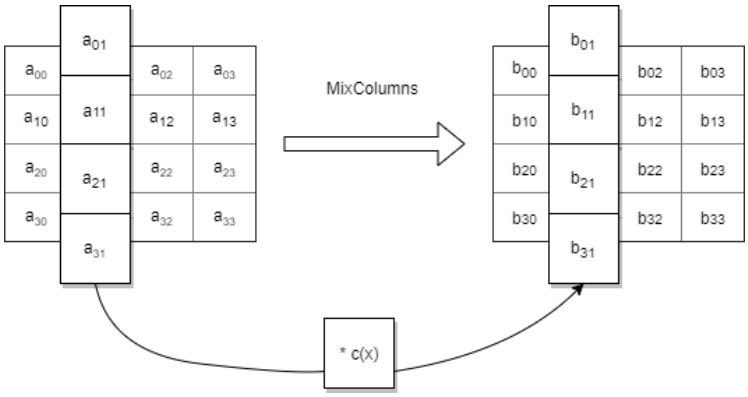


Figura 28 Función MixColumns

4.3.5 Función AddRoundKey

Esta transformación es una operación or-exclusiva (XOR) realizada entre cada bit de la matriz de Estado Intermedia en que nos encontramos y una subclave generada a partir de la clave inicial de cifrado volcada en otra matriz. Como estamos comparando bits, las operaciones XOR serían:

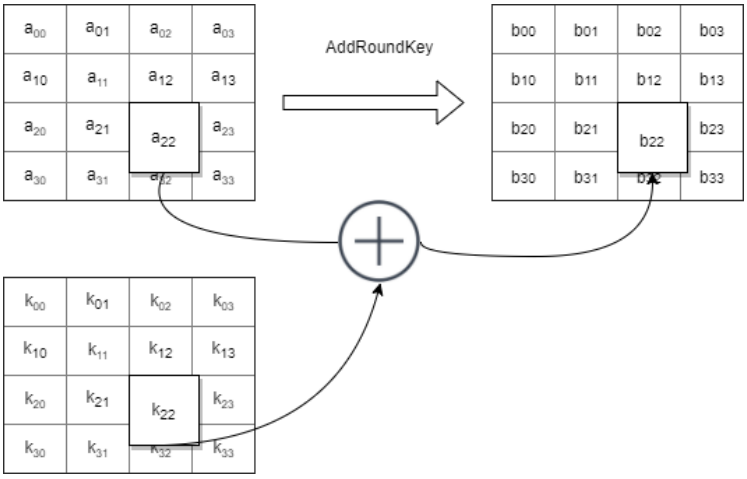


Figura 29 Función AddRoundKey

4.3.6 Descifrado AES

El proceso de descifrado aplica las mismas operaciones que el cifrado, pero de forma inversa, utilizando las mismas subclaves generadas en orden inverso; además se utiliza una matriz distinta en la operación MixColumns de manera de obtiene la inversa de la transformación lineal aplicada en el proceso de cifrado.

4.4 RSA

La seguridad de RSA está basada en la dificultad de factorizar números grandes. La llave privada y la llave públicas son generadas o calculadas en función de un par de números primos.

Para generación del par de llaves (llave pública y llave privada) se deberán seleccionar dos números primos grandes aleatorios, *p* y *q*, y se calcula n como su producto:

$$n = p*q$$

La llave de encriptación *e* será la elegida también de manera aleatoria tal que *e* y *(p-1) (q-1)* sean primos relativos.

La llave de desencriptación d será obtenida despejando la ecuación:

$$ed = 1 \bmod (p-1)(q-1)$$

Es decir:

$$d = e^{-1} \bmod ((p-1)(q-1))$$

Los números e y n componen la llave publica, el número d corresponde a la llave privada; *p* y *q* serán descartados, pero no revelados.

La elección de estos números se lleva a cabo en la función *rsa_gen_keys.*

A la hora de encriptar un mensaje m, este deberá ser dividido en bloques más pequeños que n y cada parte del texto cifrado c será obtenida mediante:

$$c_i = m_i^e \bmod n$$

Esta es la operación que básicamente se lleva a cabo en la función *rsa_encrypt.*

Para la desencriptación, cada parte o bloque del texto cifrado se tomará para calcular:

$$m_i = c_i^d \bmod n$$

Esta es la operación que básicamente se lleva a cabo en la función *rsa_decrypt.*

4.5 MD5

Este algoritmo funciona a través de 4 pasos:

Paso 1: El mensaje será extendido hasta que su longitud en bits sea congruente con 448, módulo 512. Esto es, si se le resta 448 a la longitud del mensaje tras este paso, se obtiene un múltiplo de 512. Esta extensión se realiza siempre, incluso si la longitud del mensaje es ya congruente con 448, módulo 512. El relleno se realiza añadiendo un bit 1 seguido de 0 hasta completar el tamaño.

Paso 2: Una representación de 64 bits de la longitud del mensaje original (antes de aplicar el relleno) se añade al resultado del paso 1. Estos dos pasos permiten hacer coincidir la longitud del mensaje en un múltiplo exacto de 512 bits de longitud (requerido para el resto del algoritmo), en tanto adicionalmente se garantiza que mensajes distintos no serán iguales después del complemento de los bits.

Paso 3: Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Aquí cada una de las letras A, B, C, D representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bytes de menor peso primero:

A: 01 23 45 67

B: 89 AB CD EF

C: FE DC BA 98

D: 76 54 32 10

Paso 4: El mensaje se procesa en bloques de 512 bits a la vez a través de 16 bloques de 32 bits cada uno, para lo cual las cuatro variables de concatenación se copian en variables distintas: a = A, b = B, c = C y d = D.

La parte medular del algoritmo es una función de compresión que consta de cuatro rondas, las cuales tienen una estructura similar, pero cada una utiliza operaciones distintas durante 16 iteraciones; cada operación realiza una función no lineal sobre tres de las variables a, b, c y d y el resultado es sumado a la cuarta variable que no fue elegida, un sub-bloque del texto y una constante.

A ese resultado se le aplica una rotación circular a la izquierda un número variable de bits y se suma el resultado a una de las variables a, b, c o d. Finalmente el resultado reemplaza a una

de las variables a, b, c o d. La salida de la cuarta ronda se suma a la entrada de la primera en una operación modular 2^{32}.

Hay cuatro operaciones no lineales utilizadas, una para cada ronda:

F(b,c,d) = (b AND c) OR (NOT b AND b)

G(b,c,d) = (b AND d) OR (c AND NOT d)

H(b,c,d) = b XOR c XOR d

I(b,c,d)= c XOR (b OR NOT d)

Estas funciones son designadas de forma que si los bits que corresponden a, b, c y d son independientes y no perjudiciales, cada bit del resultado también será independiente y no perjudicial. La función **F** es una función condicional: *If* b *then* c *else* d; de manera similar **G**: *If* d *then* b *else* c, en tanto la función **H** genera un bit de paridad.

Si M_j representa el j-ésimo sub-bloque del mensaje (desde 0 hasta 15), *y*<<<*s* representa un cambio circular a la izquierda de s bits; entonces las cuatro operaciones son:

$FF(a,b,c,d,M_j,s,t_i)$ denota $a = b+((a+F(b,c,d)+M_j+t_i)<<<s)$

$GG(a,b,c,d,M_j,s,t_i)$ denota $a = b+((a+G(b,c,d)+M_j+t_i)<<<s)$

$HH(a,b,c,d,M_j,s,t_i)$ denota $a = b+((a+G(b,c,d)+M_j+t_i)<<<s)$

$II(a,b,c,d,M_j,s,t_i)$ denota $a = b+((a+I(b,c,d)+M_j+t_i)<<<s)$

Las constantes ti se seleccionaron de la siguiente manera:

En los pasos i, ti es la parte entera de 2^{32} abs(sen(i)), en donde i está en radianes.

