



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo fin de Grado

Ingeniería Informática - Tecnologías Informáticas

Un criptosistema para imágenes médicas basado en un nuevo operador

**Realizado por
Marcel Espejo Cuenca**

**Dirigido por
José Andrés Armario Sampalo**

**Departamento
Matemática aplicada I**

Sevilla, Convocatoria de junio 2021/2022

Resumen

Con la necesidad de transmitir información por la red, se dan casos donde los datos que se van a transmitir no deben ser comprendidos por alguien que, accidental o intencionadamente pueda interceptarlos. En este caso, la información a transmitir son imágenes, ya sean imágenes normales que vemos a diario, o imágenes médicas que deben ser transmitidas entre personal sanitario. Estas imágenes médicas no solo pueden contener radiografías de pacientes, lo cual ya proporciona mucha información delicada, también existen datos extra relativos al paciente directamente «incrustados» en la propia imagen, por ejemplo nombre, estatura y otros datos de interés médico. El objetivo es implementar un algoritmo de cifrado capaz de tratar imágenes naturales y estas imágenes médicas llamadas DICOM (Digital Imaging and Digital Communication In Medicine) que se codifican en 16-bits. El criptosistema propuesto hace uso de distintas herramientas matemáticas como son las secuencias caóticas, computación basada en ADN y un nuevo operador binario propuesto por un artículo científico reciente (2021) el cual será el punto de partida para implementar dicho criptosistema. Además, se comprueban empíricamente los resultados de dicho artículo. La implementación realizada se ha subido a GitHub y puede encontrarse en [13].

Agradecimientos

A toda persona que lea este documento y a quien ha ayudado a crearlo.

Índice general

Índice general	III
Índice de cuadros	V
Índice de figuras	VI
1 Introducción	1
1.1 Justificación y objetivos	2
1.2 Organización del documento	2
2 Conceptualización	4
2.1 Metodología	4
2.1.1 Antecedentes	4
2.1.2 Estudio de Viabilidad	5
2.2 Definición	5
2.2.1 Análisis de requisitos	6
2.2.2 Objetivos, alcance, calidad	6
3 Planificación	8
3.1 Análisis temporal y costes de desarrollo	8
3.1.1 Actividades, tareas e hitos	8
3.1.2 Cronograma	10
3.1.3 Estimación de costes	11
3.2 Plan de riesgos	11
3.2.1 Identificación de riesgos	11
3.2.2 Planes de contingencia	13
3.3 Planes de calidad	13
3.3.1 Indicadores	13
3.3.2 Plan de mejora	14
3.4 Planes de comunicaciones	14
4 Contexto teórico	15
4.1 Operador binario \otimes	15
4.1.1 Se cumple la propiedad conmutativa	16
4.1.2 Se cumple la propiedad asociativa	16
4.1.3 Existe el elemento identidad y es el 0	17
4.1.4 Existe un elemento inverso	17
4.1.5 El operador \otimes no es lineal	18
4.1.6 El conjunto S está cerrado bajo \otimes	18
4.2 Mapas y series caóticas	19
4.2.1 Mapa	19

4.2.2	Puntos fijos, puntos k-periódicos	19
4.2.3	Diagrama de bifurcación	20
4.2.4	Definición de caos	21
4.2.5	Medida del caos	21
4.3	Arnold's Cat Map	21
4.4	Criptografía en computación basada en ADN	23
5	Estructura del algoritmo	25
5.1	Algoritmo de encriptado	25
5.1.1	Relativo a mapas caóticos	25
5.1.2	Otros preliminares para operaciones de mezcla	27
5.1.3	Mezclado de filas y mezclado (Mixing)	28
5.1.4	Relativo a la codificación en ADN	28
5.1.5	Arnold's Cat Map modificado	29
5.2	Algoritmo de desencriptado	30
5.2.1	Operador inverso (\otimes^{-1})	30
5.2.2	Invertir las operaciones en ADN y de mezcla	32
5.2.3	Invertir Arnold's Cat Map	32
5.3	Referente a la implementación	33
5.4	Interfaz gráfica	34
6	Experiencias y dificultades durante el desarrollo	37
6.1	El problema en Key Scheduling	37
6.2	Algunas confusiones en pseudocódigos	38
6.3	Problema en Arndol's Cat Map	38
6.4	Encontrar datos de prueba	39
6.5	Algunos añadidos extra	40
7	Resultados y análisis estadístico	42
7.1	Complejidad	42
7.1.1	Complejidad según el artículo	43
7.1.2	Possible error en el cálculo de la complejidad	43
7.2	Imágenes naturales	44
7.3	Imágenes médicas	47
7.4	Tiempos de ejecución con múltiples imágenes	50
7.5	Análisis de varianza	51
7.6	Análisis de entropía	52
7.7	Imágenes Homogéneas	53
8	Mejoras y futuras vías de progreso	55
8.1	Ejemplo futuro proyecto (criptosistema universal)	55
8.2	Posibles mejoras	56
9	Conclusiones	58
9.1	Qué se ha hecho	58
9.2	Conocimiento generado	59
9.3	Conclusiones personales	59
9.4	Cierre del proyecto	60
10	Apéndices	64
.1	Acta de adjudicación de TFG	64

Índice de cuadros

3.1	Actividades, tareas e hitos	9
3.2	Resumen cronograma	12
4.1	Reglas de codificación de ADN	23
4.2	Resultados de XOR	23
4.3	Resultados de \otimes	24
7.1	Tiempos de ejecución para imágenes DICOM	50
7.2	Tabla de varianza para imágenes DICOM	52
7.3	Tabla de entropía para imágenes DICOM	52
8.1	Codificación caracteres ejemplo	56
9.1	Horas según Clockify a fecha de 20/06/2022	63

Índice de figuras

3.1	Gantt (planificación y comprensión)	10
3.2	Gantt (Ejemplo de solapamiento)	10
3.3	Gantt (Fases)	11
4.1	Diagrama de bifurcación	20
4.2	Diagrama de bifurcación resaltando zonas con Lyapunov negativo	22
4.3	Periodo Arnold's Cat Map Wikipedia	22
5.1	Algoritmo de encriptado	26
5.2	Ejemplo ACM modificado horizontal	29
5.3	Ejemplo ACM modificado vertical	30
5.4	Diagrama de flujo de desencriptado	31
5.5	Comando para ejecutar el algoritmo por consola	34
5.6	Pantalla interfaz	35
5.7	Cifrado Dicom	35
5.8	Pantalla interfaz	36
6.1	Periodos por tamaño en ACM	39
6.2	Captura de NBIA Data Retriever	40
7.1	Pseudocódigo del algoritmo ACM modificado en el artículo	43
7.2	Imagen natural ejemplo	44
7.3	Histograma de la imagen natural de ejemplo	45
7.4	Imagen natural ejemplo en escala de grises encriptada	45
7.5	Histograma de la imagen natural de ejemplo encriptada	45
7.6	Imagen natural ejemplo a color	46
7.7	Histograma de la imagen natural de ejemplo a color	46
7.8	Imagen natural ejemplo encriptada a color	46
7.9	Histograma de la imagen natural de ejemplo encriptada a color	46
7.10	Imagen dicom ejemplo	47
7.11	Histograma de la imagen dicom de ejemplo	47
7.12	Histograma de la imagen dicom de ejemplo reescalado	48
7.13	Imagen dicom de ejemplo encriptada	48
7.14	Histograma de la imagen dicom de ejemplo encriptada	49
7.15	Histogramas de resultados en el artículo	49
7.16	Histograma de imagen en blanco	53
7.17	Imagen de píxeles blancos encriptada	54
7.18	Histograma de imagen en blanco encriptada	54
9.1	Añadir tarea en Clockify	61
9.2	Diagrama de tarta de tareas por etiqueta	61

9.3	Etiquetas del diagrama de tarta de Clockify	61
9.4	Etiquetas del diagrama de tarta de Clockify	62
9.5	Tiempo invertido mensualmente hasta 20/06/2022	62
1	Acta de constitución	65

CAPÍTULO 1

Introducción

El campo de la criptografía abarca todo aquello que pretende ocultar información mediante la alteración de esta, de quienes de forma accidental o a propósito y sin ser parte del conjunto de receptores de dicha información, pretendan conocer el contenido.

En la antigüedad, la información a esconder se transmitía de forma escrita y para evitar que una persona no autorizada sea capaz de leer el texto, este se alteraba utilizando algoritmos secretos solo conocidos por el transmisor y el receptor del mensaje, se basaba en el oscurantismo. Sin embargo, realizando un salto a la época actual, no basta con esto para esconder la información, ya que la capacidad de computación con la que se cuenta ahora, se pueden descifrar fácilmente documentos cifrados con un algoritmo simple y secreto mediante distintas técnicas como la búsqueda de ciertos patrones en el mensaje cifrado, por tanto el cifrado del mensaje con un algoritmo cuya seguridad se base principalmente en el no conocimiento por parte del atacante del algoritmo utilizado, está descartado, además, hoy en día es relativamente fácil encontrar información sobre gran cantidad de sistemas de cifrado.

Por tanto, en la actualidad se considera que el atacante sabe mucho, conoce que hay un mensaje cifrado, y conoce que algoritmo se está usando para el cifrado, partiendo de ahí se busca que el algoritmo de cifrado se base en un problema matemático complicado o no resuelto para que sea seguro, por ejemplo, la factorización de números muy grandes, o la predicción de un valor en una serie caótica. Debemos tener en cuenta también que, gracias a la tecnología actual, no solo es posible cifrar textos, sino también otro tipo de datos como lo son las imágenes y es en esto en lo que se centra el proyecto. Las imágenes tienen una complicación extra a la hora de cifrar frente al texto escrito, muy a menudo, existen una gran cantidad de patrones, el ejemplo más claro sería el de cifrar una imagen toda del mismo color.

Debido a todo el avance que se ha hecho en el campo de las telecomunicaciones, existe la necesidad de transmitir todo tipo de datos sensibles, entre estos datos sensibles se encuentran imágenes médicas que contienen, por ejemplo, las radiografías de los pacientes, estas imágenes se han estandarizado en DICOM (Digital Imaging and Communication in Medicine) [3], la característica principal que diferencia la imágenes DICOM de las imágenes naturales es que utilizan una profundidad de pixel de 16-bits, las imágenes naturales por otro lado son de 8-bits (256 tonos de gris).

1.1– Justificación y objetivos

Actualmente existen múltiples criptosistemas capaces de cifrar imágenes naturales con distintas herramientas matemáticas, sin embargo, a la hora de cifrar imágenes DICOM, los criptosistemas propuestos tienen uno o más de los siguientes problemas:

- Su complejidad es muy alta, realizan una gran cantidad de cálculos y es necesario hacer muchas iteraciones sobre el propio algoritmo.
- Se basan en la subdivisión de la imagen DICOM de 16-bits en dos imágenes de 8-bits con permutaciones entre ellas utilizando dos claves, esto causa que se duplique el tamaño de la imagen cifrada respecto a la original.
- Solo son capaces de cifrar imágenes DICOM y no imágenes naturales.
- Se basan en el operador XOR, el cual presenta una gran desventaja, operar un elemento consigo mismo obtiene el elemento neutro, por ejemplo para una variable x , $x \oplus x = 0$, y para dos variables x e y , se tiene que $x \oplus y \oplus y = x$. Por lo general, esto puede causar problemas de modo que aparecen correlaciones claras entre la imagen original y la imagen ya encriptada.

Este proyecto consiste en la implementación de un criptosistema propuesto en un artículo científico reciente [18] para imágenes médicas y naturales que utiliza su propio operador binario distinto a XOR, el cual, según el artículo y como se pretende comprobar, no cuenta con las debilidades que presenta el operador XOR como se mostrará más adelante.

1.2– Organización del documento

El resto del documento se estructura en las siguientes partes:

- **Conceptualización y planificación** (Capítulos 2 y 3). Es importante seguir una metodología a la hora de realizar un proyecto, el TFG es un proyecto más y por tanto se pueden seguir unas buenas prácticas, guías o pautas propuestas por personas especializadas en la gestión de proyectos.
- **Contexto teórico** (Capítulo 4). En este capítulo se hace una pequeña introducción a algunos de los campos que se utilizan como herramienta para implementar el criptosistema.
- **Estructura del algoritmo** (Capítulo 5). Se explica cuál es la forma que tiene el algoritmo del criptosistema y se detalla cada parte del mismo.
- **Experiencias y dificultades durante el desarrollo** (Lecciones aprendidas). Durante el desarrollo del proyecto aparecen distintos eventos que dificultan o ayudan al progreso del mismo, se ha tratado de reflejar esto en el capítulo 6.
- **Resultados y análisis estadístico** (Capítulo 7). Los resultados finales tras implementar por completo el criptosistema, también se obtienen algunos valores estadísticos que ayudan a comprobar si el criptosistema funciona como debe. Previo a esto se incluye un estudio de la complejidad temporal.
- **Mejoras y futuras vías de progreso** (Capítulo 8). Tras acabar con la implementación, pueden surgir ideas nuevas para mejorar aún más el criptosistema, o se abre una nueva vía a explorar para crear otro criptosistema distinto, una investigación sobre algún tema, etc.

- **Conclusiones.** Se recogen las conclusiones finales tras realizar el TFG.
- **Apéndices y referencias.** Al final del documento se pueden encontrar los apéndices y las distintas fuentes de conocimiento que se han consultado a la hora de realizar el proyecto.

Como anotación extra, se quiere destacar que los capítulos 4 y 5 son la solución presentada durante el trabajo.

CAPÍTULO 2

Conceptualización

Durante los próximos dos capítulos se pretende enfocar el Trabajo de Fin de Grado como un proyecto informático. Esto es, seguir una metodología de gestión de proyectos que permita a quienes trabajan en el proyecto ser más eficientes, es decir, se podrán prever gastos innecesarios de tiempo, dinero o de cualquier tipo y, además, se estandariza el proyecto de modo que en caso de que un profesional, o el propio autor, que decida seguir con el proyecto en un futuro, pueda hacerlo con menos esfuerzo.

2.1– Metodología

Como es de esperar de un TFG, los conocimientos adquiridos durante los cuatro años de carrera se van a reflejar constantemente durante todo el proyecto, es por esto que se utilizaran herramientas usadas en esos años como lenguajes de programación mejor aprendidos, técnicas de algoritmia, etc. Pero lo principal ahora es destacar que se utilizaran en gran medida, los conocimientos adquiridos en la asignatura de Planificación y Gestión de Proyectos Informáticos (PGPI), para seguir una metodología concreta de gestión de proyectos. En esta asignatura se enseña como su propio nombre indica, todo lo necesario para planificar y gestionar un proyecto, y para ello se sigue la guía de PMBOK [20], la cual presenta estándares y buenas prácticas para la gestión de proyectos informáticos.

Por otra parte, el seguimiento de esta guía o metodología, no será absolutamente estricta, cada proyecto tiene su propia naturaleza, y en el caso de este TFG, el cual tiene una finalidad didáctica para el alumno que lo realiza y de investigación, tiene algunas diferencias respecto a un proyecto informático "normal". Por poner un ejemplo antes de entrar en materia, al hablar de costes, no tiene sentido estimar lo que cuesta que una persona realice el trabajo que está realizando el alumno que realiza el TFG ya que realmente nadie va a cobrar por dicho trabajo más allá de una nota numérica, y los costes del proyecto en sí, serían los de las máquinas para programar, matrícula universitaria, etc. Son costes que están pagados ya. (Una posible alteración de esto se trata en el apartado de plan de riesgos).

2.1.1. Antecedentes

Lo primero a tener en cuenta como en todo proyecto es identificar los clientes, el caso del Trabajo de Fin de Grado es, en opinión propia, un proyecto con una característica especial ya que

el cliente es el Tutor del Trabajo de Fin de Grado y el tribunal al que se presenta dicho trabajo, esto es así debido a que ambos son personas u organizaciones que aprobarán el resultado del proyecto.

Para identificar a los patrocinadores tenemos en cuenta a todo aquel que aporta apoyo y recursos al proyecto, y en este caso encontramos una vez más al Tutor del Trabajo de Fin de Grado ya que proporciona apoyo al alumno mediante consejos y guías, también es posible que proporcione algunos recursos importantes como podría ser algún «dataset» para la implementación del criptosistema, el departamento de Matemática Aplicada, y encontramos también a la familia.

Por otro lado, el alumno es el jefe de proyecto ya que es quien gestiona y planifica todo el trabajo con la ayuda de los consejos de su Tutor.

El principal proveedor en este caso es la Universidad de Sevilla ya que mediante un “contrato”, en este caso la matrícula universitaria, se obtienen distintos recursos y la validez y reconocimiento del trabajo realizado durante la carrera, incluyendo los 12 créditos de los que consta la asignatura del Trabajo de Fin de Grado (300 horas).

Los interesados son todos aquellos que se ven afectados por el proyecto como la propia Universidad de Sevilla, entorno cercano del alumno que realiza el proyecto... En cuanto a los usuarios podemos nos encontramos a quienes estén interesados en salvaguardar la integridad de imágenes médicas, DICOM, o incluso imágenes naturales, por ejemplo, un hospital.

