



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Prototipo de sistema híbrido para la portabilidad de datos médicos usando cifrado por hardware y tecnología RFID”

2019-B086

Presentan

**Hernández Castellanos César Uriel
Martínez Islas Mauricio Joel
Reyes Valenzuela Alejandro**

Directores

**M. en C. Josué Rangel
González**

**M. en C. Cervantes de
Anda Ismael**



5 de junio de 2020

Índice

1. Estructura del documento	10
2. Introducción	12
2.1. Antecedentes	12
2.2. Planteamiento del problema	14
2.3. Propuesta de solución	14
2.4. Objetivos	15
2.4.1. Objetivo general	15
2.4.2. Objetivos específicos	15
2.5. Justificación	15
2.6. Alcances y limitaciones	16
3. Estado del arte	17
3.1. Comparativa con trabajos internos	17
3.1.1. TT-12-1-0009: Historial Clínico para el Departamento de Servicio Médico de ESCOM	17
3.1.2. TT-2014-B011: Sistema Móvil de Registros de Enfermería implementando Tecnología RFID	17
3.1.3. TT-2012-B005: Sistema de Expediente Clínico con Adquisición Automática de Datos	18
3.1.4. TT-2012-A054: Expediente Clínico Electrónico	18
3.2. Comparativa con trabajos externos	18
3.2.1. Using RFID Technology for Managing Patient Medical File	18
3.2.2. RFID for Inventory of Medical Records	18
3.2.3. Utilización de tecnología RFID para optimizar la operatividad y administración de una institución hospitalaria	19
4. Marco teórico	19

4.1. Visión general de la criptología	19
4.2. Campos de la criptología	20
4.3. Criptografía ligera	20
4.4. Algoritmos de cifrado ligero	21
4.5. Proceso de estandarización de NIST	21
4.6. ISO/IEC 29192-2:2019	21
4.7. Lightweight Encryption Algortihm (LEA)	21
4.8. Características	21
4.9. Seguridad	22
4.10. Eficiencia	22
4.11. Especificación de LEA	22
4.12. Notación	22
4.13. Key Schedule	23
4.13.1. Constantes	23
4.13.2. Key Schedule con 128 bits	24
4.13.3. Key Schedule con 192 bits	24
4.14. Proceso de cifrado	24
4.14.1. Inicialización	25
4.14.2. Iteración de rondas	25
4.15. Finalización	25
4.16. RSA	25
4.16.1. Introducción	25
4.16.2. Funcionamiento	26
4.17. Historia de los sistemas RFID	27
4.18. Tecnología RFID	29
4.18.1. Funcionamiento de RFID	29

4.18.2. Arquitectura de un sistema RFID	29
4.18.3. Tipos de etiquetas RFID	31
4.18.4. Aplicaciones de la tecnología RFID	32
4.19. Lenguajes de descripción de hardware	35
4.20. Características de los lenguajes de descripción de hardware [42].	35
4.21. Lenguaje de descripción VHDL (VHSIC Hardware Description Language)	35
4.22. Ventajas de VHDL [42].	36
4.22.1. Serial Peripheral Interface (SPI)	36
4.22.2. Descripción de las líneas de comunicación de SPI	37
4.22.3. Características de SPI	38
4.22.4. Funcionamiento de SPI	38
4.23. FPGA	39
4.24. Servicios web	39
4.25. REST	40
4.26. Python	40
4.27. Flask	40
4.28. MySQL	40
5. Análisis	41
5.1. Comparativa de FPGAs	41
5.2. Comparativa de placas de desarrollo	41
5.3. Comparación de los algoritmos en ISO/IEC 29192-7:2019	41
5.4. Proceso de selección de algoritmo criptográfico	42
5.4.1. Comparación de implementaciones en hardware de los algoritmos en ISO/IEC 29192-2:2019	42
5.4.2. Análisis/ataques que se han realizado a los algoritmos en ISO/IEC 29192-2:2019	42

5.5.	Etiquetas RFID	44
5.5.1.	Proceso de selección de etiqueta RFID	44
5.6.	Interfaces de comunicación	44
6.	Diseño del sistema	45
6.1.	Metodología de diseño	45
6.2.	Requerimientos funcionales	46
6.2.1.	Requerimientos de Software	46
6.2.2.	Requerimientos de Hardware	47
6.3.	Requerimientos no funcionales	47
6.3.1.	Requerimientos de Software	47
6.3.2.	Requerimientos de Hardware	48
6.4.	Reglas de negocio	48
6.5.	Diseño de alto nivel	49
6.6.	Diseño en detalle	50
6.6.1.	Proceso de generación de llaves	50
6.6.2.	Proceso de cifrado	53
6.6.3.	Proceso de descifrado	54
6.6.4.	Unidad de control	57
6.7.	Diseño de puente SPI	60
6.8.	Diseño de arquitectura Web	61
6.9.	Vistas propuestas para el sistema web	61
6.10.	Delimitación de perfiles de usuario	66
6.11.	Diagrama de clases	66
6.12.	Casos de uso	66
6.13.	Trayectorias del sistema	66

7. Desarrollo del sistema	67
7.1. Planificación de actividades por iteración	67
8. Pruebas y resultados	69
8.1. Implementación de LEA en software	69
8.2. Pruebas del Key Schedule de LEA	70
8.3. Pruebas iniciales del software de prueba	75
9. Conclusiones	78
10. Trabajo a futuro	78
11. Anexos	78
12. Glosario	79

Índice de cuadros

1.	Trabajos similares realizados en ESCOM	17
2.	Rangos de frecuencia de operación de las etiquetas RFID.	32
3.	Comparación de implementaciones en hardware de los algoritmos de ISO/IEC 29192-2:2019	41
4.	Análisis que se le ha hecho a cada cifrado en ISO/IEC 29192-2:2019	42
5.	Comparación de etiquetas RFID	44
6.	Características de diferentes interfaces de comunicación	44

Índice de figuras

1.	Campos de la criptología [15]	19
2.	Instrumento Theremin	27
3.	Operación de "La cosa"	27
4.	Funcionamiento de RFID	29
5.	Lector RFID	30
6.	Lector RFID RC522	30
7.	Etiqueta RFID pasiva	31
8.	Etiqueta RFID activa	31
9.	Etiqueta RFID semi-pasiva	32
10.	Pasaporte electrónico	33
11.	Proceso de identificación de ganado por RFID	33
12.	Sistema de acceso RFID	33
13.	Carro de supermercado con tecnología RFID incorporada	34
14.	Seguimiento de pacientes por tecnología RFID	34
15.	Pago de peaje automático	34
16.	Esquema general de SPI	37
17.	Funcionamiento de SPI	39
18.	Esquema básico de un FPGA	39
19.	Modelo en V para el diseño de sistemas embebidos	45
20.	Arquitectura de sistema	49
21.	Módulo de Key Schedule	50
22.	Módulo de cifrado	53
23.	Módulo de descifrado	54
24.	Carta ASM propuesta para la unidad de control	57

25.	Diagrama a bloques de unidad de control	58
26.	Explicación de los bloques de unidad de control	58
27.	Pseudocódigo	59
28.	Arquitectura de LEA	59
29.	Arquitectura de puente SPI	60
30.	Arquitectura web propuesta	61
31.	Login de sistema	61
32.	Menú de administradores	62
33.	Creación de nuevo registro de médico	62
34.	Modificación de registro de médico	63
35.	Eliminación de registro de médico	63
36.	Menú de usuario médico	64
37.	Captura de datos de paciente para su posterior vaciado a etiqueta RFID	64
38.	Guardado de datos en etiqueta RFID	65
39.	Lectura de datos en etiqueta RFID	65
40.	Modificación de datos de paciente	66
41.	Prueba de LEA en software (1)	69
42.	Prueba de LEA en software (2)	69
43.	Prueba de LEA en software (3)	69
44.	Prueba de LEA en software (4)	69
45.	Prueba de LEA en software(5)	70
46.	Primera ronda de llave, generación de T[0],T[1],[2] y T[3] para i = 0	70
47.	T=K. Inicialización de memoria T	70
48.	Primera ronda de llave, generación de T[0],T[1],[2] y T[3] para i = 0	71
49.	Última ronda de llave, generación de T[0],T[1],[2] y T[3] para i = 23	71
50.	Verificación de los resultados anteriores	71

51.	Simulación de sumador	72
52.	RTL del sumador	72
53.	Simulación de ROL 1	72
54.	Simulación de ROL 2	72
55.	Diagrama RTL de ROL	73
56.	Simulación de la memoria de constantes de ronda	73
57.	Diagrama RTL de la memoria de constantes de ronda	73
58.	Prueba de la memoria de constantes delta	74
59.	Diagrama RTL de la memoria de constantes delta	74
60.	Simulación de la Memoria de llave, Memoria T y Buffer T	74
61.	Diagrama RTL de la memoria	75
62.	Prueba número 1	75
63.	Prueba número 2	76
64.	Prueba número 3	76
65.	Prueba número 4	77
66.	Prueba número 5	77
67.	Prueba número 6	78

1. Estructura del documento

El presente trabajo se encuentra estructurado como se muestra a continuación:

- **Capítulo 1 (Estructura del documento):** Se describe de manera breve el contenido de cada uno de los capítulos y apartados.
- **Capítulo 2 (Introducción):** Se ilustra una panorámica general del prototipo de sistema, una perspectiva histórica sobre el robo de información, estadísticas sobre el robo de información, planteamiento del problema, propuesta de solución al problema, objetivo general y objetivos específicos.
- **Capítulo 3 (Estado del arte):** En el tercer capítulo se realiza una revisión histórica acerca de los trabajos similares desarrollados tanto nivel interno o externo del instituto para su posterior comparativa en diferentes aspectos con el trabajo realizado en este documento. Además se muestra la comparación de los diferentes algoritmos criptográficos, placas de desarrollo, tecnologías y periféricos.
- **Capítulo 4 (Marco teórico):** Se abordan los fundamentos teóricos de los elementos que conforman el prototipo de sistema, así como una revisión histórica de las diferentes tecnologías y disciplinas usadas en este prototipo de sistema.
- **Capítulo 5 (Justificación):** Se presenta el sustento que respalda el desarrollo del proyecto, se presentan diferentes estadísticas del robo de información en el sector médico y sus principales implicaciones, así como las medidas que se proponen para auxiliar al sector médico.
- **Capítulo 6 (Análisis del sistema):** Se presenta un estudio sobre las tecnologías y plataformas a usar para dar solución a la problemática.
- **Capítulo 7 (Diseño del sistema):** En el séptimo capítulo se identifican los requerimientos funcionales y no funcionales tanto de hardware y software, además se muestran los diagramas de cada componente del prototipo de sistema, además de que se muestra la maquetación de las diferentes interfaces de usuario.
- **Capítulo 8 (Desarrollo del sistema):** Se describe el funcionamiento de cada uno de los módulos que componen el prototipo del sistema.
- **Capítulo 9 (Pruebas y resultados del sistema):** Se presentan cada una de las pruebas unitarias del prototipo de sistema y la integración de cada uno de los componentes.
- **Capítulo 10 (Conclusiones):** Parte final del trabajo terminal en el cual se hace un breve resumen de los puntos abordados en el trabajo, se exponen los resultados y se destacan los hallazgos más importantes.
- **Capítulo 11 (Trabajo a futuro):** Se describen los posibles aplicaciones con los avances presentados en el trabajo terminal.
- **Capítulo 12 (Anexos):** Se muestran los agregados del trabajo terminal, los cuales aportan información adicional que contiene el documento principal.

- **Glosario:** Listado de términos que son poco usuales para el público en general incorporando además su respectiva definición.
- **Referencias:** Listado de fuentes consultadas que auxiliaron en el desarrollo de la documentación y del prototipo de sistema.

2. Introducción

2.1. Antecedentes

El robo de información se ha encontrado presente a lo largo de la historia, sin embargo al surgimiento de las tecnologías de la información éste problema se ha agravado de manera significativa [1].

Desde los inicios de las tecnologías de la información, se ha llegado a la conclusión de que no existe un sistema que presuma de ser infalible. Por lo tanto, en un mundo en el que tantas organizaciones cuentan con acceso a una gran cantidad de información personal, el riesgo del robo de información se convierte en un factor a considerar, pues entre las posibles consecuencias se encuentran las siguientes: Persuasión comercial y política, fraude cibernético y robo de identidad, discriminación a partir de datos sensibles, entre otros.

En la actualidad, el uso de la informática, es algo que se encuentra alineado con la misma vida moderna, hoy toda actividad involucra aplicaciones informáticas para compartir datos, ya sea tanto de forma privada, como pública, por eso la importancia de proteger los activos de la información.

En México durante 2017, 92 % de empresas reportaron incidentes informáticos, 27 % de ellos fue debido a equipos robados con datos sensibles. Durante el mismo año, 85 % de empresas fueron afectadas por fraude. El tipo más común fue robo de datos con un 38 %, con los principales objetivos siendo datos de empleados y de clientes [2].

En los últimos años en México, el derecho a la privacidad o a la protección de datos personales se ha consolidado como un derecho fundamental y ha sido elevado a un rango institucional [3].

Dicho derecho se ha extendido al sector salud, donde en todo momento se maneja información personal catalogada como sensible. La transgresión de dicho derecho puede ocasionar diferentes consecuencias, especialmente en el mundo moderno en el que nos encontramos.

Un ejemplo de transgresión de dichos derechos se suscitó en 2010, cuando la aseguradora Dominion National reportó un robo de información médica durante nueve años en sus servidores, que potencialmente violó los datos de 2.96 millones de pacientes.

Una alerta interna avisó de diferentes accesos no autorizados a sus sistemas, lo que provocó una investigación por parte de las autoridades. Las cuales encontraron que el acceso no autorizado comenzó el 25 de agosto de 2010, casi nueve años de que se detectara la violación en abril de 2019. [4]

Un suceso similar sucedió en enero de 2019 cuando una base de datos mal configurada expuso información médica de 1.57 millones de pacientes de Inmediata Health Group.

La base de datos comprometida se descubrió, cuando personal de la organización descubrieron que un motor de búsqueda permitía indexar páginas web internas de la organización [4].

Finalmente en febrero de 2018, UW Medicine comenzó a notificar a 974,000 pacientes que sus datos se encontraron expuestos durante tres semanas debido a un servidor mal configurado. Ésta violación

de datos se descubrió en diciembre de 2018, cuando un paciente realizó una búsqueda con su propio nombre y encontró un archivo que contenía su información médica. Éste es un ejemplo en donde un solo individuo ha sido afectado [4].

Trasladando éste impacto a nivel industria se observa que en el caso de Estados Unidos, el costo total promedio de la violación de datos en el sector salud durante 2018 fue de 6.45 mdd, lo cual es 65 % más que el costo total promedio de una violación de datos [3]. De aquí podemos extraer que la implementación de medidas de seguridad al momento de estar tratando con información médica es una prioridad dentro del ámbito. Por otro lado, el costo por violación de datos de un archivo médico fue de \$429 [5].

Los afectados del sector salud por estos hechos reportaron dificultad para retener clientes después de haber reportado una violación de datos. 7 % de clientes se mudan a otro competidor después de que se han violado sus datos. Esto posiciona a la industria de la salud como el sector más afectado por éste tipo de siniestros [2].

Las estadísticas anteriores nos muestran el impacto económico que provoca al sector salud el robo de información, sin embargo las consecuencias también pueden afectar de manera significativa a los clientes de las organizaciones afectadas.

La revelación de información médica a entidades no autorizadas puede acarrear diferentes consecuencias a nivel personal, como la empleabilidad de una persona al solicitar empleo, ya que se sabe que existen ciertos genes que predisponen al cáncer de mama o enfermedad de Alzheimer. Si esos datos se conocen o se exigen al solicitar un trabajo, dichas condiciones médica pueden resultar desfavorables algunos candidatos.

Un caso más sobre la importancia de la confidencialidad de los datos médicos se encuentra suscitando en Alemania, donde la canciller Angela Merkel ha sido señalada por diferentes medios de comunicación respecto a su estado de salud [6], lo que de resultar ser cierto y ser revelado a la opinión pública, podría derivar en consecuencias políticas, económicas y de gobernabilidad.

Los ejemplos anteriores retratan la importancia de la confidencialidad médica de las personas. Sin embargo, la revelación de información médica se da en casos excepcionales cuando entra en juego la vida [7]. Cuando se trata de un accidente es de vital importancia que los profesionales de la salud tengan acceso a la información médica del paciente, para que este pueda recibir una atención adecuada y no comprometa aún más su estado de salud.

Con la finalidad de auxiliar a pacientes de centros médicos, se propone desarrollar un prototipo de sistema híbrido médico, donde se podrá visualizar y realizar los registros médicos por medio de la identificación del paciente.

El tag implementado será leído por un lector RFID, el cual se encontrará en comunicación con un dispositivo programable (Altera EP4CE10) que se encontrará dedicado a la ejecución de un algoritmo de cifrado.

Para esto se pretende que la persona interesada en obtener un dispositivo RFID, acuda a un centro médico en donde sea evaluado por un profesional de la salud que tiene la tarea de recopilar sus datos médicos. Ya recopilados, estos serán ingresados a un sistema de software, para su posterior cifrado,

utilizando un módulo de hardware externo, encargado de implementar un algoritmo de cifrado. Éstos datos cifrados se escriben en el dispositivo RFID para que el paciente cuente con un archivo médico cifrado portátil que podrá ser visualizado en centros médicos.

Por último en el escenario que por alguna razón los datos médicos contenidos en el dispositivo RFID requieran ser sobreescritos, se requiere la asistencia del paciente a un centro médico con dicho dispositivo, junto con su llave privada para que una persona autorizada pueda modificar sus datos médicos[1].

2.2. Planteamiento del problema

La industria de la salud tiene el costo más alto por violaciones de datos y es el sector más vulnerable ante clientes que deciden irse una vez enterados de tales ataques.

Mundialmente durante 2017, el 75 % de organizaciones de cuidado de la salud, farmacología y biotecnología reportaron alguna especie de fraude. De ésta cifra, el 23 % lidiaron con casos de robo de información [6]. El siguiente año (2018), por causas de robo de información, se dieron 47 incidentes reportados los cuales afectaron 771,656 archivos de pacientes. Por razones de pérdida de información, hubo un total de 11 incidentes reportados que afectaron 23,559 archivos de pacientes. Ésto nos da un total de 795,215 archivos afectados durante 2018 debido a robo o extravío de datos [6].

Para HIPAA, en el caso de ePHI (electronic Protected Health Information) durante 2018, se reportaron once violaciones de datos en los cuales los datos no estaban cifrados [7]. Si la información hubiera estado cifrada, algún tipo de protección en contra de estos ataques estaría presente.

En el caso de información de la salud dentro de México, en 2018 sucedió que Bob Diachecko, un investigador en el área de seguridad descubrió una base de datos perteneciente a Hova Health conteniendo los datos personales y médicos de 2,373,764 pacientes. Esto nos permite ver que en México existen casos de robo de información médica y el evidente descuido que lo rodea [8].

2.3. Propuesta de solución

Con la finalidad de auxiliar a pacientes de centros médicos, se propone desarrollar un prototipo de sistema híbrido médico, donde se podrá visualizar y realizar los registros médicos por medio de la identificación del paciente.

La gestión de los registros médicos será llevada en una aplicación de software. Para la identificación de los pacientes se empleará un tag de tecnología RFID (Higgs 3, 512 bits), el cual será de utilidad en ocasiones donde se requiera identificar a un paciente o conocer detalles médicos sobre él. Dicho tag será leído por un lector RFID, el cual se encontrará en comunicación con un dispositivo programable (Altera EP4CE10 FPGA) que se encontrará dedicado a la ejecución de un algoritmo de cifrado.

Para esto se pretende que la persona interesada en obtener un dispositivo RFID, acuda a un centro médico en donde sea evaluado por un profesional de la salud que tiene la tarea de recopilar sus datos

médicos. Ya recopilados, estos serán ingresados a un sistema de software, para su posterior cifrado, utilizando un módulo de hardware externo, encargado de implementar un algoritmo de cifrado. Éstos datos cifrados se escriben en el dispositivo RFID para que el paciente cuente con un archivo médico cifrado portátil que podrá ser visualizado en centros médicos.

Por último en el escenario que por alguna razón los datos médicos contenidos en el dispositivo RFID requieran ser sobreescritos, se requiere la asistencia del paciente a un centro médico con dicho dispositivo, junto con su llave privada para que una persona autorizada pueda modificar sus datos médicos.

2.4. Objetivos

2.4.1. Objetivo general

Implementar un prototipo de sistema híbrido para la portabilidad y protección de datos médicos que permita a pacientes tener su información cifrada por hardware dentro de un dispositivo RFID.

2.4.2. Objetivos específicos

- Implementar una interfaz que tenga acceso al servidor de sistema de datos médicos y al módulo de cifrado y descifrado.
- Implementar un servidor de base de datos en el cual se encontrará la información médica de los pacientes.
- Implementar un módulo de cifrado y descifrado en una FPGA Altera EP4CE10, para el procesamiento de la información médica del paciente y posterior transmisión a un dispositivo RFID.
- Desarrollar el módulo de software que gestione los registros médicos de los pacientes y credenciales de profesionales de la salud.

2.5. Justificación

La industria de la salud tiene el costo más alto por violaciones de datos y es el sector más vulnerable ante clientes que deciden irse una vez enterados de tales ataques.

Mundialmente durante 2017, el 75 % de organizaciones de cuidado de la salud, farmacología y biotecnología reportaron alguna especie de fraude. De ésta cifra, el 23 % lidiaron con casos de robo de información [6]. El siguiente año (2018), por causas de robo de información, se dieron 47 incidentes reportados los cuales afectaron 771,656 archivos de pacientes. Por razones de pérdida de información, hubo un total de once incidentes reportados que afectaron 23,559 archivos de pacientes. Ésto nos da un total de 795,215 archivos afectados durante 2018 debido a robo o extravío de datos[8].

Para HIPAA, en el caso de ePHI (electronic Protected Health Information) durante 2018, se reportaron 11 violaciones de datos en los cuales los datos no estaban cifrados [9]. Si la información hubiera estado cifrada, algún tipo de protección en contra de estos ataques estaría presente.

En el caso de información de la salud dentro de México, en 2018 sucedió que Bob Diachecko, un investigador en el área de seguridad descubrió una base de datos perteneciente a Hova Health conteniendo los datos personales y médicos de 2,373,764 pacientes. Esto nos permite ver que en México existen casos de robo de información médica y el evidente descuido que lo rodea [10].

Una posible medida ante el robo de información médica que depende del paciente es el cifrado de la misma, dado que en caso de robo, los datos recolectados serían ilegibles para agentes externos que deseen hacer uso de dicha información.

Ante esto, se propone la elaboración de un prototipo de sistema híbrido médico centrado en el cifrado y descifrado de datos médicos con el propósito de brindarles a los pacientes un archivo de médico portátil.

El enfoque principal del prototipo es la protección de datos cuando se tenga un archivo médico dentro de la tarjeta. Si la tarjeta se llega a perder o clonar, el atacante solo obtendrá información cifrada, no los datos médicos del paciente. El cifrado brinda una capa de protección en estos casos, a diferencia de un dispositivo con datos visibles.

Para el desarrollo de este proyecto se propone utilizar un lenguaje de descripción de hardware VHDL y su sintetización en un FPGA para la implementación de un algoritmo de cifrado, además de la habilitación de periféricos para su comunicación con el lector/escritor RFID (RC522) y con el módulo de software.

2.6. Alcances y limitaciones

El trabajo terminal se encuentra limitado a los siguientes puntos:

- El diseño e implementación de la arquitectura se hace con base en la tarjeta de desarrollo Altera EP4CE10 FPGA.
- La herramienta de desarrollo a utilizar es Quartus 13.1 y ModelSim Altera como herramienta de simulación.
- Se desarrollan cada uno de los módulos de la arquitectura, sin embargo se tomaron algunos de los elementos propios del lenguaje de descripción de hardware para una mayor agilización del desarrollo.
- Este trabajo terminal se centra únicamente en la arquitectura que implementa AES y en la gestión de información médica.
- El proceso de generación de llaves se limita a nivel software.