Paso 5: Al final de todos los ciclos, a, b, c y d son sumados a A, B, C y D respectivamente y el algoritmo continúa con el siguiente bloque de datos. Y después de que todos los bloques de 512 bits (cada uno) han sido procesados, se obtiene la salida final, esto es, la concatenación de A, B, C y D que produce un bloque de 128 bits.

5 Resultados estudio comparativo de los algoritmos.

Uno de los objetivos de nuestro proyecto era realizar un estudio de la valoración computacional.

Se han tenido en cuenta tres criterios:

- El tiempo necesario para la encriptación o desencriptación de la información.
- Memoria RAM necesaria para encriptar mensajes de distintos tamaños en los distintos algoritmos y sus distintos modos.
- Memoria flash necesaria para cada uno de los algoritmos.

5.1 Tiempo.

Para llevar a cabo el estudio comparativo del tiempo necesario para encriptar o desencriptar con los diversos algoritmos y los diversos modos que tiene cada algoritmo, se ha tomado el tiempo antes de comenzar el cifrado o descifrado y una vez ha terminado dicha tarea, para textos de 256B, 1024B, 10240B.

5.1.1 Tiempos DES

En la siguiente tabla podemos observar los tiempos recogidos para el algoritmo DES con textos de tamaños 256B, 1024B y 10240B, tanto para cifrar como para descifrar.

	DES									
	Encriptar					Desencriptar				
	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC
256B	0,102	0,104	0,1	0,1	0,101	0,102	0,099	0,103	0,101	0,097
1024B	0,382	0,381	0,383	0,384	0,385	0,383	0,393	0,384	0,381	0,384
10240B	3,752	3,761	3,762	3,758	3,765	3,753	3,758	3,763	3,757	3,766

Tabla 21 Tiempos de ejecución algoritmo DES en segundos

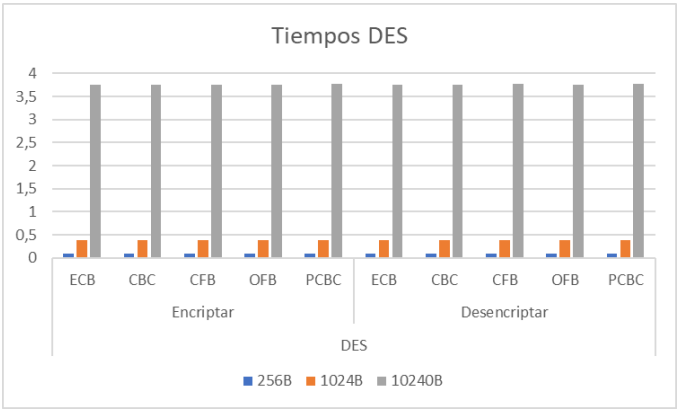


Figura 30 Tiempos de ejecución algoritmo DES en segundos

Como podemos observar, entre los diversos modos no hay mucha diferencia de tiempos de ejecución, casi todo tienen tiempos parecidos. Lo que sí influye es el tamaño del mensaje, a un mayor tamaño, mayor número de bloques que tienen que ser procesados y, por lo tanto, un mayor tiempo. Además, podemos apreciar que el tiempo crece proporcionalmente, o casi, al tamaño del mensaje procesado.

También podemos apreciar que el tiempo necesario para cifrar o descifrar un mensaje del mismo tamaño requiere un tiempo similar, esto es porque el algoritmo DES utiliza las mismas funciones tanto a la hora de cifrar como para descifrar.

5.1.2 Tiempos TDES

Como vemos en la siguiente tabla, al igual que en el caso de DES, los tiempos son muy similares en todos los modos que se han ejecutado, sin apenas cambios de tiempo de un modo a otro.

	TDES									
	Encriptar					Desencriptar				
	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC
256B	0,462	0,463	0,466	0,466	0,464	0,465	0,469	0,464	0,469	0,466
1024B	1,84	1,837	1,839	1,838	1,837	1,836	1,84	1,837	1,836	1,844
10240B	18,352	18,339	18,334	18,335	18,35	18,332	18,337	18,337	18,333	18,344

Tabla 22 Tiempos de ejecución algoritmo TDES en segundos

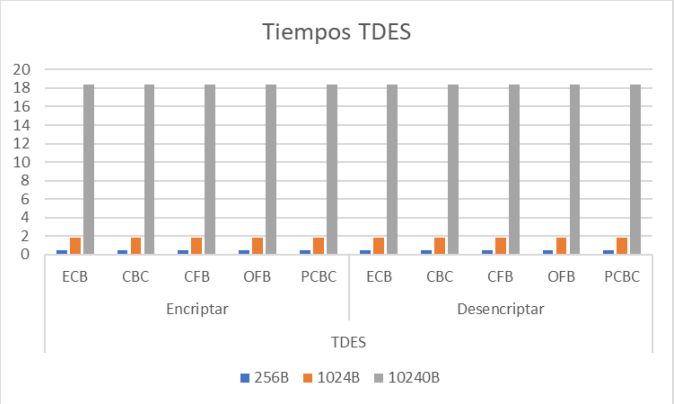


Figura 31 Tiempos de ejecución algoritmo DES en segundos

Al igual que en DES, sí que varían los tiempos con el tamaño del mensaje que se desea cifrar; a mayor tamaño mayor número de bloques y, por lo tanto, es necesario un mayor tiempo de ejecución. Como pasaba en el ejemplo anterior, este tiempo crece proporcionalmente al tamaño del mensaje que se desea cifrar.

Los tiempos de cifrado y descifrado para un mismo tamaño de mensaje apenas varían; al igual que sucedía en DES, las operaciones de descifrado son las mismas que en el cifrado.

También podemos apreciar si comparamos los tiempos de cifrado y descifrado de DES y TDES, que el tiempo de ejecución de este último es mucho mayor que el de DES; esto es debido a que el algoritmo TDES se basa en la ejecución de DES 3 veces seguidas. Además se tienen que calcular también las tres claves para su ejecución lo que ralentiza el algoritmo. Aunque debería ser tres veces mayor, los tiempos son aún mayores; supongo que esto será por el tiempo de gestión de los bloques que hace que los tiempos sean aún mayores.

5.1.3 Tiempos AES Software

En la siguiente tabla podemos ver los tiempos de ejecución para encriptar textos de 256B, 1024B y 10240B en AES, con contraseñas de 128, 192 y 256 bit con los modos ECB, CBC,CFB, OFB y PCBC.

	AES(Sotfware)														
	Encriptar														
	128					192					256				
	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC
256B	0,006	0,007	0,006	0,006	0,006	0,007	0,007	0,008	0,008	0,007	0,008	0,009	0,009	0,009	0,009
1024B	0,024	0,026	0,024	0,025	0,025	0,028	0,029	0,029	0,03	0,029	0,034	0,033	0,033	0,033	0,035
10240B	0,22	0,226	0,227	0,224	0,237	0,265	0,276	0,27	0,27	0,278	0,305	0,313	0,312	0,313	0,324

Tabla 23 Tiempos de cifrado AES software en segundos

	AES(Sotfware)														
	Desencriptar														
	128					192					256				
	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC
256B	0,007	0,008	0,006	0,006	0,008	0,009	0,009	0,008	0,007	0,009	0,011	0,01	0,008	0,009	0,012
1024B	0,028	0,03	0,026	0,024	0,03	0,035	0,036	0,029	0,028	0,036	0,042	0,043	0,034	0,035	0,041
10240B	0,27	0,277	0,223	0,225	0,285	0,335	0,338	0,274	0,267	0,338	0,385	0,391	0,312	0,312	0,395

Tabla 24 Tiempos de descifrado de AES Software en segundos.