2.1.2. Estudio de Viabilidad

Desde el punto de vista técnico, se utiliza el lenguaje de programación Python con el fin de implementar el criptosistema con las librerías necesarias, también se utilizará un dataset descargando algún dataset público de internet. En concreto se utilizará una colección de imágenes médicas de The Cancer Imaging Archive (TCIA), concretando aún más, se utilizará la colección COVID-19-AR de 2020 [2]. Se tratará con más detalle en el apartado 6.4.

Desde un punto de vista económico, el proyecto no supone ningún gasto monetario, solo se invierte tiempo para adquirir los conocimientos necesarios y para la implementación del criptosistema. Esto podría variar en función de los riesgos, puesto que en caso de pérdida de algún material de trabajo habría que reponerlo suponiendo, ya sí, un coste monetario.

Sin embargo, no se esperan demasiados riesgos a la hora de realizar el proyecto y los riesgos que podrían esperarse son los siguientes:

- Retraso del proyecto por enfermedad, o lesión grave que pueda sufrir el alumno.
- Dificultad de comprensión del artículo en el que se basa el proyecto.
- Dificultad de la implementación del criptosistema.
- Falta de capacidad de computación durante la implementación del criptosistema.
- Pérdida de material necesario para programar y compilar el programa.

2.2– Definición

Una vez están claros los antecedentes y se ha realizado el estudio de viabilidad, y se ha situado el contexto, podemos detallar los requisitos, objetivos y, en resumen, qué se pretende realizar.

2.2.1. Análisis de requisitos

El cliente (tutor) ha propuesto ciertos requisitos para este Trabajo de Fin de Grado:

- Sobre el artículo a estudiar:
 - Alto nivel de comprensión de dicho artículo al finalizar el proyecto.
 - Las herramientas matemáticas que se utilizan deben quedar bien documentadas (secuencias caóticas, computación basada en ADN, ...).
- Sobre el criptosistema a implementar:
 - Debe ser capaz de encriptar imágenes DICOM.
 - Debe ser capaz de encriptar imágenes naturales.
 - Comprobar la robustez del algoritmo de cifrado.
 - Hacer eficiente en la máxima medida que sea posible el algoritmo.
 - Comprobación experimental de lo descrito en el artículo.

Por otra parte, se deben completar las 300 horas de trabajo definidas por los créditos de los que consta la asignatura del Trabajo de Fin de Grado, documentando debidamente cómo y en qué se han invertido dichas horas para mostrar el tiempo invertido al final del proyecto.

2.2.2. Objetivos, alcance, calidad

El objetivo final de este proyecto consiste en implementar un criptosistema para imágenes médicas basado en un nuevo (2021) operador binario [18]. A su vez, el criptosistema utiliza algunas herramientas matemáticas como funciones caóticas y computación basada en ADN que deben ser bien documentadas, todo ello tomando como punto de partida un artículo internacional reciente[referencia]. Se espera también comprobar empíricamente los resultados que aparecen en el documento.

En resumen, el principal objetivo consiste en superar los requisitos propuestos por el cliente y además de maximizar la comprensión del artículo sobre el que se basará el criptosistema, además, obtener finalmente un software que pueda ser utilizado por personal médico capaz de encriptar las imágenes DICOM.

En cuanto al alcance se debe obtener un software capaz de cifrar imágenes naturales y médicas como indican los requisitos y los objetivos, tras ello se debe explicar adecuadamente el funcionamiento del criptosistema documentando todos los posibles aspectos implementados en código.

Es muy probable que durante el proyecto se encuentren oportunidades como posibles adiciones al proyecto que mejorarían el criptosistema, se descubran otras aplicaciones a las herramientas utilizadas que pueden ser documentadas, etc. Discutir si se llevarán a cabo estas actividades “extra” deberán ser estudiadas durante la ejecución del proyecto por el propio alumno realizando un nuevo estudio tanto de coste y del cronograma como de los riesgos, en caso de que la adición al proyecto conlleve un gran coste en tiempo u ofrezca una ampliación de contenido, se organizará una reunión con el cliente (tutor) para discutir si es posible llevar a cabo esa adición y los términos en los que debe llevarse a cabo.

La posibilidad de que sucedan estos añadidos al proyecto surgen de la compleja y amplia naturaleza del tema tratado en el trabajo y del tiempo disponible para realizarlo, desde algo antes de febrero hasta mitad de mayo tenemos cerca de tres meses y medio, contando con que se invierten cinco horas de lunes a viernes, obtenemos que se van a invertir en el proyecto veinte-cinco horas

semanales, contando con que cada mes tiene más de 4 semanas, se invierten más de 100 horas al mes, y añadiendo los tres meses y medio obtenemos más de 350 horas de trabajo, esto a pesar de que el mínimo exigido sean 300 horas.

Una vez presentada una estimación a grandes rasgos del número de horas, hay que destacar también que se pretende maximizar la calidad, superando los objetivos iniciales de ser posible. Por lo general un proyecto que llega a la fase de cierre habiendo introducido mayor cantidad de funcionalidades de las indicadas en los requisitos es un proyecto insatisfactorio, pero en este caso particular de proyecto se deja abierta la posibilidad de introducir nuevos aspectos que se crean necesarios.

En este proyecto es indispensable realizar una memoria, y en cuanto a su redacción y presentación se debe:

1. Componer una memoria perfectamente legible, lo cual implica que los contenidos sean sintácticamente correctos, la ausencia de errores ortográficos y un uso adecuado de los signos de puntuación.
2. Se deben incluir los elementos típicos de una memoria como son: una portada, un índice, páginas enumeradas, justificación de márgenes y como es de esperar, hay que cumplir con el estándar establecido en la normativa de trabajo de fin de grado.
3. Incluir una numeración correcta y organizada de las figuras, esquemas, diagramas, etc.
4. Por supuesto es importante señalar las referencias que sean necesarias citando las mismas de forma adecuada, esto incluye bibliografía, sitios de internet, etc.

Respecto al contenido y calidad de la memoria:

1. Se indicará que se utilizan recursos específicos para realizar el trabajo dejando constancia de la capacidad de absorber conceptos y al mismo tiempo adaptar estos conceptos aprendidos para aplicarlos en el proyecto.
2. Se debe justificar la relevancia del proyecto aportando una pequeña introducción sobre algunas técnicas criptográficas que se utilizan en la actualidad.
3. Informar al lector de la memoria sobre los objetivos que se proponen en el TFG.
4. Relacionado con la necesidad de mantener coherente el contenido de la memoria nos encontramos la necesidad de exponer las aportaciones realizadas en el trabajo sin incurrir en el desorden o en un exceso de florituras sin aporte. Esto se aplica a las gráficas, figuras, diagramas, etc.
5. Debe quedar claro en cada punto de la memoria los aspectos del proyecto que se han resuelto y los que quedan por resolver.
6. Redactar un pequeño capítulo o anexo donde se indique el método de uso del criptosistema en caso de que se consiga desarrollar el mismo como una herramienta.

Tras dejar claros los apartados previamente tratados, se procede a firmar un Acta de Constitución, los firmantes son, el alumno que realiza el TFG y el tutor del alumno.

CAPÍTULO 3

Planificación

Prestando atención ahora a la planificación, tenemos que tratar el desglose de tareas que deberán ser realizadas teniendo en cuenta también cuáles serán los hitos que supondrán progresos destacables del proyecto que se mostrarán al cliente (el tutor) para poder valorar el progreso del proyecto.

También será útil plantear un cronograma en el cual se situarán las distintas tareas descritas anteriormente, indicando también así, el orden de realización de las tareas, cuales son compatibles y paralelizables, etc. Gracias a esto se podrá obtener el camino crítico y por tanto atender a las tareas que, de retrasarse, retrasarían el proyecto en su totalidad.

Hay que tener en cuenta también una estimación de costes en el proyecto, aunque como se dijo en un principio en la fase de inicio, es muy posible que no haya costes económicos más allá de los que pueda haber en caso de perder la máquina en la que se trabajara el proyecto ya que esto supondría una sustitución de la misma (este sería un ejemplo de posible gestión de riesgo).

Es en este apartado donde se deben considerar los distintos planes de gestión, tanto los de gestión de calidad, de recursos humanos y de adquisiciones como de interesados y comunicaciones.

3.1– Análisis temporal y costes de desarrollo

En este apartado se tratarán las distintas tareas que deberán ser realizadas y su posición en el cronograma.

3.1.1. Actividades, tareas e hitos

Para realizar una estimación del costo y el tiempo del proyecto se subdivide el proyecto en fases, estas fases son:

- Planificación. (puliendo lo que queda de planificación)
- Comprensión y asimilación del artículo.
- Desarrollo del criptosistema.
- Escritura de la documentación/memoria.

A su vez estas fases del proyecto tendrán distintas tareas y actividades, este proceso se ha realizado en “sucio” y tras obtener un desglose de las tareas necesarias se ha utilizado la herramienta Excel para poder visualizar todo el coste del tiempo en su totalidad de forma rápida. Las fases del proyecto aparecen en verde, las actividades las cuales constan de subtareas aparecen en azul y las tareas se muestran en blanco. Las horas dedicadas a las fases son el resultado de la suma de horas dedicadas a sus actividades y tareas, las horas dedicadas a las actividades son la suma de las horas dedicadas a sus subtareas y el resultado total de horas es el sumatorio de las horas dedicadas a todas las tareas (representadas con el color blanco).

En la versión en Latex de este documento se muestra sin colores.

Cuadro 3.1: Actividades, tareas e hitos

No.	Tarea	Horas
1	Plan de riesgos, calidad, comunicaciones, ...	3
2	<i>Comprensión y asimilación del artículo</i>	93
3	Lectura profunda de lo que se propone en el artículo	40
4	Operador binario. (Lectura)	12
5	Computación basada en ADN (Lectura)	17
6	Series caóticas (Lectura)	11
7	Ampliación de conocimiento profundizando en referencias	8
8	Esquematización de conceptos	5
9	<i>Desarrollo del criptosistema</i>	195
10	Algoritmos	141
11	Obtener vector inicial (Algoritmo)	2
12	Obtener el mapa de sustituciones de ADN (Algoritmo)	13
13	Mezclado de filas (Algoritmo)	8
14	Modificación ACM (Algoritmo)	15
15	Mezclado (Algoritmo)	9
16	Codificación ADN (Algoritmo)	27
.	Hito. Control de progreso	0
17	Sustitución (Algoritmo)	15
18	Decodificación ADN (Algoritmo)	23
19	Encriptado (Algoritmo)	29
.	Hito. Control de progreso	0
20	Comprobación de eficiencia	9
21	Comprobación de resultados	15
22	Comprobación de integridad de imagen tras cifrado y descifrado	12
.	Hito. Control de progreso	0
23	Tests de comprobación de confidencialidad	18
24	<i>Escritura de la documentación/memoria</i>	56
25	Explicación de técnicas utilizadas	9
26	Documentar el flujo del algoritmo	5
27	Progreso de desarrollo de algoritmos	22
28	Estudio de complejidad y eficiencia	8
29	Muestras de resultados con imágenes	6
30	Muestras de tests sobre confidencialidad	6
.	Hito. Revisión final	0
.	TOTAL =	307

Hay que destacar que esta es una estimación de lo que conllevaría como mínimo cada tarea, pero, como se ha indicado en el alcance del proyecto, hay posibilidad de añadir nuevas tareas o

subtareas al proyecto durante la ejecución del mismo.

3.1.2. Cronograma

Para tener una previsión del tiempo que supondrá realizar este proyecto se ha utilizado un diagrama de Gantt teniendo en cuenta ciertas cosas:

- Se trabajará en el proyecto de lunes a viernes, un día de trabajo equivale a 4 horas. (Ya que se dedicarán otras 3 horas para otras asignaturas, y los fines de semana para recuperar retrasos y descansar)
- Cuando dos tareas se superpongan, se realizarán sesiones de 12 horas de trabajo ese día para evitar en la medida de los posibles los retrasos. (Esto es posible porque habrá pocas veces donde dos tareas se solapen)
- En principio, solo se contará con un recurso (el propio alumno que realiza el trabajo), por lo que el camino crítico está definido por todas las tareas del proyecto. Esto es así ya que la mayor parte de la realización de las tareas será secuencial.

A continuación, se muestran las tareas correspondientes a planificación y comprensión del artículo.

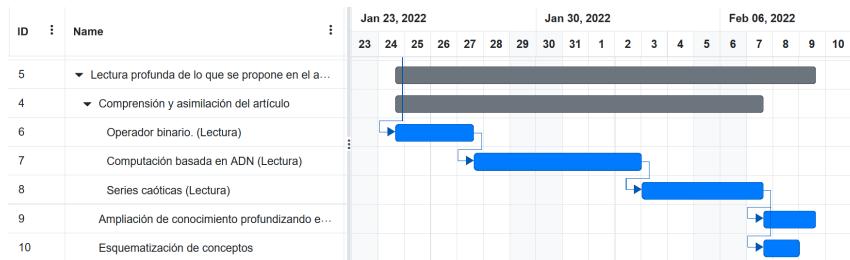


Figura 3.1: Gantt (planificación y comprensión)

A parte de en las tareas de “Ampliación de conocimiento” y “Esquematización de conceptos”, no se producen más solapamientos ya que solo se realizan prácticamente todas las tareas de forma secuencial por la limitación de recursos y para no sobreasignar el recurso disponible en exceso (el alumno). En caso de producirse solapamientos se hubiesen producido en las fases de test, en caso de poder contar con una segunda máquina para ejecutar código, se utilizaría esta para ejecutar código en paralelo asignando así cada máquina a una de las tareas. El resultado en el diagrama se vería así.



Figura 3.2: Gantt (Ejemplo de solapamiento)

Se muestra además una vista de la duración de las distintas fases.

Para realizar los diagramas de Gantt se ha buscado una alternativa gratuita a MSProject, la página en cuestión es "www.onlinegantt.com". [4].

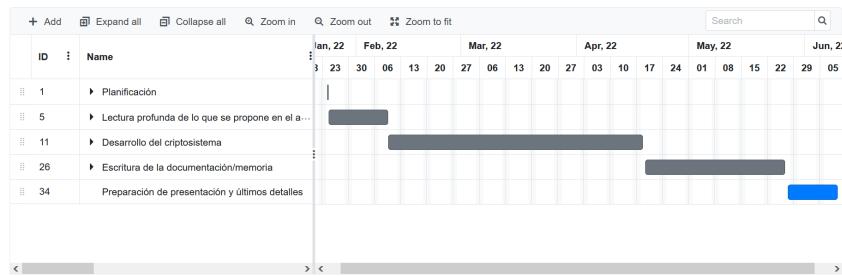


Figura 3.3: Gantt (Fases)

Como puede comprobarse, el plazo de finalización del proyecto está adelantado a la fecha de entrega del proyecto, esto es así porque restaría por añadir una última actividad que consista en revisión de todo el contenido, y posibles añadidos que surjan durante la implementación del criptosistema.

Los recursos con los que se cuenta son, el alumno que realiza el TFG, y la máquina que posee para programar y realizar el documento. Podría ser posible la adición de un segundo recurso material (otro PC), pero como no es seguro no se contabiliza para el cronograma por el momento.

A parte, no se contabilizan días festivos de descanso a parte de sábado y domingo, que como ya se ha comentado, también se especifica por último que la idea inicial es cumplir con el cronograma, aunque parezca que es exigente, es posible cumplirlo adecuadamente, una vez cumplido, y haber finalizado la fase de ejecución seguimiento y control, se procedería a utilizar ratos libres para ultimar detalles del trabajo, depurar, etc.

A continuación se muestra en formato de tabla un resumen del cronograma, si es posible, se entregará al final de este trabajo el documento original en Excel 3.2.

3.1.3. Estimación de costes

En un principio y sin que haya nada que demuestre lo contrario, en este proyecto no se presentan costes que asumir, puesto que todo el está realizado por una sola persona (alumno) y esto no supondrá coste alguno.

Sin embargo, es posible que durante la ejecución del proyecto se produzcan problemas por los cuales habrá que pagar el precio necesario, esto puede verse a continuación en el plan de riesgos.

3.2– Plan de riesgos

A continuación, se deben identificar los posibles riesgos (tanto amenazas como oportunidades) que pueden presentarse durante la ejecución del proyecto, y tras esto, se crearán los planes de contingencia necesarios para gestionar esos riesgos más adelante.