3. Estado del arte

Para tener una idea acerca de los trabajos que han sido desarrollados , así como de los elementos que los conforman, hemos decidido realizar nuestro estudio completo donde no solo se comparan de manera general trabajos similares, si no que adicionalmente, se presentan las comparativas de los dispositivos programables, ambientes de desarrollo y algoritmos que se tomaron en cuenta para el desarrollo de este trabajo terminal.

A continuación vamos a realizar una comparación de los trabajos tanto internos como externos que han sido presentados anteriormente que poseen características similares al descrito en este documento.

3.1. Comparativa con trabajos internos

Características/ Trabajo terminal	TT-2019-B086	TT-12-1-0009 [9]	TT-2014-B011 [10]	TT-2012-B005 [11]	TT-2012-A054 [10]
Tipo de aplicación	Web	Web	Móvil	Móvil	Web
Uso de tecnología RFID	Sí	No	Sí	No	No
Norma oficial mexicana utilizada	NOM-024-SSA3-2012	NOM-168-SSA1-1998	NOM-024-SSA3-2012 NOM-168-SSA1-1998	NOM-168-SSA1-1998 NOM-024-SSA3-2010	NOM-024-SSA3-2010
Cifrado por hardware	Sí	No	No	No	No
Orientada a una institución en particular	No	Sí	No	Sí	No
Tipo de implementación	Software y hardware	Software	Software y hardware	Software y hardware	Software
Tipo de acceso al sistema	Manual	Manual	Tecnología RFID	Manual	Manual
Expediente clínico electrónico	No	Sí	No	Sí	Sí

Cuadro 1: Trabajos similares realizados en ESCOM

3.1.1. TT-12-1-0009: Historial Clínico para el Departamento de Servicio Médico de ESCOM

Trabajo terminal realizado en la Escuela Superior de Cómputo en el año 2010, el cual va enfocado al manejo de historiales clínicos del departamento de servicio médico de la misma escuela. El manejo de la información médica está basado en la Norma Oficial Mexicana NOM-168-SSA1-1998. El sistema está basado en una arquitectura cliente servidor desarrollada en una plataforma web, la cual puede ser accedida desde un dispositivo móvil por el cliente. El servidor hace uso de un sistema gestor de base de datos para almacenar la información con la que se trabaja [9].

3.1.2. TT-2014-B011: Sistema Móvil de Registros de Enfermería implementando Tecnología RFID

Trabajo terminal realizado en la Escuela Superior de Cómputo en el año 2014, el cual consiste en el desarrollo de un prototipo de sistema híbrido que va a permitir la realización de las funciones de registro del área de enfermería haciendo uso de tecnología RFID para almacenar la información de los pacientes que van a ser atendidos, además de dispositivos móviles para el registro de información

en las hojas de enfermería. El manejo de la información médica está basado en la Norma Oficial Mexicana NOM-168-SSA1-1998, NOM-004-SSA3-2012 y NOM-024-SSA3-2012. Se hace uso de un microcontrolador para realizar la lectura del RFID, cuya información va a ser transmitida a la aplicación móvil de las enfermeras mediante UART, que a su vez, va a ser almacenada en un servidor [10].

3.1.3. TT-2012-B005: Sistema de Expediente Clínico con Adquisición Automática de Datos

Trabajo terminal realizado en la Escuela Superior de Cómputo en el año 2012, el cual consiste en el desarrollo de un sistema de expediente clínico basado en la Norma Oficial Mexicana NOM-168-SSA1-1998, el cual va a ser actualizado de manera constante mediante el uso de sensores que van a comunicarse a través de bluetooth con el dispositivo móvil que contendrá los datos del expediente [11].

3.1.4. TT-2012-A054: Expediente Clínico Electrónico

Trabajo terminal realizado en la Escuela Superior de Cómputo en el año 2012, el cual consiste en el desarrollo de un sistema de expediente clínico basado en la Norma Oficial Mexicana NOM-024-SSA3-2010, para la atención de pacientes del Centro Interdisciplinario de Ciencias de la Salud (CICS), Unidad Santo Tomás [10].

Se hace uso de una plataforma web en una arquitectura en tres capas para la automatización de procesos tanto de manejo de información de los pacientes como del personal médico.

3.2. Comparativa con trabajos externos

3.2.1. Using RFID Technology for Managing Patient Medical File

Trabajo realizado en la World University of Bangladesh, ubicada en la ciudad de Dhaka, Bangladesh en el año 2018. Este consiste en un sistema sencillo donde se hace uso de un escáner RFID para la información médica que va a ser contenida en etiquetas del mismo tipo. Los datos obtenidos van a ser procesados desde una computadora. Se hace la comparación de el manejo de archivos físicos y con código de barras para demostrar que el uso de etiquetas resulta ser más rápido y eficiente [12].

3.2.2. RFID for Inventory of Medical Records

Proyecto realizado en la California Polytechnic State University, ubicada en San Luis Obispo, Estados Unidos en el año 2012. Este consiste en la implementación de un sistema RFID en un hospital para personas de bajos recursos, denominados CMSP (County Medical Service Program) para el manejo de diversos expedientes médicos ,donde se expone que el tiempo de espera de varios pacientes es demasiado, además de que existe la posibilidad de cometer errores debido a que existen

varios documentos sin ordenar. Al final se demuestra que a pesar de reducir los tiempos de espera con la implementación del sistema, algunos empleados tuvieron problemas al cambiar del sistema tradicional al propuesto en este trabajo [13].

3.2.3. Utilización de tecnología RFID para optimizar la operatividad y administración de una institución hospitalaria

Proyecto desarrollado en el Instituto Tecnológico de Buenos Aires, Argentina en el año 2016, el cual consiste en la implementación de dispositivos RFID para la mejora de procesos logísticos, tales como el manejo de la información sensible de los pacientes, control de inventarios y la automatización de procesos [14].

4. Marco teórico

En el presente capítulo se desarrolla la teoría para sustentar el prototipo de sistema. En las próximas secciones se describirá los orígenes y funcionamiento de los diferentes algoritmos criptográficos candidatos a implementar a nivel hardware, también se describe la historia y funcionamiento de la tecnología RFID, además descripción general de las tecnologías utilizadas en este trabajo terminal.

4.1. Visión general de la criptología

Es posible que al escuchar la palabra criptografía la primera asociación que hagamos sea con el cifrado de correos electrónico, accesos seguros a sitios web, pequeñas tarjetas para aplicaciones de banco o bien nos podemos remontar a sucesos históricos, como el rompimiento del código enigma alemán por parte de Alan Turing y su máquina [15].

En general la criptología se divide en dos ramas, por una parte tenemos el criptoanálisis y por otro la criptografía.

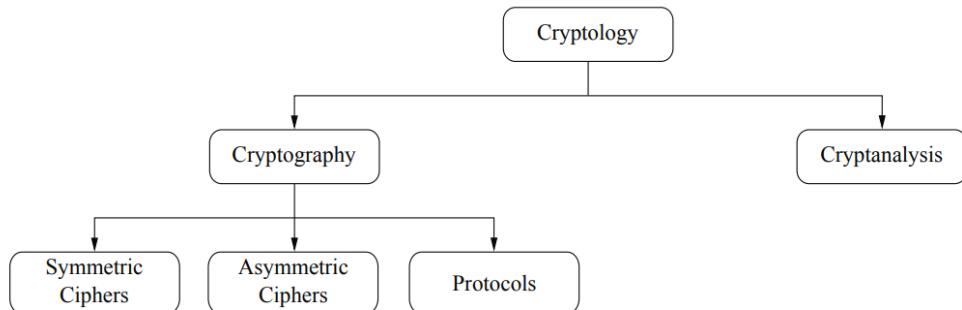


Figura 1: Campos de la criptología [15]

4.2. Campos de la criptología

De acuerdo con [15]:

- **Criptografía:** Es la ciencia de la escritura, con el principal objetivo de ocultar el significado de los mensajes.
- **Criptoanálisis:** Es la ciencia de romper criptosistemas, el cual es de gran importancia para los criptosistemas modernos, porque de esta manera es posible conocer si un criptosistema es realmente seguro.

La criptografía a su vez se divide en tres ramas: criptografía simétrica, criptografía de llave pública y protocolos criptográficos

- **Criptografía simétrica:** es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar.
- **Criptografía asimétrica:** Es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que recibirá el mensaje.
- **Protocolos criptográficos:** Hace referencia a la aplicación de algoritmos criptográficos donde en la mayoría de aplicaciones se combina la criptografía simétrica y asimétrica.

4.3. Criptografía ligera

Según [16], los algoritmos de criptografía convencional como AES (Cifrado), SHA-256 (Hash) y RSA/Curvas elípticas (firma digital) trabajan de manera adecuada en sistemas que tienen un buen poder de procesamiento y capacidades de memoria, sin embargo, estos no se adaptan de manera adecuada en sistemas embebidos y en redes de sensores, por lo que se han propuesto métodos criptográficos ligeros para superar muchos de estos problemas en sistemas con capacidades limitadas.

Es posible dividir la criptografía en dos grupos, por un lado la criptografía convencional y la ligera los cuales son mayormente ocupados en los dispositivos que se listan a continuación:

- **Criptografía convencional:** Servidores, Computadoras portátiles y de escritorio, Tablets y smart phones.
- **Criptografía ligera:** Sistemas embebidos, RFID y nodos de sensores.

4.4. Algoritmos de cifrado ligero

Existen diversos algoritmos de cifrado ligero, pero esto no significa que sean seguros. Hoy en día existen fuentes oficiales que aprueban a ciertos algoritmos para su uso en diversas aplicaciones, como [17] y [18]. En éste trabajo solamente profundizamos un poco en los 3 algoritmos mencionados en [17].

4.5. Proceso de estandarización de NIST

NIST, a partir del 20 de julio de 2015 ha comenzado un proceso para *solicitar, evaluar y estandarizar algoritmos de cifrado ligeros que son aptos para su uso dentro de ambientes limitados*. Éste proceso todavía no termina, pero se puede ver el transcurso de él en [19].

4.6. ISO/IEC 29192-2:2019

Éste estándar *especifica a tres cifrados por bloques aptos para aplicaciones que requieran implementaciones de criptografía ligera*[17]. Posteriormente se compararán para decidir cuál es el algoritmo más apto a las necesidades planteadas por éste trabajo.

4.7. Lightweight Encryption Algorithm (LEA)

LEA es un cifrador ligero por bloques surcoreano creado por Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu y Dong-Geon Lee. El algoritmo provee tamaños de bloque de 128 bits y tamaños de llave de 128, 192 y 256 bits. El algoritmo fue diseñado para dispositivos de recursos limitados y gadgets que son participes del internet de las cosas. Provee una rápida encriptación en procesadores de propósito general además de ser más rápido que AES en plataformas de Intel, AMD, ARM y ColdFire además de ser seguro ante todos los ataques existentes de cifradores por bloque.

4.8. Características

La estructura de LEA tiene las siguientes características:

- LEA consiste en únicamente operaciones ARX(Adición modular, Rotación y XOR) para palabras de 32 bits. Estas operaciones son bien soportadas y rápidas por plataformas de 32 bits y 64 bits. Además de que el uso de procesadores de 32 bits y 64 bits crecerá rápidamente a comparación de los procesadores de 8 bits y 16 bits.
- Las operaciones ARX contribuyen a un cifrado y proceso de generación de llaves en una forma eficiente y paralela.

- La última ronda es generada de la misma manera que las demás rondas mientras que muchos cifradores de bloque como DES y AES tienen rondas especiales para su última ronda.

4.9. Seguridad

El objetivo para la seguridad de LEA es obtener resistencia a todos los ataques de cifradores de bloque conocidos, para lograr éste objetivo se encontró el número de rondas mínimos para LEA que resistiera todas las técnicas de criptoanálisis para cada tamaño de llave.

4.10. Eficiencia

LEA provee un cifrado rápido en múltiples plataformas, los experimentos midieron la velocidad de cifrado para un bloque en plataformas de Intel,AMD,ARM y ColdFire lo cual arrojó que incluso a un nivel de C LEA es muy rápido. Incluso la versión optimizada de 128 bits de LEA es más rápida que AES-128 [20] reportadas públicamente.

También LEA puede ser implementado con un código reducido. Las implementaciones de LEA en forma reducida es útil en sistemas donde se tiene memoria limitada. LEA-128 es implementado con 600 y 750 bytes en plataformas de ARM926EJ-S y ColdFire MCF5213 respectivamente, mientras que AES-128 es implementado con 2400 y 960 bytes en plataformas ARM7TDMI y ColdFirev2 respectivamente.

4.11. Especificación de LEA

LEA es un cifrador por bloques con un tamaño de bloque de 128 bits y tamaños de llave de 128,192 y 256 bits. El número de rondas es 24 para llaves de 128 bits, 28 para las de 192 bits y 32 para las de 256 bits.

4.12. Notación

- **P** : Texto plano de 128 bits el cual consiste en cuatro palabras de 32 bits $P = (P[0], P[1], P[2], P[3])$
- **C** : Texto cifrado de 128 bits el cual consiste en cuatro palabras de 32 bits $C = (C[0], C[1], C[2], C[3])$
- **Len(x)** : Tamaño en bits de una cadena x
- **K** : Llave maestra que es denotada por la concatenación de palabras de 32 bits
 $K = (K[0], K[1], K[2], K[3])$ cuando **Len(K) = 128**;
 $K = (K[0], K[1], \dots, K[5])$ cuando **Len(K) = 192**;
 $K = (K[0], K[1], \dots, K[7])$ cuando **Len(K) = 256**;

- **RK** : Denota la concatenación de todas las rondas de llave.

$$RK = (RK_0, RK_1, \dots, RK_{r-1})$$

Donde RK_i es la llave de ronda de 192 bits de la ronda i . Cada RK_i se encuentra conformada por seis palabras de 32 bits.

$$RK_i = (RK_i[0], RK_i[1], \dots, RK_i[5])$$

- **X \oplus Y**: Or exclusiva entre dos cadenas x & y de misma longitud.
- **X \boxplus Y**: Adición módulo 2^{32} de dos cadenas x & y
- **ROL_i(x)** : Rotación de i bits a la izquierda de una palabra x de 32 bits.
- **ROR_i(x)** : Rotación de i bits a la derecha de una palabra x de 32 bits.
- **X_i** : Valores intermedios de 128 bits (Entradas de la i -ésima ronda en la función de cifrado), consiste en 4 palabras de 32 bits

$$X_i = (X_i[0], X_i[1], \dots, X_i[5])$$

4.13. Key Schedule

El proceso de generación de llaves genera i llaves de ronda de 192 bits cada una.

4.13.1. Constantes

El proceso de generación de llaves usa diferentes constantes para generar llaves de ronda, las cuales se encuentran definidas como:

$$\delta[0] = 0xc3efe9db, \delta[1] = 0x44626b02,$$

$$\delta[2] = 0x79e27c8a, \delta[3] = 0x78df30ec,$$

$$\delta[4] = 0x715ea49e, \delta[5] = 0xc785da0a.$$

$$\delta[6] = 0xe04ef22a, \delta[7] = 0xe5c40957.$$

Las constantes anteriores son obtenidas de $\sqrt{766995}$, donde 76,69 y 95 son los valores ASCII de 'L', 'E', 'A'.

4.13.2. Key Schedule con 128 bits

Primeramente se partitiona la llave maestra en palabras de 32 bits, en el caso de 128 bits quedaría de la siguiente manera:

$$K = (K[0], K[1], K[2], K[3])$$

Se iguala la variable $T[i] = K[i]$ for $0 \leq i < 4$.

Posteriormente se genera la ronda de llave $RKi = (RKi[0], RKi[1], \dots, RKi[5])$ de la siguiente manera:

Algoritmo 1: Key Schedule (128 bits)

```
for i ← 0 to 24 do
    T[0] ← ROL01(T[0] ⊕ ROLi+0(δ[i mod 4])),
    T[1] ← ROL03(T[1] ⊕ ROLi+1(δ[i mod 4])),
    T[2] ← ROL06(T[2] ⊕ ROLi+2(δ[i mod 4])),
    T[3] ← ROL11(T[3] ⊕ ROLi+3(δ[i mod 4])),
    RKi ← (T[0], T[1], T[2], T[1], T[3], T[1]).
```

4.13.3. Key Schedule con 192 bits

Primeramente se partitiona la llave maestra en palabras de 32 bits, en el caso de 192 bits quedaría de la siguiente manera:

$$K = (K[0], K[1], \dots, K[5])$$

Se iguala la variable $T[i] = K[i]$ for $0 \leq i < 6$.

Posteriormente se genera la ronda de llave $RKi = (RKi[0], RKi[1], \dots, RKi[5])$ de la siguiente manera:

4.14. Proceso de cifrado

El proceso de cifrado de LEA consiste en 24 rondas para llaves de 128 bits, 28 rondas para llaves de 192 bits, y de 32 para llaves de 256 bits. Para r rondas se cifran 128 bits $P = (P[0], P[1], P[2], P[3])$ es convertido a un texto cifrado de 128 bits $C = (C[0], C[1], C[2], C[3])$

4.14.1. Inicialización

Se inicializa X_0 con 128 bits del texto plano P, posteriormente se hace el proceso de generación de llaves para generar r llaves de ronda.

4.14.2. Iteración de rondas

Los 128 bits de salida de $X_{i+1} = (X_{i+1}[0], X_{i+1}[1], \dots, X_{i+1}[3])$ de la i ronda se calcula de la siguiente manera:

Algorithm 2:

```
for  $i \leftarrow 0$  to  $r$  do
     $X_{i+1}[0] \leftarrow ROL_{09}((X_i[0] \oplus RK_i[0]) \boxplus (X_i[1] \oplus RK_i[1])),$ 
     $X_{i+1}[1] \leftarrow ROR_{05}((X_i[1] \oplus RK_i[2]) \boxplus (X_i[2] \oplus RK_i[3])),$ 
     $X_{i+1}[2] \leftarrow ROR_{03}((X_i[2] \oplus RK_i[4]) \boxplus (X_i[3] \oplus RK_i[5])),$ 
     $X_{i+1}[3] \leftarrow X_i[0],$ 
```

4.15. Finalización

El texto cifrado es producido por X_r y se asigna de la siguiente manera:

$$C_0 \leftarrow X_r[0], C_1 \leftarrow X_r[1], C_2 \leftarrow X_r[2], C_3 \leftarrow X_r[3]$$

$$C \leftarrow (C_0, C_1, C_2, C_3)$$

4.16. RSA

4.16.1. Introducción

RSA es un sistema criptográfico de llave pública desarrollado en 1979 por Ron Rivest, Adi Shamir y Leonard Adleman en el Instituto de Tecnología de Massachusetts. En este sistema, tanto la llave pública como la llave privada pueden ser utilizadas para cifrar un mensaje, siendo la llave opuesta la utilizada para el proceso de descifrado. Para ello, hace uso de la factorización de números primos de gran tamaño, de tal forma que tomaría un tiempo indefinido para encontrar los números utilizados para la generación de llaves.

4.16.2. Funcionamiento

Dados dos números primos generados, p y q , se obtiene un número n , el cual se obtiene al multiplicar p y q . Este número es que el que va a ser utilizado tanto por la llave pública como por la llave privada, cuya longitud en bits determinará la longitud de la llave.

La llave pública es generada con el número n y un exponente público e (generalmente ajustado a 65537), el cual al ser público, no necesita ser un número secreto, mientras que la llave privada es generada por el número n y un exponente secreto d , el cual debe de ser calculado mediante el algoritmo de Euclídes Extendido para encontrar el inverso multiplicativo dado el número φ de la función indicatriz de Euler, que representa el número de enteros positivos menores o iguales al número n . Para calcularlo, se pueden considerar tres casos:

- Si n es primo, entonces $\varphi = p - 1$
- Si $n = p * q$, entonces $\varphi = (p - 1) * (q - 1)$
- Si $n = p^n$, entonces $\varphi = p^n - p^{n-1}$

Por ejemplo, se va a suponer el escenario en el que Bob deseó mandarle un mensaje a Alicia, para ello Alicia debe de generar sus llaves mediante números primos. En ese caso se decide que $p = 11$ y $q = 13$. Por ende, el módulo n resultante de la operación $p * q$ es igual a 143. Ahora, calcularemos el número φ .

Dado que n es el resultado del producto de los números p y q , utilizamos el modelo matemático del segundo caso, donde $\varphi = (p - 1) * (q - 1)$. Al sustituir tenemos que $\varphi = (11 - 1) * (13 - 1) = 120$. Ahora, Alicia elige el número 7 para que con él se genere su llave pública, para calcular su llave privada, se utiliza el algoritmo de Euclídes Extendido, lo cual da como resultado 103.

Ahora, Bob le enviará el mensaje M a Alicia. Para ello, debe de conocer su llave pública, lo cual hace mediante el par de números previamente conocidos, 143 y 7.

El mensaje a cifrar será el número 9 (puede ser cualquier mensaje), por lo que tenemos que:

$$M^e \bmod n = 9^7 \bmod 143 = 48 = C$$

Siendo C el mensaje cifrado.

Ahora, para descifrar dicho mensaje, Alicia debe de realizar el proceso inverso con su llave privada, esto es:

$$C^d \bmod n = 48^{103} \bmod 143 = 9 = M$$

4.17. Historia de los sistemas RFID

La tecnología RFID ha sido desarrollada por múltiples investigadores a lo largo de la historia. La primera aparición de RFID sucedió con Leon Theremin, quien fue un inventor ruso que en 1929 creó un instrumento musical, el cual bautizó como theremin. Éste se encontraba compuesto por un par de antenas contenidas en una caja, lo cual permitía ajustar el tono del instrumento y su volumen, este instrumento era operado al acercar y alejar las manos de las antenas [37].



Figura 2: Instrumento Theremin

Posterior a su invención Leon Theremin desarrolló una herramienta de espionaje para el gobierno ruso el cual fue nombrado como "La cosa". Éste dispositivo fue el primer micrófono oculto pasivo que usaba transmisión por radiofrecuencia, dicho dispositivo fue utilizado para espiar al embajador americano en Rusia.

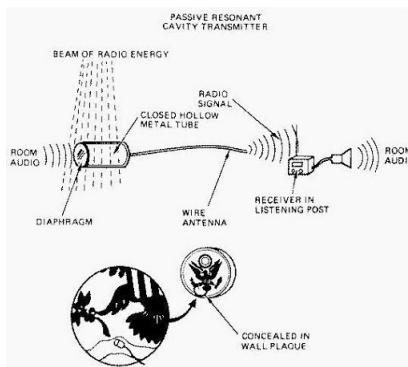


Figura 3: Operación de "La cosa"

"La cosa" hacia uso de inducción electromagnética para la transmisión de una señal de audio, siendo esta activada por ondas electromagnéticas (microondas) desde una fuente externa al recinto, lo que hacia casi imposible su detección.

Durante la segunda guerra mundial los británicos hicieron uso de un sistema que permitía identificar las aeronaves aliadas que regresaban de la costa, por lo que cada aeronave contaba con su propio identificador que era catalogado como enemigo o aliado, dicho sistema llevó como nombre IFF.

El avance de la tecnología RFID tuvo un salto importante durante la década de los 60's, se publicaron numerosos artículos, y además la actividad comercial en este campo comenzó a surgir. Uno de los más importantes investigadores en el área fue Roger F. Harrington quien en sus investigaciones "Theory of loaded" y "Field measurements using active scatters" estudió la teoría electromagnética relacionada a la tecnología RFID, por su parte los inventores se dedicaron a aplicar la tecnología de radiofrecuencia a los dispositivos destinados con distinto propósito al militar, algunos de los inventores más destacados que emplearon dicha tecnología fueron: J. P . Vinding's con "Interrogator-responder identification system" en 1967, J. H. Vogelman's con "Passive data transmission techniques utilizing radar beams" en 1968, Robert Richardson's con "Remotely activated radio frequency powered devices" en 1963 y Otto Rittenback's con "Communication by radar beans".