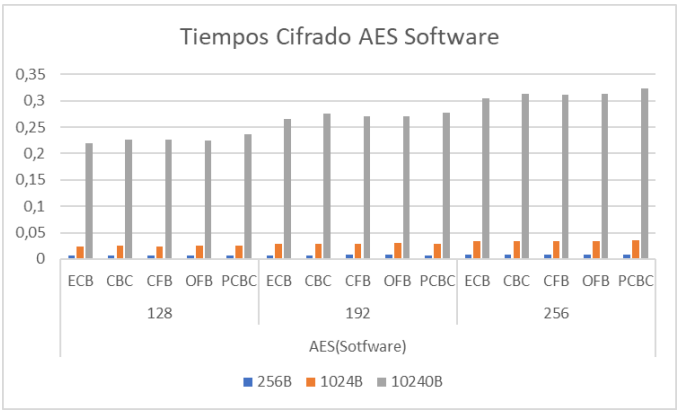


Figura 33 Tiempos de cifrado AES Software

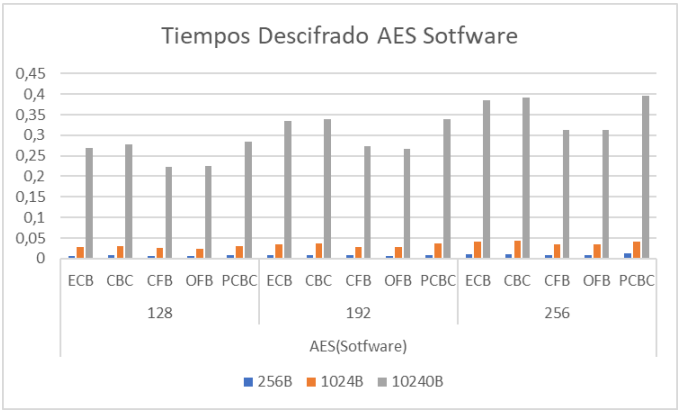


Figura 32 Tiempos de descifrado AES Software

Como se viene observando en el resto de algoritmos, el tiempo de cifrado aumenta con el volumen de datos; al igual que en el resto de casos el aumento también es proporcional al aumento de los datos.

Aunque los tiempos necesarios para el cifrado son muy similares en todos los modos, son algo inferiores en el modo ECB; esto tiene su lógica, puesto que es el que menor número de operaciones requiere a la hora de gestionar los bloques. En cambio con el modo PCBC los tiempos son algo superiores a los recogidos en el resto de modos.

El algoritmo AES puede funcionar con tres tamaños diferentes de clave: 128, 192 y 256 bits y, como vemos, los tiempos obtenidos aumentan con el aumento del tamaño de la clave; puesto

que a mayor tamaño de la clave aumenta el número de rondas necesarias para el cifrado, lo que hace más seguro el algoritmo.

En este caso se puede apreciar que los tiempos para descifrar aumentan en relación a los tiempos obtenidos a la hora de cifrar, excepto en los modos CFB y OFB, que usan la operación de encriptar también para descifrar, lo que hace que los tiempos tengan que ser muy similares entre cifrado y descifrado. En cuanto al resto de modos, se puede apreciar que la desencriptación es algo más lenta que el cifrado, aunque las operaciones son similares a las aplicadas a la hora de cifrar.

En relación con el cifrado DES o TDES se puede apreciar que los tiempos con este algoritmo de cifrado son significativamente inferiores a los recogidos por DES o TDES, esto puede ser debido al menor número de rondas necesarias para cifrar así uso de operaciones más rápidas en cada ronda.

5.1.4 Tiempos de AES Hardware

En la siguiente tabla podemos ver los tiempos de ejecución para encriptar textos de 256B, 1024B y 10240B en AES, con contraseñas de 128, 192 y 256 bits con los modos ECB, CBC, CFB, OFB y PCBC, usando el acelerador de la placa.

	AES(Hardware)														
	Cifrado														
	128					192					256				
	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC
256B	0	0,001	0	0	0,001	0	0,001	0	0	0,001	0,001	0,001	0,001	0	0,001
1024B	0,001	0,002	0,002	0,001	0,002	0,002	0,002	0,002	0,002	0,003	0,001	0,002	0,002	0,002	0,002
10240B	0,011	0,018	0,017	0,017	0,024	0,012	0,018	0,018	0,018	0,025	0,019	0,021	0,02	0,019	0,026

Tabla 25 Tiempos de Cifrado AES Hardware

	AES(Hardware)														
	Descifrado														
	128					192					256				
	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC	ECB	CBC	CFB	OFB	PCBC
256B	0	0	0,001	0	0,001	0,001	0	0	0,001	0,001	0	0,001	0,001	0	0,001
1024B	0,001	0,002	0,002	0,002	0,003	0,002	0,002	0,002	0,002	0,003	0,001	0,002	0,002	0,002	0,002
10240B	0,011	0,018	0,018	0,017	0,024	0,012	0,02	0,018	0,019	0,025	0,02	0,02	0,019	0,019	0,026

Tabla 26 Tiempos Descifrado AES Hardware

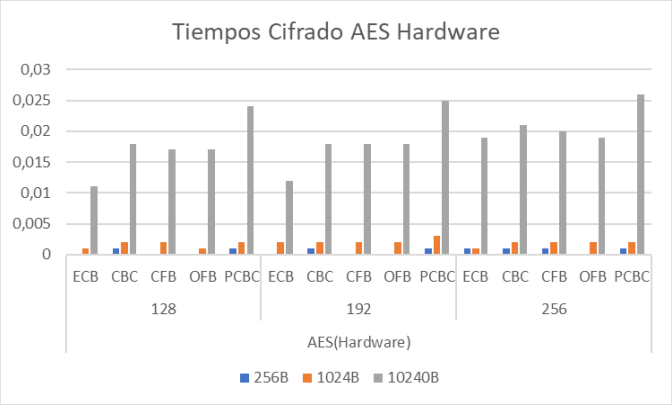


Figura 35 Tiempos de cifrado de AES (Hardware)

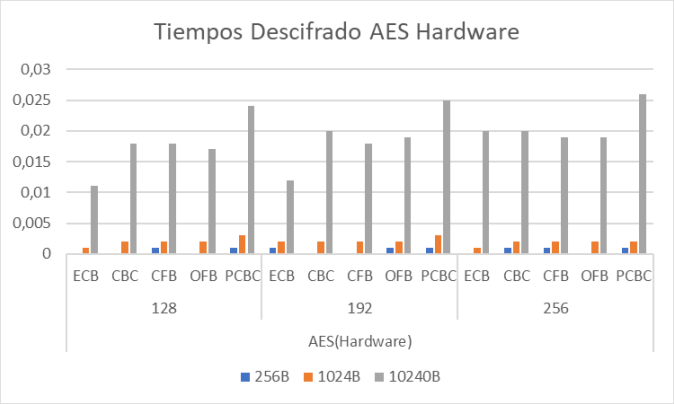


Figura 34 Tiempos de descifrado de AES (Hardware)

En primer lugar, indicar que aunque los tiempos recogidos con textos pequeños son cero, esto es debido a que los tiempos han sido recogidos mediante la función xTaskGetTickCount, que devuelve el número de tick y luego estos son transformados a segundos; si las operaciones se terminan antes que el tick aumente, este será cero y, por lo tanto también sus tiempos.