3.2.1. Identificación de riesgos

A continuación, se indican los posibles riesgos del proyecto, tanto amenazas como oportunidades, para cada riesgo se indica la probabilidad de que suceda en una escala del 1 (menor posibilidad de que suceda) al 10 (mayor posibilidad de que suceda):

- Amenazas:

Cuadro 3.2: Resumen cronograma

ID	Name	Start Date	End Date	Duration
1	Planificación	Jan 24, 2022	Jan 24, 2022	0.75 days
2	Plan de riesgos, calidad, comunicaciones, ...	Jan 24, 2022	Jan 24, 2022	0.75 days
5	Lectura profunda del artículo	Jan 24, 2022	Feb 09, 2022	12 days
4	Comprensión y asimilación del artículo	Jan 24, 2022	Feb 07, 2022	10 days
6	Operador binario. (Lectura)	Jan 24, 2022	Jan 27, 2022	3 days
7	Computación basada en ADN (Lectura)	Jan 27, 2022	Feb 02, 2022	4.25 days
8	Series caóticas (Lectura)	Feb 03, 2022	Feb 07, 2022	2.75 days
9	Ampliación de conocimiento	Feb 07, 2022	Feb 09, 2022	2 days
10	Esquematización de conceptos	Feb 07, 2022	Feb 08, 2022	1.25 days
11	Desarrollo del criptosistema	Feb 09, 2022	Apr 18, 2022	48.25 days
12	Algoritmos	Feb 09, 2022	Mar 30, 2022	35.25 days
13	Obtener vector inicial (Algoritmo)	Feb 09, 2022	Feb 10, 2022	0.5 days
14	Mapa de sustitución ADN (Algoritmo)	Feb 10, 2022	Feb 15, 2022	3.25 days
15	Mezclado de filas (Algoritmo)	Feb 15, 2022	Feb 17, 2022	2 days
16	Modificación ACM (Algoritmo)	Feb 17, 2022	Feb 23, 2022	3.75 days
17	Mezclado (Algoritmo)	Feb 23, 2022	Feb 25, 2022	2.25 days
18	Codificación ADN (Algoritmo)	Feb 25, 2022	Mar 08, 2022	6.75 days
19	Sustitución (Algoritmo)	Mar 08, 2022	Mar 11, 2022	3.75 days
20	Decodificación ADN (Algoritmo)	Mar 14, 2022	Mar 21, 2022	5.75 days
21	Encriptado (Algoritmo)	Mar 21, 2022	Mar 30, 2022	7.25 days
22	Comprobación de eficiencia	Mar 31, 2022	Apr 04, 2022	2.25 days
23	Comprobación de resultados	Apr 04, 2022	Apr 07, 2022	3.25 days
24	Comprobación de integridad tras cifrado	Apr 07, 2022	Apr 12, 2022	3 days
25	Tests de comprobación de confidencialidad	Apr 12, 2022	Apr 18, 2022	4.5 days
26	Escritura de la documentación/memoria	Apr 19, 2022	May 26, 2022	28 days
27	Explicación de técnicas utilizadas	Apr 19, 2022	Apr 21, 2022	2.25 days
28	Documentar el flujo del algoritmo	Apr 21, 2022	Apr 22, 2022	1.25 days
29	Progreso de desarrollo de algoritmos	Apr 22, 2022	Apr 29, 2022	5.5 days
30	Estudio de complejidad y eficiencia	May 02, 2022	May 03, 2022	2 days
31	Muestras de resultados con imágenes	May 04, 2022	May 05, 2022	1.5 days
32	Muestras de tests sobre confidencialidad	May 05, 2022	May 06, 2022	1.5 days
33	Depuración de código	May 09, 2022	May 26, 2022	14 days
34	Preparación de presentación, últimos detalles	May 27, 2022	Jun 09, 2022	10 days

- Pérdida de equipo necesario para desarrollar el criptosistema y/o la creación de la documentación del TFG. Esto puede deberse a una perdida literal del mismo, lo cual es muy poco probable, o a la destrucción material de las máquinas por accidente. (Riesgo bajo, 1)
- Retraso del proyecto, en general por una posible enfermedad o lesión grave que pueda sufrir el alumno. (Riesgo bajo, 2)
- Dificultad de comprensión del artículo en el que se basa el proyecto, lo cual también puede llegar a causar algún retraso en el cronograma. (Riesgo medio, 6)
- La posibilidad de encontrar complicaciones para desarrollar el criptosistema debido a que no se encuentren herramientas de programación adecuadas, por ejemplo, algoritmos para realizar ciertas operaciones sin que ello incurra en un exceso de complejidad. (Riesgo alto, 8)
- Falta de capacidad de computación durante la implementación del criptosistema, por

ejemplo, aún no está claro si el sistema deberá procesar múltiples imágenes, por ejemplo, cifrar múltiples imágenes a la vez. (Riesgo medio, 6)

- **Oportunidades:**

- La adición de un nuevo recurso, una segunda máquina para compilar y ejecutar código. (Riesgo alto, 7)
- Mejoras que pueden añadirse al proyecto. (Riesgo alto, 8)

3.2.2. Planes de contingencia

Para cada caso la forma de actuar sería la que se muestra a continuación:

1. Se tendría que proceder a el posible segundo recurso del que se ha hablado anteriormente, si no fuese posible tampoco acceder a él, habría que adquirir nuevo equipo.
2. Se perderán días de trabajo en el peor de los casos, por lo general, no se pone en riesgo la salud del alumno que realiza el TFG, pero en caso de suceder no hay más remedio que tratar de recuperarse lo más rápido posible.
3. En caso de que las búsquedas por internet no den resultado, se acudiría a expertos para tratar de comprender mejor el artículo, por ejemplo, el tutor del TFG y otras personas con conocimientos profundos en matemáticas. Debido a que ha sido requisito en muchas asignaturas de la carrera el hecho de enfrentarse frontalmente con artículos de investigación, ya se cuenta con cierta experiencia para sortear problemas de este tipo.
4. Si tras buscar en internet no se encuentra ninguna solución, se buscaría mas ayuda consultando con el tutor del TFG o con personas habituadas a programar algoritmos complejos.
5. Se trataría de conseguir el segundo recurso del que se ha hablado anteriormente (PC) para paralelizar ejecución. Se debe tratar la limpieza del código y hacer al mismo lo más eficiente posible en términos de complejidad.
6. No sería complicado y podría aportar beneficios por lo que es muy posible que se trate de conseguir.
7. Una vez detectada dicha mejora, es importante realizar un estudio del cronograma, consultar con los clientes y medir bien los tiempos que ocuparía dicha actualización o mejora, a no ser que sea obligatorio introducir esa mejora en el tiempo designado para crear el proyecto definido en un inicio, se incluirá esta mejora después de terminar el proyecto completo. (Esto es tras la fecha en la que se prevé que acabará el proyecto)

3.3– Planes de calidad

3.3.1. Indicadores

- Conclusiones obtenidas durante las reuniones que se proponen llevar a cabo como hitos.
- Contraste con los resultados que se indican en el artículo.
- Comparación de resultados con otros criptosistemas en ámbitos similares.
- Legibilidad y comprensión a primera vista del programa desarrollado.

- Comprobación de los objetivos propuestos inicialmente.
- Estudio de complejidad de los algoritmos.

3.3.2. Plan de mejora

Se debe prestar especial atención a las conclusiones obtenidas tras cada reunión, bien sea de reuniones de hitos o de cualquier otra índole.

3.4– Planes de comunicaciones

Para el plan de comunicaciones entre los actores del proyecto se propone:

Reuniones de seguimiento entre el alumno que realiza el TFG y el tutor del alumno para discutir novedades que puedan ser introducidas al proyecto en caso de haber detectado alguna, comprobar si el curso que lleva el proyecto es correcto, detectar errores, etc.

Reuniones con personas que tengan conocimiento sobre las áreas de las que trata el proyecto con el fin de ampliar conocimiento y generar posteriormente una documentación más sólida.

CAPÍTULO 4

Contexto teórico

Previo al desarrollo del criptosistema, es necesario realizar un estudio teórico de los elementos que aparecen en el artículo en el que se basa este TFG [18]. Con esto se quiere definir las distintas herramientas matemáticas principales de modo que una persona que lea este documento, pueda acudir a este capítulo en caso de no conocer dichas herramientas. Por otro lado, también se presentará el operador binario propuesto por el artículo que puede ser usado en criptosistemas enfocados en imágenes de 8 o 16 bits, y en codificación basada en ADN.

Para realizar este capítulo se ha recogido el material teórico más relevante de los artículos que más han influido en la creación de este proyecto. Se pretende que este TFG sea lo más autocontenido posible y por ello, se tratan de explicar a continuación los conceptos necesarios para la construcción del criptosistema. Se ha tratado de comunicar este contenido teórico sin utilizar un lenguaje puramente matemático. Tampoco se pretende que este TFG sea excesivamente informal y por ello son necesarias algunas demostraciones matemáticas.

4.1– Operador binario \otimes

Teniendo en cuenta nuestro contexto, el cifrado de imágenes, podremos encontrarnos con tres tipos de codificación de dichas imágenes.

La primera es la codificación de 8-bits, las imágenes naturales, si nuestra imagen está en escala de grises, tenemos una matriz 2D donde cada pixel representa un tono de gris, estos píxeles están compuestos de una cadena de 8 elementos y estos elementos son el 1 y el 0. De forma paralela, podemos decir que tenemos un lenguaje, y un lenguaje se compone de palabras que, a su vez, se componen de un conjunto de símbolos que en nuestro caso es el conjunto $\{0,1\}$. Teniendo en cuenta que nos encontramos en la codificación de 8-bits, nuestras palabras serán de tamaño 8, es decir, una cadena de ocho unos y ceros, los cuales están representando en realidad números decimales del 0 al $2^8 - 1 = 255$ mediante números binarios.

La segunda codificación posible que nos vamos a encontrar es la de 16-bits que utilizan las imágenes DICOM, de forma similar al caso anterior, tenemos palabras que se componen de ciertos símbolos pertenecientes al conjunto $\{0,1\}$. La diferencia en este caso es el tamaño de cada palabra, que sigue siendo fijo pero ahora es de tamaño 16, es decir, una cadena de 16 símbolos. Cada palabra representa un número decimal del 0 al $2^{16} - 1 = 65535$.

El tercer caso es el de tener la imagen almacenada como nucleótidos de ADN. Solo es necesario saber por el momento que podemos codificar una imagen usando el conjunto de símbolos

$\{A,C,T,G\}$ a los que podemos mapear como $\{A,C,T,G\} \rightarrow \{0,1,2,3\}$. Lo cual será útil para la implementación del criptosistema.

Podemos considerar que cada uno de estos tres casos es un lenguaje distinto (cadena de simbolos) para comunicar el contenido de una imagen. Cada lenguaje tiene su propio alfabeto, por ejemplo en el caso del lenguaje natural que usamos a diario en países de Europa occidental tenemos el conjunto de símbolos $\{A,B,\dots,Y,Z\}$. En el caso de querer comunicar el contenido de una imagen en un lenguaje basado en secuencias de ADN, el conjunto de símbolos es $\{A,C,T,G\}$. El lenguaje basado en 8 bits contiene el conjunto de símbolos $\{0,1,\dots,255\}$ y si está basado en 16 bits el conjunto es $\{0,1,\dots,65535\}$.

El objetivo del artículo [18] es demostrar que el operador binario 4.1 que proponen es útil para operar con los símbolos múltiples lenguajes S que pueden describir el contenido de una imagen. Un detalle extra muy importante para que el operador binario funcione correctamente, el número de símbolos existentes en el lenguaje debe ser un primo menos uno. Por ejemplo, para el lenguaje basado en ADN tenemos 4 símbolos, que es justo $5 - 1$, siendo 5 un primo. Lo mismo sucede con $256 = 257 - 1$ y $65536 = 65537 - 1$.

$$a \otimes b = \{(a + 1) \times (b + 1) \pmod p\} - 1; a, b \in S \quad (4.1)$$

Este operador forma un grupo Abierno en el conjunto $Z_p - \{p-1\} = S$, lo cual significa que cumple:

- La propiedad conmutativa
- La propiedad asociativa
- Existe un inverso para los elementos de S (símbolos)
- Existe un elemento identidad o neutro
- Es cerrado sobre si mismo

Es importante comprobar que el operador cumple las condiciones que se han comentado en la lista anterior.

4.1.1. Se cumple la propiedad conmutativa

Queremos comprobar que al realizar $a \otimes b$ obtenemos el mismo resultado que si realizamos $b \otimes a$. Es fácil comprobar que al cambiar el orden de a y de b en $a \otimes b$ se obtiene el mismo resultado ya que cambiamos de orden los factores de una multiplicación, operación conmutativa.

$$\begin{aligned} a \otimes b &= \{(a + 1) \times (b + 1) \pmod p\} - 1 \\ &= \{(b + 1) \times (a + 1) \pmod p\} - 1 \\ &= b \otimes a \end{aligned}$$

4.1.2. Se cumple la propiedad asociativa

Queremos comprobar que al realizar la operación $a \otimes (b \otimes c)$ obtenemos el mismo resultado que si realizamos $(a \otimes b) \otimes c$. Podemos demostrarlo desarrollando $a \otimes (b \otimes c)$ tratando de llegar al resultado que queremos, $a \otimes b) \otimes c$.

$$a \otimes (b \otimes c)$$

$$\begin{aligned}
&= (a + 1) \times [\{(b+1) \times (c+1) \text{ mod } p\} - 1 + 1] \text{ mod } p - 1 \\
&= (a + 1) \times \{(b+1) \times (c+1) \text{ mod } p\} \text{ mod } p - 1 \\
&= (a + 1) \times (b+1) \times (c+1) \text{ mod } p - 1 \\
&= [\{(a + 1) \times (b + 1)\} \times (c+1)] \text{ mod } p - 1 \\
&= [\{(a + 1) \times (b + 1) \text{ mod } p\} \times (c+1)] \text{ mod } p - 1 \\
&= [\{(a + 1) \times (b + 1) \text{ mod } p\} - 1 + 1] \times (c+1) \text{ mod } p - 1 \\
&= (a \otimes b) \otimes c
\end{aligned}$$

Por tanto $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.

4.1.3. Existe el elemento identidad y es el 0

Queremos comprobar si existe algún valor del conjunto de símbolos S que al operarlo con otro valor cualquiera $a \in S$, se obtenga de nuevo a. Suponemos inicialmente el caso en el que S es el conjunto más pequeño que podemos construir, es decir $S = \{0\}$, solo incluimos el 0 y nuestro $p = 2$, en este caso es fácil comprobar que dado un $a \in S$ (es decir, $a=0$):

$$0 \otimes a = a \otimes 0 = 0 = a$$

Para cualquier otro caso de S, tenemos lo siguiente:

$$\begin{aligned}
&0 \otimes a \\
&= \{(0 + 1) \times (a + 1) \text{ mod } p\} - 1 \\
&= \{1 \times (a + 1) \text{ mod } p\} - 1 \\
&= \{(a + 1) \text{ mod } p\} - 1 \\
&= \{(a \text{ mod } p) + (1 \text{ mod } p)\} \text{ mod } p - 1 \\
&= \{(a \text{ mod } p) + 1\} \text{ mod } p - 1 \\
&= a \text{ mod } p + 1 - 1 \\
&= a \text{ mod } p = a
\end{aligned}$$

Si contemos con que ya hemos demostrado que se cumple la propiedad commutativa, podemos aplicar a la demostración anterior que $0 \otimes a = a \otimes 0 = a$.

4.1.4. Existe un elemento inverso

Buscamos la existencia de un valor a^{-1} que pertenece al conjunto de símbolos S, este valor es el elemento inverso de a, al operar un elemento con su inverso obtenemos el neutro, es decir, 0. Empezamos nuestra demostración asumiendo que existe este valor a^{-1} y seguimos con 4.2 tenemos $(a + 1) \times (a^{-1} + 1) \equiv 1 \pmod{p}$.

$$\{(a + 1) \times (a^{-1} + 1) \pmod{p}\} - 1 = 0 \implies (a + 1) \times (a^{-1} + 1) \equiv 1 \pmod{p} \quad (4.2)$$

Por tanto, si definimos un $x = (a + 1)$ y un $y = (b + 1)$, tenemos que hay dos números, que multiplicados, son congruentes con 1 mod p. 4.3

$$x \times y \equiv 1 \pmod{p}. \quad (4.3)$$

Si nos apartamos momentaneamente y observamos el pequeño teorema de Fermat, este nos dice que para un primo p , y un entero a que no es divisible por p , tenemos que $a^{p-1} - 1$ es un entero múltiplo de p . Es decir 4.4.

$$a^{p-1} \equiv 1 \pmod{p} \quad (4.4)$$

Tenemos que x es menor que p y mayor que 0, por tanto, x no es divisible por p , así que podemos aplicar en este caso el pequeño teorema de Fermat del siguiente modo 4.5.

$$x^{p-1} \equiv 1 \pmod{p} = x \times x^{p-2} \equiv 1 \pmod{p} \quad (4.5)$$

Ahora uniendo ambos resultados tenemos 4.6.

$$y = x^{p-2} \pmod{p} \implies (a^{-1} + 1) = (a + 1)^{p-2} \pmod{p} \implies a^{-1} = \{(a + 1)^{p-2} \pmod{p}\} - 1 \quad (4.6)$$

Ahora que tenemos definido el inverso queda comprobar que este pertenezca al conjunto S . Sabemos que $(a^{-1} + 1) = (a + 1)^{p-2} \pmod{p}$, y por la naturaleza de la aritmética modular, $(a^{-1} + 1) \in \{1, \dots, p - 1\}$, por tanto $(a^{-1}) \in \{1, \dots, p - 2 = n - 1\} = S$

4.1.5. El operador \otimes no es lineal

Queremos demostrar que $r(a \otimes b) \neq ra \otimes rb$. Para ello empezamos a desarrollar ambas partes y comprobamos que se obtienen resultados distintos.

$$\begin{aligned} r(a \otimes b) &= r[\{(a+1) \times (b+1) \pmod{p}\} - 1] \\ &= \{(rab + ra + rb + r) \pmod{p}\} - r \\ &= r(rab + a + b) \pmod{p} \end{aligned}$$

Por otro lado tenemos que:

$$\begin{aligned} (ra \otimes rb) &= \{(ra + 1) \times (rb + 1) \pmod{p}\} - 1 \\ &= (r^2ab + ra + rb + 1) \pmod{p} - 1 \\ &= r(rab + a + b) \pmod{p} \neq r(a \otimes b) \end{aligned}$$

4.1.6. El conjunto S está cerrado bajo \otimes

Con cerrdo sobre si mismo se quiere decir que si operamos dos elementos del conjunto S , obtendremos otro elemento del conjunto S .