Años posteriores las compañías Knogo, Sensormatic y Checkpoint desarrollaron sistemas de vigilancia electrónica de artículos, dichos dispositivos almacenaban un bit de información y su única función era identificar la presencia o ausencia del artículo, esto auxilio a evitar robos.

Durante la década de los setenta diferentes instituciones académicas, investigadores independientes y empresas de laboratorios gubernamentales comenzaron a desarrollar la tecnología RFID. El enfoque principal de la época fue hacia los animales, el cobro de peaje, seguimiento de vehículos y la automatización de la fábrica. La primer patente realizada que tuvo que ver con tecnologías RFID se dió un 23 de enero de 1973 a la compañía ComServ, la cual se dió como finalizada a principios de los 90's.

Durante la próxima década, la tecnología RFID se encontraba inmersa en los Estados Unidos, se enfocó en el control de accesos, aplicaciones de transporte y empezaron a tener menor presencia en los animales. En ciertos países europeos se centraron mayormente en aplicaciones industriales y sistemas de corto alcance para el control de animales.

Posteriormente, IBM empezó a incursionar en la tecnología RFID, hasta obtener una patente de un sistema RFID de ultra-alta frecuencia (UHF), éste sistema ofrecía mayor alcance de lectura (Hasta 20 pies) y la transferencia de datos era más rápida. En la década de los 90's RFID empieza a ser ampliamente usado por empresas a nivel mundial y consumidores, tanto así que empezaron a surgir las primeras normas. En Europa se comenzaron a usar sistemas de microondas e inductivos para control de accesos y billetes electrónicos y aplicaciones para autopistas, por otra parte en Estados Unidos se comenzó a controlar de manera electrónica el peaje, en autopistas de Houston y Oklahoma, además de estas partes del mundo, también la tecnología RFID se fue extendiendo por Asia, África, Suramérica y Australia.

El RFID de alta frecuencia fue altamente apoyado en el año 1999, cuando Uniform Code Council, EAN Internacional, Procter & Gamble y Gillete invirtieron fondos para dar apertura al Centro de Auto-ID en el MIT (Massachusetts Technology Institute). Dos profesores de dicho instituto estaban realizando investigaciones para tener la posibilidad de colocar etiquetas RFID de bajo costo en todos los productos, esto para realizar un seguimiento en las cadenas de suministro. Durante los años 1999 y 2003, el Centro Auto-ID obtuvo el apoyo de más de 100 empresas de usuario final, además del departamento de defensa del gobierno estadounidense.

En los siguientes años se abrieron laboratorios de investigación en Australia, el Reino Unido, Suiza, Japón y China en los cuales se desarrolló dos protocolos de interfaz de aire, el código electrónico-

co de producto (EPC) de numeración de esquema, y una arquitectura de red utilizada para buscar datos relacionados en una etiqueta RFID en Internet. En el año 2003 el centro Auto-ID retuvo sus actividades que se relacionaban con la tecnología RFID.

4.18. Tecnología RFID

La tecnología RFID (Radio Frequency Identification), permite la identificación por medio de una onda emisora que transmite por radiofrecuencia los datos identificativos del objeto. [38].

4.18.1. Funcionamiento de RFID

El funcionamiento de la tecnología RFID se basa en dos dispositivos, por un lado el lector RFID y por el otro la etiqueta RFID. La etiqueta RFID es la encargada de almacenar la información de identificación, mientras que el lector se encuentra emitiendo una serie de ondas de radiofrecuencia las cuales se les da el nombre de señal de interrogación, cuando una etiqueta se encuentra en el rango de estas ondas de radiofrecuencia es energizada, dicha energía es utilizada para alimentar los circuitos internos de la etiqueta, para su posterior transmisión de la información almacenada. La información recibida por el lector. [38].

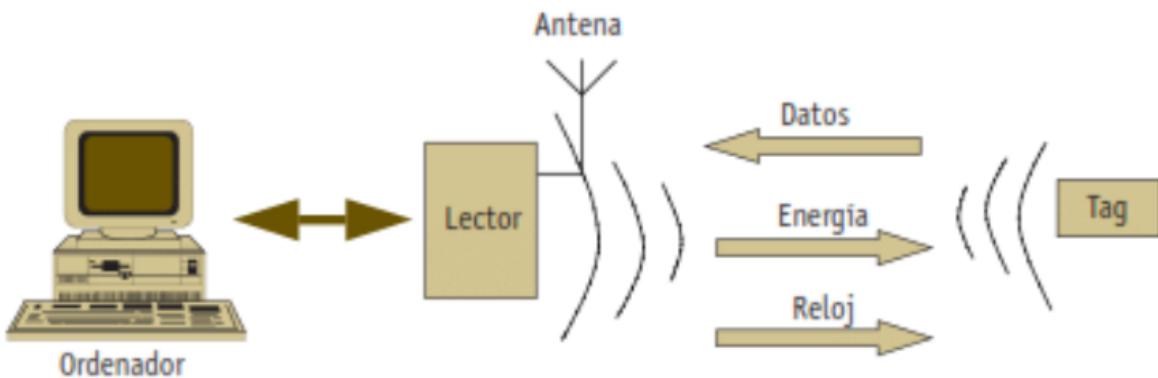


Figura 4: Funcionamiento de RFID

4.18.2. Arquitectura de un sistema RFID

Un sistema RFID se compone por cuatro elementos principales [39].

- **Etiqueta RFID:** Una etiqueta RFID, tags o transpondedores hace referencia a dispositivos que permiten almacenar y enviar información a través de ondas de radiofrecuencia.

Las etiquetas RFID se encuentran compuestas por tres elementos: Una antena, un transductor y un microchip.

La antena es la encargada de transportar la información que identifica a la etiqueta, mientras el transductor es el que transforma la información proveniente de la antena.

El microchip contiene una memoria interna que almacena el número de identificación de la etiqueta RFID y en algunos casos contiene datos.

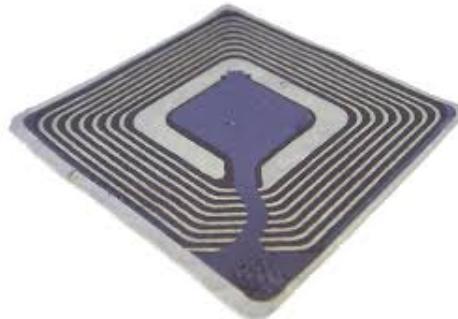


Figura 5: Lector RFID

- **Lector RFID:** El lector RFID es el encargado de recibir información proveniente de las etiquetas y su posterior transferencia al subsistema de procesamiento de datos. El lector RFID se compone por tres elementos: Antena, Transceptor y Decodificador.



Figura 6: Lector RFID RC522

- **Subsistema de procesamiento de datos:** Se trata de un software alojado en un servidor y que es un intermediario entre el lector y la aplicación. Es el encargado de filtrar información que recibe un lector, de tal manera de recibir información de utilidad.
- **Programadores RFID:** Son los dispositivos encargados de realizar la escritura de información sobre una etiqueta RFID, lo que quiere decir, codifican la información en el microchip situado en la etiqueta RFID.

4.18.3. Tipos de etiquetas RFID

En el mercado se encuentran una gran variedad de etiquetas RFID, las cuales varían en tamaño, fuente de alimentación, capacidad de almacenamiento, frecuencia, entre otros.

Las etiquetas RFID se dividen por tipo de etiqueta y frecuencia a la que trabajan: [40].

- **Etiquetas pasivas:** No poseen ninguna fuente de alimentación, obtiene su energía a partir de las ondas electromagnéticas emitidas por el lector RFID, por lo que la etiqueta no tiene la capacidad de comunicarse hasta que entra en interacción, trabajan a una distancia de usualmente unos milímetros hasta 6 o 7 metros.

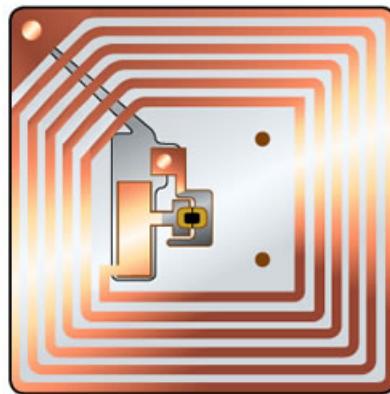


Figura 7: Etiqueta RFID pasiva

- **Etiquetas activas:** Contienen una batería integrada que les permite comunicarse de manera autónoma, por lo que su alcance, potencia y almacenamiento es mayor, son más confiables y pueden trabajar en entornos especializados

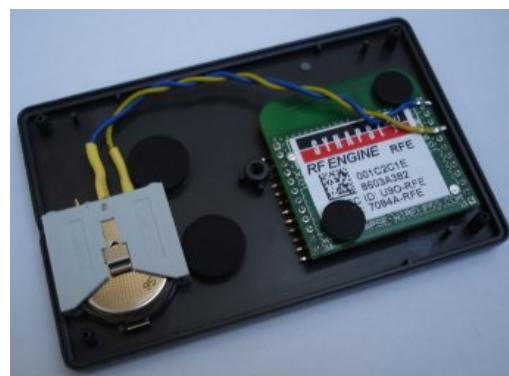


Figura 8: Etiqueta RFID activa

- **Etiquetas semi-pasivas:** Son una mezcla de etiquetas activas y pasivas, ya que posee internamente una batería, pero también aprovecha el campo electromagnético generado por el lector RFID generado por el lector RFID.



Figura 9: Etiqueta RFID semi-pasiva

Se puede clasificar de acuerdo a la frecuencia que trabajan, como baja, alta, ultra alta frecuencia y microondas. La frecuencia de operación determina características de la etiqueta como la capacidad de transmisión de datos, velocidad, tiempo de lectura, radio de cobertura y el costo. En la tabla siguiente es posible apreciar el rango de frecuencias utilizadas por los sistemas RFID.

Frecuencia	Denominación	Rango
125 KHz - 134 KHz	LF (Baja frecuencia)	Hasta 45 [cm]
13.553 MHz - 13.567 MHz	HF (Alta frecuencia)	De 1 [m] a 3 [m]
400 MHz - 1000 MHz	UHF (Ultra Alta Frecuencia)	De 3 [m] a 10 [m]
2.45 GHz - 5.4 GHz	Microondas	Más de 10 [m]

Cuadro 2: Rangos de frecuencia de operación de las etiquetas RFID.

4.18.4. Aplicaciones de la tecnología RFID

En la actualidad la tecnología RFID tiene una amplia aplicación en el mundo moderno, ya que se encuentra en una etapa de masificación, esto debido a la constante reducción de costos de fabricación de dispositivos RFID, lo que ha facilitado su incorporación con dispositivos electrónicos, animales, ropa, libros, pulseras, productos de supermercado, etc. [41].

Entre las aplicaciones actuales se encuentran las que se listan a continuación:

- **Pasaportes:** En múltiples países en los pasaportes se implementan chips RFID, el cual almacena información de la persona.

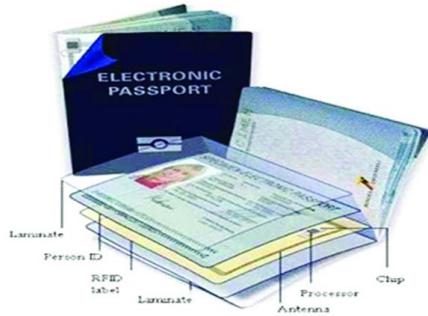


Figura 10: Pasaporte electrónico

- **Control de ganado:** Las etiquetas RFID son implementadas en la oreja de los animales con la finalidad de llevar a cabo una identificación adecuada.

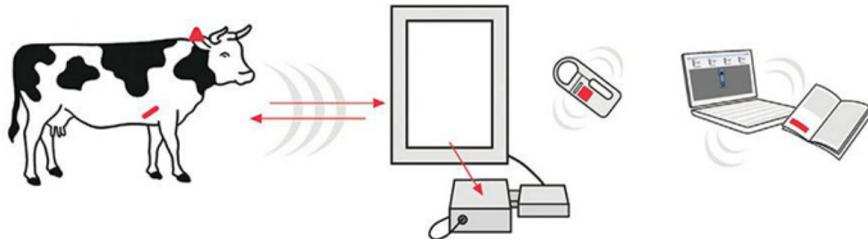


Figura 11: Proceso de identificación de ganado por RFID

- **Control de acceso:** Es de las aplicaciones más comunes de la tecnología RFID la cual es posible encontrarla en zonas residenciales, hoteles, oficinas, estacionamientos, escuelas, etc.



Figura 12: Sistema de acceso RFID

- **Supermercados:** En los supermercados se comienzan a implementar dispositivos RFID, esto para poder llevar a cabo una facturación de los productos.



Figura 13: Carro de supermercado con tecnología RFID incorporada

- **Sector salud:** En el sector salud se usan los sistemas RFID para llevar a cabo el inventario de los medicamentos, muestras médicas, seguimiento de pacientes, controles de acceso a los recintos, etc.



Figura 14: Seguimiento de pacientes por tecnología RFID

- **Pago de peaje:** Prácticamente en la mayoría de las autopistas se encuentra implementado un sistema RFID, el cual brinda un mejor flujo de vehículos por las autopistas.

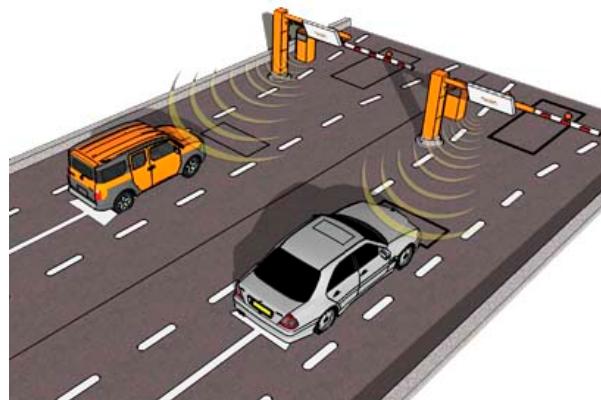


Figura 15: Pago de peaje automático

4.19. Lenguajes de descripción de hardware

Son lenguajes que permiten modelar o describir un circuito digital. [42].

4.20. Características de los lenguajes de descripción de hardware [42].

- Se usan como medio de intercambio entre diseñadores y fabricantes de circuitos
- Son medio de comunicación entre componentes de software CAD (Captura, Síntesis, Simulación)
- Soportan jerarquía en el diseño.
- Son independientes de la tecnología.
- Permiten distintas técnicas de descripción (Booleanas, FSM, Descripciones algorítmicas)
- Son estándares IEEE y ANSI lo que nos garantiza portabilidad.
- Permiten al menos tres niveles de abstracción para la descripción (Estructural, RTL o flujo de datos y comportamiento)
- No hay restricciones en el tamaño del código.s
- Facilidades para manejar diseños grandes.
- Permiten generar archivos con 2 contenidos diferentes (Modelo del circuito y banco de pruebas)
- Se pueden usar bibliotecas de vendedores (IP)

4.21. Lenguaje de descripción VHDL (VHSIC Hardware Description Language)

Fue creado a principios de los años 80 por el departamento de defensa de los Estados Unidos, con el fin de desarrollar circuitos integrados de manera rápida. El lenguaje soporta tres niveles: [42].

- Descripción por comportamiento: Permite la codificación de sentencias propias de lenguajes de programación comunes, tales como `if` , `for` , `case`. En esta parte del código es donde se encuentran los errores de simulación.
- Descripción a niveles lógicos de transferencia: Permite hacer descripciones mediante el conocimiento del funcionamiento del circuito.
- Descripción a nivel de componentes: Este nivel se basa enteramente en que se tenga un completo conocimiento del funcionamiento del circuito. El código que se escribe de esta forma es totalmente sintetizable.

4.22. Ventajas de VHDL [42].

El lenguaje VHDL presenta las siguientes ventajas:

- Se encuentra disponible de forma pública, ya que se trata de un estándar sin patente.
- Existe la suficiente documentación, soporte y estabilidad.
- Es posible realizar cualquier diseño, ya que permite la independencia de tecnología y del proceso de fabricación por lo que es posible trabajar tecnologías MOS, BICMOS entre otras sin hacer ninguna especificación.
- Se tiene una independencia de la tecnología de diseño, soportando diversas tecnologías como PLD, FPGA, CPLD o ASIC y circuitos secuenciales, combinacionales, sincronos o asincronos.
- Portabilidad entre proveedores, los circuitos son sintetizables para hardware de Altera como de Xilinx y para otros, siempre que no se haga uso especial de los recursos de los dispositivos.
- Permite la reutilización de código, ya que es un estándar que permite utilizar código que ya ha sido generado en diversos proyectos.
- Cuenta con la capacidad de realizar

4.22.1. Serial Peripheral Interface (SPI)

SPI es un hardware/firmware para protocolo de comunicaciones síncrona, que fue desarrollado por Motorola, que permite la comunicación entre diferentes microcontroladores o periféricos. En SPI es posible encontrar que las líneas de comunicación se encuentran separadas en dos partes: El pulso de reloj y la transmisión de datos.

SPI cuenta con un bus de cuatro líneas que son utilizadas por diversos microcontroladores para establecer control y comunicación con otros dispositivos; este bus fue originalmente diseñado para la transmisión e datos entre diferentes circuitos integrados de altas velocidades. Debido a esto, las líneas del bus no pueden ser demasiado largas, ya que la reactancia tiende a incrementarse y con esto el bus puede quedar inservible, aunque dicho bus es posible utilizarlo a bajas velocidades.

La transmisión se realiza por paquetes de 8 bits cada uno. Los dispositivos son definidos como maestros y esclavos. Un maestro es aquel que inicializa la transferencia de información sobre el bus y además genera las señales de reloj y control. Un esclavo es aquel que es controlado por un maestro.[43].

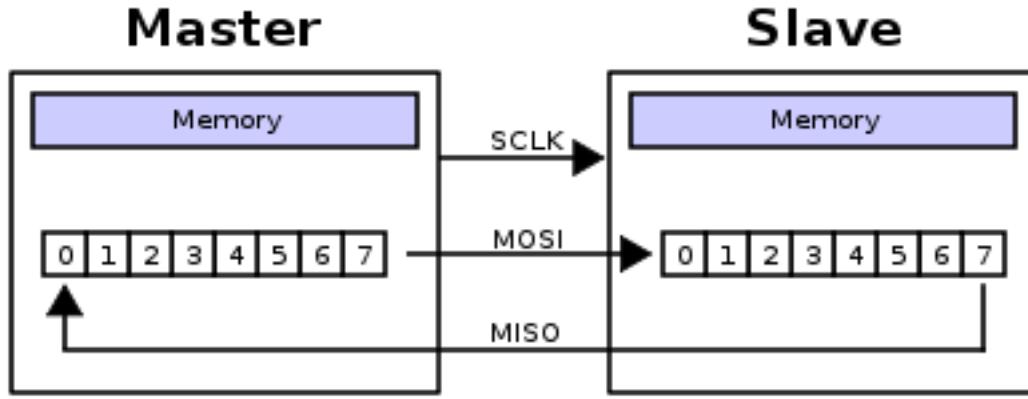


Figura 16: Esquema general de SPI

4.22.2. Descripción de las líneas de comunicación de SPI

SPI consta de cuatro líneas de conexión, cada una de estas con un cierto propósito y tiene una forma de interactuar dependiendo del rol que desempeñe (maestro o esclavo). A continuación describiremos cada una de estas líneas de comunicación.[43].

- **Master In Slave Output (MISO):** Es la linea de comunicación configurada como entrada al dispositivo maestro y salida del dispositivo esclavo. Es una de las dos líneas de transmisión de datos en serie en una sola dirección, enviando primeramente el bit más significativo y así sucesivamente con el resto de los bits. En el caso de que un dispositivo cumpla el rol la linea de MISO es conectada en alta impedancia.
- **Master Out Slave In (MOSI):** Se trata de una linea de comunicación que al igual que MISO, envía los datos bit por bit durante la transmisión, con una diferencia de que tiene configurada la salida al dispositivo maestro y la entrada al dispositivo esclavo.
- **Serial Clock (SCK):** La linea de reloj se utiliza para sincronizar la transferencia de datos en ambas direcciones por medio de las líneas de comunicación MISO y MOSI. Ambos dispositivos son capaces de transferir un byte de información en una sola secuencia de 8 ciclos de reloj.
- **Slave Select (SS):** Esta linea sirve para seleccionar al esclavo, permitiendo la transferencia de datos a un dispositivo esclavo en particular.

4.22.3. Características de SPI

- Operaciones sobre dispositivos maestros o esclavos.
- Bandera de interrupción que indica el fin de la transmisión.
- Bandera de protección contra colisiones en la escritura.
- Comunicación full dúplex.
- Reloj programable en polaridad y fase.
- Protección contra fallos en el modo. maestro-maestro.
- Facilidad de expansión con otros dispositivos que manejan comunicación SPI.

4.22.4. Funcionamiento de SPI

Primeramente el maestro es el que inicializa la comunicación en todo momento, además de configurar primero el reloj, utilizando siempre una frecuencia que sea menor o igual a la frecuencia máxima soportada por los dispositivos esclavos. El maestro se encarga de seleccionar un esclavo para poder establecer la comunicación, esto por medio de su linea de selección. Los esclavos que aún no han sido seleccionados por el maestro utilizaran la linea de selección, obtendrán en sincronía la señal del reloj y las señales MOSI del maestro, pero no tendrán activa la línea MISO. Esto quiere decir que el maestro únicamente selecciona un esclavo a la vez..

En la mayoría de los dispositivos se cuentan con salidas tri-estado en el que se involucrado el estado de alta impedancia cuando no ha sido seleccionado el dispositivo. Los dispositivos sin una salida de alta impedancia no pueden compartir bus SPI con otros dispositivos, esto porque la linea de comunicación Slave Select de un esclavo no puede quedar inactivo.

Como se mencionó anteriormente SPI soporta transmisión full dúplex, dicha transmisión es posible que ocurra durante cada ciclo de reloj. Esto quiere decir que cada ocasión que el maestro envía un bit en la linea MOSI, el esclavo seleccionado lee dicho y bit y de manera simultánea el esclavo envía un bit en la línea MISO y el maestro lee el bit de dicha linea.

La transferencia de información se lleva a cabo por el uso de registros de desplazamiento con un byte de tamaño para ambos. El maestro cambia los valores de registro a través de la línea MOSI, mientras el esclavo modifica los datos en su registro de intercambio.