Al igual que sucede con el resto de algoritmos, el tamaño del texto influye significativamente en los tiempos de cifrado; como podemos observar, los tiempos de cifrado son significativamente menores cuanto menor es el archivo a cifrar.

Al igual que sucedía en AES por software, el modo que mejores tiempos obtiene es el modo ECB; esto es debido a que es el más sencillo y el que tiene que ejecutar menos operaciones para la encriptación.

También podemos apreciar que los tiempos son algo mayores en los modos PCBC, que tienen que ejecutar mayor número de operaciones a la hora de cifrar.

En cuanto al tamaño de la clave, podemos observar, como los tiempos son progresivamente mayores cuanto mayor es la longitud de la clave; esto es porque cuanto mayor es la clave se realizan un mayor número de rondas, lo cual hace que sea algo más lenta la encriptación.

Como podemos ver, los tiempos de cifrado y descifrado son prácticamente iguales.

Salta a la vista que los tiempos en esta tabla son mucho menores que en las tablas anteriores; esto es debido a que todas las operaciones de encriptado y desencriptado se realizan mediante hardware, lo que acelera las operaciones de cifrado.

5.2 Tiempos RSA

En este caso no ha sido posible realizar la medición de tiempos para 10240B ya que el algoritmo RSA esta poco optimizado y el texto cifrado ocupa 50120B lo que hace que se desborde la memoria.

La siguiente tabla muestra los tiempos tomados para este algoritmo:

	RSA	
	Encriptar	Desencriptar
256B	0,018	0,014
1024B	0,063	0,059
10240B	-	-

Tabla 28 Tiempos RSA

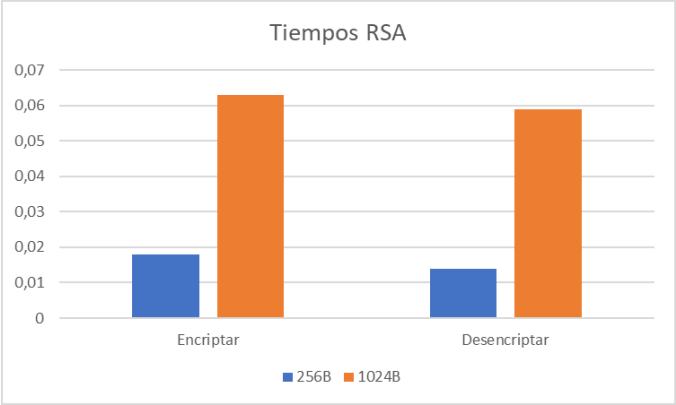


Tabla 27 Tiempos RSA

Como se puede ver los tiempos dependen del tamaño del mensaje, a mayor tamaño de mensaje más tiempo es necesario para el cifrado.

En el caso de este algoritmo los tiempos dependen mucho de las claves elegidas puesto que depende de estas el número de cálculos que hay que hacer para cifrar el texto.

Aunque sólo se han usado 32bits para su cifrado los tiempos de cifrado son mayores que los tiempos de cifrado de AES, si se hubieran usado más bit seguramente los tiempos de cifrado serian aún más lentos.

5.3 Tamaño RAM

El tamaño en la RAM se ha medido a partir de las variables globales utilizadas por los algoritmos, las variables globales del módulo cli, las variables usadas en las funciones de encriptar o desencriptar y las funciones utilizadas para generar la clave si fuera necesario. No se han tenido en cuenta otras funciones, ya que en estas funciones es donde más RAM es necesaria y, por lo tanto, no habrá que tener en cuenta otras funciones puesto que esta se libera tras la ejecución de cada función. La RAM necesaria se ha calculado para textos de tamaños de 256B, 1024B y 10240B.

En la siguiente tabla podemos ver un resumen de los consumos de RAM en bits utilizada por cada algoritmo. En el anexo I se podrán encontrar las tablas detalladas del consumo de RAM de cada algoritmo.

	256B	1024B	10240B
DES	13760	26048	173504
TDES	13960	26248	173704
AES 128 (Software)	7008	19296	166752
AES 192 (Software)	7072	19360	166816
AES 256 (Software)	7136	19424	166880
AES 128 (Hardware)	4840	17128	164584
AES 192 (Hardware)	4904	17192	164648
AES 256 (Hardware)	4968	17256	164712
RSA Encriptar	18832	74128	737680
RSA Desencriptar	12608	49472	491840

Tabla 29 Consumos de RAM (en bits)

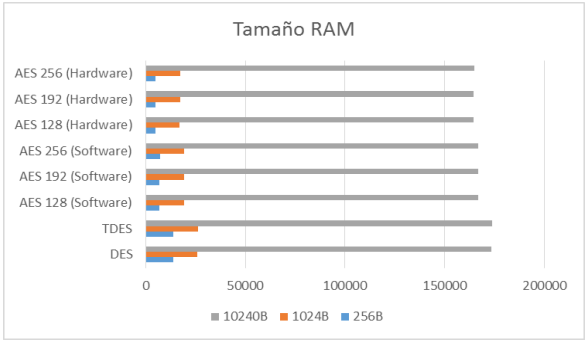


Figura 36 Tamaño RAM de los diferentes algoritmos de cifrado

Como podemos observar en la tabla, el algoritmo que menos consume es el AES Hardware, esto es porque únicamente necesita memoria RAM para la gestión de la información y de los bloques antes y después de ser cifrada, puesto que el cifrado se realiza por el hardware de la placa.

En segundo lugar, podemos apreciar que el siguiente algoritmo que menos RAM requiere es AES software; este algoritmo guarda todas las tablas que necesita el algoritmo para realizar los cálculos como constantes, con lo cual se reduce drásticamente el tamaño de RAM necesaria a diferencia de DES, que las guarda en la RAM.

En tercer lugar está el algoritmo DES, cuya diferencia no es demasiado grande con TDES; esto es porque la ejecución de TDES aprovecha las variables que guardan los bloques y el resto de variables de gestión de datos para pasárselas a la siguiente ejecución del algoritmo DES, con lo cual el consumo de RAM se mantiene, únicamente crece por las variables de la clave.

También podemos apreciar que el consumo de la RAM está muy influido por el tamaño del mensaje que se debe cifrar, exactamente crece el doble de lo que crece el tamaño del mensaje; esto es porque tenemos una variable con el mensaje original y otra que guarda temporalmente el mensaje encriptado o desencriptado.

Los algoritmos AES también están influidos por el tamaño de la clave, que debe guardarse en la RAM.

En cuanto al algoritmo RSA se aprecia que es el que más ocupa esto es porque no esta optimizado el módulo que utilizamos en su algoritmo. En este algoritmo se puede apreciar que también depende del tamaño del mensaje, cuanto mayor es el mensaje mayor es el tamaño necesario para procesarlo. Las claves elegidas influyen también en la memoria RAM necesaria, puesto que el se deben trocear en partes menores que el n y en ocasiones hay que usar relleno para que los bloques se aproximen los más posible a n.