Sabemos que $S = \{0, 1, \dots, n-1\}$. Si realizamos la siguiente operación $z = \{(a+1) \times (b+1) \pmod{p}\}$, por la naturaleza de la aritmética modular, z está comprendida entre 0 y $p-1$. Sabemos que $p = n + 1$, por tanto el rango de valores que puede tomar z es de 0 a n . Por otro lado, para que z valga p , tenemos que $t = \{(a+1) \times (b+1)\}$ con $a, b \in S$ debe ser: 0, p , o $k \times p$. Es imposible conseguir que t sea 0 ya que para ello, o bien $(a+1)$ debe ser 0, o bien $(b+1)$ debe ser 0. No es posible que sea p ya que para ello $(a+1)$ debe ser 1 y $(b+1)$ debe ser p o viceversa, sabemos que a y b son parte del conjunto S que llega hasta $n-1$, y p es $n+1$, por tanto es imposible este caso y también, es imposible obtener un múltiplo de p (ya que seguiría siendo necesario tener a p en uno de los dos factores). Eso hace que el rango de valores de z sea de 1 a n , viendo la fórmula completa del operador binario, nos queda por añadir un -1 a z , por lo que z es un valor comprendido entre 0 y $n-1$, que es justo S .

4.2– Mapas y series caóticas

A partir de 1990, comienzan a aparecer una serie de criptosistemas basados en caos que tratan de utilizar la idea propuesta por Louis M. Pecora y Thomas L. Carroll [17] de sincronizar sistemas caóticos. Esto abre paso a un nuevo horizonte en el campo de la criptografía con la aparición de novedosos criptosistemas como el de M. S. Baptista [12] que propone cifrar texto mediante un abecedario y utilizar propiedades de los mapas caóticos para encriptar. A pesar de la gran expectativa puesta en la teoría del caos aplicada a la criptografía no se ha producido un gran avance en el campo. Sin embargo, aún se proponen múltiples criptosistemas basados en caos a nivel académico y de investigación.

Como podrá comprobarse más adelante, el criptosistema implementado está fuertemente basado en caos, su “columna vertebral” está compuesta por un mapa caótico que afecta a casi todas las partes del algoritmo. Es por ello que se debe introducir en este apartado el concepto de mapa y se realiza una pequeña introducción a la teoría del caos con el fin de crear un documento lo más autocontenido posible.

4.2.1. Mapa

Definimos un mapa como una función cuyo dominio y rango son el mismo espacio, es decir, el conjunto de posibles valores de entrada es el mismo que el conjunto de los valores de salida [14]. Es decir, $f: D \rightarrow D$.

Por ejemplo la función $f: R \rightarrow R$ definida por $f(x) = -2x^2$ es un mapa.

Si tenemos una ecuación de la forma

$$x_{(k+1)} = f(x_k) \quad \text{para } k = 0, 1, \dots \quad (4.7)$$

siendo f un mapa definido sobre un conjunto D , tenemos un sistema dinámico discreto. La forma de representar este sistema dinámico discreto es (D, f) .

Si tenemos un x_0 perteneciente a un conjunto D , se dice que la órbita positiva de X_0 bajo f es el conjunto ordenado de puntos $x_0, f(x_0), f^2(x_0), \dots$, siendo x_0 el punto inicial de la órbita, f^2 siendo la composición de funciones $f \circ f$ y f^n la composición de funciones $n-1$ veces.

Como ejemplo podemos tener la función

$$f: R \rightarrow R \quad f(x) = 3x^2$$

y teniendo $x_0 = 1$, la órbita positiva de x_0 bajo f es $3, 81, 19683, \dots$. La órbita negativa se define a partir de la posibilidad de invertir el mapa, debido a que no vamos a entrar en eso, a partir de ahora cuando se nombra una órbita, será una órbita positiva.

Nuestro mapa anterior era de una dimensión, pero podemos tener mapas de dos o más dimensiones.

De este modo se puede empezar a ver la utilidad de los mapas para crear series de elementos del conjunto D a partir de las órbitas obtenidas desde un punto semilla inicial x_0 .

4.2.2. Puntos fijos, puntos k -periódicos

Tenemos un sistema dinámico discreto (D, f) , si dado un punto X_0 de D , se tiene que $f(x_0) = x_0$, entonces estamos ante un punto fijo. Si dado otro valor inicial distinto a x_0 se alcanza en su órbita ese x_0 , entonces la órbita queda invariante a partir de ese momento. Los puntos fijos pueden ser atractivos (si para cada punto en un entorno suficientemente cercano al punto fijo, tras cada

iteración de la órbita nos encontramos cada vez más cerca del punto fijo) o atractivos (el caso contrario).

$$f(x_k) = (1/2)x - 1 \quad (4.8)$$

La órbita de $x_0=6$ es 6, 2, 0, 0, 0, ...

Tenemos también que un punto puede ser k-peródico si tras k iteraciones en la órbita del punto k periódico, nos encontramos otra vez con dicho punto.

4.2.3. Diagrama de bifurcación

Si tenemos un sistema dinámico discreto con un parámetro como puede ser el caso del mapa logístico

$$x_{(k+1)} = rx_k(1 - x_k) \quad (4.9)$$

Con el parámetro r, tenemos una familia de sistemas dinámicos discretos. Es decir, para cada posible valor de r tenemos un sistema dinámico discreto distintos, y cada sistema puede tener sus propios puntos fijos y sus puntos periódicos. Una forma de comprobar visualmente el comportamiento de una familia de sistemas es creando un diagrama de bifurcación. 4.1

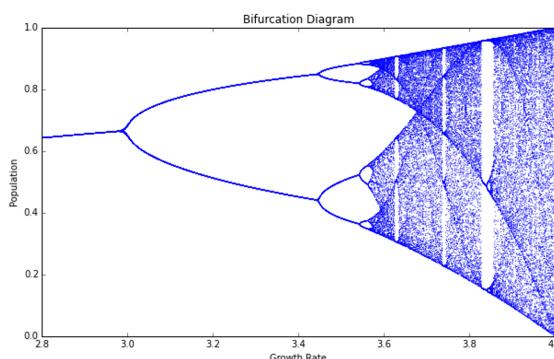


Figura 4.1: Diagrama de bifurcación

El eje vertical representa los posibles valores de imágenes y el eje horizontal representa los distintos valores del parámetro r. Para cada valor del parámetro r se definen muchos x_0 distintos, se itera el mapa una cierta cantidad de veces y se representa con un punto el valor obtenido en la última iteración.

De este modo puede verse representado en la imagen que para $r=2.9$ tenemos que todos los puntos han acabado en el mismo sitio exacto, es decir, cuando $r=2.9$ tenemos un punto atractivo cerca de 0.6. Si tenemos que $r = 3.2$, podemos comprobar que esta vez los puntos han acabado distribuidos en dos lugares, por tanto, se ha producido una bifurcación y cuando $r=3.2$ tenemos dos puntos fijos atractivos. Si nos vamos a $r=3.8$ tenemos que los puntos no tienden a ningún lugar en concreto, esto se debe a que hay una gran cantidad de puntos repulsivos, haciendo que todas las órbitas intenten escapar de todas partes, este es el primer síntoma que encontramos del caos.

Como observación, podemos indicar que muchas veces sucede como en la imagen anterior, en los primeros valores de los parámetros uno o más puntos atractivos que se van bifurcando conforme aumentan los valores de los parámetros, y poco a poco se ve como aparece el caos para

que de repente todo se reordene y volvamos a tener una serie de puntos atractivos solo para acabar cayendo de nuevo en el caos.

4.2.4. Definición de caos

Un sistema dinámico discreto (D, f) es caótico si cumple con las siguientes tres propiedades.

- Es sensible a las condiciones iniciales.
- Es topológicamente transitivo.
- Los puntos periódicos de f son densos en D .

Cuando decimos que es sensible a las condiciones iniciales queremos decir que aplicando un cambio muy pequeño en x_0 , obtenemos grandes diferencias en la órbita tras iterar una cierta cantidad de veces. Pequeños cambios causan grandes diferencias en el futuro.

Con topológicamente transitivo decimos que, dado dos subintervalos cualesquiera de D , si tenemos un punto perteneciente a uno de esos dos intervalos, tras iterar alguna cantidad de veces, llegamos al otro intervalo. Con esto se intenta expresar que dado un punto x_0 cualquiera, iterando varias veces en el mapa, podemos llegar a cualquier zona de D . (Ergodicidad).

Decir que los puntos periódicos de f son densos en D significa que, cogiendo un subintervalo de D cualquiera, siempre encontramos un punto periódico.

4.2.5. Medida del caos

En la mayoría de los artículos consultados, cuando se busca como de caótico es un sistema dinámico discreto, la herramienta más utilizada es el exponente de Lyapunov.

El exponente de Lyapunov es una medida que caracteriza el grado de separación de dos trayectorias infinitesimalmente cercanas. Es decir, mide la diferencia entre trayectorias. Si la diferencia entre las trayectorias es muy grande, entonces tenemos mayor caos. Si el exponente de Lyapunov es mayor que 0 decimos que el sistema es caótico.

Este exponente de Lyapunov se calcula con el siguiente límite:

$$\lambda(x_0) = \lim_{n \rightarrow \infty} \left(\frac{1}{n} \sum_{i=0}^{n-1} \ln(\|f'(x_i)\|) \right) \quad (4.10)$$

Para las zonas resaltadas en la siguiente imagen, obtendríamos un exponente de Lyapunov positivo. 4.2

4.3– Arnold's Cat Map

Arnold's Cat Map (ACM) es un mapa caótico [19], en 1960 Vladimir Arnold demostró empíricamente los efectos de este mapa sobre la imagen de un gato, de ahí viene su nombre.

La transformación ACM de $\mathbb{R}^2/\mathbb{Z}^2$ en $\mathbb{R}^2/\mathbb{Z}^2$ viene definida como sigue.

$$\Gamma : (x, y) \rightarrow (2x + y, x + y) \bmod 1 \quad (4.11)$$

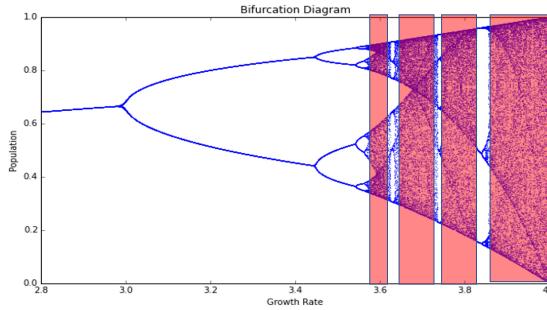


Figura 4.2: Diagrama de bifurcación resaltando zonas con Lyapunov negativo

Cuando hablamos de $\mathbb{R}^2/\mathbb{Z}^2$, hablamos de \mathbb{T}^2 como un espacio cociente. En nuestro caso, utilizaremos esta herramienta para modificar las posiciones de los píxeles en imágenes utilizando la notación matricial.

$$\Gamma \left(\begin{bmatrix} x \\ y \end{bmatrix} \right) = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \bmod N \quad (4.12)$$

Esta herramienta está solo en matrices cuadradas, por tanto, cuando tratamos imágenes, solo es posible utilizar esta herramienta directamente en imágenes que sean cuadradas, sin embargo, como se verá mas adelante, se ha modificado para que sea capaz de reordenar los píxeles de cualquier imagen dado un tamaño N cualquiera. Es importante también señalar que existe un periodo tras el cual es vuelve al estado original, este periodo es variable en función del tamaño de la matriz.

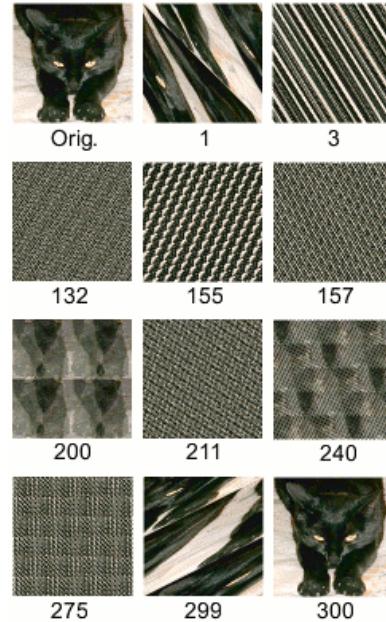


Figura 4.3: Periodo Arnold's Cat Map Wikipedia

4.4– Criptografía en computación basada en ADN

La computación basada en ADN ha estado desarrollándose en los últimos años, el interés en su estudio se debe a sus propiedades de gran capacidad de almacenamiento, gran capacidad de paralelización, gran densidad de información y bajo coste energético (Medical Image Encryption Based on Hybrid Chaotic DNA Diffusion). Cada secuencia de ADN está compuesta por las cuatro bases nitrogenadas: Adenina (A), Timina (T), Citosina (C) y Guanina (G). Para codificar estas secuencias se asocia a cada base un número entero, A → 0, C → 1, G → 2, T → 3, que si representamos en binario corresponden con A → 00, C → 01, G → 10, T → 11. De este modo, si tomamos el operador XOR para utilizarlo sobre este conjunto, se cumple la complementariedad para las imágenes binarias de las cuatro bases, es decir, A(00) y T(11) son complementarios y C(01) y G(10) son también complementarios.

Podemos utilizar otras codificaciones binarias de modo que se mantengan iguales los complementarios, en concreto podemos obtener ocho reglas distintas.

Cuadro 4.1: Reglas de codificación de ADN

Reglas de codificación de ADN								
Valor binario	Regla 1	Regla 2	Regla 3	Regla 4	Regla 5	Regla 6	Regla 7	Regla 8
00	A	A	C	G	C	G	T	T
01	C	G	A	A	T	T	C	G
10	G	C	T	T	A	A	G	C
11	T	T	G	C	G	C	A	A

A la hora de operar con XOR sobre las cuatro bases tenemos los resultados de la siguiente tabla.

Cuadro 4.2: Resultados de XOR

Resultados de XOR				
XOR	A	G	C	T
A	A	G	C	T
G	G	A	T	C
C	C	T	A	G
T	T	C	G	A

Utilizando la codificación de ADN, cada 8 bits de una imagen en escala de grises pueden codificarse como una secuencia de ADN de longitud 4. Por ejemplo, para un pixel de una imagen natural en escala de grises $p = 01111000$ tenemos que dicho pixel puede codificarse como CTGA, de forma similar si tenemos una imagen médica de 16 bits por pixel, podemos codificar cada pixel como una secuencia de 8 bases.

En este trabajo, en lugar de utilizar el operador XOR como suele ser el caso, se utilizará el operador propuesto por el artículo principal en el que se basa el TFG, el operador \otimes . Manteniendo las relaciones de complementarios, tenemos que al operar las distintas bases, se tienen los siguientes resultados si estamos utilizando la regla de codificación 1.

En realidad, lo que pretende el artículo es mostrar como el operador propuesto es capaz de actuar en conjunción a otras herramientas que puedan usarse en criptografía, como puede ser el caso en el que se utilice codificación basada en ADN.

Cuadro 4.3: Resultados de \otimes

Resultados de \otimes				
\otimes	A	C	G	T
A	A	C	G	T
C	C	T	A	G
G	G	A	T	C
T	T	G	C	A

CAPÍTULO 5

Estructura del algoritmo

En los capítulos 2 y 3 se ha hecho un repaso de la metodología que ha ayudado a desarrollar el proyecto del trabajo de fin de grado. En este capítulo se estudia la estructura del criptosistema, el cual abarca tanto el cifrado como el descifrado de las imágenes. Se presentará un diagrama de flujo que indicará la estructura del cifrado, y otro diagrama que indicará la estructura del descifrado.

5.1– Algoritmo de encriptado

El objetivo inicial con respecto al algoritmo propuesto por el artículo en el que se basa este TFG [18], era implementarlo tal y cómo se describe en el artículo. Esto ha sido finalmente posible y a continuación se detalla, este criptosistema.