Los datos son intercambiados empezando por el bit más significativo (MSB), continuendo con el siguiente bit y así sucesivamente hasta llegar al bit menos significativo (LSB). Una vez terminado este proceso el maestro y esclavo terminaron de transferir información.[43]

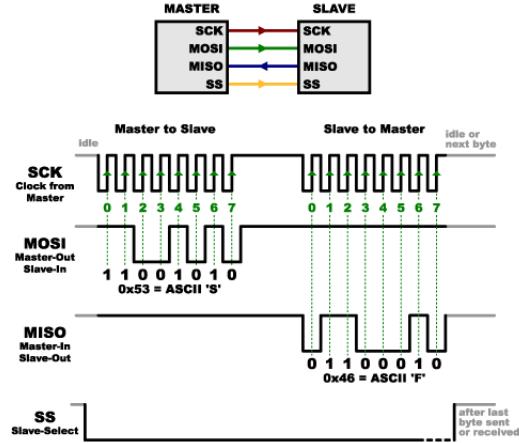


Figura 17: Funcionamiento de SPI

4.23. FPGA

Las FPGA (Field Prgrammable Gate Array) fue introducida por Xilinx en el año 1985, se trata de dispositivos que contienen una matriz bidimensional de bloques lógicos configurables los cuales pueden ser interconectados por segmentos de pista de diferente longitud. Dichos bloques lógicos contienen un flip-flop tipo D y una LUT (Lookup Table) que serán usados dependiendo de la naturaleza del circuito digital.[44]

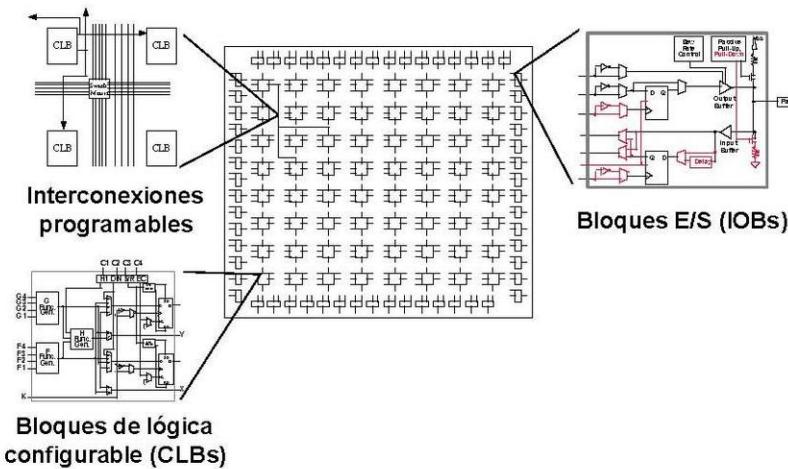


Figura 18: Esquema básico de un FPGA

4.24. Servicios web

La World Wide Web Consortium (W3C) define a los servicios web como un conjunto de aplicaciones o tecnologías para interoperar en la web. Con esto proporciona una comunicación entre el

cliente y el servidor por medio de mensajes XML. Dichos servicios facilitan los estándares de comunicación entre aplicaciones y proporcionan información al usuario final [21].

4.25. REST

REST es un acrónimo de Representational State Transfer, es un estilo de arquitectura para sistemas de hipermédia distribuidos. Hace uso de peticiones HTTP para la obtención de información en formato JSON o XML, tales como GET, POST, DELETE y PUT. Los objetos con los que se trabajan son manejados mediante identificadores de recursos uniformes, los cuales son únicos entre sí. Otra característica es que ni el cliente ni el servidor necesitan conocer el estado de las peticiones que se envían entre sí, por lo que podemos configurar la aplicación de tal forma que se envíen respuestas predeterminadas ante ciertas peticiones [22].

4.26. Python

Es un lenguaje de programación interpretado, ya que no hace uso de ningún compilador. Fue diseñado en 1991 por Guido Van Rossum. Python permite el desarrollo de servidores para la creación de aplicaciones web, al igual que puede usarse para conectarse a bases de datos para la manipulación de la información contenida en la misma. Adicionalmente, puede ser implementado en diferentes entornos (tales como Windows, Mac, Linux, Raspberry Pi, entre otros) [23].

4.27. Flask

De acuerdo con [24], es un microframework escrito en Python. Es denominado de esa manera debido a que no se hace uso de herramientas o librerías externas. Esto le permite al desarrollador tener libertad acerca de los elementos que van a hacer utilizados para el desarrollo de la aplicación.

Está basado en los proyectos Werkzeug y Jinja2, siendo utilizados para el desarrollo de interfaces que permiten generar páginas web de manera personalizada y dinámica.

4.28. MySQL

Es un sistema gestor de bases de datos relacionales de código abierto que trabaja con un modelo cliente servidor. Es utilizado para la creación y administración de bases de datos. Éste hace uso del lenguaje de consulta estructurado SQL para el manejo de las sentencias con las que el servidor va a interactuar junto con la base de datos. Permite la realización de operaciones específicas, tales como consulta, manipulación, identificación y control de acceso de datos [25].

5. Análisis

5.1. Comparativa de FPGAs

Para la implementación en hardware del proyecto se evaluaron diferentes FPGA de diferentes compañías, así como sus características técnicas como número de elementos lógicos totales, multiplicadores, I/O máximas, RAM, Celdas lógicas y número de flip flops.

En la siguiente tabla es posible apreciar a detalle las características de cada una de las FPGA que se consideraron.

FPGA/características	Elementos lógicos	Multiplicadores 18x18	I/O máximas	RAMI	Celdas lógicas	Flip flops
Intel Cyclone IV EP4CE10 [26]	10,000	23	179	414K	N/A	N/A
Xilinx SPARTAN-6 XC6SLX9 [27]	N/A	N/A	200	648K	3,840	4,800
Microchip IGLOO2 M2GL010 [28]	12,084	22	233	912K	N/A	N/A
Lattice MachXO3D-9400 [29]	N/A	N/A	335	505K	9,400	N/A
GOWIN GW1N-9 [30]	8,640	20	276	485K	N/A	6,480
QuickLogic QL1P1000 [31]	N/A	N/A	244	221K	7,680	7,680

Una vez comparadas la FPGA se prosiguió a buscar placas de desarrollo que tuvieran las características técnicas necesarias para el proyecto, así como un rango asequible de precios.

El resultado de la investigación arrojó los datos que se muestran a continuación:

5.2. Comparativa de placas de desarrollo

Placa de desarrollo/Características	FPGA	FLASH	SDRAM	SRAM	Interfaz VGA	Interfaz tarjeta SD	USB-UART	Precio (MXN)
New Start EP4C FFGA BOARD V1.0 [15]	Intel Cyclone IV EP4CE10	M2SP16 (16 Mbit)	W9825GOKH-6 (256 Mbit)	N/A	Si	Si	CH340G	1,115.71
ALINX AX301 [16]	Intel Cyclone IV EP4CE10	M2SP16 (16 Mbit)	H57V2562GTR (256 Mbit)	N/A	Si	Si	N/A	1,217.74
ALINX AX309 [17]	XILINX SPARTAN-6 XC6SLX9	M2SP16 (16 Mbit)	H57V2562GTR (256 Mbit)	N/A	Si	Si	CP2102	1,340.78
勇敢的芯 FPGA [18]	XILINX SPARTAN-6 XC6SLX9	M2SP40 (4 Mbit)	N/A	IS62LV256 (256 Kbit)	Si	Si	PL2303HXD	1,027.12
RZ-XSP6Z-FFGA [19]	XILINX SPARTAN-6 XC6SLX9	16 Mbit	256 Mbit	N/A	Si	N/A	CP2102	870.76
M2GL-EVAL-KIT [20]	Microchip IGLOO2 M2GL010	64 Mbit	512 Mbit (LPDDR)	N/A	Si	N/A	No especificado	7,843.06
LCMIX0LF-9400C-ASC-B-EVN [21]	Lattice MachXO3D-9400	16 Mbit	N/A	N/A	N/A	N/A	N/A	1,610.83
DK-START-GW1N9 [22]	GOWIN GW1N-9	64 Mbit	N/A	N/A	N/A	N/A	N/A	1,951.38

5.3. Comparación de los algoritmos en ISO/IEC 29192-7:2019

De acuerdo con [32], las implementaciones en hardware de los algoritmos ISO/IEC 29192-2:2019 producen los resultados de la tabla 5.3

Cifrado	Tamaño de llave	Tamaño de bloque	Latencia (ciclos/bloque)	Rendimiento a 100KHz	Área (GE)
CLEFIA [32]	128	128	176	76	2678
LEA	128	128	24	533.33	5426
PRESENT	128	64	32	200	1570

Cuadro 3: Comparación de implementaciones en hardware de los algoritmos de ISO/IEC 29192-2:2019

5.4. Proceso de selección de algoritmo criptográfico

5.4.1. Comparación de implementaciones en hardware de los algoritmos en ISO/IEC 29192-2:2019

Algoritmo	PRESENT	CLEFIA	LEA
Año de publicación	2007	2007	2013
Tamaños de llave (en bits)	80 y 128	128, 192 y 256	128, 192 y 256
Tamaño de bloque (en bits)	64	128	128
Número de rondas	31	18, 22 y 26	24, 28 y 32
Área (Número de compuertas lógicas equivalentes)	1570	2488	3826
Número de ciclos de reloj (por bloque)	32	328	168

5.4.2. Análisis/ataques que se han realizado a los algoritmos en ISO/IEC 29192-2:2019

De acuerdo con [32], los siguientes ataques/análisis se han realizado a los 3 cifrados. Éstos ataques se detallan en la tabla 4

Cifrado	Ataques
PRESENT	Ataques de canales laterales, ataque de relación de llaves en la décimo séptima ronda, Análisis de situaciones improbables, Ataques basados en grafos completos bipartitos en todas las rondas del algoritmo, Análisis por truncamiento de bits en ronda veintiséis
CLEFIA	Análisis de diferencial improbable en rondas reducidas
LEA	Análisis de consumo de energía

Cuadro 4: Análisis que se le ha hecho a cada cifrado en ISO/IEC 29192-2:2019

Debido a las limitaciones de recursos con la que cuentan los sistemas embebidos es de gran prioridad optimizar las funcionalidades a implementar en éste, para el caso particular del actual trabajo se decidió elegir un algoritmo criptográfico ligero en lugar de los algoritmos clásicos.

En la actualidad se encuentran estandarizados tres algoritmos ligeros, entre los cuales podemos encontrar (PRESENT, CLEFIA y LEA). Dichos algoritmos son nuestros tres principales candidatos a implementar de manera definitiva en el proyecto, para elegir alguno de los algoritmos se llevó a cabo una investigación acerca de las características que nos brindan, así como los diferentes ataques perpetrados a los algoritmos de cifrado ligero.

En los cuadros anteriores es posible apreciar las características de cada uno de nuestros algoritmos candidatos, para esto se decidió tomar en cuenta los tamaños posibles de llave que soporta cada algoritmo, esto porque un tamaño de llave considerable nos garantizará un nivel de seguridad apropiado para la naturaleza de información que se está manejando.

Uno de los parámetros también considerados fue el tamaño de bloque, ya que el tamaño de bloque entre más pequeño sea existe una probabilidad mayor de producir bloques repetidos lo que podría afectar a la integridad de los datos.

Además del tamaño de llave y bloque se consideran el número de rondas de cada algoritmo, esto debido a que el número de rondas y su complejidad de cada una de estas son importantes al momento de resistir los ataques conocidos.

A pesar de que los tres algoritmos son estandarizados por ISO/IEC 29192-2:2019, la diferencia de tiempo entre la fecha de publicación de los tres ha provocado que existan más investigaciones realizadas sobre CLEFIA y PRESENT que de LEA.

Aunque PRESENT muestre ser un algoritmo óptimo en todos los sentidos y existen muchas implementaciones de éste, según [20], una implementación de LEA optimizada en rendimiento resulta ser más rápida que ciertas versiones optimizadas de PRESENT. Un ejemplo de esto se puede ver en [1].

A pesar de que PRESENT tiene un rendimiento óptimo descuida algunas de las características importantes de los algoritmos criptográficos, como lo es el tamaño de llave y de bloque.

LEA presenta un número de rondas óptimas para proveer resistencia en contra de todos los ataques conocidos a cifradores por bloque y proveer un nivel suficiente de seguridad.

LEA provee un rápido cifrado en múltiples plataformas de Intel, AMD, ARM y Cold e incluso a un nivel de implementación de C.

CLEFIA y LEA presentan tamaños de llave idénticos. Dichos algoritmos soportan tamaños de llave mayores que los soportados por PRESENT. Por último, los números de rondas que soporta LEA son mayores lo que garantiza una mayor resistencia a los ataques conocidos.

El rendimiento que puede alcanzar y el tamaño de llaves que soporta LEA nos ha llevado a determinar que LEA es el cifrador ligero más adecuado para el actual proyecto.

Por último cabe mencionar que el número de implementaciones de CLEFIA es mayor al de LEA, lo que nos lleva a implementar LEA de manera final.

5.5. Etiquetas RFID

El módulo de lectura/escritura RFID que se utilizará es un MFRC522 el cual soporta ISO/IEC 14443 A/MIFARE y NTAG. Por razones de conveniencia y disponibilidad, se decidió centrarse en etiquetas que soporten ISO/IEC 14443 A/MIFARE [33].

5.5.1. Proceso de selección de etiqueta RFID

En el cuadro 5 comparamos 3 candidatos de etiquetas RFID a utilizar.

Etiqueta/Características	Almacenamiento	Precio (USD)
RRHFSM11 [34]	4Kb	39.65
UHF Monza 4 [35]	1Kb	11.28
FUDAN F1108 [36]	1Kb	2.86

Cuadro 5: Comparación de etiquetas RFID

5.6. Interfaces de comunicación

Las interfaces de comunicación permiten a los dispositivos el intercambio de información con otros dispositivos, como puede ser un microcontrolador, computadora, fpga, etc.

A continuación se presente un cuadro con las diferentes características de algunas de las interfaces de comunicación.

Interfaz	Tipo de serial	Señales	Modo	Velocidad de transmisión máximo	Topología
UART	Asincrono	Rx,Tx	Full-duplex	1 Mb/s	Point-to-Point
SPI	Síncrono	MOSI,MISO,SCK,SS	Full-duplex	60 Mb/s	Bus Master-Slave
I2C	Síncrono	SDA,SCL	Half-duplex	400 Kb/s	Bus Master-Slave
USB	Síncrono	D+,D-(diferencial)*	Half-duplex	480 Mb/s	Point-to-Point Host-Device

Cuadro 6: Características de diferentes interfaces de comunicación

En el presente trabajo terminal se hace uso de SPI, esto por la compatibilidad con el módulo RFID que se emplea.

6. Diseño del sistema

Las subsecciones siguientes describen los diagramas de cada componente del prototipo de sistema, además de que se muestra la maquetación de las diferentes interfaces de usuario.

6.1. Metodología de diseño

Se propone trabajar con la metodología en V, debido a que permite especificar sistemas embebidos de manera clara. En la figura de a continuación se muestran sus etapas de desarrollo.

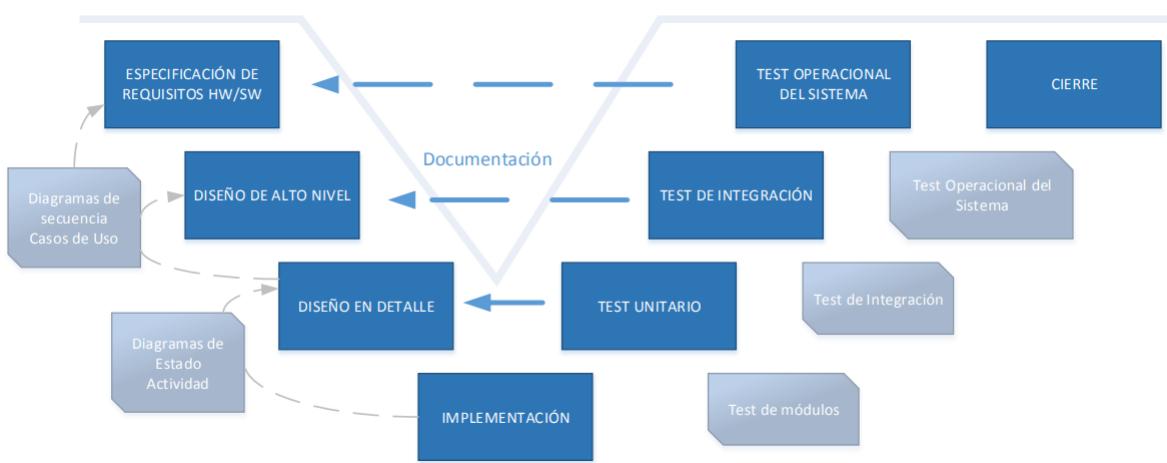


Figura 19: Modelo en V para el diseño de sistemas embebidos

El modelo en V para el diseño de sistemas embebidos propone las siguientes etapas de desarrollo:

- **Especificación de requerimientos de software y hardware:** Se definen y documentan los diferentes requerimientos del sistema, identificando los valores numéricos más concretos posible. Entre ellos debe estar la especificación del nivel de integridad o SIL, en caso de que sea requerido.
- **Diseño de alto nivel:** Su objetivo es obtener un diseño y visión general del sistema.
- **Diseño en detalle:** Consiste en detallar cada uno de los bloques de la fase anterior.
- **Implementación:** En ésta fase se materializa el diseño en detalle.
- **Pruebas unitarias:** En ésta fase se verifican cada uno de los módulos de software y hardware de manera individualizada, comprobando su correcto funcionamiento.
- **Integración:** En éste fase se unen los distintos módulos que conforman al sistema. Como en el caso anterior, ha de generarse un documento de pruebas. Por una parte debe de comprobarse el funcionamiento correcto, y por otra, en caso de que se trate de un sistema tolerante a fallos,

debe verificarse que ante la presencia de un fallo persiste el correcto funcionamiento. Se comprueba el cumplimiento de todos los requerimientos.

- **Test operacional del sistema:** Se realizan las últimas pruebas pero sobre un escenario real, en su ubicación final, anotando una vez más las pruebas realizadas y los resultados obtenidos.

6.2. Requerimientos funcionales

6.2.1. Requerimientos de Software

Requerimiento	Nombre	Descripción
RFS1	Perfiles de usuario	Existen dos perfiles, administrador y médico
RFS2	Inicio de sesión	Los perfiles de usuario deben de acceder al sistema mediante un nombre de usuario y una contraseña
RFS3	Alta médico	Los administradores serán capaces de dar de alta a los médicos usuarios del sistema
RFS4	Modificar médico	Los administradores serán capaces de modificar a los médicos usuarios del sistema
RFS5	Eliminar médico	Los administradores serán capaces de eliminar a los médicos usuarios del sistema
RFS6	Alta paciente	Los médicos serán capaces de dar de alta pacientes, los cuales contarán con su información cifrada en su etiqueta RFID
RFS7	Modificar pacientes	Los médicos serán capaces de leer etiquetas RFID para desplegar la información de los pacientes para su posterior modificación
RFS8	Eliminar paciente	Los médicos serán capaces de leer etiquetas RFID para desplegar la información de los pacientes para su posterior eliminación
RFS9	Base de datos	El sistema contará con un servidor de base de datos para el almacenamiento de la información relacionada con el mismo.

6.2.2. Requerimientos de Hardware

Requerimiento	Nombre	Descripción
RFH1	Cifrado y descifrado	El proceso de cifrado y descifrado de información se hará mediante el algoritmo LEA
RFH2	Protocolo	Se utilizará el estándar SPI para la comunicación con el RC522
RFH3	Almacenamiento	La información cifrada será almacenada dentro de la etiqueta RFID, para que la información sea descifrada por el FPGA
RFH4	Almacenamiento	La información cifrada será almacenada dentro de la etiqueta RFID, para que la información sea descifrada por el FPGA

6.3. Requerimientos no funcionales

6.3.1. Requerimientos de Software

Requerimiento	Nombre	Descripción
RNFS1	Restricción de usuarios	Los administradores no podrán realizar acciones de médicos, y visceversa
RNFS2	Etiqueta RFID	El uso de etiqueta RFID se limitará a uno por paciente
RNFS3	Cifrado y descifrado	El sistema deberá de trabajar únicamente con la información descifrada una vez que haya sido cifrada anteriormente

6.3.2. Requerimientos de Hardware

Identificador	Nombre	Descripción
RNFH1	Bajo consumo del dispositivo programable	El consumo de potencia del FPGA no debe sobrepasar los 5v.
RNFH2	Espacio en memoria TAG	El tag utilizado para guardar la información de los pacientes deberá de contar con el almacenamiento suficiente para poder realizar el almacenamiento de la misma
RNFH3	Comunicación entre dispositivos	Se deberá de establecer un estándar de comunicación entre los elementos del prototipo de sistema
RNFH4	Periféricos	La FPGA utilizada deberá de contar con los periféricos suficientes para poder comunicarse con los otros elementos del prototipo de sistema
RNFH5	Elementos lógicos	La FPGA utilizada deberá de contar con el número de elementos lógicos necesarios para soportar la arquitectura a implementar.

6.4. Reglas de negocio

Identificador	Descripción
RN1	Los registros de los pacientes sólo podrán ser llenados por los médicos.
RN2	El médico responsable será el único capaz de manipular la información de sus pacientes
RN3	Para la modificación o eliminación de un registro es necesario presentar la respectiva etiqueta RFID
RN4	La información de las etiquetas RFID será estrictamente cifrada.
RN5	Sólo los médicos responsables podrán descifrar la información de las etiquetas
RN6	En caso de que se bloquee alguna cuenta, se deberá acudir con un administrador

6.5. Diseño de alto nivel

En la figura siguiente se presenta tanto la conexión lógica como física, entre los componentes que intervienen.

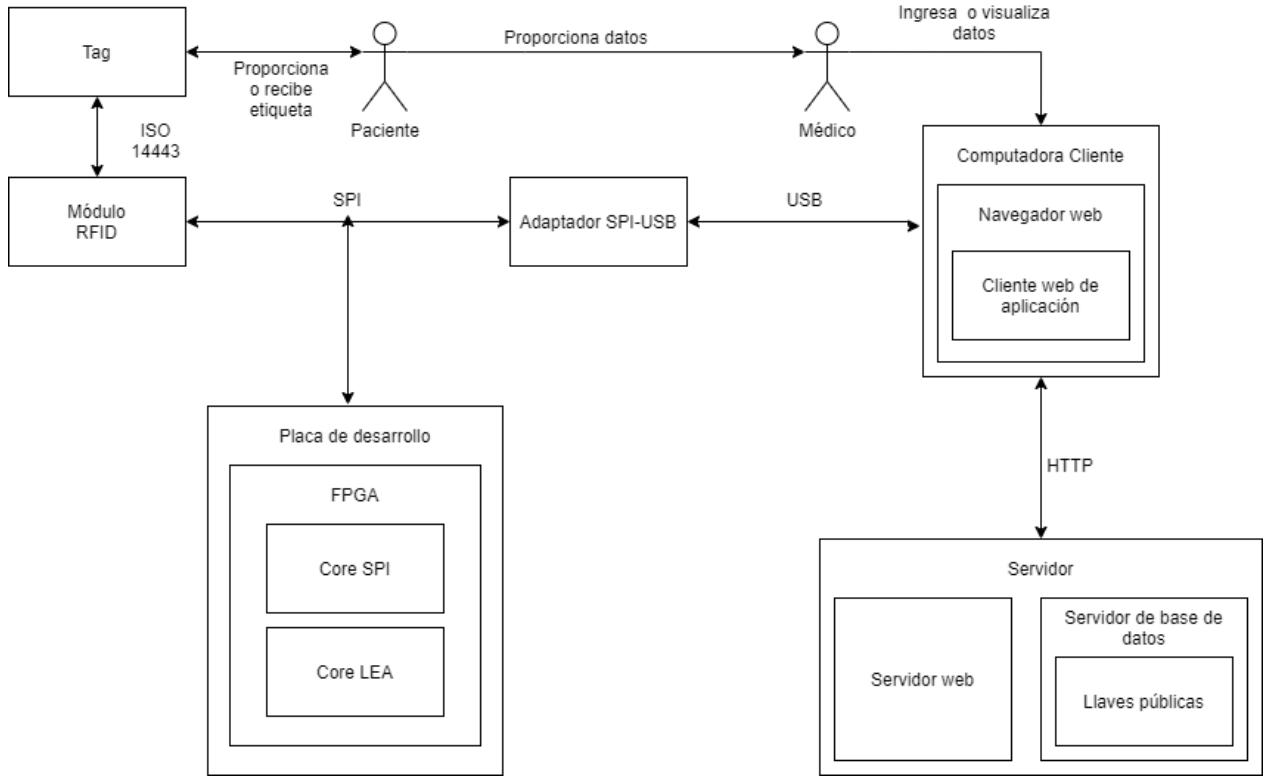


Figura 20: Arquitectura de sistema

El sistema se encuentra compuesto por un módulo rfid, un tag MFRC522, por la placa de desarrollo ALINX AX301, un adaptador SPI-USB y finalmente tenemos un software de prueba que capta o muestra los datos de los pacientes.