5.4 Tamaño en la memoria flash

Para medir este parámetro se ha compilado la aplicación de la placa con y sin el módulo correspondiente a la codificación, teniendo en cuenta que hay funciones comunes como las que se encargan de recibir y de enviar la información a la aplicación de escritorio. El tamaño en bits de dichas funciones se ha desglosado como se puede ver en la siguiente tabla.

FreeRtos + Librerías	35.066
CLI y MD5	13.220
DES	4.664
3DES	1.660
AES Software	7.992
AES Hardware	1.628
RSA	5.696
Sistema Login	560
Tamaño Total	70.486

Tabla 30 Tamaño en la Flash

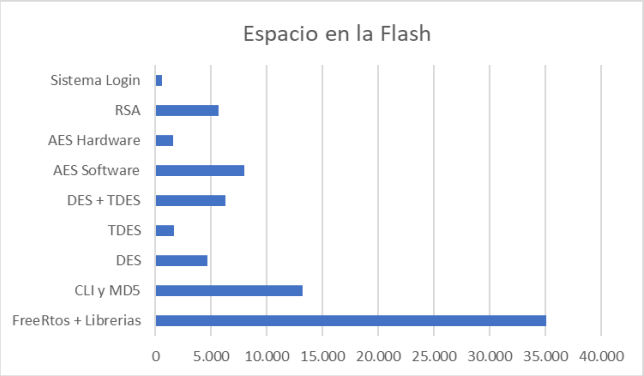


Figura 37 Tamaño en la Flash

Como es natural, los módulos que más ocupan son las librerías de FreeRtos y las librerías que utiliza este para la gestión de los recursos en la placa.

Las librerías encargadas de la comunicación con la placa, es decir, las encargadas de enviar y recibir los datos desde la placa, CLI y MD5, son las siguientes que más memoria utilizan dentro de la placa. Estas librerías y las de FreeRtos son comunes para todos los algoritmos, ya que son utilizadas por todas ellas, por lo cual no las tenemos en cuenta en cada uno de los algoritmos.

El algoritmo que menos espacio flash necesita es el AES Hardware, esto es lógico, ya que es la placa la encargada del cifrado y descifrado; el espacio utilizado por este algoritmo es el de las funciones que se encargan de dividir el mensaje en bloques y de aplicar los modos al algoritmo.

El siguiente algoritmo que menos espacio utiliza es el DES, ya que a la hora de programar se han utilizado variables en lugar de constantes, con lo cual el espacio utilizado es menor que en el caso de AES software, cuyos vectores son constantes y, por lo tanto, estas variables no ocupan espacio en la flash sino en la memoria RAM.

Aunque el tamaño del módulo TDES parece pequeño, hay que tener en cuenta que este algoritmo hace uso del algoritmo DES, por lo tanto, el tamaño total del algoritmo será el del módulo TDES y DES.

El algoritmo RSA ocupa un mayor espacio en flash que DES puesto que el algoritmo RSA almacena como constantes los números primos que puede utilizar en lugar de generarlos mediante alguna función.

AES software tiene un tamaño mayor que el resto, ya que al programarlo se han utilizado constantes en vez de utilizar variables; estas constantes se almacenan en disco en vez de hacer uso de la RAM.

5.5 Conclusiones:

Se ha demostrado como se produce un incremento del tiempo de cifrado proporcional al aumento del tamaño del mensaje.

El algoritmo más rápido es el AES por Hardware, debido al uso del hardware que acelera las operaciones necesarias para el cifrado y descifrado, como ya hemos comentado.

En cuanto a los algoritmos de software, el algoritmo más lento es 3DES y el más rápido AES. Por tanto, AES será el algoritmo que habrá que elegir en cualquier caso, ya que además, proporciona un mayor nivel de seguridad.

El modo de operación más rápido para cualquier algoritmo ha sido ECB, lo que se entiende dado que no requiere de retroalimentación para procesar el cifrado aunque, de otra forma, es el más vulnerable. El modo de operación más lento suele ser PCBC.

En los algoritmos AES se observa un aumento en el tiempo de cifrado de forma escalonada con respecto al tamaño de la clave, debido al mayor de número de rondas utilizadas cuanto mayor es la clave.

El tamaño del mensaje que se quiere encriptar influye mucho en la memoria RAM necesaria para el cifrado.

6 Viabilidad técnica

El presente proyecto es viable técnica y económicamente, ya que se basa en el desarrollo de aplicaciones sobre hardware existente, de fácil acceso, bajo coste y ampliamente documentado, basados en estándares de la industria y de código abierto. Los montajes y circuitos adicionales necesarios son de fácil realización u obtención.

Puntos Fuertes

- Se trata de un sistema sencillo capaz de adaptarse a casi cualquier sistema embebido.
- Bajo coste del material.
- Fácil instalación y puesta en marcha.

Puntos Débiles.

- El código, aunque comprobado, puede no estar lo suficientemente depurado y siempre cabe la posibilidad de tener bug o fallos imprevistos.
- Existen librerías de software libre en el mercado que ofrecen herramientas con más opciones que la nuestra a la hora de encriptar.

7 Valoración económica

Cabe destacar que el fin del proyecto no es económico, sino esta más orientado a la investigación, lo que servirá para dilucidar cuál es el algoritmo más apropiado para trabajar sobre los sistemas empotrados, ya que estos sistemas tienen limitaciones en cuanto a memoria y procesos. En cualquier caso, se refleja el presupuesto que se ha empleado para realizar dicho proyecto:

Concepto	Precio Unidad	Cantidad	Total
Placa MSP432P401R	12,20€	1	12,20€
Code Composer Studio 7.3.0	Gratuito	1	0€
FreeRTOS	Gratuito	1	0€
Horas de Trabajo	15€	315	4725€
Total			4737,20€

Figura 38 Valoración económica

Como ya se ha indicado anteriormente, el producto consiste en una prueba de concepto y nos ha servido para el estudio de las limitaciones de los diversos algoritmos de encriptación sobre sistemas encastrados y por lo cual no procede llevar a cabo la fabricación en masa para la comercialización.

8 Conclusiones

8.1 Lecciones aprendidas

Durante la realización de este trabajo he podido aprender sobre sistemas encastrados, un mundo que siempre me había llamado la atención, aunque desconocido para mí; puede que un proyecto de estas características no haya sido la mejor manera de iniciarse en este mundo, ya que se me ha hecho bastante cuesta arriba, sobre todo el inicio.

Este proyecto también me ha servido para aprender sobre criptología y los algoritmos más conocidos y utilizados en el mundo de la encriptación, un tema muy interesante y del que seguro sacaré partido más adelante.

También me ha venido bien para desempolvar el lenguaje C, el cual tenía olvidado desde los primeros cursos de la Ingeniería Técnica.

Otra de las lecciones que he podido aprender es que con esfuerzo y tesón todo se puede conseguir. Llegar a este punto no ha sido nada fácil, ya que se he tenido que invertir muchas horas y mucho esfuerzo para llegar a conseguir este resultado.

Y este proyecto también me ha servido para adquirir una buena base para la realización de futuros proyectos, ya sea dentro de los sistemas encastrados o en otras materias.

8.2 Autoevaluación.

8.2.1 Una reflexión crítica sobre el logro de los objetivos

Desde mi punto de vista, se pueden dar como satisfechos los objetivos marcados para este proyecto, aunque siempre habrá partes que se podrían mejorar o en las que se podría hacer más hincapié como, por ejemplo, la encriptación RSA, que no funciona del todo bien y que se podría pulir aplicando algo más de tiempo al proyecto.