Usando el lenguaje de programación Python, se ha seguido la estructura del diagrama de flujo indicado en la figura del siguiente modo:

5.1.1. Relativo a mapas caóticos

En esta subsección se van a tratar las partes destacadas en la figura.

Se propone utilizar un mapa caótico específico propuesto por los autores de otro artículo [21], útil para desarrollar criptosistemas, este mapa caótico que es hiper-caótico, esto en resumen nos dice que el comportamiento que va a tener este mapa caótico en las sucesivas iteraciones es extremadamente complicado de predecir. Este comportamiento es demostrado por los autores del artículo en cuestión mediante distintas herramientas de teoría de caos como, el estudio de atractores, el exponente de Lyapunov, la dimensión de correlación y la entropía de Kolmogorov.

El mapa propuesto es un "2D-Infinite Collapse Map"(2D-ICM), un ICM es un mapa caótico en una dimensión que tiene el mejor rendimiento entre los mapas de 1D caóticos. Se define como

$$x_{n+1} = \sin\left(\frac{a}{x_n}\right) \quad (5.1)$$

donde x_n es la entrada y x_{n+1} es la salida, a es un número real distinto de 0. Sin embargo, los mapas caóticos de una dimensión suelen tener una estructura relativamente simple predecir su comportamiento con algunas tecnologías actuales. A partir de aquí se define el 2D-ICM como

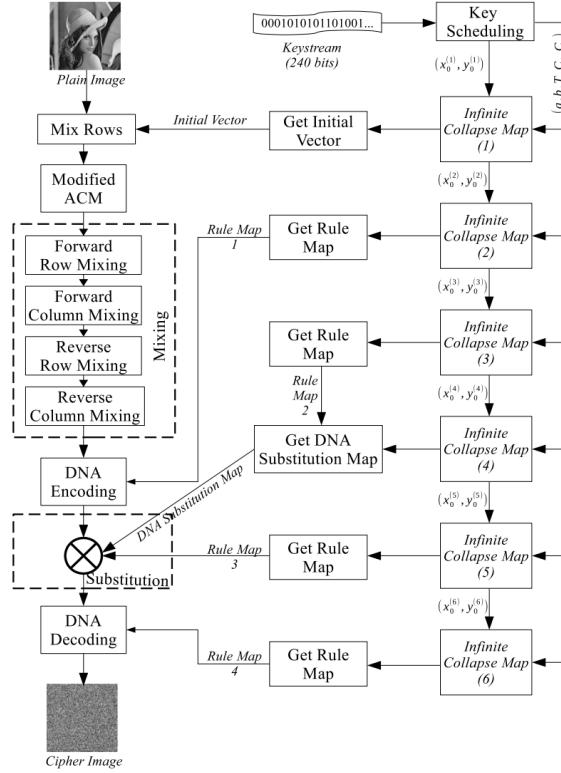


Figura 5.1: Algoritmo de encriptado

$$x_{n+1} = \sin\left(\frac{a}{y_n}\right) \times \sin\left(\frac{b}{x_n}\right) \quad (5.2)$$

$$y_{n+1} = \sin\left(\frac{a}{x_n}\right) \times \sin\left(\frac{b}{y_n}\right) \quad (5.3)$$

Siendo x_{n+1} e y_{n+1} las salidas y x_n e y_n las entradas, a y b son dos números reales distintos de 0.

Como es de esperar, este mapa se utilizará para obtener secuencias de números pseudoaleatorios con el fin de reordenar y alterar los pixeles de modo que solo alguien que conozca las condiciones iniciales del sistema sea capaz de revertir el proceso. Estas condiciones iniciales vienen dadas por una clave que tomará como entrada el algoritmo del criptosistema, una clave de 240 bits que será procesada por el algoritmo denominado Key Scheduling.

El objetivo de Key Scheduling es obtener los valores de a, b, x_0 e y_0 , para ello se divide la clave de 240 bits en las siguientes 7 partes {a,b,x,y,T,C₁,C₂}, cogiendo 40 bits para los valores de a, b, x, y, t en el mismo orden, y para los otros dos valores se utilizan 20 bits y los últimos 20 bits. A partir de aquí, según el artículo de Cao, se obtienen los valores iniciales de la siguiente forma:

$$a = (a_0 + T + C_1) \bmod 5 + 16 \quad (5.4)$$

$$b = (b_0 + T + C_2) \bmod 5 + 16 \quad (5.5)$$

$$x_0 = (x + T + C_1) \bmod 2 - 1 \quad (5.6)$$

$$y_0 = (y + T + C_2) \bmod 2 - 1 \quad (5.7)$$

Esto sin embargo, ha causado distintos problemas durante la implementación del algoritmo debido a ciertas contradicciones y a la limitada posibilidad de combinaciones que esto supone. Se hablará sobre esto en profundidad más adelante en el capítulo 6.

Durante la explicación que se ofrece en el artículo de las distintas funciones que debe tener el criptosistema se habla en múltiples ocasiones sobre la expansión del mapa caótico al tamaño $m \times n$, aunque no viene explicado en ningún lugar a que se refiere, se ha interpretado que se refiera realizar $m \times n$ iteraciones en el mapa caótico, y en cada k -ésima iteración, almacenar el k -ésimo par de valores obtenido en la k -ésima posición de un array que represente una matriz $m \times n$, los valores de la matriz son pares de números reales comprendidos entre -1 y 1 debido a la naturaleza senoidal del mapa caótico propuesto.

Se puede apreciar en el diagrama que existen distintos ICM, sin embargo, todos ellos son realmente la misma instancia del mismo mapa caótico, solo que las condiciones iniciales del mapa siguiente, son las condiciones finales del mapa anterior.

La función de «Get Rule Map» está directamente relacionada con la parte de codificación en ADN, esta nos permite obtener una matriz con valores del 1 al 8. Estos valores identifican una de las 8 reglas que se definen para codificar cada par de bits de la imagen. Esto se consigue utilizando el mapa caótico que se acaba de describir. El mapa caótico se expande al tamaño de la imagen $m \times n$ y se procesa de modo que se obtienen valores pseudoaleatorios en una matriz $m \times (n \cdot (p/2))$. Siendo p la profundidad de pixel de la imagen, 8 o 16.

Sobre la obtención de la clave de 240 bits se hablará en detalle en el capítulo 6.

5.1.2. Otros preliminares para operaciones de mezcla

Aquí se presentarán lo referente a los algoritmos de "Get Initial Vector" "Get DNA Substitution Map". Ambos necesarios para realizar operaciones que nos permitirán mezclar los pixeles de la imagen de modo que se elimine la mayor cantidad de información visual de la misma. En este caso y en la mayor parte del criptosistema, la palabra mezcla no se utiliza en el sentido de reordenación (tener un elemento A y un elemento B, cambiar la posición de A por la posición de B), sino en el sentido de, teniendo un elemento A, y un elemento B, obtener un elemento C en la posición del elemento A o B. Estas mezclas serán realizadas utilizando el operador binario propuesto en el artículo. En criptografía, la palabra aquí referida como mezcla suele ser confusión (sustitución) y la palabra reordenación suele ser difusión (transposiciones).

En el caso de la función de obtención de un vector inicial, simplemente se expande el mapa caótico al tamaño $1 \times m$, y se obtiene un vector del mismo tamaño donde a cada i -ésimo elemento del array se le asigna el siguiente valor:

$$IV(1, i) = |x_i| \times |y_i| \times (2^{31} - 1) \bmod (2^p) \quad (5.8)$$

El factor $(2^{31} - 1)$ se utiliza varias veces durante la implementación del criptosistema, por el uso que se le da, se ha intuido que sirve para representar el número pseudoaleatorio obtenido en un número de 32 bits.

Este vector será útil para hacer una mezcla de todos los pixeles de la imagen por filas, utilizando como valor inicial de mezcla para cada fila, un valor del vector.

Algo similar sucede con la función de «Get DNA Substitution Map», en este caso se expande el mapa caótico, y para cada i -ésimo valor obtenido, se transforma este en binario, y por cada dos bits de ese valor binario, se le asigna un valor del conjunto {A,C,G,T}, el valor asignado dependerá del i -ésimo valor obtenido (un número del 1 al 8) en el Rule Map que toma como entrada esta función.

Del mismo modo, la matriz obtenida a partir de este algoritmo generará una mezcla de la imagen utilizando el operador nuevo operador binario, sin embargo, esta vez se opera con la codificación de hebras de ADN artificiales.

5.1.3. Mezclado de filas y mezclado (Mixing)

Hasta ahora, no se ha tocado ningún pixel de la imagen, a partir de ahora todos los algoritmos supondrán un cambio directo de forma visual e incluso habrá cambios de formato.

El primer paso para realizar cambios en la imagen se hace mediante la función definida como «Mix Rows», esta función toma el primer pixel de cada i-ésima fila x_i , el i-ésimo valor del vector inicial obtenido anteriormente IV_i , y realiza la siguiente operación para obtener el nuevo valor del pixel en esa posición x_{nuevo} :

$$x_{nuevo} = x_i \otimes IV_i \quad (5.9)$$

A partir de aquí, la mezcla se propaga a los siguientes valores de cada fila, de modo que el siguiente nuevo valor de la imagen, será el resultado de operar el valor antiguo, con el valor nuevo del pixel anterior.

Tras esto, como se puede comprobar en la figura, se realiza un paso denominado «Modified ACM», solo es necesario saber por el momento que este paso realiza un reordenamiento de los pixeles obtenidos al realizar el mezclado de filas, se detallará su función más en detalle en la siguiente subsección. Por ahora nos centramos en el paso llamado «Mixing» en la figura, toma como entrada los pixeles de la imagen mezclados por filas y reordenados, y realiza de nuevo mezclas entre los píxeles.

Este paso de mezcla no requiere de ninguna información previa, es decir, no depende de ninguna clave proporcionada al criptosistema, ni de ninguna secuencia de números pseudoaleatorios, sin embargo, es útil para eliminar información visual, también ayuda a prevenir ataques diferenciales. Sigue una filosofía similar a la mezcla de filas que acabamos de ver, solo que se realiza en 4 pasos:

- Forward row mixing: Se realiza una mezcla igual que en el mezclado de filas visto anteriormente, tomando como valor invariante el primer pixel de cada fila, y propagando la mezcla a partir del segundo valor de cada fila utilizando el nuevo operador binario.
- Forward column mixing: Esta vez se propaga la mezcla por columnas, es decir, el primer pixel de cada columna queda invariante, y se propaga la mezcla a partir del segundo pixel de cada columna.
- Reverse row mixing: Es la misma mecánica que la de "Forward row mixing", pero esta vez los píxeles invariantes que inicializan el proceso de mezcla son los últimos píxeles de cada fila.
- Reverse column mixing: Mismo caso que "Forward column mixing" pero los pixeles que no cambian de valor son los últimos de cada columna, y se propaga la mezcla hacia los pixeles anteriores de cada columna.

5.1.4. Relativo a la codificación en ADN

En este apartado se describe cómo funciona la codificación de ADN en el criptosistema implementado. En resumen, lo que se realiza en el algoritmo de cifrado del criptosistema es, codificar el

resultado de las mezclas anteriores con nucleótidos de ADN utilizando la tabla de reglas. La regla que se utilizará dependerá del mapa de reglas que se tiene como entrada, obtenido a partir de la función anteriormente descrita «Get Rule Map». De este modo, una imagen p-bits por pixel, en nuestro caso $p = 8$ o $p = 16$, se transformará en una cadena de $p/2$ nucleótidos por cada pixel.

Tras codificar la imagen, llega el momento de nuevo de hacer mezclas, esta vez se utiliza el operador binario con el primo $p = 5$ ya que estamos en el conjunto $\{A,C,G,T\}$ mapeado en $S = \{0, 1, 2, n-1 = 3\}$. En este caso tomamos como entrada un mapa de sustitución de ADN, un mapa de reglas y la imagen con el formato de cadena de ADN. El proceso es el siguiente, cada nucleótido de la cadena que codifica la imagen se traduce a un número perteneciente a S según corresponda con la regla obtenida del mapa de reglas, se hace lo mismo con el nucleótido en la misma posición del mapa de sustitución, se operan ambos valores con \otimes y se traduce el resultado a un nuevo nucleótido utilizando la regla adecuada del mapa de reglas. De este modo se ha hecho una nueva mezcla.

El resultado de la secuencia caótica de nucleótidos que tenemos ahora, depende del mapa de sustitución de ADN y del mapa de reglas, si no conocemos uno de los dos elementos es muy complicado descifrar la imagen o tratar de conocer la clave original.

Una vez tenemos esta cadena de nucleótidos caótica, volvemos al formato de pixeles original utilizando otro mapa de reglas distinto, de este modo, la imagen actual no tiene correlación aparente con la imagen original.

5.1.5. Arnold's Cat Map modificado

Se va a describir por último, un mapa caótico cuya función será la de reordenar los píxeles de la imagen. El mapa caótico en cuestión es el siguiente.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} (\text{mod } N) \quad (5.10)$$

donde x e y son las posiciones de cada pixel y x' y y' son las nuevas coordenadas asignadas a cada pixel. El principal problema de utilizar Arnold's Cat Map (ACM) es que está definido solo para imágenes cuadradas, es decir, de tamaño $n \times n$, y es muy habitual encontrar tanto en imágenes médicas e imágenes naturales, que no son cuadradas, es más, lo común es que no sean cuadradas. Por eso la parte complicada de incluir ACM en el criptosistema, es conseguir hacer una versión modificada que sea capaz de procesar imágenes de cualquier forma.

El modo de solucionar este problema consiste en subdividir la imagen en distintas partes cuadradas todas del mismo tamaño, y en cada subimagen, se aplica ACM por separado.



Figura 5.2: Ejemplo ACM modificado horizontal

Si el tamaño de la imagen es $m \times n$, el tamaño del lado de cada cuadrado será $\min(m,n)$, siendo \min la función que determina el mínimo de un par.

De este modo, el problema se define como, encontrar en que posiciones irá cada cuadrado y determinar el número de cuadrados. El número de cuadrados $\alpha = \lceil \max(m,n)/N \rceil$ donde \max es la función que retorna el máximo de un par.

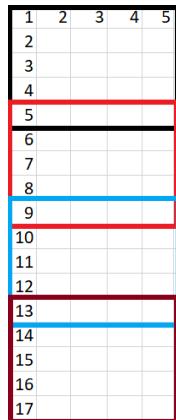


Figura 5.3: Ejemplo ACM modificado vertical

Es muy posible que m y n no sean múltiplos el uno del otro, por tanto, se producirá un solapeamiento entre cuadrados, el tamaño de este solapamiento \mathcal{O} se calcula del siguiente modo.

$$\mathcal{O} = \lfloor \{N - (\max(m, n) \bmod N)\}/(\alpha - 1) \rfloor \quad (5.11)$$

Y así, el primer cuadrado se posiciona en los primeros píxeles de la imagen, el segundo se posiciona empezando en el límite donde acaba el primero, pero \mathcal{O} píxeles por detrás, así hasta el penúltimo cuadrado, el último abarca desde la última fila o columna de píxeles hasta donde abarque.

5.2– Algoritmo de desencriptado

Como es de esperar, el desencriptado se implementa siguiendo el camino inverso, es decir, encontrado las operaciones inversas de cada una de las herramientas que se han definido para la encriptación. En este caso partimos de la imagen cifrada y de una clave de 240 bits, la cual debe ser la misma que se utilizó para la encriptación, y terminamos con la imagen original.

En el caso de las partes relativas al 2D-ICM, la obtención del vector inicial y el mapa de sustitución de ADN se mantienen igual puesto que nos ofrecen los datos que originaron las mezclas posteriores durante la encriptación de la imagen.