Tarjeta de desarrollo: Es el elemento principal del trabajo terminal, ya que es el encargado de realizar el cifrado o descifrado de la información del paciente, además de dirigir el flujo de datos hacia el lugar adecuado.

Software de prueba: Es el encargado de mostrar o recibir la información del paciente.

Adaptador SPI-USB: Se utiliza para lograr la comunicación entre el CORE de SPI y el software de prueba.

Tag: Es el encargado de almacenar los datos cifrados del paciente.

6.6. Diseño en detalle

6.6.1. Proceso de generación de llaves

El proceso de generación de llaves se lleva a cabo por medio del siguiente módulo, donde encontramos un total de dos memorias ROM las cuales almacenan las constantes que son proporcionadas por el mismo algoritmo y que son necesarias para el cumplimiento de las tareas de éste módulo.

Además de diferentes circuitos combinacionales (ROL, adición modular y sumador) que son necesarios para llevar a cabo las operaciones que dan origen a las rondas de llave.

También se tienen dos memorias que sirven para almacenar temporalmente los valores de T. Por una parte la Memoria T almacena los 4 valores necesarios para conformar la ronda de llave i-ésima, y por otra parte el buffer T sirve para almacenar el valor de T que es necesario para llevar a cabo la siguiente iteración.

Se tiene un archivo de registros RK y una unidad de control. El archivo de registros RK es el encargado de almacenar las 24 llaves de ronda generadas durante éste proceso. Esto para poder usarlas en el módulo de cifrado y descifrado.

Finalmente se tiene una unidad de control que coordina los procesos necesarios para poder llevar a cabo la generación de llaves de manera adecuada.

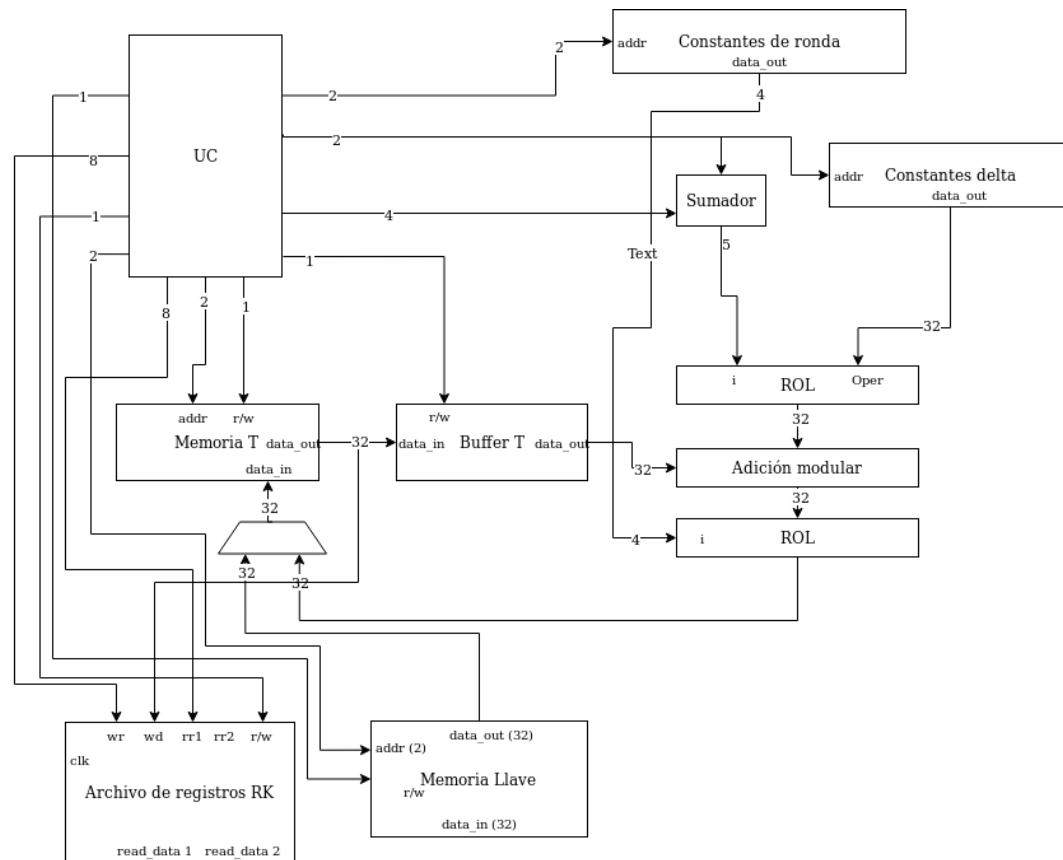
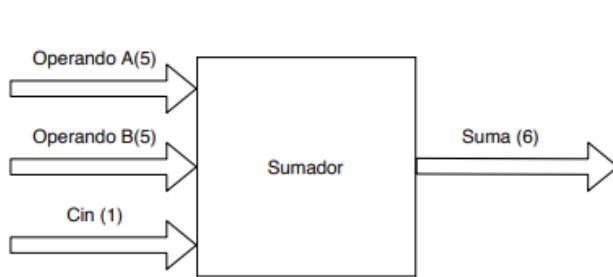


Figura 21: Módulo de Key Schedule

A continuación se detallan cada uno de los módulos que son propios del Key Schedule, en un apartado más adelante se especifican los módulos comunes.



MÓDULO SUMADOR

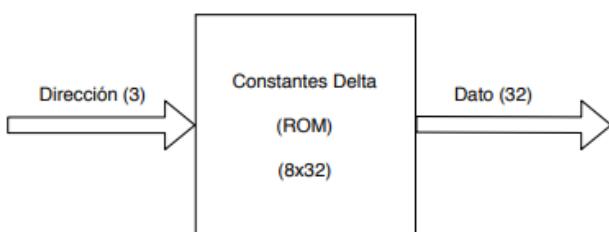
Sumador encargado de obtener el índice de la operación ROL.

Operando A: será un número que variará del 0 al 23

Operando B: será un número que viene de la memoria constantes de ronda (0,1,2 y 3) son los posibles valores

Cin: permanecerá siempre en 0

Suma: es el resultado de la operación y viene con el acarreo de salida anexado.



MÓDULO CONSTANTES DELTA

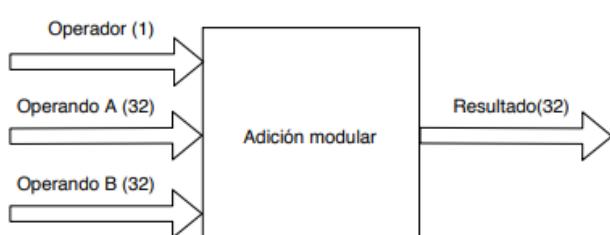
El módulo constantes delta se encarga de proporcionar las constantes definidas en el algoritmo LEA .

Dirección: La dirección de la memoria proviene del resultado de la operación módulo entre el contador y 4, los números van a ir variando del 0 al 3

Dato: Es la constante proporcionada por el algoritmo, éste dato puede variar entre los siguientes valores:

delta = [0xe3efe9db, 0x44626b02, 0x79e27e8a, 0x78df30ec, 0x715ea49e, 0xcx785da0a, 0xe04ef22a, 0xe5c40957]

Cada uno de los números anteriores tiene un tamaño máximo de 32 bits.



MÓDULO ADICIÓN MODULAR

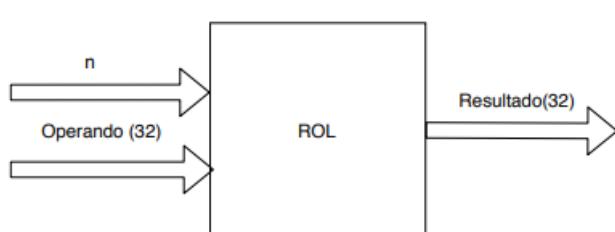
El módulo adición modular es el encargado de realizar la operación $(A+B) * 2^{32}$ o $(A-B) * 2^{32}$, para esto se contempla tener 2 buses de entrada que serán los dos números a operar y una señal de modo que nos obtendrá la suma o resta modular.

Operador: Señal que determina la operación a realizar (0: adición, 1:sustracción)

Operando A: Primer operador que proviene del módulo ROL.

Operando B: Segundo operador que proviene de la memoria que almacena las llaves de ronda.

Resultado: Es un bus de 32 bits que muestra la adición o sustracción modular, éste resultado será entrada de un módulo ROL.



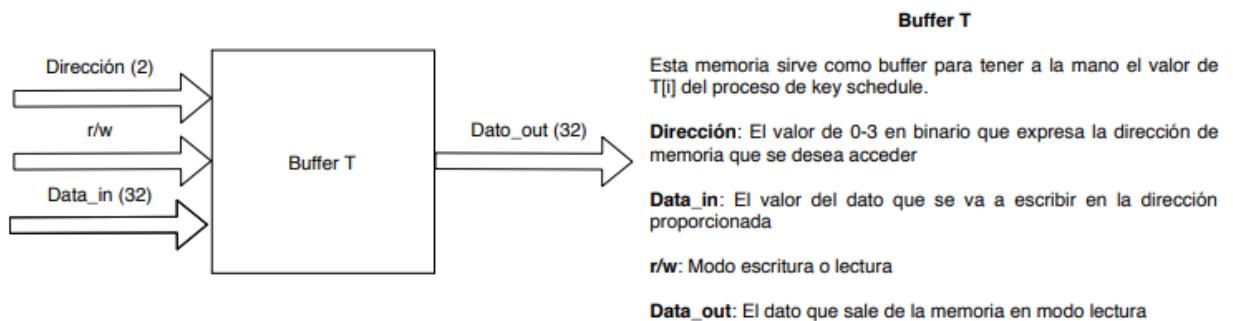
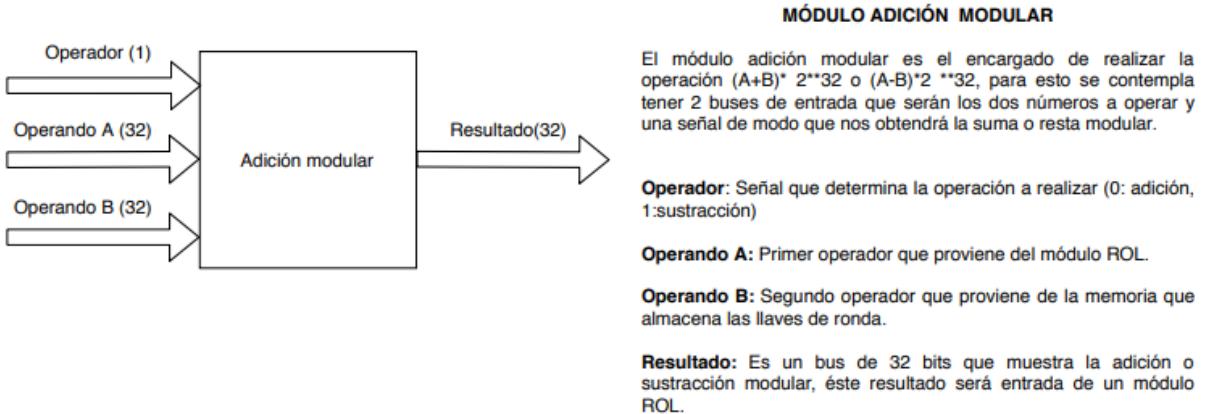
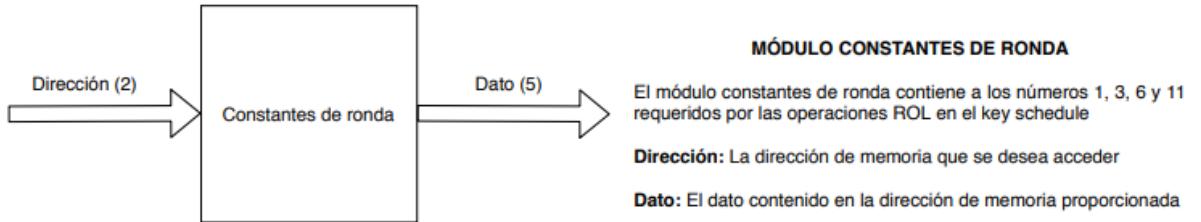
MÓDULO ROL (rotación izquierda)

El módulo ROL rota a la izquierda i-bits del operando que se le proporcione

n: Número de bits a rotar.

Operando: Palabra a rotar

Resultado: Resultado de rotar n bits.



6.6.2. Proceso de cifrado

El proceso de cifrado se lleva a cabo por el siguiente módulo, donde se observan múltiples circuitos combinacionales que hacen posible esta tarea, como dos módulos XOR, uno de adición modular, un rol, un ror, los cuales son operaciones definidas en el algoritmo.

Además se hacen uso de dos multiplexores, el primero para seleccionar entre la operación ROR o ROL, ya que en ciertos cálculos se requieren en algunos casos ROR y en otro ROL.

También en éste módulo se hace uso del archivo de registros RK que previamente fue llenado con las llaves de ronda provenientes del key schedule.

Un segundo archivo de registros X, se encarga de almacenar y leer los valores intermedios de 128 bits del proceso de cifrado.

Finalmente tenemos la Memoria C y una unidad de control, la primera es donde se almacena el texto cifrado y el segundo módulo se encarga de coordinar a los elementos digitales.

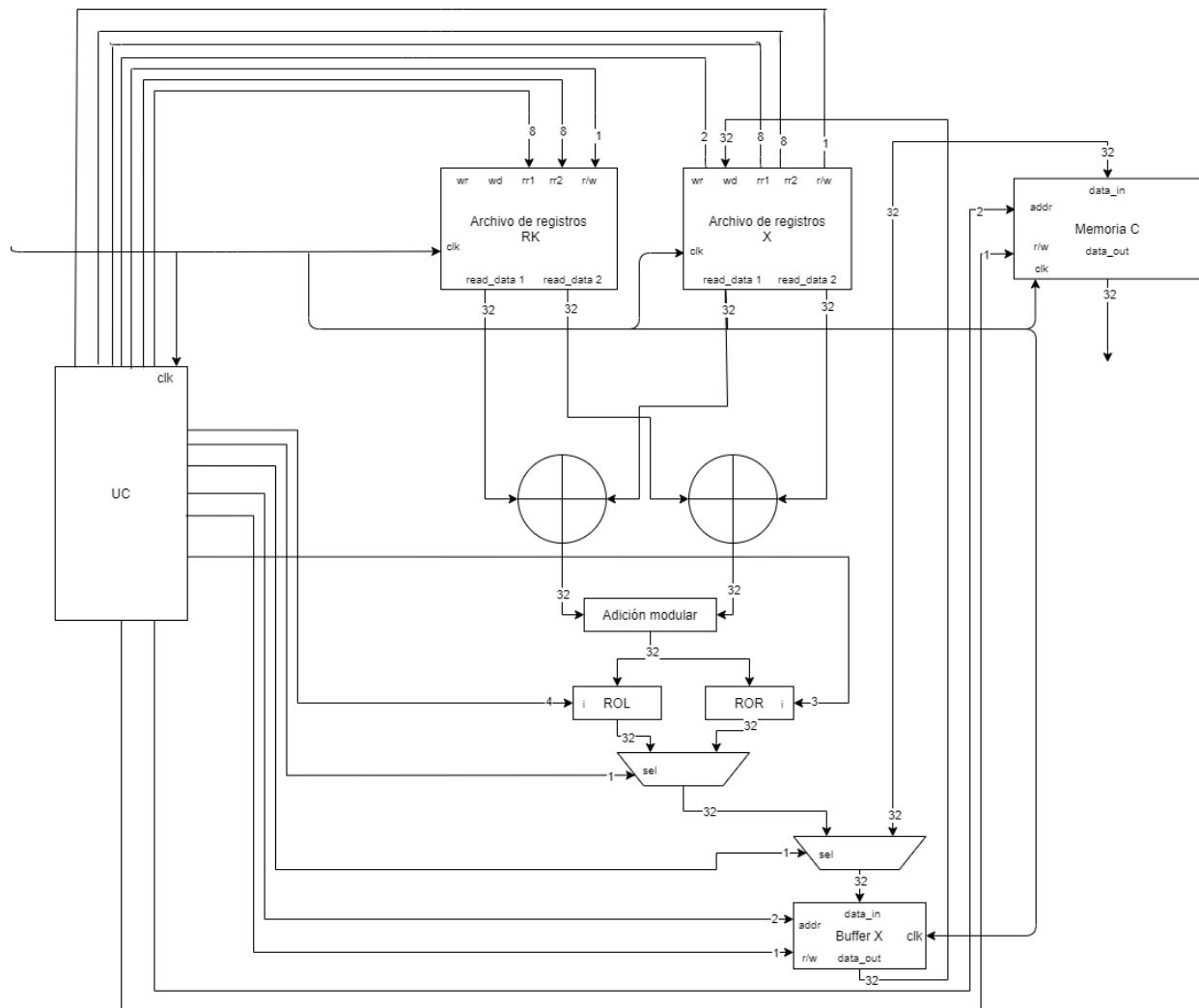


Figura 22: Módulo de cifrado

En la siguiente sección se detallan los módulos usados en el cifrado, en un apartado más adelante se especifican los módulos comunes.

6.6.3. Proceso de descifrado

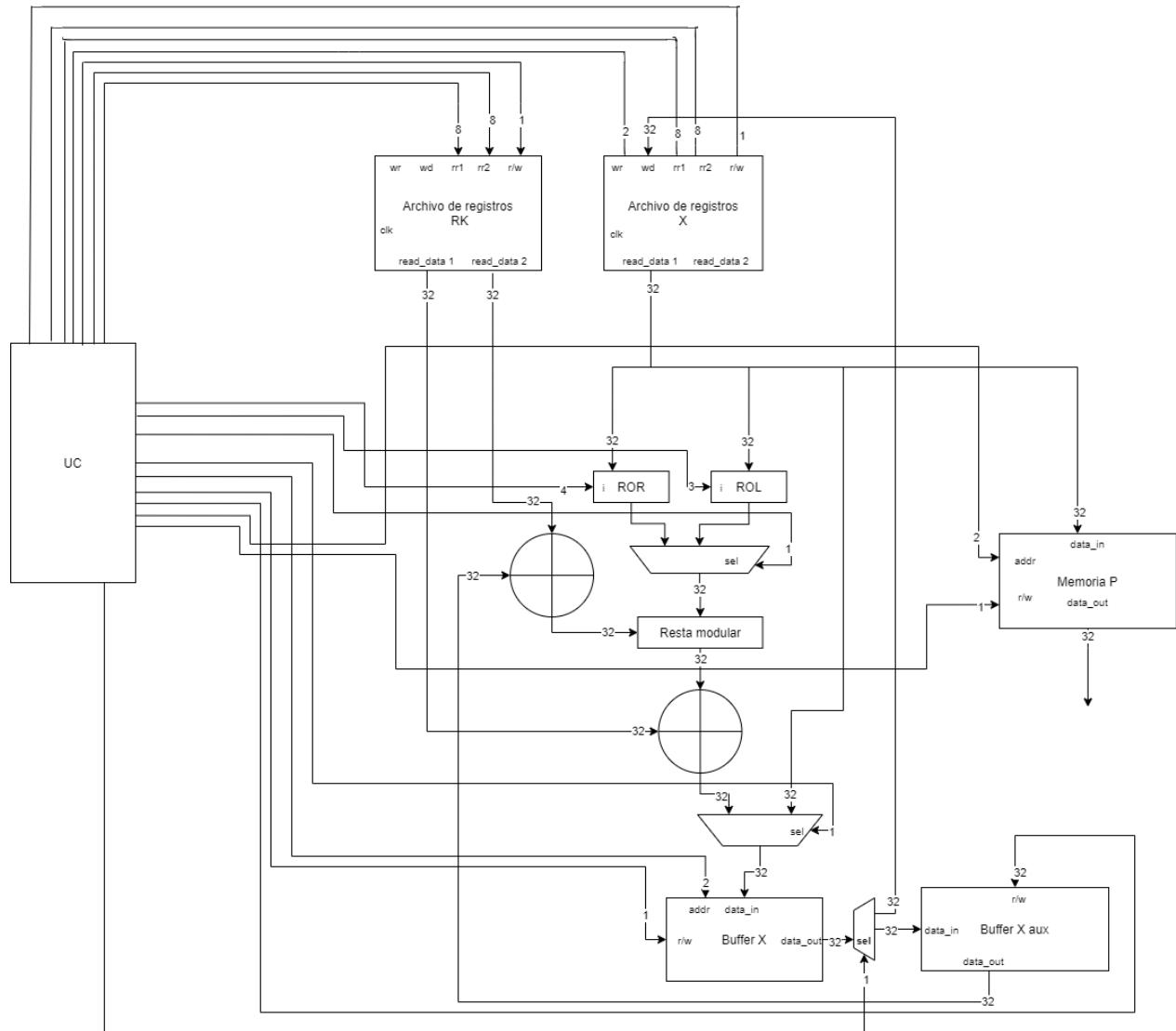
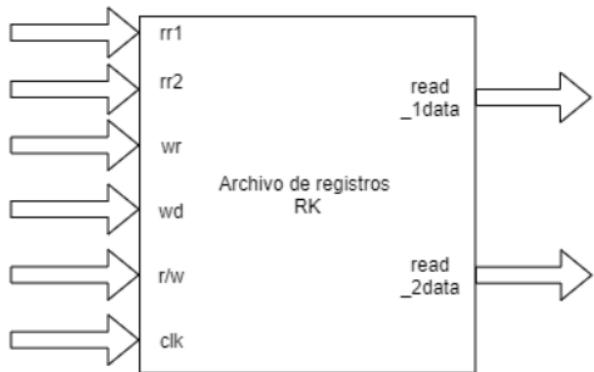


Figura 23: Módulo de descifrado



XOR

Módulo encargado de llevar a cabo la compuerta xor entre dos señales de 32 bits.



Archivo de registros RK

Este módulo es el encargado de almacenar las llaves de ronda generadas por el key schedule, contando con un máximo de 24 rondas de llave, tal como lo indica la especificación del algoritmo.

rr1: La dirección del registro 1 a leer. Los datos de éste salen por **read_1data**. 8 bits para direcciones 0 a 255.

rr2: La dirección del registro 1 a leer. Los datos de éste salen por **read_1data**. 8 bits para direcciones 0 a 255.

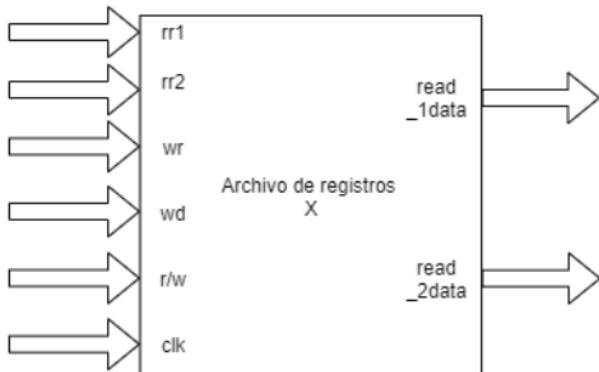
wr: La dirección del registro al que se va a escribir. 8 bits para direcciones 0 a 255.

wd: Los datos a escribir en el registro señalado por wr. 32 bits de palabra.

r/w: 1 bit para señalar si es lectura o escritura.

read_1data: El contenido del registro señalado por rr1. 32 bits de salida.

read_2data: El contenido del registro señalado por rr1. 32 bits de salida.



Archivo de registros X

Este módulo es el encargado de almacenar el estado en el que se encuentran los datos que se están cifrando o descifrando.

rr1: La dirección del registro 1 a leer. Los datos de éste salen por **read_1data**. 8 bits para direcciones 0 a 255.

rr2: La dirección del registro 1 a leer. Los datos de éste salen por **read_1data**. 8 bits para direcciones 0 a 255.