Como se ha comentado en la entrega del código final y a lo largo de esta memoria, el objetivo secundario de implementar el algoritmo RSA, en ocasiones, dependiendo de las claves de cifrado seleccionadas, falla, ya que se pueden sobrepasar los límites de las variables soportadas por C. Además, al no estar optimizado el uso de la memoria en este algoritmo, al introducir textos excesivamente largos se puede producir un overflow de las variables y colgar la placa.

En definitiva, considero que los objetivos del proyecto han quedado satisfechos.

8.2.2 Un análisis crítico del seguimiento de la planificación

Durante el proyecto se ha tratado de seguir la planificación que teníamos planteada, si bien es verdad que ha habido tareas que han sufrido retrasos con respecto a la planificación inicial; en las entregas se han presentado finalizadas casi todas las tareas de dicha fase y aquellas que no se han podido terminar a tiempo se han recuperado en las siguientes fases.

8.3 Las líneas de trabajo futuro

Hay varias líneas de trabajo que podrían ser explotadas en el futuro como, por ejemplo, pueden ser:

La ampliación del algoritmo RSA: El algoritmo RSA que hemos implementado en este proyecto tiene grandes limitaciones puesto que utiliza los tipos numéricos estándar de C, lo que hace que la longitud de la clave esté limitada a 32 bits. Como hemos visto, para que actualmente una clave RSA sea segura tiene que tener como mínimo 1024 bits. Se podría trabajar en la creación de nuevos tipos numéricos y sus operaciones para poder trabajar con valores numéricos de esta longitud.

Ampliación de los algoritmos de cifrado: Aunque nuestro estudio se ha llevado a cabo con los algoritmos de cifrado más populares, se han quedado en el tintero otros que podrían ser incluidos en líneas de trabajo futuras como pueden ser IDEA, Blowfish, Twofish, RC4, El Gamal o Diffie-Hellman, entre otros.

Conexión WiFi del Sistema: se podría ampliar los tipos de conexión para poder conectarse vía WiFi y no únicamente por el puerto serie, ampliándose así las posibilidades de nuestro sistema de encriptación.

9 Glosario

3DES: Triple DES, se le llama al algoritmo que hace triple cifrado del DES.

ADC: Siglas en Inglés de convertidor analógico-digital, encargado de muestrear la señal a su entrada en valores discretos aptos para su uso en sistemas digitales.

AES: Advanced Encryption Standard también conocido como Rijndael, es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el Gobierno de los Estados Unidos.

DES: Data Encryption Standard, es un algoritmo de cifrado ampliamente propagado.

DMA: El acceso directo a memoria (DMA, del inglés direct memory access) permite a cierto tipo de componentes de una computadora acceder a la memoria del sistema independientemente de la CPU**.**

EEPROM: Electrically Erasable Programmable Read-Only Memory (Memoria de Solo Lectura Programable y Borrable Eléctricamente); tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente.

FreeRTOS: Sistema operativo libre en tiempo real utilizado en el proyecto. Es el encargado de la planificación de CPU, así como de la sincronización multitarea.

I2C: Circuito interintegrado (I²C, del inglés Inter-Integrated Circuit), es un bus serie de datos desarrollado en 1982.

LED: Diodo emisor de luz (Light-emitting diode).

MD5: (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5), es un algoritmo de reducción criptográfico de 128 bits ampliamente usado.

MSP432: El MSP432 es una familia de microcontroladores de señal mixta de Texas Instruments.

RSA: Es un sistema criptográfico de clave pública desarrollado en 1977.

SPI: (Serial Peripheral Interface), es un estándar de comunicaciones usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

UART: Son las siglas en inglés de Universal Asynchronous Receiver-Transmitter, Transmisor-Receptor Asíncrono Universal, es el dispositivo que controla los puertos y dispositivos serie.

10 Bibliografía

[1] Libro: Maiorano, A. (2010). Criptografía. Madrid: Ra-Ma.

[2] Libro: Hernández Encinas, L. (2016). La criptografía. Madrid: Editorial CSIC Consejo Superior de Investigaciones Científicas.

[3] Artículo: NIST, Data Encryption Standard (DES), Computer Security, Cryptography, US. Department of Commerce, 1999.

[4] Artículo: NIST, Advanced Encryption Standard, AES, Computer Security Standard, Cryptography, 2001.

[5] Artículo: Barry, Richard. Mastering the FreeRTOS Real Time Kernel – A Hands on tutorial guide, Pre-release 161204 Edition, Real Time Engineers Ltd. 2016

[6] Artículo: Amazon Web Services. FreeRTOS V10.0.0 Reference Manual, Amazon.com, 2017.

[7] Web: Wiki Sistemas Embebidos: http://cv.uoc.edu/webapps/xwiki/wiki/matembeddedsystemslabhome/view/Material/StartMSP432 [Consulta:15 noviembre 2017].

[8] Web: MSP432P401R LaunchPad, http://www.ti.com/tool/MSP-EXP432P401R [Consulta: 7 enero 2018].

[9] Web: Cifrado por bloques, https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation [Consulta: 7 enero 2018].

[10] Web: Data Encryption Standard, https://es.wikipedia.org/wiki/Data_Encryption_Standard [Consulta: 8 enero 2018].

[11] Web: Cifrado mediante DES, http://es.ccm.net/contents/130-introduccion-al-cifrado-mediante-des [Consulta: 8 enero 2010].

[12] Web: Triple DES, https://es.wikipedia.org/wiki/Triple_DES [Consulta: 8 enero 2010].

[13] Web: Advanced Encryption Standard, https://es.wikipedia.org/wiki/Advanced_Encryption_Standard [Consulta: 8 enero 2010].

[14] Web: RSA, https://es.wikipedia.org/wiki/RSA [Consulta:]. [Consulta: 8 enero 2018].

[15] Web: The MD5 Message-Digest Algorithm, http://www.ietf.org/rfc/rfc1321.txt [Consulta: 10 enero 2018].

[16] Web: MD5, http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/index.php/ [Consulta: 8 enero 2018].

[17] Web: Algoritmo de Euclides, https://es.wikipedia.org/wiki/Algoritmo_de_Euclides [Consulta: 13 diciembre 2017].

[18] Web: Exponenciación modular https://es.wikipedia.org/wiki/Exponenciaci%C3%B3n_modular [Consulta: 13 diciembre 2017].

[19] Web: Análisis comparativo de las placas Arduino, https://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/ [Consulta: 7 enero 2018].

[20] Web: Wolfssl, https://www.wolfssl.com/products/wolfssl/ [Consulta: 4 enero 2018]

[21] Web: OpenSSL, https://www.openssl.org/docs/ [Consulta: 4 enero 2018]

[22] Web: arm MBEB, https://tls.mbed.org/core-features [Consulta: 4 enero 2018]

11 Anexos

11.1 Consumo de RAM de los Algoritmos de cifrado

En el presente anexo se recogen los consumos de RAM de los distintos algoritmos de cifrado estudiados en este proyecto.