5.2.1. Operador inverso (\otimes^{-1})

Una de las herramientas más importantes del criptosistema es precisamente la operación inversa del operador \otimes . Dado que el artículo no menciona en ningún momento el proceso de desencriptar, tampoco hace mención de la necesidad de encontrar esta operación inversa \otimes^{-1} . Para $a, b, c \in S$, si realizamos la operación:

$$a \otimes b = c \quad (5.12)$$

Se pretende encontrar una operación que dados a y c , se obtenga b .

$$a \otimes^{-1} c = b \quad (5.13)$$

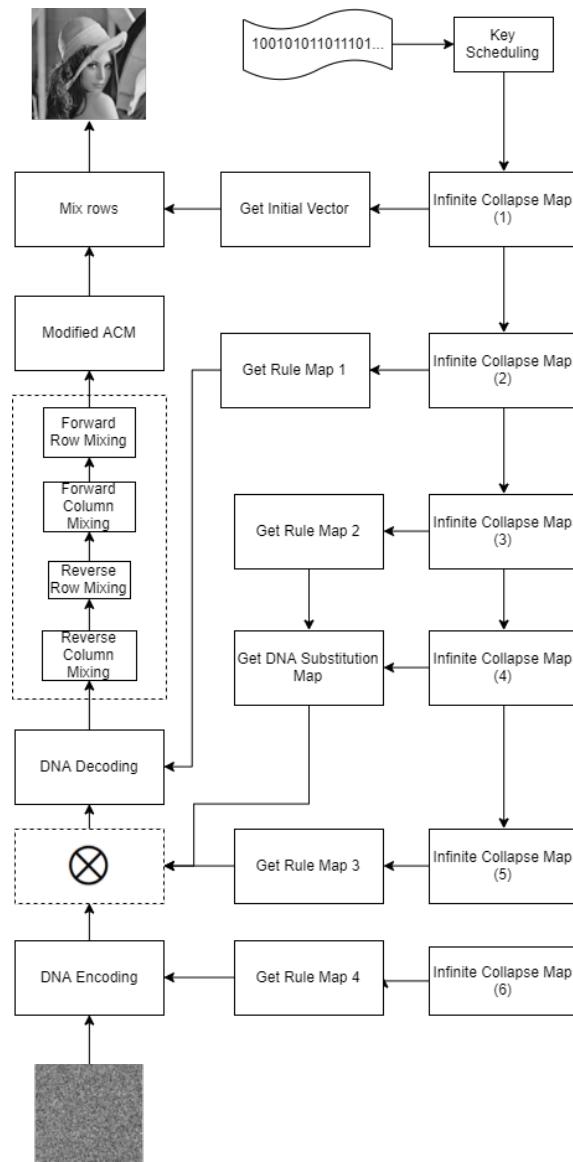


Figura 5.4: Diagrama de flujo de desencriptado

Lo complicado de encontrar esta operación inversa, es el hecho de que el operador \otimes utiliza el modulo p. Una de las rutas más exploradas ha sido tratar de definir el inverso asumiendo de forma intuitiva lo siguiente.

$$a \otimes c^{-1} = b \quad (5.14)$$

Esto no ha funcionado en general, sin embargo, haciendo una gran cantidad de pruebas, me he dado cuenta de que el resultado obtenido no era b, sino b^{-1} . Por tanto, en ese momento he asumido que la operación inversa estaba definida de la siguiente forma.

$$a \otimes^{-1} c = (a \otimes c^{-1})^{-1} = b \quad (5.15)$$

Por suerte, esto ha paecido ser correcto al menos para los casos en los que el número primo utilizado en la operación \otimes es 5, 257 y 65537 que son los tres casos posibles en el criptosistema. La forma de comprobar que la operación es válida para esos valores, ha sido comprobar por fuerza bruta si se obtiene el resultado correcto en todas las posibles combinaciones de valores de a,b y c.

Aunque he intentado proponer una demostración matemática para asegurar que el operador inverso propuesto es válido para cualquier valor de p, no he sido capaz de demostrarlo formalmente. Por tanto, no aconsejo utilizarlo para otros valores sin un estudio previo del mismo para el valor de p necesario.

5.2.2. Invertir las operaciones en ADN y de mezcla

La operación inversa de codificar en ADN es precisamente decodificar en ADN, así que a la hora de hacer el proceso inverso en esta ocasión, se debe primero codificar la imagen cifrada en una cadena de nucleótidos de ADN, se debe realizar el proceso inverso de sustitución y después descifrar la cadena de nucleótidos resultante.

El proceso de sustitución de nucleótidos para el encriptado de la imagen se realizó utilizando el operador binario \otimes , por tanto, para deshacer el cambio se utiliza el operador inverso \otimes^{-1} con p = 5 ya que S = {0,1,2,3}.

En el caso de las operaciones de mezcla por filas y de mezcla de toda la imagen, se sigue la misma mecánica, se recorre la imagen en el orden inverso en el que se recorrió inicialmente, y se aplica el operador inverso para volver a los estados originales de la imagen.

5.2.3. Invertir Arnold's Cat Map

En este caso he tratado de seguir dos caminos distintos para volver al estado original de la matriz de píxeles. El primer camino consiste en tratar de utilizar la propiedad especial de este algoritmo para, tras un determinado número de iteraciones, volver a su estado original. Esto conlleva investigar algún modo de calcular el periodo asociado a la imagen cuadrada que viene determinado por el tamaño de la misma. He encontrado algunas menciones en artículos científicos [19], [periodacm], [15] que hacen referencia al cálculo del periodo, pero al tratar de comprobar experimentalmente lo que se dice en ellos, no he obtenido resultados satisfactorios. Finalmente, tampoco he sido capaz de encontrar por mi cuenta una posible solución a este problema, por lo que este camino quedó atrás y el resultado que si proporciono soluciones válidas fue el que seguí.

Conociendo la definición del mapa caótico utilizando la matriz:

$$\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \quad (5.16)$$

Es posible realizar el cálculo de la matriz inversa, la cual es:

$$\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \quad (5.17)$$

Si aplicamos el mismo número de iteraciones al encriptar, pero utilizando esta matriz inversa a cada uno de las subimágenes en las que se haya tenido que dividir la imagen, en el orden inverso, podemos volver al estado original fácilmente.

5.3– Referente a la implementación

La implementación del critposistema ha sido llevada a cabo utilizando el lenguaje de programación Python en su versión 3.8.3. A parte de utilizar las técnicas de programación adquiridas durante la carrera, se han utilizado distintas funciones de las siguientes librerías que hay que se deben instalar para poder ejecutar el programa.

- **numpy[7]**: Se trata de una librería para trabajar con números y matrices de forma más precisa. Se puede instalar mediante el comando 'pip install numpy'.
- **pydicom[9]**: Esta librería permite leer imágenes dicom, para usar algunas de sus funciones se necesita tener instalado previamente numpy, esto se debe a que se utilizan arrays de numpy. Se puede instalar mediante el comando 'pip install pydicom'.
- **opencv[8]**: De modo similar a pydicom, esta librería se utiliza para trabajar con imágenes, también necesita tener instalado previamente numpy y puede instalarse mediante el comando 'pip install opencv-python'.
- **matplotlib[6]**: Utilizada en este proyecto principalmente para la generación de gráficas y para mostrar algunas imágenes. Se instala utilizando el comando 'pip install matplotlib'.
- **sympy[10]**: Una librería para trabajar con matemáticas simbólicas, se utiliza en este proyecto principalmente para identificar si un número es primo. Se puede instalar con el comando 'pip install sympy'.

El código fuente se estructura de forma sencilla en seis archivos de Python. Aunque hay otros archivos de Python que son para pruebas, los más relevantes y que de verdad aportan funcionalidad al programa son los siguientes.

- **Cifrado.py**: Este es un script que permite encriptar una imagen utilizando los siguientes parámetros por consola: ruta de la imagen (cadena), contraseña (cadena), ruta de guardado del resultado (cadena), escala de grises (booleano), ruta para guardar los pasos intermedios (string).
- **Criptosistema.py**: En este archivo se unifican todos los algoritmos necesarios para componer el criptosistema. Es aquí donde se definen las funciones para encriptar y desencriptar.
- **CriptosistemaInterfaz.py**: Una versión de «Criptosistema.py» que es utilizado por la interfaz creada.
- **Descifrado.py**: Este es un script que permite desencriptar una imagen utilizando los siguientes parámetros por consola: ruta de la imagen (cadena), contraseña (cadena), ruta de guardado del resultado (cadena), escala de grises (booleano), ruta para guardar los pasos intermedios (string).

- **funcionesCripto.py:** Todos los algoritmos necesarios para implementar el criptosistema se encuentran en este archivo.
- **funcionesEstadisticas.py:** Las funciones creadas para obtener datos estadísticos se encuentran aquí.
- **ICM.py:** Se ha creado una clase que define el mapa caótico.
- **main.py:** Un script que permite cifrar o descifrar imágenes desde la línea de comandos como se ve en la figura 5.5. El orden de los parámetros es el siguiente: Modo cifrar o descifrar (0 ó 1), ruta de la imagen original, contraseña que se utilizará como clave, ruta de la imagen resultado, escala de grises o a color (1 ó 0).

```
>py main.py 0 ../planeImage.png 1234 ../prueba.png 1
```

Figura 5.5: Comando para ejecutar el algoritmo por consola

A parte de la implementación del criptosistema en Python, también se ha creado una interfaz gráfica para permitir la utilización del programa a un abanico más amplio de usuarios. Este interfaz permite cambiar entre los modos de encriptar y desencriptar mediante un botón. Tras seleccionar los parámetros como la ruta de la imagen, escala de grises, etc; se muestra el proceso de encriptar y desencriptar. Esto incluye un histograma por cada paso y la varianza y entropía original y final.

Se puede acceder al código descrito a través de mi perfil de GitHub [13].

5.4– Interfaz gráfica

Se ha tratado de hacer lo más simple posible de modo que requiera pocos conocimientos para ser utilizado, sin embargo, es posible acceder a prácticamente todas las funcionalidades programadas. Se debe señalar que la implementación de esta interfaz gráfica no era uno de los objetivos iniciales, por lo que recibe mayor importancia en la implementación del propio criptosistema y el análisis del artículo[18]. Se ha hecho uso de HTML, PHP, JavaScript y CSS para crear dicha interfaz, para hacer pruebas en local se ha utilizado la herramienta XAMPP utilizada en algunas asignaturas de la carrera, esta proporciona el servidor APACHE y PHP. Como se ha comentado anteriormente, se ha tratado de mantener la mayor simplicidad posible, la interfaz consta de una sola pantalla 5.6 con distintas partes diferenciables.

- **A Título:** Nombre del proyecto.
- **B Cifrar/Descifrar:** Podemos cambiar entre ambos modos mediante el botón verde.
- **C Selección de parámetros:** En este apartado podemos seleccionar una o más imágenes del mismo formato (png o dcm) para nuestra prueba del criptosistema. Podemos seleccionar una contraseña para que sea utilizada como clave para encriptar o desencriptar las distintas imágenes. Es posible seleccionar si queremos tratar la imagen a color o en escala de grises, en caso de ser una imagen Dicom no debemos marcar la opción a color.
- **D Log/info:** Se muestra un registro de lo que está sucediendo mientras el algoritmo está siendo ejecutado, es útil para comprobar el progreso e informar de cualquier error que suceda.

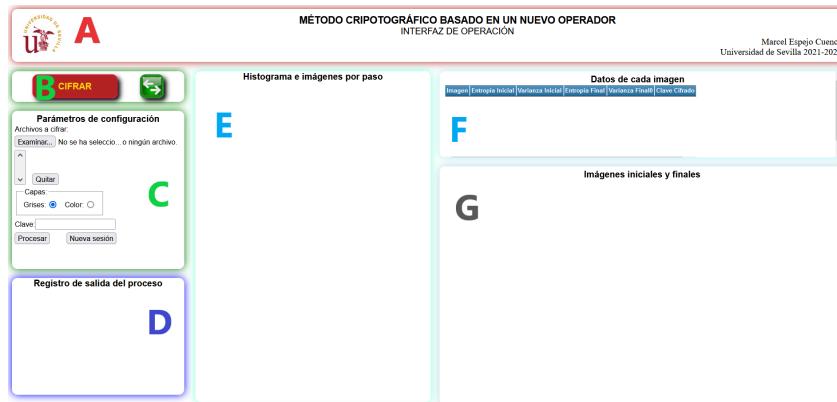


Figura 5.6: Pantalla interfaz

- E Pasos de ejecución:** La imagen sufre varios cambios visibles durante la ejecución del criptosistema, los cambios visibles se presentan en esta parte junto con un histograma. Por ejemplo, si tenemos una imagen en escala de grises, obtendremos 3 pasos con sus 3 histogramas, son exactamente 3 pasos porque son los pasos que podemos observar como una imagen normal 5.1.
- F Información estadística:** Tras la ejecución del algoritmo se recoge en una tabla la entropía y la varianza de la imagen al inicio y al final.
- G Resultados:** Obtenemos la imagen original con su histograma y el resultado de encriptar o desencriptar con su histograma.

También tenemos la posibilidad de almacenar cualquier imagen presentada en la interfaz haciendo click sobre ella. De este modo podemos encriptar una imagen, guardar el resultado final, y posteriormente iniciar el proceso para desencriptar. Un ejemplo de cifrado en una imagen Dicom puede verse en la figura 5.7, por otro lado, en la figura 5.8 se muestra el proceso inverso pero en una imagen natural a color.

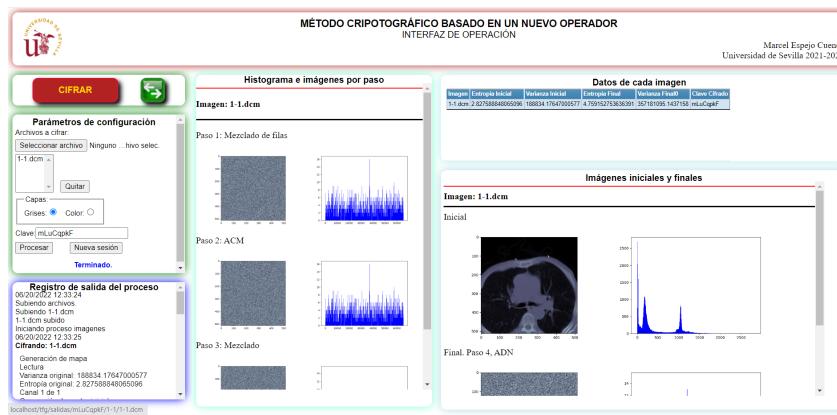


Figura 5.7: Cifrado Dicom

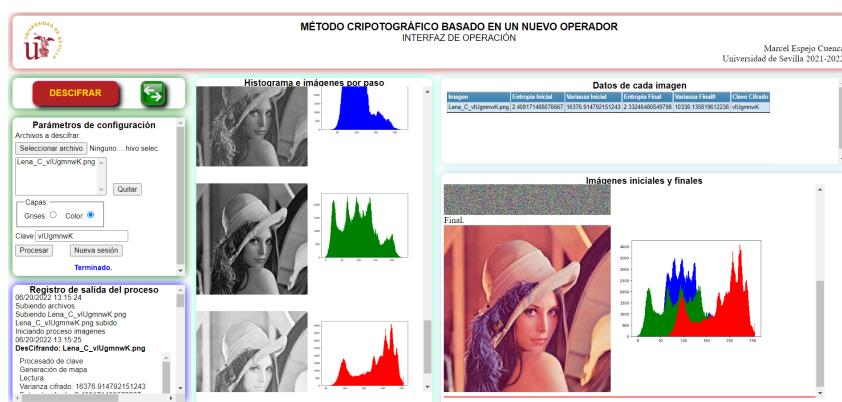


Figura 5.8: Pantalla interfaz

CAPÍTULO 6

Experiencias y dificultades durante el desarrollo

En este capítulo se hace mención a los distintos problemas a los que ha sido necesario enfrentarse durante el desarrollo del proyecto. Realizar un TFG o en general, tratar de investigar y de llevar a cabo lo que propone un artículo científico conlleva cierto riesgo. Al menos por lo que he visto en estos años de carrera, los artículos de investigación también llamados «paper» son documentos que, en muchas ocasiones, siguen la línea de investigación de otros artículos, esto hace que muchos de los términos utilizados y la teoría intrínseca que hay que dominar para comprender en profundidad el documento, se suponga ya conocido para quien lo lee. En ocasiones, a pesar de que estos artículos son revisados antes de ser publicados contienen erratas o incluso el contenido publicado es directamente erróneo, y es esto lo que hace que sea aún más arriesgado enfocar el TFG así. Sin embargo, es una buena forma de aprender a tomar decisiones y encontrar alternativas a lo que se propone en el artículo, también se aprende a consultar con expertos cuando uno se queda atascado en distintas situaciones.

Es también muy útil realizar este tipo de proyectos como entrenamiento para un ingeniero informático, ya que en el campo de la informática uno puede enfrentarse a muchas situaciones. Es posible que un día sea necesario implementar un software de simulación que utilice más tarde un arquitecto para la construcción de un puente, el ingeniero informático que diseñe este software deberá estudiar todo lo relacionado con la física que actúa en los distintos tipos de puente, deberá conocer características de los distintos materiales que podrían utilizarse, etc. Pero también es posible que a ese mismo ingeniero informático un tiempo después, debe encargarse de crear un programa capaz de simular una orquesta para un compositor de música, en este caso se deberá estudiar todo lo relacionado con la teoría musical necesaria para ello.

Por suerte en este caso, parte de la matemática aplicada en el criptosistema ya me era conocida. Para las partes en las que está implicada la teoría del caos, se han utilizado conocimientos impartidos en la asignatura de Matemática Computacional impartida en la escuela.

6.1– El problema en Key Scheduling

El artículo principal habla del método utilizado en otro artículo (Cao-Mao) para realizar todo el proceso relacionado con los 2D-ICM, nos dice que la clave de 240 bits se divide en varias partes y que con esas partes se calculan las condiciones iniciales del mapa caótico, el problema es que no nos dice directamente como se realiza ese cálculo, solo nos indica que se realiza como en el otro artículo. Al consultar el otro artículo nos encontramos con que los cálculos son los mencionados en la subsección 5.1.1, las fórmulas 5.4, 5.5, 5.6 y 5.7. Como puede verse, en la fórmula 5.6,

se realiza el módulo 2 de ciertas operaciones de suma de números enteros positivos, con esto obtenemos un número que será el 0 o el 1, tras esto se resta 1 por lo que los valores posibles de x_0 son -1 y 0, de forma similar y_0 solo puede tener los valores -1 y 0. Para los valores de a y b tenemos que se hace el modulo 5 de una suma de enteros, por lo que tenemos un total de 5 opciones para los valores de a y de b para todas las claves posibles. No hace falta realizar muchos cálculos para darse cuenta de que tenemos una cantidad de estados iniciales para nuestro sistema caótico extremadamente pequeña.