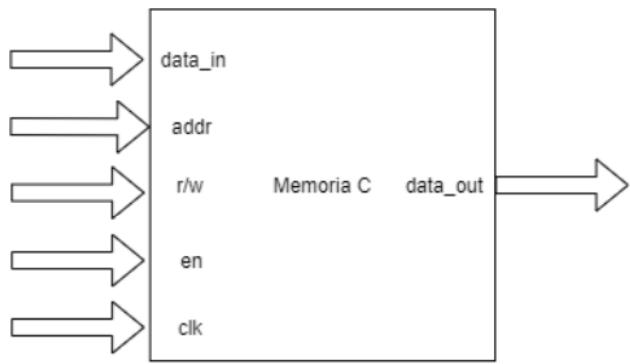
wr: La dirección del registro al que se va a escribir. 8 bits para direcciones 0 a 255.

wd: Los datos a escribir en el registro señalado por wr. 32 bits de palabra.

r/w: 1 bit para señalar si es lectura o escritura.

read_1data: El contenido del registro señalado por rr1. 32 bits de salida.

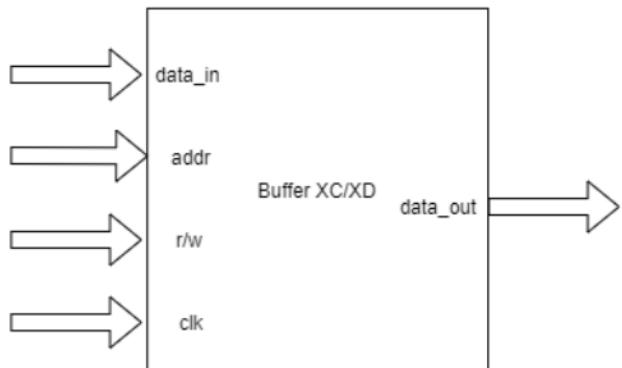
read_2data: El contenido del registro señalado por rr1. 32 bits de salida.



Memoria C

La Memoria C, nos va a servir a almacenar los bloques de texto cifrado que se vayan generando

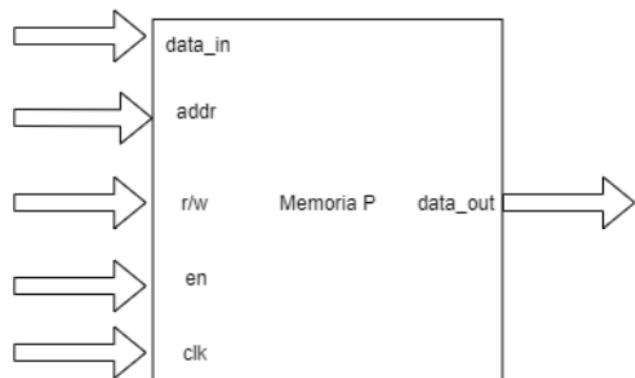
data_in: Dato de entrada que puede ser escrito en la dirección de indique la señal addr.
addr: Dirección de lectura o escritura de la memoria C.
r/w: Señal que indica si se va a leer o escribir en la memoria.
data_out: Dato resultante de la lectura en la dirección señalada por addr
en: señal que indica si la memoria se encuentra activada o no.



Buffer XC/XD

Este módulo nos va a auxiliar en almacenar temporalmente el estado del texto cifrado o descifrado.

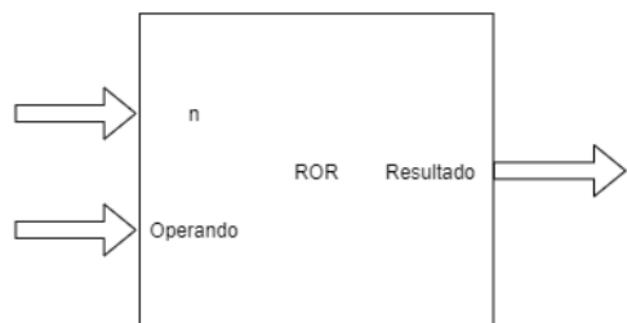
data_in: Dato de entrada a ser escrito en el buffer.
addr: Dirección donde va a ser escrito el dato de entrada.
r/w: Señal que selecciona entre escritura o lectura.
data_out: Dato resultante de la lectura en la dirección señalada por addr



Memoria P

La Memoria P sirve para almacenar texto plano

data_in: Dato de entrada que puede ser escrito en la dirección de indique la señal addr.
addr: Dirección de lectura o escritura de la memoria C.
r/w: Señal que indica si se va a leer o escribir en la memoria.
data_out: Dato resultante de la lectura en la dirección de la señal addr
en: señal que indica si la memoria se encuentra activada o no.

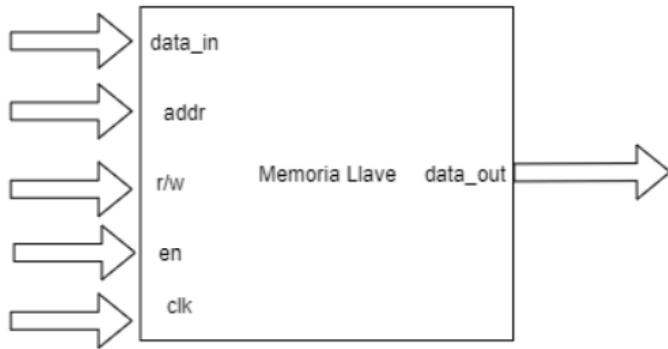


ROR

Este módulo se encarga de rotar n bits a la derecha.

Operando: Palabra a rotar
n: Número de bits a rotar.
Resultado: Resultado de rotar n bits.

Memoria Llave



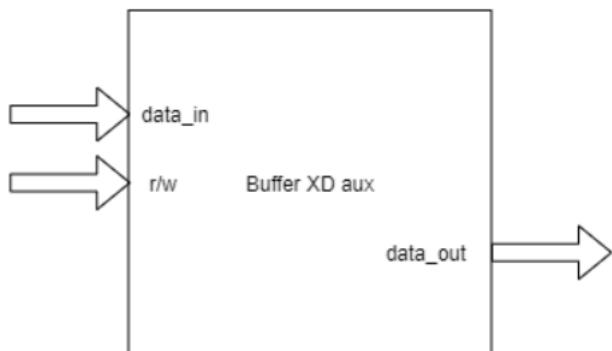
La Memoria Llave sirve para almacenar una llave.

data_in: Dato de entrada que puede ser escrito en la dirección de indique la señal **addr**.
addr: Dirección de lectura o escritura de la memoria C.

r/w: Señal que indica si se va a leer o escribir en la memoria.

data_out: Dato resultante de la lectura en la dirección de la señal **addr**

en: señal que indica si la memoria se encuentra activada o no.



Buffer XD aux

Este módulo sirve para almacenar temporalmente cambios hechos al valor en el buffer XD. Uso exclusivo en descifrado.

data_in: Dato de entrada a ser escrito en el buffer.

r/w: Señal que selecciona entre escritura o lectura.

data_out: Dato resultante de la lectura en la dirección señalada por **addr**

6.6.4. Unidad de control

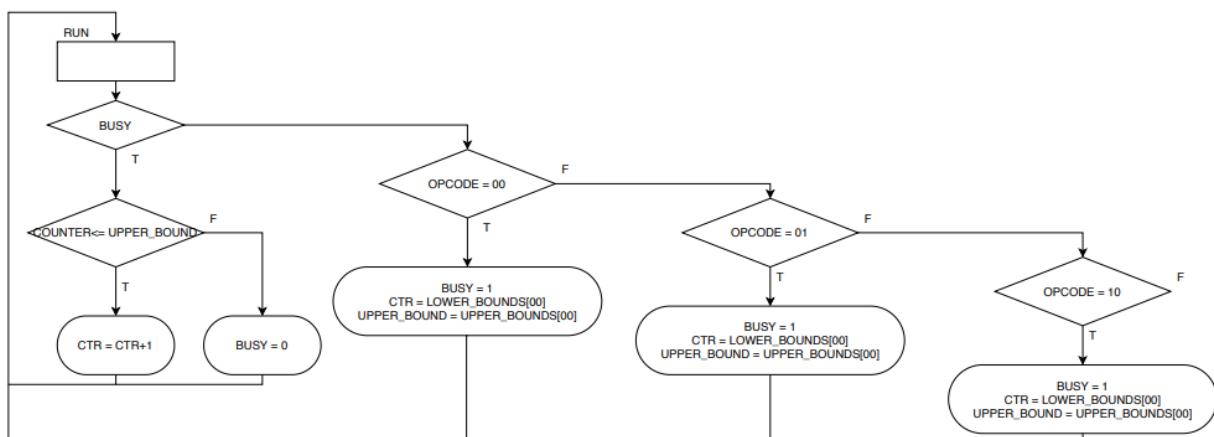


Figura 24: Carta ASM propuesta para la unidad de control

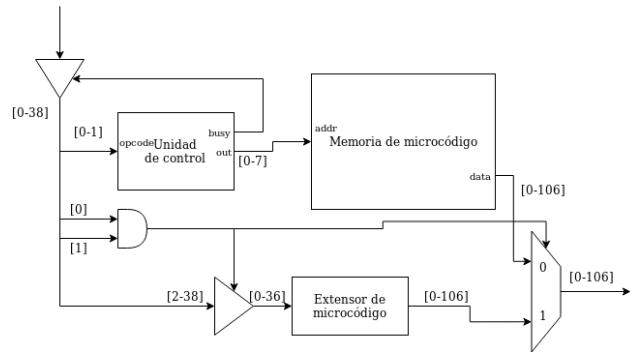


Figura 25: Diagrama a bloques de unidad de control

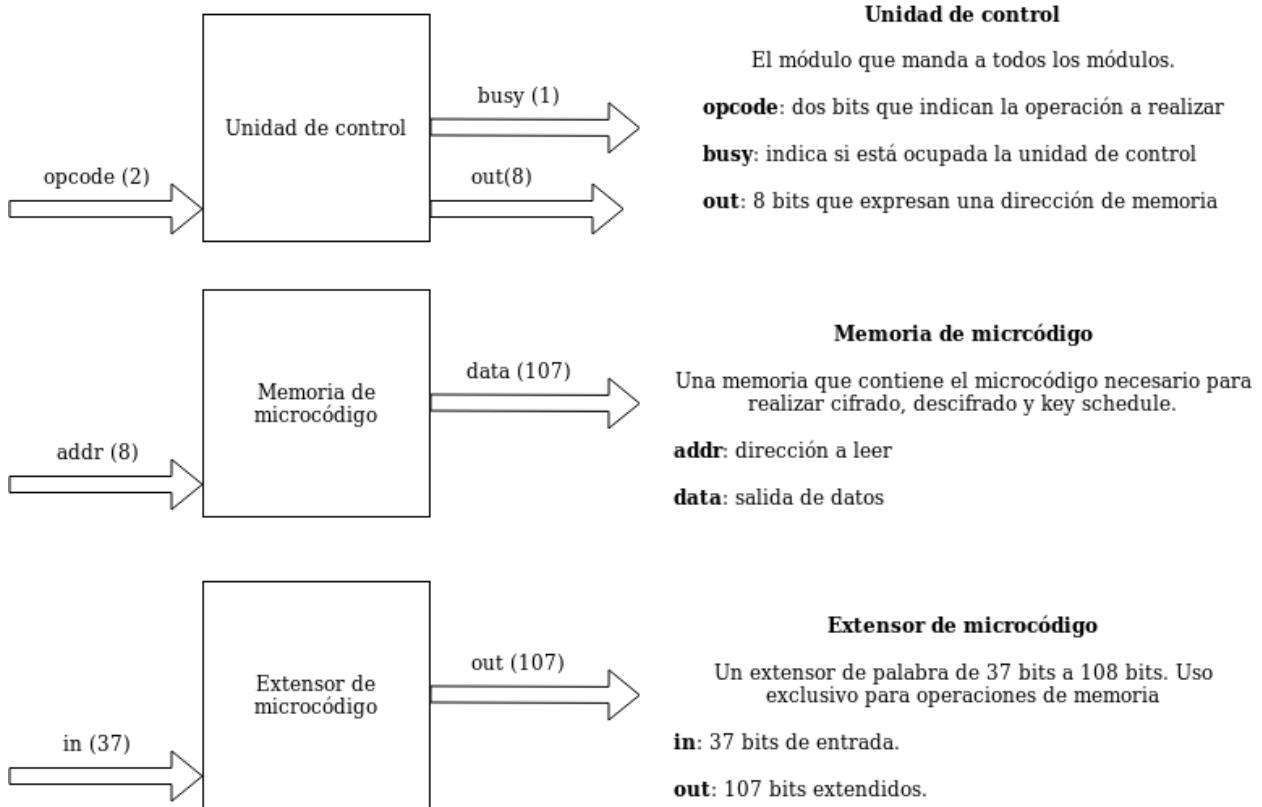


Figura 26: Explicación de los bloques de unidad de control

```

# todos los counters tienen un lower_bound preestablecido

while 1:
    if busy == 0:
        if opcode == 00: #cifrado
            busy = 1
            counter = cifrado_lower_bound # LOWER_BOUNDS[00]
            upper_bound = cifrado_upper_bound # UPPER_BOUNDS[00]
        if opcode == 01: # descifrado
            busy = 1
            counter = descifrado_lower_bound # LOWER_BOUNDS[01]
            upper_bound = descifrado_upper_bound # UPPER_BOUNDS[01]
        if opcode == 10: #key schedule
            busy = 1
            counter = keyschedule_lower_bound # LOWER_BOUNDS[10]
            upper_bound = keyschedule_upper_bound # UPPER_BOUNDS[10]
    else:
        pass
    if busy == 1:
        if counter <= upper_bound
            counter++
        else:
            busy = 0

```

Figura 27: Pseudocódigo

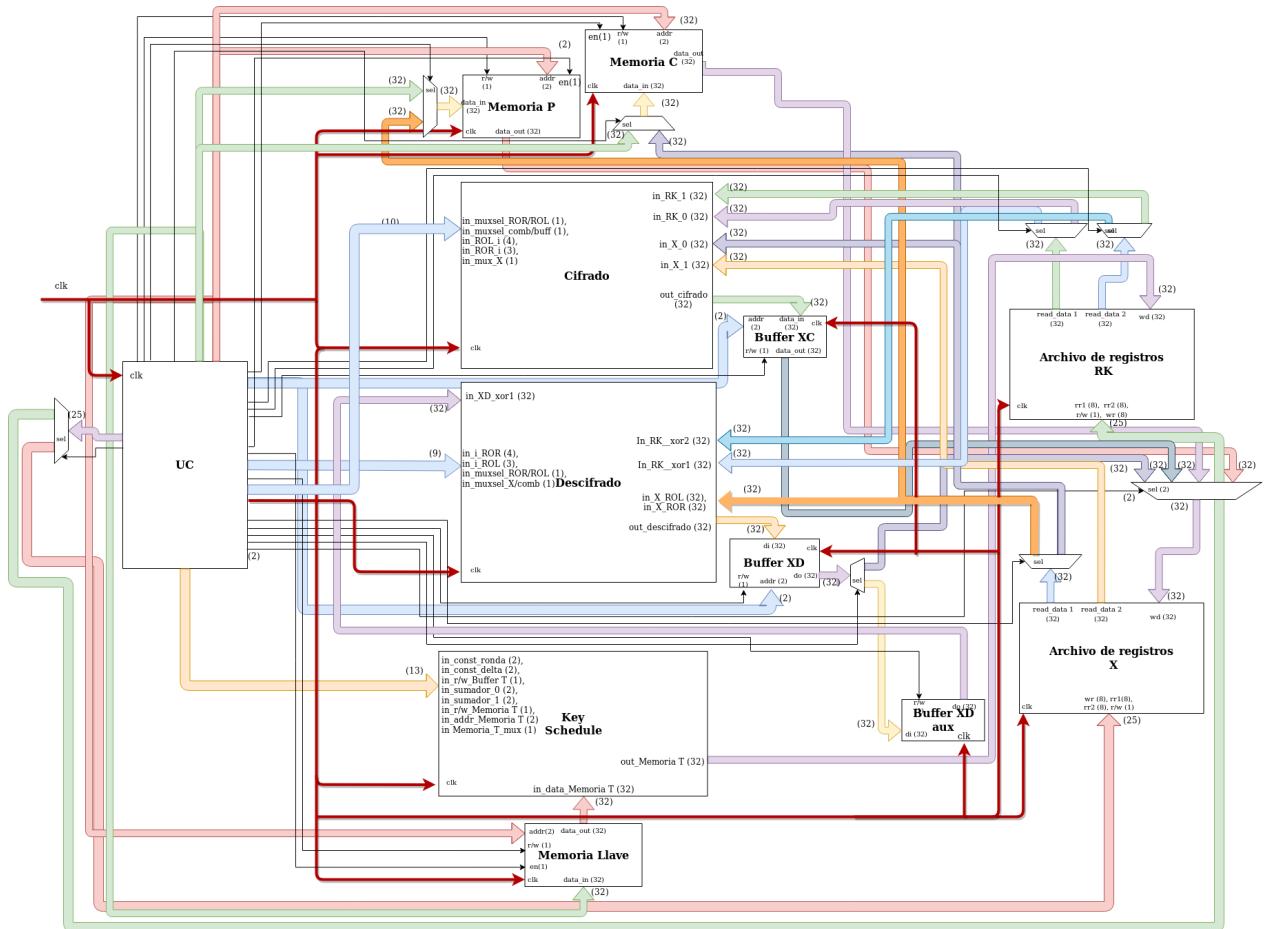


Figura 28: Arquitectura de LEA

6.7. Diseño de puente SPI

El puente que va entre el bus SPI y la arquitectura de LEA.

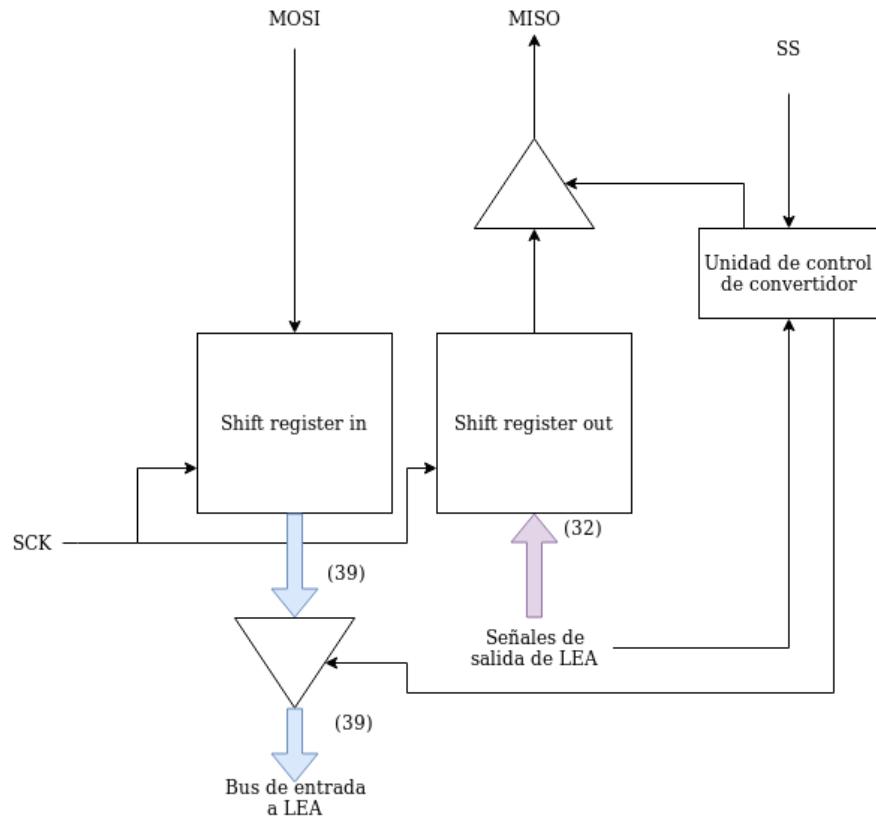


Figura 29: Arquitectura de puente SPI

6.8. Diseño de arquitectura Web

Para la aplicación web se propone utilizar una arquitectura estilo cliente-servidor, esto con el fin de mantener un control acerca de las peticiones que los clientes realicen, para su posterior almacenamiento en la base de datos correspondiente.

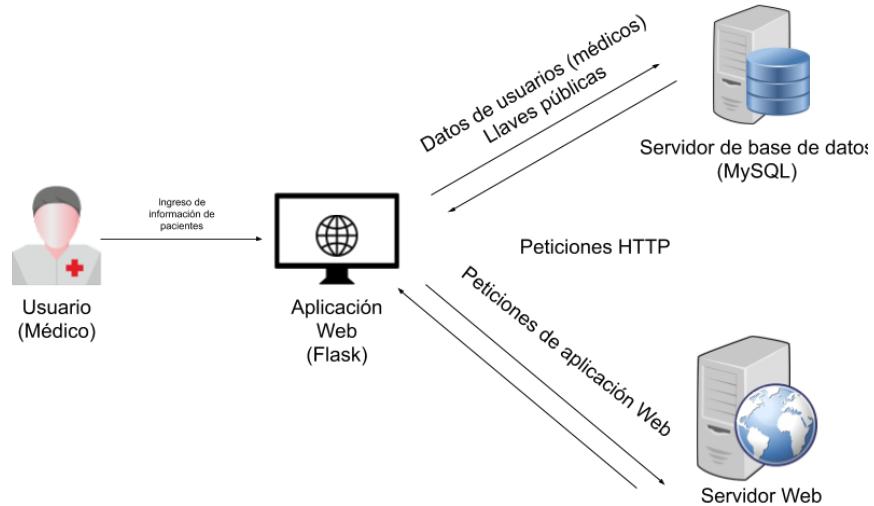


Figura 30: Arquitectura web propuesta

6.9. Vistas propuestas para el sistema web

La captura de pantalla muestra la interfaz de login de un navegador web. El encabezado dice 'Navegador Web'. En la barra de direcciones se ve la URL <http://servidormedicorfid.com>. El formulario de login tiene dos campos: 'Ingrese su nombre de usuario' con el valor 'user1' y 'Ingrese su contraseña' con el valor '*****'. Abajo del formulario hay un botón 'Acceder'.