	256B				1024B				10240B			
	Tam.	Tipo	Tamaño Variable	Tamaño total	Tam.	Tipo	Tamaño Variable	Tamaño total	Tam.	Tipo	Tamaño Variable	Tamaño total
Variables Globales DES:												
initial_key _permutation	56	int	8	448	56	int	8	448	56	int	8	448
initial_message_permutation	64	int	8	512	64	int	8	512	64	int	8	512
key_shift_sizes	17	int	8	136	17	int	8	136	17	int	8	136
sub_key_permutation	48	int	8	384	48	int	8	384	48	int	8	384
message_expansion	48	int	8	384	48	int	8	384	48	int	8	384
S1	64	int	8	512	64	int	8	512	64	int	8	512
S2	64	int	8	512	64	int	8	512	64	int	8	512
S3	64	int	8	512	64	int	8	512	64	int	8	512
S4	64	int	8	512	64	int	8	512	64	int	8	512
S5	64	int	8	512	64	int	8	512	64	int	8	512
S6	64	int	8	512	64	int	8	512	64	int	8	512
S7	64	int	8	512	64	int	8	512	64	int	8	512
S8	64	int	8	512	64	int	8	512	64	int	8	512
right_sub_message_permutation	32	int	8	256	32	int	8	256	32	int	8	256
final_message_permutation	64	int	8	512	64	int	8	512	64	int	8	512
Total Variables Globales:				**6728**				**6728**				**6728**
Variable globales CLI												
datos	256	uint8_t	8	2048	1024	uint8_t	8	8192	10240	uint8_t	8	81920
clave	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
vector	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Totales Variables Globales CLI				**2248**				**8392**				**82120**
Función prvDESCommand												
block	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
block_aux	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
block_process	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
resultado	256	uint8_t	8	2048	1024	uint8_t	8	8192	10240	uint8_t	8	81920
key_sets	17	key_Set	128	2176	17	key_Set	128	2176	17	key_Set	128	2176
init_vector	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
i	1	int	8	8	1	int	8	8	1	int	8	8
Totales Función prvDESCommand				**4488**				**10632**				**84360**
Funcion process_message:												
i	1	int	8	8	1	int	8	8	1	int	8	8
k	1	int	8	8	1	int	8	8	1	int	8	8
shift_byte	1	u. char	8	8	1	u. char	8	8	1	u. char	8	8
l	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
r	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
ln	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
rn	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
er	6	u. char	8	48	6	u. char	8	48	6	u. char	8	48
ser	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
pre_end_permutation	8	u. char	8	64	8	u. char	8	64	8	u. char	8	64
Totales Funcion process_message:				**296**				**296**				**296**
TOTAL				**13760**				**26048**				**173504**

Tabla 31 Memoria RAM para cifrar con DES

	256B				1024B				10240B			
	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total
Variables Globales DES:												
initial_key_permutation	56	int	8	448	56	int	8	448	56	int	8	448
initial_message_permutation	64	int	8	512	64	int	8	512	64	int	8	512
key_shift_sizes	17	int	8	136	17	int	8	136	17	int	8	136
sub_key_permutation	48	int	8	384	48	int	8	384	48	int	8	384
message_expansion	48	int	8	384	48	int	8	384	48	int	8	384
S1	64	int	8	512	64	int	8	512	64	int	8	512
S2	64	int	8	512	64	int	8	512	64	int	8	512
S3	64	int	8	512	64	int	8	512	64	int	8	512
S4	64	int	8	512	64	int	8	512	64	int	8	512
S5	64	int	8	512	64	int	8	512	64	int	8	512
S6	64	int	8	512	64	int	8	512	64	int	8	512
S7	64	int	8	512	64	int	8	512	64	int	8	512
S8	64	int	8	512	64	int	8	512	64	int	8	512
right_sub_message_permutation	32	int	8	256	32	int	8	256	32	int	8	256
final_message_permutation	64	int	8	512	64	int	8	512	64	int	8	512
Totales Variables Globales DES				**6728**				**6728**				**6728**
Variable globales CLI												
datos	256	uint8_t	8	2048	1024	uint8_t	8	8192	10240	uint8_t	8	81920
clave	24	uint8_t	8	192	24	uint8_t	8	192	24	uint8_t	8	192
vector	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Totales Variable globales CLI				**2376**				**8520**				**82248**
Función prvDESCommand												
block	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
block_aux	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
block_process	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
resultado	256	uint8_t	8	2048	1024	uint8_t	8	8192	10240	uint8_t	8	81920
init_vector	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
i	1	int	8	8	1	int	8	8	1	int	8	8
Totales Función prvDESCommand				**2312**				**8456**				**82184**
Funcion process_message:												
i	1	int	8	8	1	int	8	8	1	int	8	8
k	1	int	8	8	1	int	8	8	1	int	8	8
shift_byte	1	u. char	8	8	1	u. char	8	8	1	u. char	8	8
l	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
r	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
ln	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
rn	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
er	6	u. char	8	48	6	u. char	8	48	6	u. char	8	48
ser	4	u. char	8	32	4	u. char	8	32	4	u. char	8	32
pre_end_permutation	8	u. char	8	64	8	u. char	8	64	8	u. char	8	64
Totales process_message:				296				296				296
Funcion TDES												
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
clave	8	uint8_t	8	64	8	uint8_t	8	64	8	uint8_t	8	64
key_sets	17	key_set	128	2176	17	key_set	128	2176	17	key_set	128	2176
Total Funcion process_message				**2248**				**2248**				**2248**
TOTALES				13960				26248				173704

Tabla 32 Memoria RAM necesaria cifrar con TDES

	256B											
	128				192				256			
	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total
aes_decrypt	16	BYTE	8	128	16	BYTE	8	128	16	BYTE	8	128
Total aes_decrypt				128				128				128
Variable globales CLI												
datos	256	uint8_t	8	2048	256	uint8_t	8	2048	256	uint8_t	8	2048
clave	16	uint8_t	8	128	24	uint8_t	8	192	32	uint8_t	8	256
vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Total Variable globales CLI				2376				2440				2504
Función prvAESCommand												
numbloques	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
block	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_aux	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_process	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
resultado	256	uint8_t	8	2048	256	uint8_t	8	2048	256	uint8_t	8	2048
init_vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
key_schedule	60	WORD	32	1920	60	WORD	32	1920	60	WORD	32	1920
i	1	int	8	8	1	int	8	8	1	int	8	8
tam_aes	1	int	8	8	1	int	8	8	1	int	8	8
Total Función prvAESCommand				4504				4504				4504
TOTALES				7008				7072				7136

Tabla 33 Memoria RAM para cifrar un texto de 256B con AES software

	256B											
	128				192				256			
	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total
aes_decrypt	16	BYTE	8	128	16	BYTE	8	128	16	BYTE	8	128
Total aes_decrypt				128				128				128
Variable globales CLI												
datos	256	uint8_t	8	2048	256	uint8_t	8	2048	256	uint8_t	8	2048
clave	16	uint8_t	8	128	24	uint8_t	8	192	32	uint8_t	8	256
vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Total Variable globales CLI				2376				2440				2504
Función prvAESCommand												
numbloques	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
block	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_aux	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_process	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
resultado	256	uint8_t	8	2048	256	uint8_t	8	2048	256	uint8_t	8	2048
init_vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
key_schedule	60	WORD	32	1920	60	WORD	32	1920	60	WORD	32	1920
i	1	int	8	8	1	int	8	8	1	int	8	8
tam_aes	1	int	8	8	1	int	8	8	1	int	8	8
Total Función prvAESCommand				4504				4504				4504
TOTALES				7008				7072				7136