Esto es un gran problema porque la seguridad de nuestro criptosistema depende prácticamente por completo de esto ya que es de esta parte de donde dependerán los números pseudoaleatorios obtenidos al expandir nuestro mapa caótico. Un problema extra y que imposibilita del todo utilizar lo que se propone en el artículo de Cao [21], es el hecho de que una de las dos posibles opciones para x_0 e y_0 es 0, esto hace que en la ecuación 5.2 se divida por 0. La solución finalmente ha sido prescindir de la operación módulo y transformar las fórmulas anteriores en las siguientes.

$$a = (a_0 + T + C_1) \quad (6.1)$$

$$b = (b_0 + T + C_2) \quad (6.2)$$

$$x_0 = x + 1 \quad (6.3)$$

$$y_0 = y + 1 \quad (6.4)$$

Ahora los posibles estados iniciales de nuestro sistema caótico son muchísimos más, además, se imposibilita el problema de la división por 0.

6.2– Algunas confusiones en pseudocódigos

El algoritmo para encriptar imágenes está descrito con un pseudocódigo por cada parte del diagrama de flujo, en ocasiones, en estos pseudocódigos se reflejan algunas operaciones específicas de la implementación realizada por los autores del artículo sin comunicar exactamente el motivo de esto, y en otros casos, parece que hay operaciones que es necesario realizar pero que no son descritas en pseudocódigo ni en el artículo en sí.

Un ejemplo de una de estas situaciones de confusión, en el pseudocódigo del algoritmo para obtener un mapa de reglas se utiliza una variable x la cual no es inicializada en ningún momento ni se habla de a que hace referencia exactamente. Por eso en estos casos hay que tratar de usar el sentido común y encontrar el significado más probable para esta variable x , en este caso, esta variable se utiliza como un índice para una matriz, y parece que en la implementación original del criptosistema, las matrices están representadas por arrays de una sola dimensión, por ello aparece la x sin que sea realmente necesario utilizarla ni tampoco inicializarla, puesto que realmente no existe esta variable en el código original.

Por otro lado, en ciertas partes de los pseudocódigos parece haber algunas erratas, por ejemplo a la hora de describir el algoritmo de Arnold's Cat Map modificado 7.1. En la línea 18 y 20 se muestra el modo de identificar el primer subíndice del siguiente cuadro a procesar. El pseudocódigo nos dice en la línea 20 que el valor de y_0 depende de x_0 en lugar de y_0 , este error puede causar que el algoritmo no funcione correctamente para imágenes con $m < n$.

6.3– Problema en Arndol's Cat Map

En ocasiones, algunos problemas nos los complicamos nosotros mismos, y este ha sido uno de esos casos. En el artículo principal se hace mención a la existencia de un número de iteraciones k

de ejecución de ACM tras el cual, la imagen vuelve a su estado original, esto sucede para todas las imágenes. El número de iteraciones k que necesita una imagen para volver a su estado original depende del tamaño de la imagen, sin embargo, esto no significa que cuanto más grande sea la imagen, más iteraciones tarde en volver a su estado primigenio. Es posible que una imagen más grande tenga un periodo menor que una imagen más pequeña, y puede suceder también al contrario.

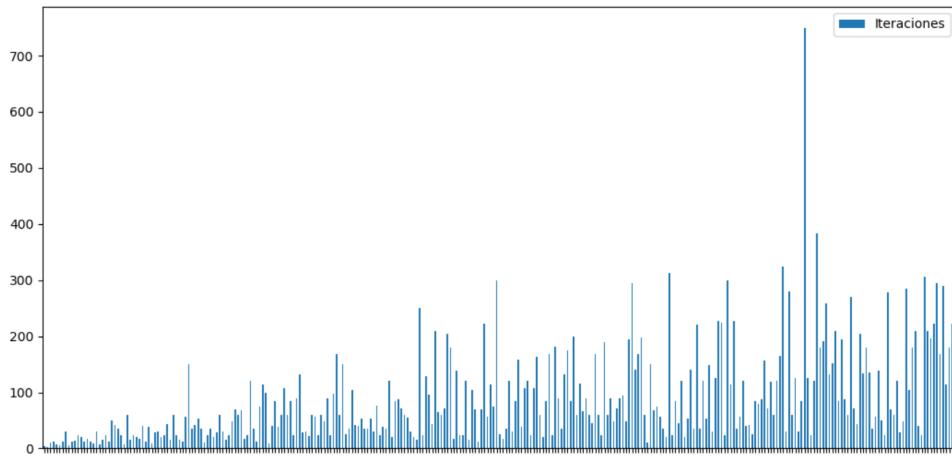


Figura 6.1: Periodos por tamaño en ACM

Para comprobar esto de modo más visual, he creado un histograma donde se representa el número de periodo frente al tamaño de la imagen para los tamaños de 3 a 299.61

Según el artículo principal, hay otros artículos que mencionan que el periodo podría calcularse del siguiente modo.

$$\Pi(N) = \begin{cases} 3N & \text{si } N = 25^k, k = 1, 2, 3, \dots \\ 2N & \text{si } N = 5^k \text{ o } N = 65^k, k = 1, 2, 3, \dots \\ \leq \frac{12N}{7} & \text{en otro caso} \end{cases}$$

Sin embargo, esto no encajaba con los resultados que yo obtenía de periodo, por eso finalmente dejé de investigar esto y me dediqué a buscar otra solución. Por supuesto me estaba complicando y lo más simple era simplemente calcular la inversa de la matriz que define ACM, haciendo así los pasos inversos.

6.4– Encontrar datos de prueba

Aunque hay veces que es complicado obtener buenos datos para hacer pruebas, como suele suceder en muchas ocasiones en proyectos de inteligencia artificial, en este caso, no ha sido demasiado complicado debido a que nuestro criptosistema debe ser capaz de encriptar cualquier tipo de imagen natural o médica, las imágenes naturales las tenemos en gran cantidad en casi cualquier parte, y las imágenes médicas han sido obtenidas de en su gran mayoría de un repositorio público proporcionado por TCIA (The Cancer Imaging Archive)[11], donde podemos encontrar grandes cantidades de TB de imágenes DICOM públicas.

Si queremos descargar un conjunto de datos de muchos GB, lo más probable es que sea necesario descargar primero un programa proporcionado por la propia página llamado NBIA Data Retriever, el cual permite iniciar la descarga de los datos por partes, gracias a este programa podemos iniciar la descarga y pausarla si es necesaria en el proceso.

En este caso, para realizar pruebas con múltiples imágenes DICOM, he descargado un conjun-

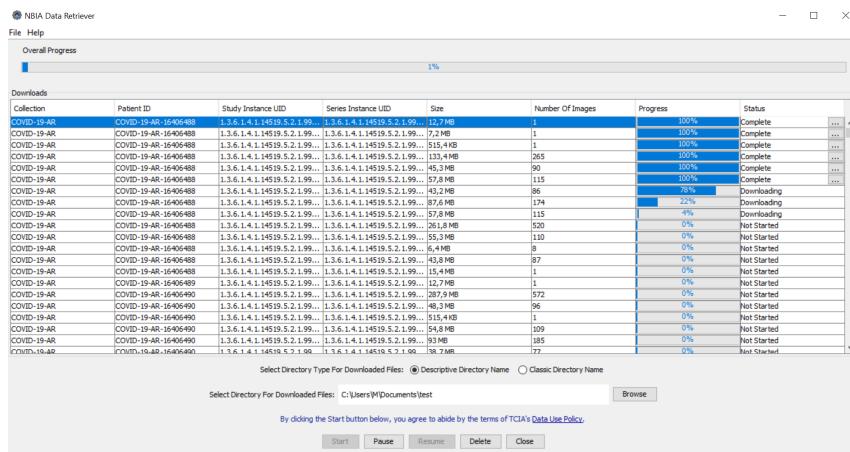


Figura 6.2: Captura de NBIA Data Retriever

to de datos llamado COVID-19-AR [2], contiene múltiples imágenes de los pulmones de distintos pacientes con COVID-19.

6.5– Algunos añadidos extra

A parte de la implementación propuesta por el artículo, se han añadido otros elementos al algoritmo que permiten pequeñas funcionalidades extra al criptosistema.

La clave de 240 bits que se da como entrada en el algoritmo es el factor más importante para la seguridad del criptosistema, ya que esta clave será la pieza que decida cuál será el camino que siga el mapa caótico, el cual decidirá el vector inicial y los distintos mapas de reglas que utilizará la parte basada en cadenas de ADN artificial. Esta es una clave simétrica, es decir, la misma clave sirve para encriptar y desencriptar la imagen. Se ha querido incorporar también un sistema simple mediante el cual sea posible introducir una contraseña (cadena de texto cualquiera) para que un futuro usuario del criptosistema pueda introducir su propia contraseña como clave para encriptar sus imágenes.

La solución simple implementada ha sido calcular el SHA-256 de la cadena introducida y tomar los primeros 240 bits de ese resultado como clave del criptosistema. Un algoritmo SHA (Secure Hash Algorithm) es una función criptográfica que permite calcular una cadena de bits que tiene siempre el mismo tamaño, en el caso de SHA-256 son 256 bits. Las características que tienen estas funciones de Hashing (resumen) que las hacen tan útiles en criptografía son:

- Dada una entrada A de información, es posible obtener una cadena de bits B mediante un algoritmo de Hashing, pero dada la cadena de bits B no es posible obtener el valor de los datos A. Esto se debe a que, en realidad, hay infinitas entradas que proporcionan la misma cadena de bits B.
- Es extremadamente complicado encontrar una colisión, es decir, es muy difícil encontrar dos entradas de información que al procesarlas mediante el algoritmo de Hashing, se obtenga la misma cadena de bits.
- Es extremadamente complicado que, al tener un dato de entrada A y su correspondiente salida del algoritmo de Hashing B, se obtengan unos datos C resultado de modificar los datos A, de modo que se obtenga el mismo resultado al ejecutar el algoritmo de Hashing.

- Son algoritmos muy rápidos, además suelen estar implementados en Hardware.

También se ha añadido la capacidad de cifrar imágenes con varias capas, por ejemplo, RGB y otras formas de representación por capas. Se realiza el mismo tratamiento a las distintas capas presentes en la imagen por separado. Sin embargo, sería posible realizar un procesamiento extra en caso de tener más de una capa en la imagen, de modo que se mezcle la información de los pixeles entre capas.

CAPÍTULO 7

Resultados y análisis estadístico

En este capítulo se trata de recopilar los resultados obtenidos tras la implementación del criptosistema. Se comprueba que el objetivo de encriptar imágenes naturales e imágenes médicas se ha cumplido, además, se comprueban algunos parámetros estadísticos para determinar el correcto funcionamiento del criptosistema. Para la ejecución del algoritmo se puede proporcionar una clave para determinar los valores iniciales de los mapas caóticos, pero dejando vacío este campo, se genera una clave aleatoria, dado que no es importante cual es la clave para el propósito de este capítulo debido a que no tiene apenas carga computacional, la clave en cada prueba será generada aleatoriamente.

El equipo utilizado para realizar las pruebas es un portátil con un procesador Intel Core i7-8550U 1.8GHz y se ha utilizado el editor de código Visual Studio Code para implementar los distintos algoritmos y realizar las pruebas.

7.1– Complejidad

Se incluye un estudio de complejidad previo a las tablas de tiempos para poder razonar con el resultado obtenido en este apartado.

Cuando decimos que un algoritmo tiene mayor complejidad respecto a otro, queremos decir que su tiempo de ejecución depende en mayor medida del tamaño de la entrada proporcionada a dicho algoritmo, por ejemplo en nuestro caso, si tenemos un criptosistema, el cual tiene una complejidad $O(1)$, significa que dada una imagen de cualquier tamaño, el algoritmo terminaría la encriptación siempre en el mismo tiempo, es decir, no depende del tamaño de la imagen. Si consideramos que el tamaño de la imagen es el número de píxeles, y este es $n = xy$ (siendo x e y los subíndices de la matriz de píxeles), la complejidad del criptosistema podría ser también $O(n)$, en este caso estamos diciendo que el tiempo de ejecución es lineal, el criptosistema terminará su ejecución en un tiempo lineal en función de su tamaño.

El artículo en el que se basa el criptosistema [18] implementado expone su propio estudio de complejidad, sin embargo, creo que hay algún detalle que no se ha tenido en cuenta, el cual puede alargar en gran medida el tiempo de ejecución del algoritmo completo.

7.1.1. Complejidad según el artículo

El algoritmo completo cuenta con 6 partes importantes que dependen de la entrada suministrada, estas partes son las encontradas en la zona izquierda de la figura que contiene el diagrama de flujo del algoritmo, es decir, el mezclado de filas, ACM, el mezclado de toda la imagen, la codificación en ADN, la sustitución de ADN y la decodificación de ADN. La complejidad será una suma de todas las complejidades. Contamos con que el tamaño de una imagen es $m \times n$, los índices de la matriz que representa los píxeles de la imagen. Para la primera parte, el mezclado de filas, tenemos una complejidad $O(mn)$. La segunda parte, ACM, tiene una parte en la que se calcula el mayor primo menor que $\min(m,n)$, y este será el número de iteraciones k que ejecutará el algoritmo de ACM, por tanto, tenemos en esta parte una complejidad de $O(\min(m,n)mn)$. La tercera parte, la mezcla de todos los pixeles, ocupa una complejidad de $O(2m + 2n) \leq O(m + n)$. Las últimas tres partes operan en cada píxel de la imagen por lo que tenemos una complejidad de $O(mn)$. Uniendo todas las partes, tenemos que la complejidad es $4 \times O(mn) + O(\min(m,n)mn) + O(m + n)$. Podemos decir que la parte con más peso en complejidad está siendo la del algoritmo de ACM, por tanto, finalmente la complejidad del algoritmo es $O(\min(m,n)mn)$.

7.1.2. Posible error en el cálculo de la complejidad

El artículo trata el algoritmo de ACM modificado como un algoritmo de ejecución de k iteraciones de ACM, sin embargo, el algoritmo que se presenta en el artículo no es un simple algoritmo de ACM, ya que está modificado para aceptar cualquier tamaño de imagen, este cambio hace que se divida la imagen en α cuadrados, y se aplique ACM en cada cuadro individual de la imagen un total de k veces. El estudio de complejidad presentado en el artículo está contando solo con el caso ideal en el que $\alpha = 1$. Esto afecta al resultado final de la complejidad debido a que esta es efectivamente la parte de mayor peso a la hora de estudiar la complejidad. Además, en el artículo parece que se está tomando $k = \min(m,n)$, cuando en realidad, k es el mayor primo menor que $\min(m,n)/2$, en el caso peor, $k = \min(m,n)/2$.

Algorithm 5: Modified ACM

```

Input : Image  $I$  of dimension  $m \times n$ 
Output: Shuffled image  $I'$ 
1 Length of side of square :  $N = \min(m, n)$ ;
2 Number of squares :  $\alpha = \lceil \max(m, n)/N \rceil$ ;
3  $L = N - (\max(m, n) \bmod N)$ ;
4  $\mathcal{O} = \lfloor L/(\alpha - 1) \rfloor$ ;
5  $k = \text{largest prime less than } (N/2)$ ;
6  $(x_0, y_0) = (1, 1)$ ;
7 for  $i = 1$  to ( $\alpha$ ) do
8    $Sq = I[x_0 : (x_0 + N - 1), y_0 : (y_0 + N - 1)]$ ;
9   Apply ACM on  $Sq$  for  $k$  iterations using Equation (11) ;
10  Append  $Sq$  to  $I'$  in appropriate place;
11  if  $i == \alpha - 1$  then
12    if  $m > n$  then
13       $x_0 = m - N$ ;
14    else
15       $y_0 = n - N$ ;
16  else
17    if  $m > n$  then
18       $x_0 = x_0 + N - \mathcal{O}$ ;
19    else
20       $y_0 = x_0 + N - \mathcal{O}$ ;
21 return  $I'$ ;

```

Figura 7.1: Pseudocódigo del algoritmo ACM modificado en el artículo

Si observamos el pseudocódigo del algoritmo presentado en el artículo, en la línea 5 tenemos el cálculo de un número k que contará más tarde el número de iteraciones que hay que realizar

para ACM, vamos a contar este cálculo con complejidad $O(1)$. En la línea 7 se inicia el bucle "grande" en el cual se encuentran las operaciones que realmente tienen peso en este algoritmo, y es un bucle de α iteraciones, estas iteraciones son las que han sido despreciadas en el estudio original de complejidad y que dan una gran carga computacional al algoritmo. Para cada iteración del bucle se realizan k iteraciones de ACM, por tanto, la complejidad del algoritmo de ACM modificado es $O(\alpha \times (\min(m,n)/2) \times m \times n)$. Sabemos que $\alpha = \lceil (\max(m,n)) / (\min(m,n)) \rceil$, por tanto, podemos decir que la complejidad es del orden $O(\lceil (\max(m,n)) / (\min(m,n)) \rceil \times \min(m,n)/2 \times m \times n)$.