Figura 31: Login de sistema

Navegador Web
http://servidormedicorfid.com/admin

Menú Administrador

Médicos registrados

Nombre	Fecha de nacimiento	Acciones
Ada Lovelace	December 10, 1815	Actualizar Eliminar
Grace Hopper	December 9, 1906	Actualizar Eliminar
Margaret Hamilton	August 17, 1936	Actualizar Eliminar
Joan Clarke	June 24, 1917	Actualizar Eliminar

Crear nuevo registro

Figura 32: Menú de administradores

Navegador Web
http://servidormedicorfid.com/admin/crear

Menú Administrador

Creación de nuevo registro

Nombre

Contraseña

Especialidad

Ciudad

Ciudad

Aceptar

Figura 33: Creación de nuevo registro de médico

Navegador Web
<http://servidormedicorfid.com/admin/modificar>

Menú Administrador

Actualización de registro

Nombre	Ricardo Lopez
Contraseña	*****
Especialidad	Otorrinolaringólogo
Ciudad	Miami
Fecha de nacimiento	4/22/2012 <input type="button" value="Calendario"/>

Figura 34: Modificación de registro de médico

Navegador Web
<http://servidormedicorfid.com/admin>

Menú Administrador

Médicos registrados

Nombre	Fecha de nacimiento	Acciones
Ada Lovelace	June 10, 1815	<input type="button" value="Eliminar"/>
Grace Hopper	December 9, 1906	<input type="button" value="Eliminar"/>
Margare Hamilton	January 11, 1915	<input type="button" value="Eliminar"/>
Joan Clarke	June 24, 1917	<input type="button" value="Actualizar"/> <input type="button" value="Eliminar"/>

Figura 35: Eliminación de registro de médico



Figura 36: Menú de usuario médico

Figura 37: Captura de datos de paciente para su posterior vaciado a etiqueta RFID

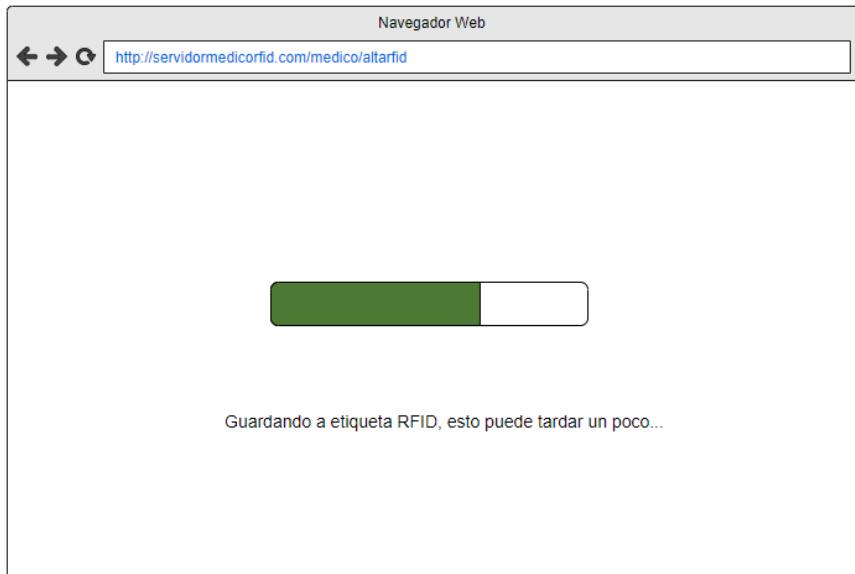


Figura 38: Guardado de datos en etiqueta RFID

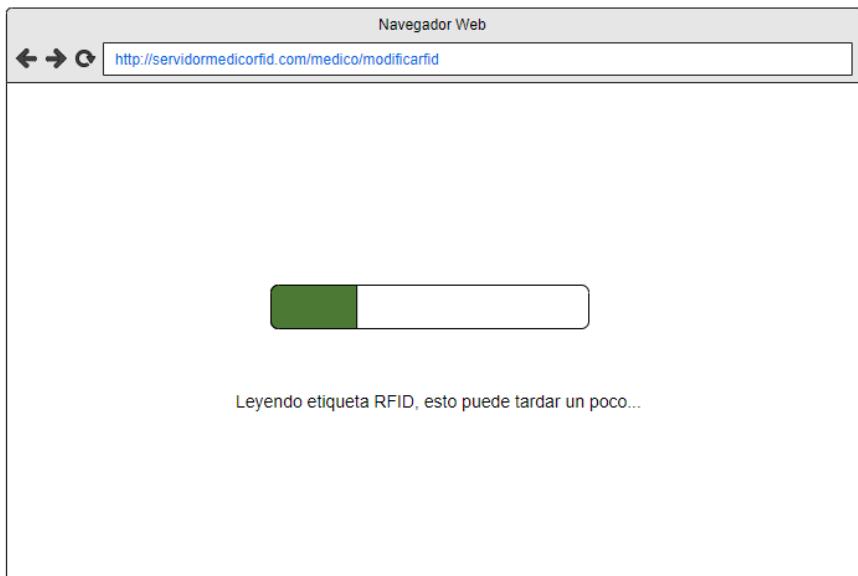


Figura 39: Lectura de datos en etiqueta RFID

Navegador Web
<http://servidormedicorfid.com/medico/modificarrfid>

Menú médico
 Ingrese los datos del paciente

Nombre*	Pedro	Condiciones médicas	Alergias	Medicamentos autorizados
Apellidos*	Perez Perez	Hipertensión	Ninguna	Ibuprofeno
CURP*	PEPP99XXX	+ Añadir nuevo	+ Añadir nuevo	+ Añadir nuevo
Tipo de sangre	AB+ *			
Teléfono personal	5511223344 *			
Teléfono emergencia	5511223345 *			
Teléfono emergencia				
Identificador pulsera 02930290 *				
Datos obligatorios *				
<input checked="" type="checkbox"/> Donador de órganos*		<input type="button" value="Cancelar"/> <input type="button" value="Actualizar etiqueta"/>		

Figura 40: Modificación de datos de paciente

6.10. Delimitación de perfiles de usuario

Para este proyecto se contemplarán dos tipos de usuario:

- **Administrador:** Será el encargado del manejo de registros relacionados con los médicos que utilizarán el sistema. En caso de que alguno de ellos presente algún problema para acceder a su cuenta, él será el responsable de devolverle el acceso a su perfil.
- **Médico responsable:** Será responsable de capturar la información de sus respectivos pacientes para que sea vertida en las etiquetas RFID. En caso de que sea necesario actualizar la información necesitará leer la etiqueta correspondiente, mediante el dispositivo de lectura.

6.11. Diagrama de clases

En desarrollo

6.12. Casos de uso

En desarrollo

6.13. Trayectorias del sistema

En desarrollo

7. Desarrollo del sistema

Al ser un proyecto híbrido el que va a ser desarrollado, se adecuará la metodología para poder llevarla a cabo tanto para el hardware como para el software que se planea entregar.

7.1. Planificación de actividades por iteración

Iteración 0 o Iteración Inicial

- Plática con directores
- Planeación inicial
- Delimitación de requerimientos de los módulos del sistema
- Planificación de actividades

1º Iteración (Arquitectura web)

- Diseño de arquitectura web
- Desarrollo de aplicación
- Pruebas
- Evaluación del entregable

2º Iteración (Etiqueta RFID)

- Proceso de selección
- Diseño de módulo de comunicación RFID
- Pruebas
- Evaluación del entregable

3º Iteración (Selección de algoritmo)

- Proceso de selección
- Diseño de módulo comunicación serial y cifrado
- Pruebas

- Evaluación del entregable

4º Iteración (Módulo de cifrado)

- Diseño de “core” de cifrado
- Pruebas
- Evaluación del entregable

5º Iteración (Módulo de descifrado)

- Diseño de “core” de descifrado
- Pruebas
- Evaluación del entregable

6º Iteración (Módulo de autenticación)

- Diseño
- Desarrollo y codificación de la terminal
- Pruebas
- Evaluación del entregable

7º Iteración (Integración de módulos).

- Desarrollo y establecimiento de comunicación entre todos los módulos desarrollados
- Pruebas finales con los módulos en conjunto

8. Pruebas y resultados

8.1. Implementación de LEA en software

Para el desarrollo del módulo de cifrado, se desarrolla un conjunto de pruebas en el lenguaje de programación python, los cuales nos sirven para comprender con más profundidad el funcionamiento del algoritmo LEA.

```
[mauricio@friednoodles tt]$ python3 lea.py
encrypted value is:
100100111100110100000101111100
1010101001000001101101100101111
00011010110011001100010000001
1001100001010111111100010100111
P's value is:
00010001000100010001000100010001
00010001000100010001000100010001
00010001000100010001000100010001
00010001000100010001000100010001
K's value is:
01110111011101110111011101110111
01110111011101110111011101110111
01110111011101110111011101110111
01110111011101110111011101110111
decrypted value is:
00010001000100010001000100010001
00010001000100010001000100010001
00010001000100010001000100010001
00010001000100010001000100010001
[mauricio@friednoodles tt]$ ]
```

Figura 41: Prueba de LEA en software (1)

```
[mauricio@friednoodles tt]$ python3 lea.py
encrypted value is:
101010101010101000110000000100000
11111000011100000010001111011110
11011001100001111111011100101110
00010110101110101011100101100001
P's value is:
00110001000100010001000100010001
00010001001111010001001111100001
00010001111110010011111100010001
11111101001111111001000100010001
K's value is:
1101011101110111011111111111111
011101110111011101110111011100000111
011101110111011101110111011101110111
0111011100000000000000000000000000
decrypted value is:
00110001000100010001000100010001
00010001001111010001001111100001
00010001111110010011111100010001
11111101001111111001000100010001
[mauricio@friednoodles tt]$ ]
```

Figura 43: Prueba de LEA en software (3)

```
[mauricio@friednoodles tt]$ python3 lea.py
encrypted value is:
11111101011011110010000010101100
10100001011101101001000010000111
11001011100011110000011010001010
11110100001111100011100010000110
P's value is:
00110001000100010001000100010001
00010001001111010001001111100001
00010001000100010001000100010001
00010001000100010001000100010001
K's value is:
1101011101110111011111111111111
01110111011101110111011100000111
01110111011101110111011101110111
01110111011101110111011101110111
decrypted value is:
00110001000100010001000100010001
00010001001111010001001111100001
00010001000100010001000100010001
00010001000100010001000100010001
[mauricio@friednoodles tt]$ ]
```

Figura 42: Prueba de LEA en software (2)

```
[mauricio@friednoodles tt]$ python3 lea.py
encrypted value is:
11111101011011110010000010101100
10100001011101101001000010000111
11001011100011110000011010001010
11110100001111100011100010000110
P's value is:
00110001000100010001000100010001
00010001001111010001001111100001
00010001000100010001000100010001
00010001000100010001000100010001
K's value is:
1101011101110111011111111111111
01110111011101110111011100000111
01110111011101110111011101110111
01110111011101110111011101110111
decrypted value is:
00110001000100010001000100010001
00010001001111010001001111100001
00010001000100010001000100010001
00010001000100010001000100010001
[mauricio@friednoodles tt]$ ]
```

Figura 44: Prueba de LEA en software (4)

- C Cipher Process
Pt= Lorem ipsum dolor sit amet, consectetur adipisci ng elit. Maecena scilicet imperdiet
HÖÖl2ÿ5ÜGóø&Øx!ØØØØL÷UØüþØ\$L ïE
Decipher Process
Dt= Lorem ipsum dolor sit amet, consectetur adipisci ng elit. Maecena scilicet imperdiet
HÖÖl2ÿ5ÜGóø&Øx!ØØØØL÷UØüþØ\$L ïE

Figura 45: Prueba de LEA en software(5)

El código de implementación de LEA, puede ser consultado en el siguiente [enlace](#)

8.2. Pruebas del Key Schedule de LEA

La primera parte del trabajo terminal contempla una prueba unitaria del proyecto, por lo que se decidió implementar el módulo de KeySchedule.

El diagrama RTL que se obtuvo una vez que se implementó el diseño propuesto en éste documento fue el siguiente:

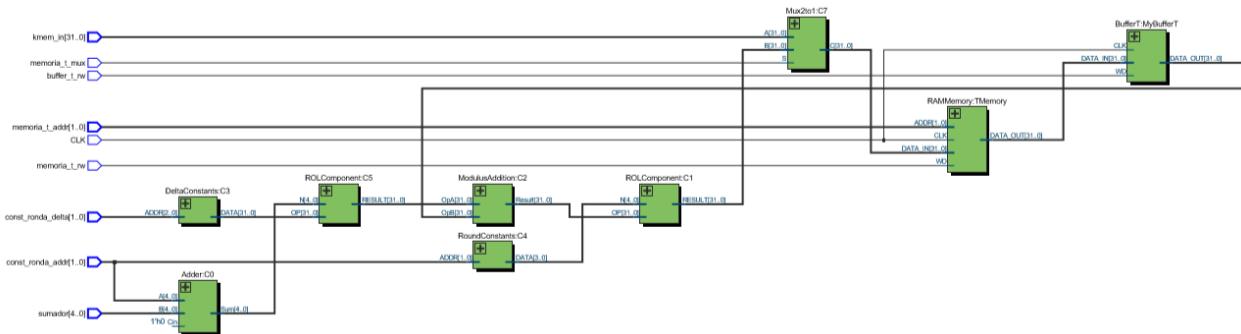


Figura 46: Primera ronda de llave, generación de $T[0], T[1], T[2]$ y $T[3]$ para $i = 0$

En la siguiente figura se muestra la inicialización de la memoria T con la llave K, esto tal como nos indica el algoritmo LEA.



Figura 47: T=K. Inicialización de memoria T

En la figura que se ve a continuación se muestra la generación de la llave de ronda con $i = 0$, estos resultados pueden ser comparados con los resultados obtenidos en la implementación en software del mismo algoritmo.

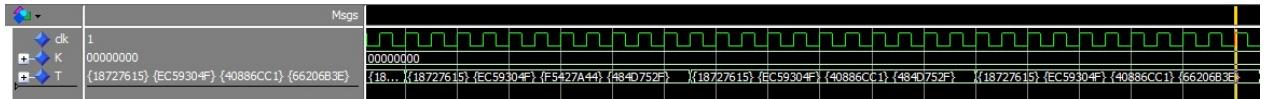


Figura 48: Primera ronda de llave, generación de $T[0], T[1], T[2]$ y $T[3]$ para $i = 0$

Finalmente tenemos las última ronda de llave, donde $i = 23$

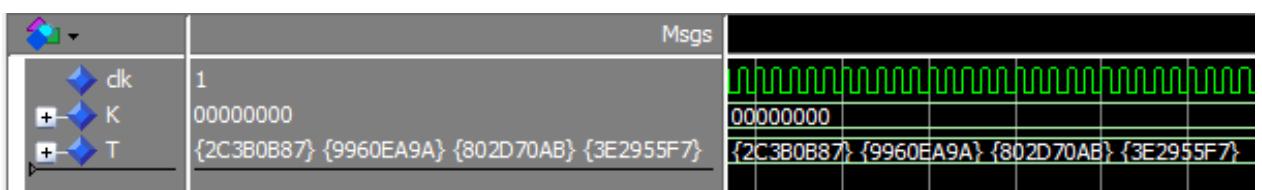


Figura 49: Última ronda de llave, generación de $T[0], T[1], T[2]$ y $T[3]$ para $i = 23$

Los resultados anteriores pueden ser comparados con la siguiente implementación en software del key schedule, donde podemos apreciar que existe una correspondencia entre los resultados.

```
main.py  saved
58
59  def key_schedule():
60
61      for i in range(0, 24):
62          T[0] = ROL(1, mod_add(T[0], ROL(i, delta[i % 4])))
63          T[1] = ROL(3, mod_add(T[1], ROL(i+1, delta[i % 4])))
64          T[2] = ROL(6, mod_add(T[2], ROL(i+2, delta[i % 4])))
65          T[3] = ROL(11, mod_add(T[3], ROL(i+3, delta[i % 4])))
66
67          if(i==0 or i==23):
68              print("i= "+str(i))
69              print(hex(int(T[0],2)))
70              print(hex(int(T[1],2)))
71              print(hex(int(T[2],2)))
72              print(hex(int(T[3],2)))
73
74          auxlist = [T[0],T[1],T[2],T[1],T[3],T[1]]
75          RK.append(auxlist)
76
77
```

```
i= 0
0x18727615
0xec59304f
0x40886cc1
0x66206b3e
i= 23
0x2c3b0b87
0x9960ea9a
0x802d70ab
0x3e2955f7
▶ □
```

Figura 50: Verificación de los resultados anteriores

A continuación se muestra la simulación y diagrama RTL de los diferentes módulos que conforman el key schedule,

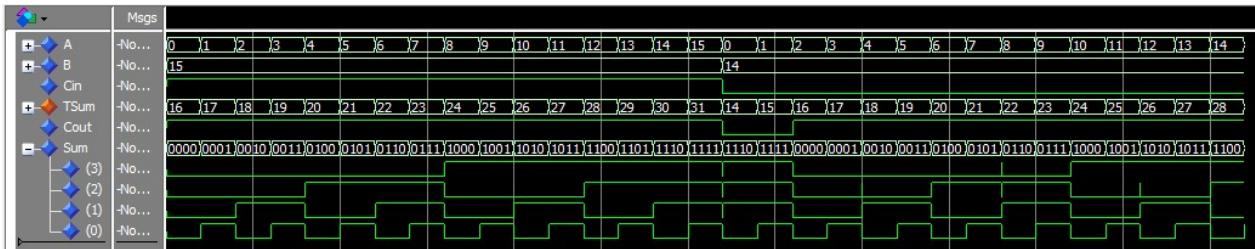


Figura 51: Simulación de sumador

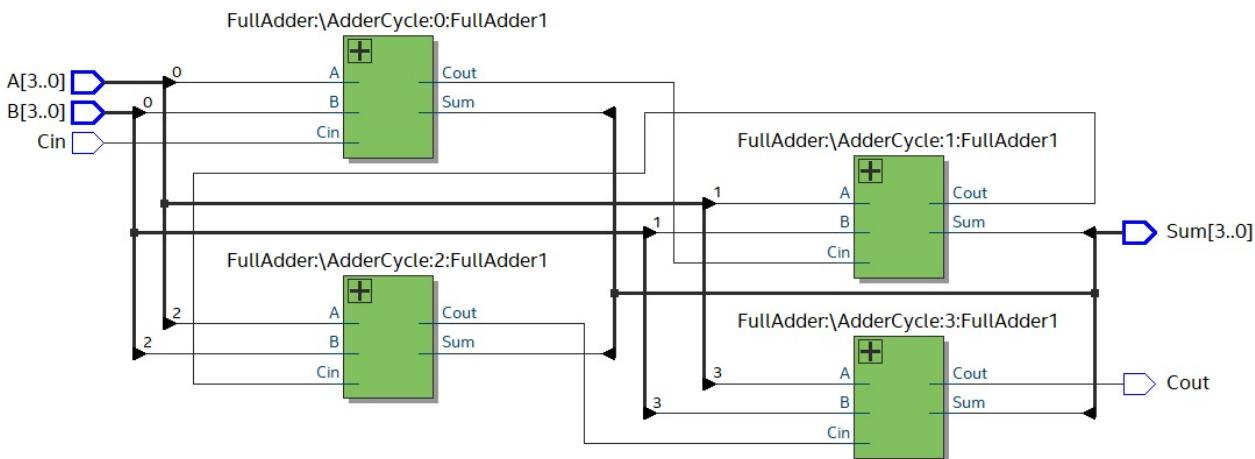


Figura 52: RTL del sumador

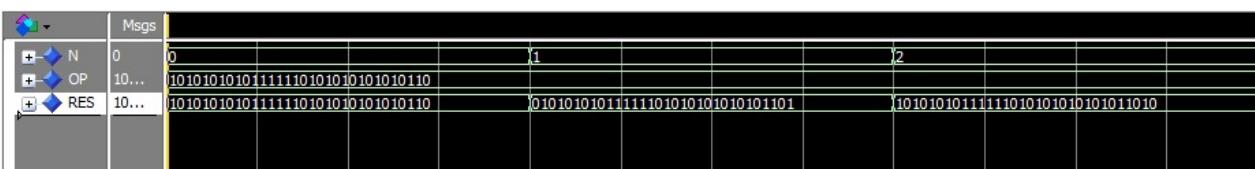


Figura 53: Simulación de ROL 1

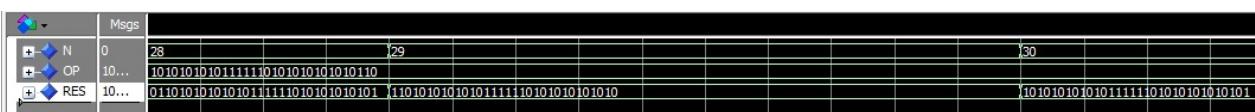


Figura 54: Simulación de ROL 2

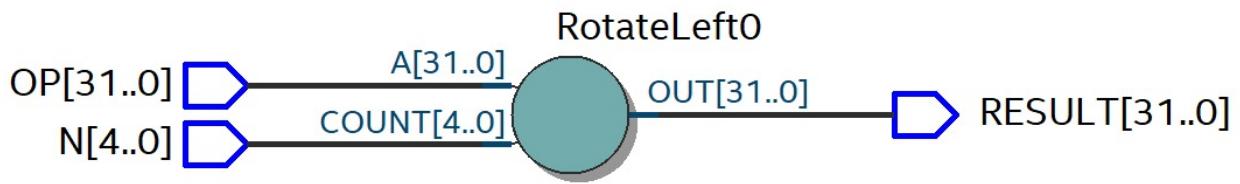


Figura 55: Diagrama RTL de ROL

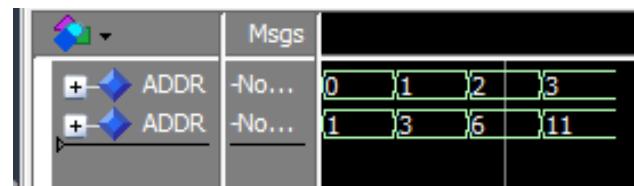


Figura 56: Simulación de la memoria de constantes de ronda

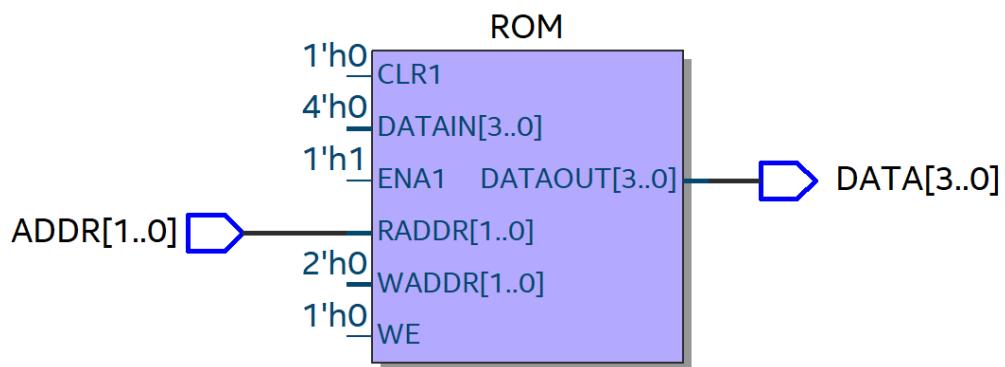
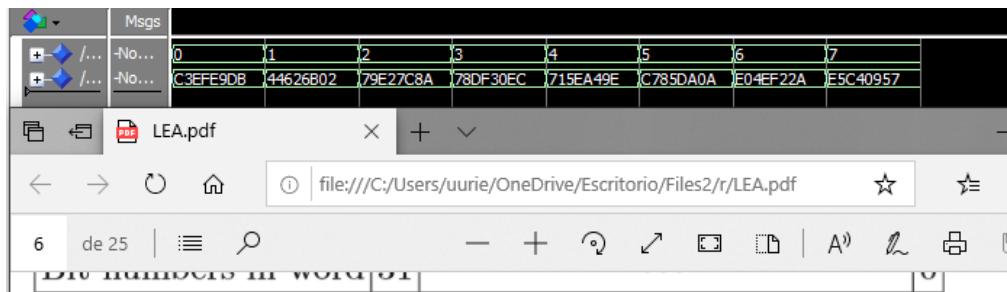


Figura 57: Diagrama RTL de la memoria de constantes de ronda



Schedule

The schedule generates a sequence of 192-bit round keys RK_i as follows:

The key schedule uses several constants for generating round keys, refined as

$$\begin{aligned}\delta[0] &= 0xc3eфе9db, & \delta[1] &= 0x44626b02, \\ \delta[2] &= 0x79e27c8a, & \delta[3] &= 0x78df30ec, \\ \delta[4] &= 0x715ea49e, & \delta[5] &= 0xc785da0a, \\ \delta[6] &= 0xe04ef22a, & \delta[7] &= 0xe5c40957.\end{aligned}$$