Tabla 34Memoria RAM para cifrar un texto de 1024B con AES software

	1024B											
	128				192				256			
	Tama	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total
aes_decrypt	16	BYTE	8	128	16	BYTE	8	128	16	BYTE	8	128
Total aes_decrypt				128				128				128
Variable globales CLI												
datos	1024	uint8_t	8	8192	1024	uint8_t	8	8192	1024	uint8_t	8	8192
clave	16	uint8_t	8	128	24	uint8_t	8	192	32	uint8_t	8	256
vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Total Variable globales CLI				8520				8584				8648
Función prvAESCommand												
numbloques	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
block	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_aux	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_process	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
resultado	1024	uint8_t	8	8192	1024	uint8_t	8	8192	1024	uint8_t	8	8192
init_vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
key_schedule	60	WORD	32	1920	60	WORD	32	1920	60	WORD	32	1920
i	1	int	8	8	1	int	8	8	1	int	8	8
tam_aes	1	int	8	8	1	int	8	8	1	int	8	8
Total Función prvAESCommand				10648				10648				10648
TOTALES				19296				19360				19424

Tabla 35 Memoria RAM para cifrar un texto de 10240B con AES software

	10240B											
	128				192				256			
	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total
aes_decrypt	16	BYTE	8	128	16	BYTE	8	128	16	BYTE	8	128
Total aes_decrypt				128				128				128
Variable globales CLI												
datos	10240	uint8_t	8	81920	10240	uint8_t	8	81920	10240	uint8_t	8	81920
clave	16	uint8_t	8	128	24	uint8_t	8	192	32	uint8_t	8	256
vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Total Variable globales CLI				82248				82312				82376
Función prvAESCommand												
numbloques	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
block	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_aux	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_process	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
resultado	10240	uint8_t	8	81920	10240	uint8_t	8	81920	10240	uint8_t	8	81920
init_vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
key_schedule	60	WORD	32	1920	60	WORD	32	1920	60	WORD	32	1920
i	1	int	8	8	1	int	8	8	1	int	8	8
tam_aes	1	int	8	8	1	int	8	8	1	int	8	8
Total Función prvAESCommand				84376				84376				84376
TOTALES				166752				166816				166880

Tabla 36 Memoria RAM para cifrar un texto de 256B con AES hardware

	1024B											
	128				192				256			
	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total
Variable globales CLI												
datos	1024	uint8_t	8	8192	1024	uint8_t	8	8192	1024	uint8_t	8	8192
clave	16	uint8_t	8	128	24	uint8_t	8	192	32	uint8_t	8	256
vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Totales Variable globales CLI				**8520**				**8584**				**8648**
Función prvAESCommand												
numbloques	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
block	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_aux	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_process	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
resultado	1024	uint8_t	8	8192	1024	uint8_t	8	8192	1024	uint8_t	8	8192
init_vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
i	1	int	8	8	1	int	8	8	1	int	8	8
tam_aes	1	int	8	8	1	int	8	8	1	int	8	8
Totales Función prvAESCommand				**8608**				**8608**				**8608**
TOTAL				**17128**				**17192**				**17256**

Tabla 37 Memoria RAM para cifrar un texto de 1024B con AES Hardware

	10240B											
	128				192				256			
	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total	Tam	Tipo	Tamaño Variable	Tamaño total
Variable globales CLI												
datos	10240	uint8_t	8	81920	10240	uint8_t	8	81920	10240	uint8_t	8	81920
clave	16	uint8_t	8	128	24	uint8_t	8	192	32	uint8_t	8	256
vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
modo	4	uint8_t	8	32	4	uint8_t	8	32	4	uint8_t	8	32
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
size	1	int	8	8	1	int	8	8	1	int	8	8
algoritmo	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	int	8	8	1	int	8	8	1	int	8	8
tam_clave	1	int	8	8	1	int	8	8	1	int	8	8
Totales Variable globales CLI				**82248**				**82312**				**82376**
Función prvAESCommand												
numbloques	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
block	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_aux	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
block_process	1	uint8_t	8	8	1	uint8_t	8	8	1	uint8_t	8	8
resultado	10240	uint8_t	8	81920	10240	uint8_t	8	81920	10240	uint8_t	8	81920
init_vector	16	uint8_t	8	128	16	uint8_t	8	128	16	uint8_t	8	128
i	1	int	8	8	1	int	8	8	1	int	8	8
tam_aes	1	int	8	8	1	int	8	8	1	int	8	8
Totales Función prvAESCommand				**82336**				**82336**				**82336**
TOTAL				**164584**				**164648**				**164712**

Tabla 38 Memoria RAM para cifrar un texto de 10240B con AES Hardware

RSA Encriptar

	256B				1024B				10240B			
	Tam	**Tipo**	**Tamaño Variable**	**Tamaño total**	**Tam**	**Tipo**	**Tamaño Variable**	**Tamaño total**	**Tam**	**Tipo**	**Tamaño Variable**	**Tamaño total**
Variables Globales CLI												
datos	1280	uint8_t	8	10240	5120	uint8_t	8	40960	51200	uint8_t	8	409600
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	Int	8	8	1	Int	8	8	1	Int	8	8
struct pub[1]	1	struct	64	64	1	struct	64	64	1	struct	64	64
private_key_class pub[1]2	1	struct	64	64	1	struct	64	64	1	struct	64	64
Total Variables Globales CLI				**10384**				**41104**				**409744**
Funciones rsa_encrypt												
encrypted	256	long	32	8192	1024	long	32	32768	10240	long	32	327680
temp	1	long	32	32	1	long	32	32	1	long	32	32
i	1	int	8	8	1	int	8	8	1	int	8	8
Total				**8232**				**32808**				**327720**
Función rsa_modExp												
b	1		32	32	1		32	32	1		32	32
e	1		32	32	1		32	32	1		32	32
m	1		32	32	1		32	32	1		32	32
retorno	1		32	32	1		32	32	1		32	32
Total Funciones rsa_encrypt				**128**				**128**				**128**
Función prvRSACommand												
encriptar	1	u. char	8	8	1	u. char	8	8	1	u. char	8	8
temp	5	char	8	40	5	char	8	40	5	char	8	40
cadena	5	u char	8	40	5	u char	8	40	5	u char	8	40
Total Función prvRSACommand				**88**				**88**				**88**
TOTAL				**18832**				**74128**				**737680**

Tabla 39 Memoria RAM para cifrar con RSA

	RSA Encriptar											
	256B				1024B				10240B			
	Tam	**Tipo**	**Tamaño Variable**	**Tamaño total**	**Tam**	**Tipo**	**Tamaño Variable**	**Tamaño total**	**Tam**	**Tipo**	**Tamaño Variable**	**Tamaño total**
Variables Globales CLI												
datos	1280	uint8_t	8	10240	5120	uint8_t	8	40960	51200	uint8_t	8	409600
tam_datos	1	int	8	8	1	int	8	8	1	int	8	8
tam_block	1	Int	8	8	1	Int	8	8	1	Int	8	8
public_key_class pub[1]	1	struct	64	64	1	struct	64	64	1	struct	64	64
private_key_class pub[1]2	1	struct	64	64	1	struct	64	64	1	struct	64	64
Total Variables Globales CLI				**10384**				**41104**				**409744**
num_boques	1	int	8	8	1	int	8	8	1	int	8	8
block	5	u. char	8	40	5	u. char	8	40	5	u. char	8	40
decrypted	256	u. char	8	2048	1024	u. char	8	8192	10240	u. char	8	81920
Total				**2096**				**8240**				**81968**
Función rsa_modExp												
b	1		32	32	1		32	32	1		32	32
e	1		32	32	1		32	32	1		32	32
m	1		32	32	1		32	32	1		32	32
retorno	1		32	32	1		32	32	1		32	32
Total Funciones rsa_encrypt				**128**				**128**				**128**
TOTAL				**12608**				**49472**				**491840**

Tabla 40 Memoria RAM para descifrar con RSA