Figura 58: Prueba de la memoria de constantes delta

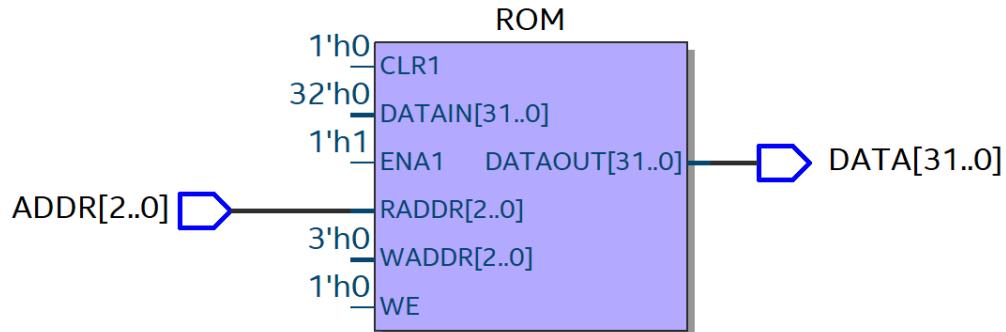


Figura 59: Diagrama RTL de la memoria de constantes delta

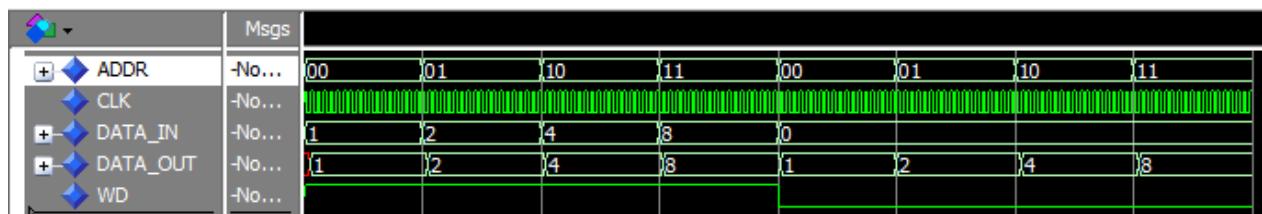


Figura 60: Simulación de la Memoria de llave, Memoria T y Buffer T

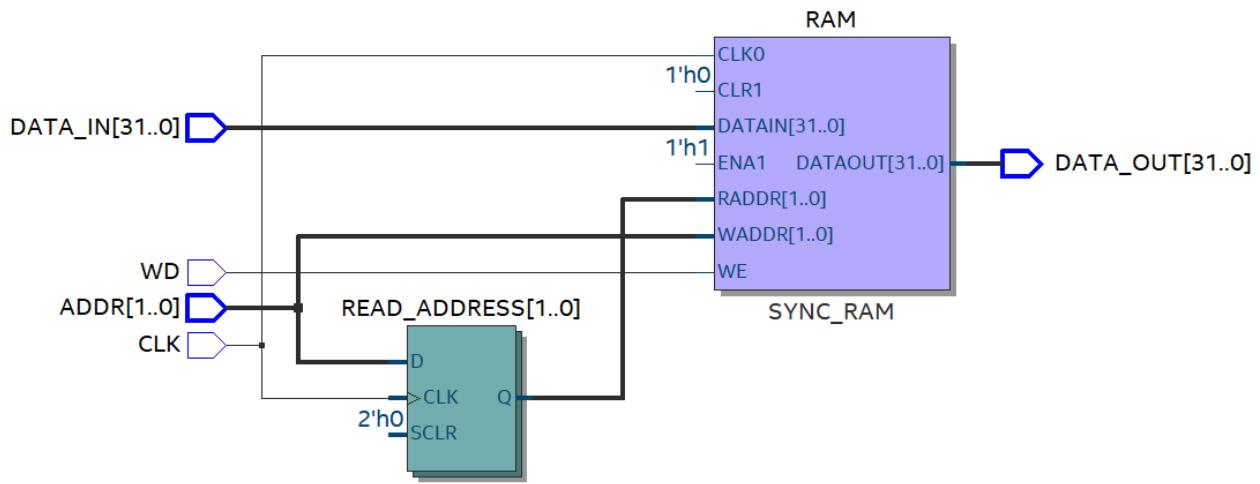


Figura 61: Diagrama RTL de la memoria

8.3. Pruebas iniciales del software de prueba

En las figuras siguientes se aprecia los avances iniciales del software de prueba, el cual servirá para visualizar información del FPGA o ingresar información a la misma.

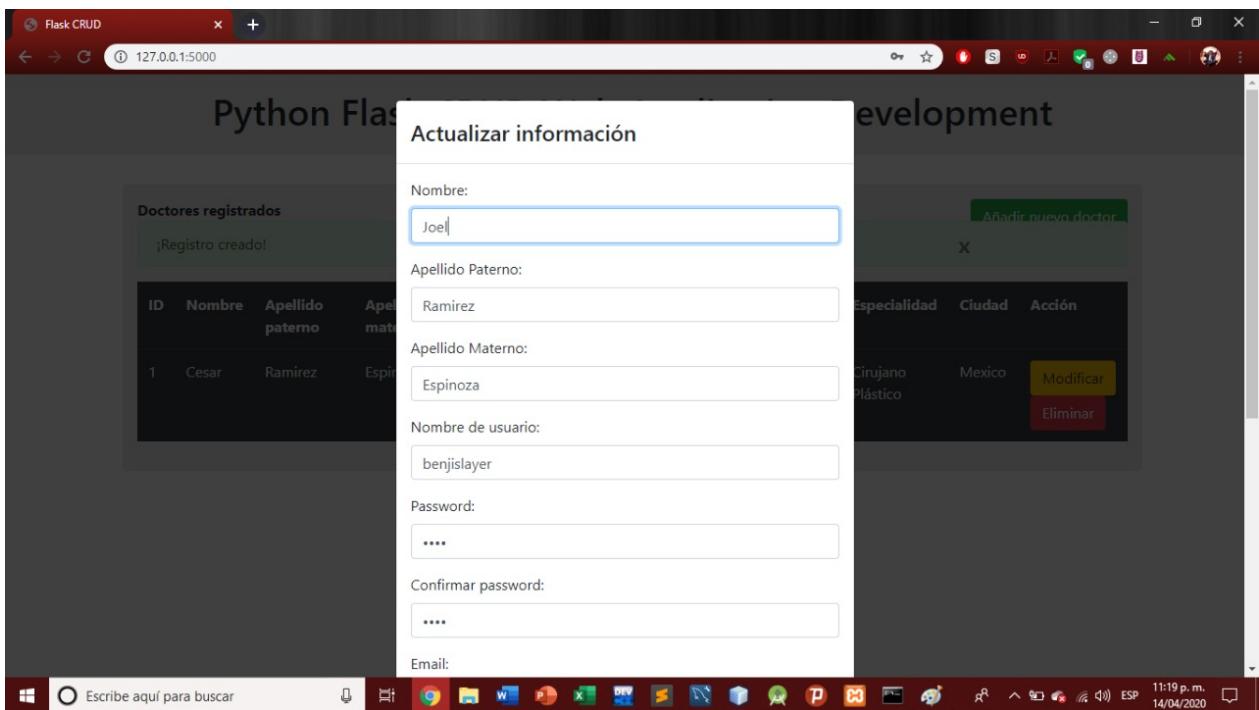


Figura 62: Prueba número 1

The screenshot shows a web browser window titled "Flask CRUD" with the URL "127.0.0.1:5000". The main content area is titled "Doctores registrados" and displays a table of doctor information. A green success message at the top states "¡Registro actualizado!". The table has columns: ID, Nombre, Apellido paterno, Apellido materno, Nombre de usuario, Contraseña, Email, Telefono, Especialidad, Ciudad, and Acción. One row is shown with values: ID 1, Nombre Joel, Apellido paterno Ramirez, Apellido materno Espinoza, Nombre de usuario benjislayer, Contraseña 1234, Email a@a.com, Telefono 32111111, Especialidad Cirujano Plástico, Ciudad Mexico, and two buttons in the Acción column: "Modificar" (yellow) and "Eliminar" (red). The bottom of the screen shows a Windows taskbar with various pinned icons.

Figura 63: Prueba número 2

The screenshot shows a web browser window titled "Flask CRUD" with the URL "127.0.0.1:5000". The main content area is titled "Doctores registrados" and displays a table of doctor information. A green success message at the top states "¡Registro eliminado!". The table has columns: ID, Nombre, Apellido paterno, Apellido materno, Nombre de usuario, Contraseña, Email, Telefono, Especialidad, Ciudad, and Acción. The table is currently empty. The bottom of the screen shows a Windows taskbar with various pinned icons.

Figura 64: Prueba número 3

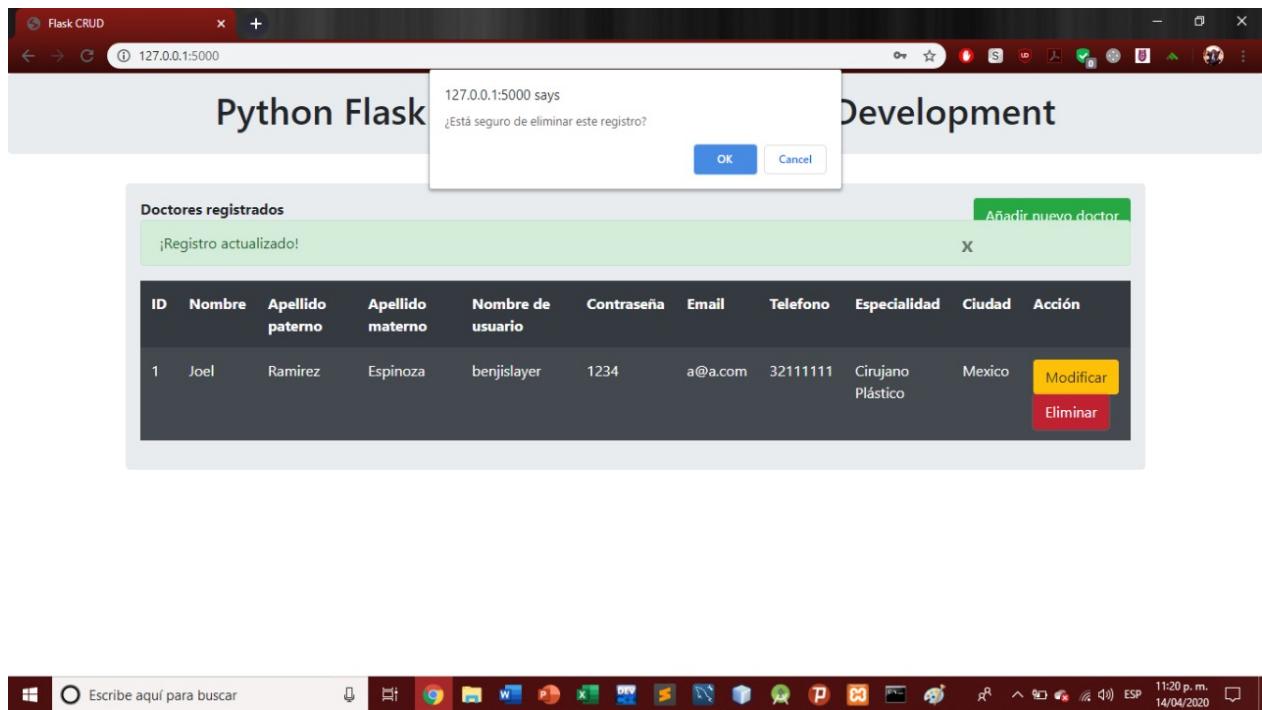


Figura 65: Prueba número 4

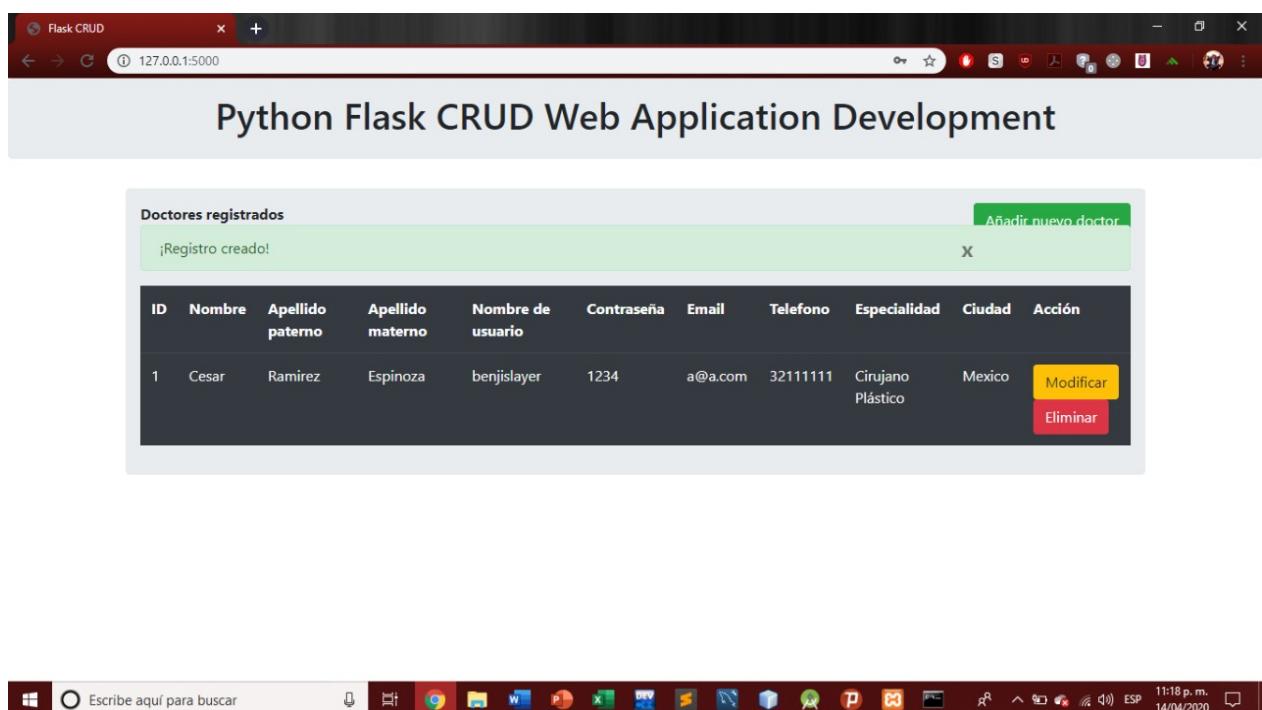


Figura 66: Prueba número 5

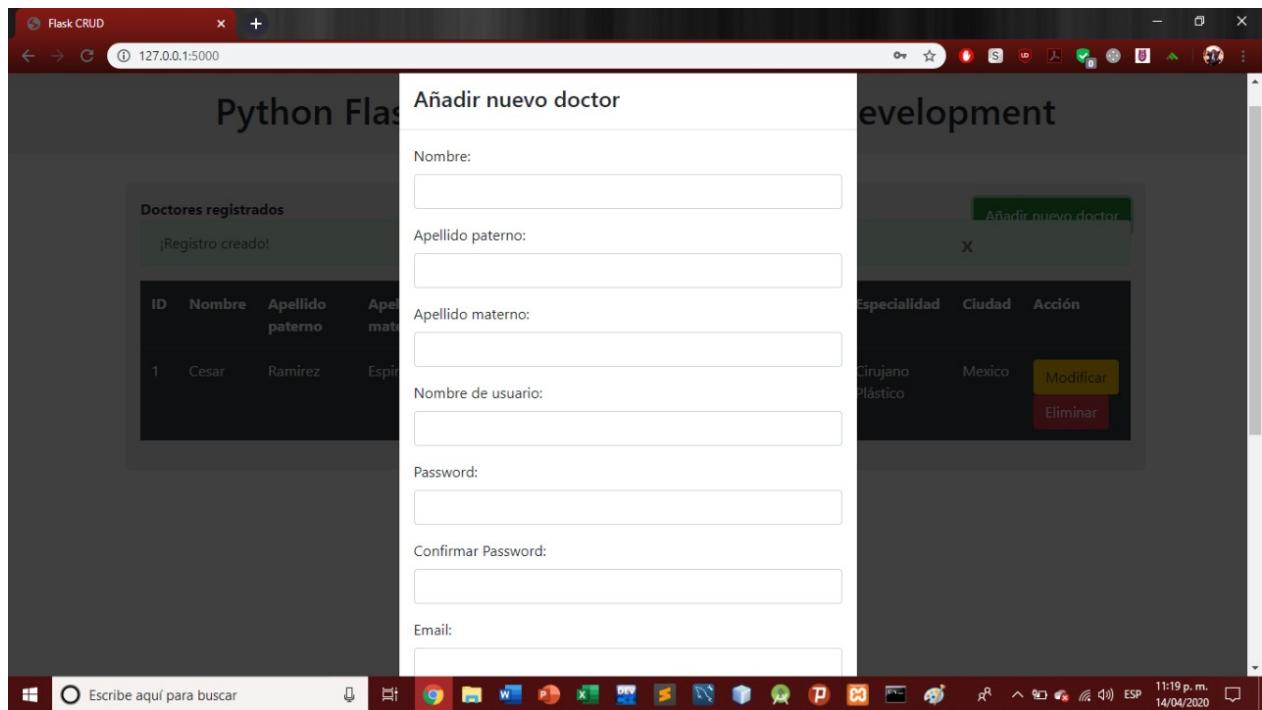


Figura 67: Prueba número 6

9. Conclusiones

10. Trabajo a futuro

11. Anexos

12. Glosario

- **FPGA:** Field-Programmable Gate Array
- **AES:** Advanced Encryption Standard
- **DES:** Data Encryption Standard
- **Blowfish:** Blowfish es un codificador de bloques simétricos
- **RFID:** (Radio Frequency Identification), tecnología de auto identificación por radiofrecuencia.
- **REST:** Advanced Encryption Standard
- **Lector RFID:** Advanced Encryption Standard
- **IP:** Intellectual Property
- **FSM:** Finite State Machine
- **SPI:** Serial Peripheral Interface, es una interfaz serial síncrona de comunicación.
- **VHDL:** VHSIC Hardware Description Language, es un lenguaje de descripción de hardware usado para el diseño de sistemas digitales.
- **Web Service:** Servicio web, es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.
- **FF:** Identification-Friend-or-Foe, sistema utilizado por los británicos durante la segunda guerra mundial para la identificación de aeronaves.
- **MSB:** *Most significant bit* ó bit más significativo. La posición del bit con mayor valor en un número binario.
- **LSB:** *Least significant bit* ó bit menos significativo. La posición del bit con menor valor en un número binario.
- **SQL:** *Structured Query Language* Lenguaje utilizado para el diseño y manejo de la información contenida en bases de datos relacionales.

Referencias

- [1] C. C. E. Madrigal. (oct. de 2010). Robo de Identidad y Consecuencias Sociales, dirección: <https://www.artima.com/intv/python.html>.
- [2] (2019). Código de ética para el ejercicio profesional del médico colegiado en México, dirección: http://www.comego.org.mx/reglamentos/codigo_etica.pdf.
- [3] I. Security. (2019). Cost of a Data Breach Report.
- [4] Kroll, *Global Fraud & Risk Report*. 2018, pág. 66.
- [5] K. Bennhold. (2019). Robo de Identidad y Consecuencias Sociales, dirección: <https://www.nytimes.com/es/2019/07/04/merkel-temblores-salud-temblando/>.
- [6] Protenus, *2019 Annual Breach Barometer Report*. 2019, págs. 10-11.
- [7] H. Journal. (). Healthcare Data Breach Statistics, dirección: <https://www.hipaajournal.com/healthcare-data-breach-statistics/>.
- [8] HealthcareITNews. (). Telemedicine vendor breaches the data of 2.4 million patients in Mexico, dirección: <https://www.healthcareitnews.com/news/telemedicine-vendor-breaches-data-24-million-patients-mexico>.
- [9] O. Jarquín García. (ago. de 2017). Historial Clínico para el Departamento de Servicio Médico de ESCOM, dirección: <https://tesis.ipn.mx/handle/123456789/22667>.
- [10] R. A. D. I. Cedeño Gómez Juan Carlos Crespo Hernández Raquel. (feb. de 2017). Sistema móvil de registros de enfermería implementando tecnología rfid, dirección: <https://tesis.ipn.mx/handle/123456789/22095>.
- [11] E. E. J. Moreno Cruz Ariadna Betzabe; Toledo Mejía. (sep. de 2017). Sistema de expediente clínico con adquisición automática de datos, dirección: <https://tesis.ipn.mx/handle/123456789/22719>.
- [12] R. MM. (mayo de 2018). Using RFID Technology for Managing Patient Medical File, dirección: <http://www.imedpub.com/articles/using-rfid-technology-for-managing-patient-medical-file.php?aid=22831>.
- [13] J. Wu. (2012). RFID for Inventory of Medical Records, dirección: <https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1032&context=bmedsp>.
- [14] M. L. Gonzalo Canepa Lautaro Palarino. (abr. de 2016). RFID for Inventory of Medical Records, dirección: <https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1032&context=bmedsp>.
- [15] J. P. Christof Paar, *Understanding Cryptography: A Textbook for Students and Practitioners*. 2010, págs. 2, 3.
- [16] W. J. Buchanan, D. S. Li y D. R. Asif, «Lightweight Cryptography Methods», pág. 1, mar. de 2018.
- [17] c. ISO/IEC JTC 1/SC 27 Information security y privacy protection. (nov. de 2019). ISO/IEC 29192-2:2019 [ISO/IEC 29192-2:2019] Information security —Lightweight cryptography —Part 2: Block ciphers.
- [18] CRYPTREC. (abr. de 2017). Cryptographic Technology Guideline(Lightweight Cryptography).

- [19] () .Lightweight Cryptography, dirección: <https://csrc.nist.gov/projects/lightweight-cryptography>.
- [20] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu y D.-G. Lee, «LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors», Y. Kim, H. Lee y A. Perrig, eds., págs. 3-27, 2014.
- [21] (), dirección: <https://aldeahost.com.mx/todo-lo-que-necesitas-saber-sobre-el-web-service/>.
- [22] (), dirección: <https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos>.
- [23] (), dirección: <https://www.python.org>.
- [24] (), dirección: <https://palletsproject.com/p/flask>.
- [25] (), dirección: <https://www.mysql.com>.
- [26] Intel. (2020), dirección: <https://www.intel.la/content/dam/www/programmable/us/en/pdfs/literature/pt/cyclone-iv-product-table.pdf>.
- [27] XILINX. (2020), dirección: <https://www.xilinx.com/products/silicon-devices/fpga/spartan-6.html>.
- [28] M. T. Incorporated. (2019), dirección: <http://ww1.microchip.com/downloads/en/DeviceDoc/00003294A.pdf>.
- [29] L. S. Corp. (feb. de 2019), dirección: <https://www.mouser.com/datasheet/2/225/FPGA-DS-02032-2-4-MachX03-Family-Data-Sheet-1022841.pdf>.
- [30] G. S. Corp. (2020), dirección: <https://gowinsemi.com/en/product/detail/2/>.
- [31] Q. Corporation. (2020), dirección: <https://www.quicklogic.com/products/fpga/fpgas-antifuse/#polarpro2>.
- [32] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou y C. Manifavas, págs. 20-23, abr. de 2017.
- [33] N. S. N.V. (abr. de 2016). Standard performance MIFARE and NTAG frontend.
- [34] Amazon. (2020), dirección: https://www.amazon.com/gp/offer-listing/B081RHDNH7/ref=dp_olp_new_mbc?ie=UTF8&condition=new.
- [35] AliExpress. (2020), dirección: https://www.aliexpress.com/item/32904956069.html?spm=a2g0o.productlist.0.0.2f0749e15HjnNt&algo_pvid=bd1d6739-8a3d-4103-a801-d28531354f74&algo_expid=bd1d6739-8a3d-4103-a801-d28531354f74-14&btsid=0ab6d59515835477456308199e5ef0&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_.
- [36] ——, (2020), dirección: https://www.aliexpress.com/item/32904956069.html?spm=a2g0o.productlist.0.0.2f0749e15HjnNt&algo_pvid=bd1d6739-8a3d-4103-a801-d28531354f74&algo_expid=bd1d6739-8a3d-4103-a801-d28531354f74-14&btsid=0ab6d59515835477456308199e5ef0&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_.