Si suponemos el caso particular donde $m=n$, entonces tenemos $O(\min(m,n)/2 \times m \times n)$, que es justo el caso que trata el artículo [18] en su estudio de complejidad.

Si suponemos $m > n$ y $\lceil \frac{m}{n} \rceil = 2$ obtenemos $O(m^2 \times n)$, este es justo el resultado obtenido en tras el estudio de complejidad del artículo[18].

Recordando que el tamaño de nuestro problema es $m \times n$, tenemos que la complejidad de nuestro algoritmo es de orden mayor que un algoritmo con complejidad $O(N)$ y mucho menor que un algoritmo con $O(N^2)$.

Finalmente si $n > m$; $O(N) < O(\frac{\alpha}{2} \times m^2 \times n) \ll O(N^2)$. Intercambiamos m por n si $m > n$.

7.2– Imágenes naturales

El criptosistema debe ser capaz de encriptar imágenes naturales. Dada una imagen en escala de grises cualquiera con los píxeles codificados con 8-bits, es posible realizar el proceso de encriptación y desencriptación obteniendo finalmente la misma imagen. A continuación, se muestra un ejemplo de una imagen natural con un histograma que representa la cantidad de píxeles que posee con cada intensidad de tono de gris. La imagen de ejemplo puede verse en las figuras 7.2 y 7.3 que muestran respectivamente la imagen original y su histograma.



Figura 7.2: Imagen natural ejemplo

Se puede apreciar cómo tras encriptar la imagen 7.4, el histograma 7.5 resultante reparte de forma mucho más equitativa los tonos de gris entre los píxeles.

Es posible además realizar la encriptación de una imagen natural que disponga de varias capas 7.6, por ejemplo una imagen en RGB 7.7, el proceso sin embargo se triplica, debido a que la forma de implementar este sistema siendo fiel al criptosistema propuesto en el artículo, es encriptando por separado las tres capas y uniéndolas posteriormente en una misma imagen.

Se aporta a continuación la salida tras la ejecución usando una imagen a color. 7.8 7.9

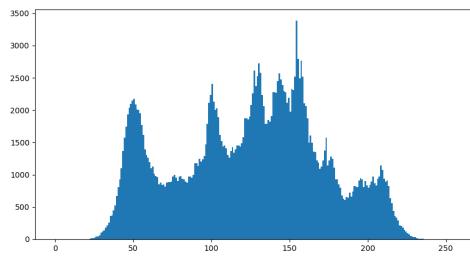


Figura 7.3: Histograma de la imagen natural de ejemplo

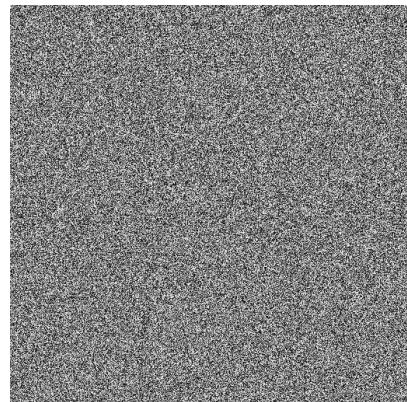


Figura 7.4: Imagen natural ejemplo en escala de grises encriptada

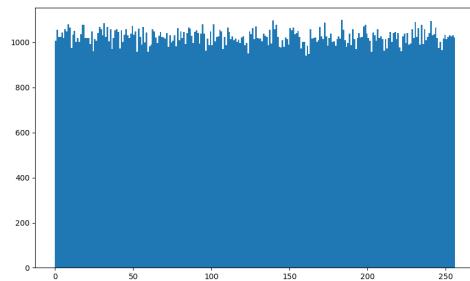


Figura 7.5: Histograma de la imagen natural de ejemplo encriptada



Figura 7.6: Imagen natural ejemplo a color

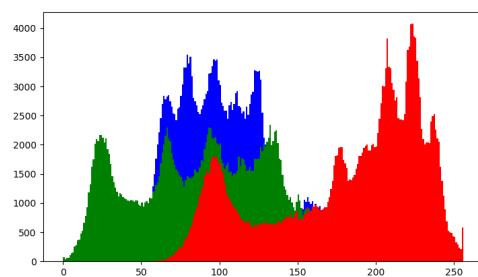


Figura 7.7: Histograma de la imagen natural de ejemplo a color

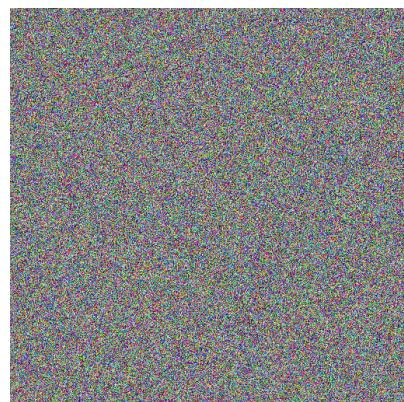


Figura 7.8: Imagen natural ejemplo encriptada a color

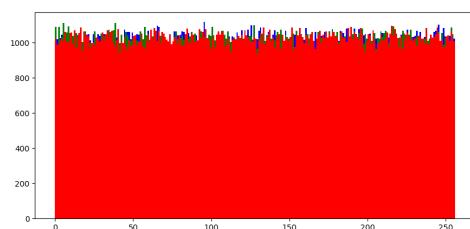


Figura 7.9: Histograma de la imagen natural de ejemplo encriptada a color

7.3– Imágenes médicas

El criptosistema debe ser capaz de encriptar imágenes médicas DICOM. Dada una imagen DICOM cualquiera con los píxeles codificados con 16-bits, usando el mismo criptosistema que el utilizado para encriptar imágenes naturales, se puede encriptar y desencriptar dicha imagen obteniendo finalmente el dato de entrada original. La imagen que se expone como ejemplo es la que se aprecia en la figura 7.10 y su respectivo histograma se encuentra en la figura 7.11.

Aunque puede parecer a primera vista que el histograma es incorrecto, no es así. El tamaño de la imagen es 2140×1760 , por tanto, tenemos un total de 3766400 de píxeles. El histograma 7.12 es una representación reescalada del histograma 7.11 para que se pueda observar con mayor detalle. Si realizamos una media de la cantidad de píxeles por cada valor en el rango de valores [200,900] obtenemos aproximadamente 5000, si multiplicamos los 700 rangos de píxel por 5000 obtenemos 3500000, por tanto, casi todos los píxeles se encuentran en ese rango. Es muy común encontrar imágenes dicom que no abarquen toda la escala de valores de la imagen. Esto se debe al propio tamaño de la imagen, no hay suficientes píxeles para tener los 65536 posibles valores de pixel. También tenemos que debido a la gran cantidad de campos de información extra que pueden añadirse, en muchas ocasiones, la mayoría de estos campos no se rellenan, por tanto, los valores de los píxeles se suelen situar en un rango muy bajo respecto al total (en esta imagen aproximadamente en [200,900] del rango total [0,65535]).

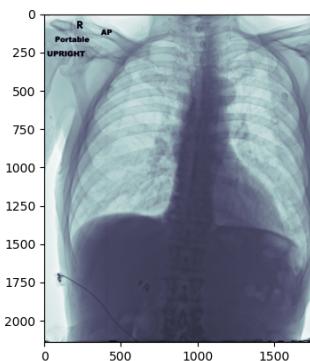


Figura 7.10: Imagen dicom ejemplo

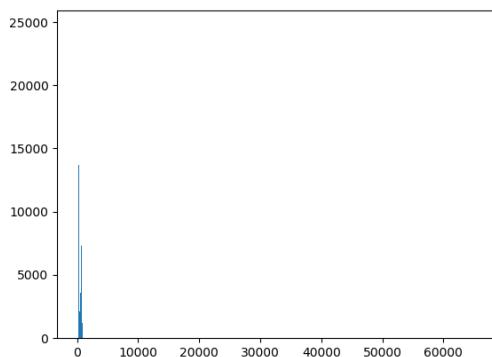


Figura 7.11: Histograma de la imagen dicom de ejemplo

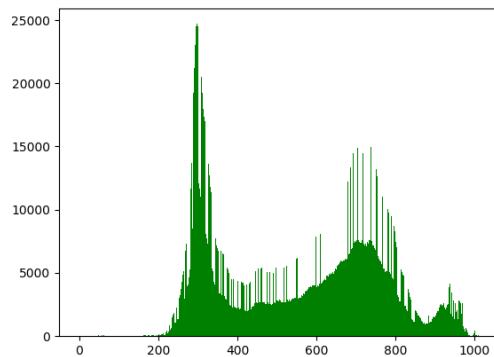


Figura 7.12: Histograma de la imagen dicom de ejemplo reescalado

Tras ejecutar el criptosistema obtenemos la imagen cifrada como se muestra en la figura 7.13, por otro lado, se puede ver en la figura 7.14 su histograma, los valores de los píxeles están repartidos de forma similar entre todos los valores posibles de pixel.

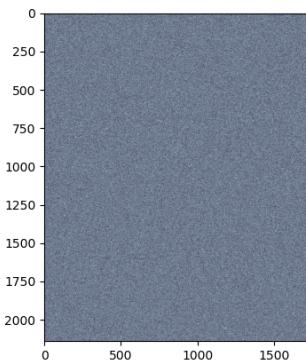


Figura 7.13: Imagen dicom de ejemplo encriptada

Esta vez, comprobamos que hay una línea blanca, esto significa que no hay ningún píxel con ese valor, además, podemos ver que los valores de los píxeles están distribuidos pero no de forma muy equitativa, esto no sucede así según el artículo 7.15 donde se ve claramente como se reparten los valores de los tonos de gris de los distintos píxeles en la imagen.

Esto puede significar tres cosas:

- **Error en mi código.** La primera y más obvia es que al programar el algoritmo he cometido al menos un error que genera una salida distinta a la esperada, sin embargo, no parece ser este el problema ya que he repasado en múltiples ocasiones dicho código.
- **Fallo en el artículo.** Es posible que se haya colocado en el artículo un histograma que no corresponde con la imagen encriptada presentada, o simplemente se han presentado resultados dónde la salida obtenida es muy buena.
- **El histograma del artículo se presenta en otro formato.** En lugar de utilizar todos los valores de píxel (65536) en una imagen DICOM, se ha transformado la imagen en una

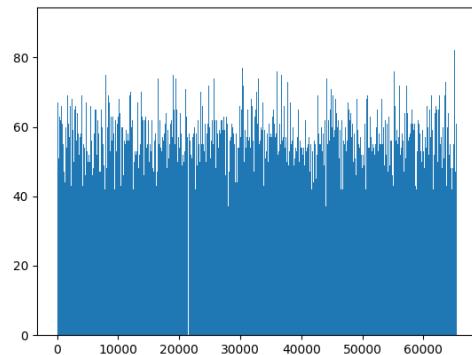


Figura 7.14: Histograma de la imagen dicom de ejemplo encriptada

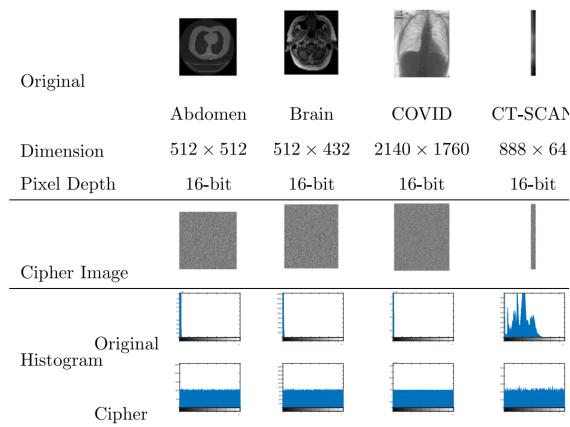


Figura 7.15: Histogramas de resultados en el artículo

imagen de 8-bits y se ha representado el resultado del histograma solo con 256 valores de pixel.

7.4– Tiempos de ejecución con múltiples imágenes

Se ha comprobado el tiempo de ejecución en múltiples imágenes médicas y se han anotado dichos tiempos en la siguiente tabla 7.1 que recoge la ruta de cada imagen, la forma que tiene cada imagen $m \times n$, el tiempo en segundos de ejecución y el tiempo de ejecución en horas. Las imágenes son las obtenidas como se explica en el apartado 6.4.

Cuadro 7.1: Tiempos de ejecución para imágenes DICOM

Path	forma	Tiempo para encriptar segundos	Tiempo horas
Path1	"(2965, 2729)"	4284	1.19
Path2	"(2140, 1760)"	3629	1.008
Path3	"(2330, 2846)"	7680	2.133
Path4	"(819, 512)"	180	0.05
Path5	"(819, 512)"	125	0.035
Path6	"(512, 512)"	102	0.028
Path7	"(512, 512)"	129	0.036
Path8	"(2330, 2846)"	8898	2.472
Path9	"(2330, 2604)"	7592	2.109
Path10	"(358, 512)"	125	0.035
Path11	"(358, 512)"	143	0.040
Path12	"(2164, 2441)"	6976	1.938
Path13	"(1760, 2140)"	7443	2.068
Path14	"(1760, 1760)"	2301	0.639
Path15	"(2133, 2846)"	6499	1.805
Path16	"(2320, 2846)"	8752	2.431
Path17	"(2330, 2451)"	13140	3.655
Path18	"(2322, 2846)"	8300	2.306
Path19	"(409, 512)"	101	0.028
Path20	"(409, 512)"	132	0.037
Path21	"(2985, 3001)"	17393	4.831

Las rutas a las que se hace referencia en la tabla 7.1 son muy largos y no caben, por tanto, se muestran en forma de lista a continuación.

- **Path1:** 01-08-2012-NA-XR CHEST PA AND LATERAL-04866/2.000000-Lateral L-70577/1-1.dcm
- **Path2:** 01-15-2012-NA-XR CHEST AP ONLY-24124/1001.000000-NA-00046/1-1.dcm
- **Path3:** 01-17-2012-NA-XR CHEST AP ONLY-65366/1.000000-Lateral L-10044/1-1.dcm
- **Path4:** 02-14-2012-NA-CT PE CHEST-63916/1.000000-SCOUT-04116/1-1.dcm
- **Path5:** 02-14-2012-NA-CT PE CHEST-63916/1.000000-SCOUT-04116/1-2.dcm
- **Path6:** 02-14-2012-NA-CT PE CHEST-63916/2.000000-locator-16446/1-1.dcm
- **Path7:** 02-14-2012-NA-CT PE CHEST-63916/4.000000-3MM-28806/1-001.dcm

- **Path8:** 01-26-2012-NA-XR CHEST AP PORTABLE-18461/1.000000-AP-77707/1-1.dcm
- **Path9:** 01-11-2012-NA-XR CHEST AP PORTABLE-28662/1.000000-AP-72624/1-1.dcm
- **Path10:** 02-05-2012-NA-CT PE CHEST-37142/1.000000-SCOUT-25496/1-1.dcm
- **Path11:** 02-05-2012-NA-CT PE CHEST-37142/1.000000-SCOUT-25496/1-2.dcm
- **Path12:** 01-29-2012-NA-XR CHEST AP PORTABLE-16862/1.000000-AP-00155/1-1.dcm
- **Path13:** 01-05-2012-NA-XR CHEST AP PORTABLE-50189/1001.000000-NA-12847/1-1.dcm
- **Path14:** 01-04-2012-NA-XR CHEST AP PORTABLE-18268/1001.000000-NA-45815/1-1.dcm
- **Path15:** 01-14-2012-NA-XR CHEST AP PORTABLE-76275/1.000000-AP-39520/1-1.dcm
- **Path16:** 01-16-2012-NA-XR CHEST AP ONLY-35226/2.000000-Lateral L-07806/1-1.dcm
- **Path17:** 01-07-2012-NA-XR CHEST AP PORTABLE-58729/1.000000-AP-50053/1-1.dcm
- **Path18:** 01-26-2012-NA-XR CHEST AP PORTABLE-46730/1.000000-AP-35790/1-1.dcm
- **Path19:** 02-22-2012-NA-CT CHEST W CONTRAST-94020/1.000000-Surview-97077/1-1.dcm
- **Path20:** 02-22-2012-NA-CT CHEST W CONTRAST-94020/1.000000-Surview-97077/1-2.dcm
- **Path21:** 01-12-2012-NA-XR CHEST PA AND LATERAL-39042/1.000000-PA-95803/1-1.dcm

La información que obtenemos de la tabla 7.1 se obtiene de observar el tamaño de las imágenes en relación al tiempo que tarda el algoritmo de encriptación en ejecutar. Como puede verse no es demasiado eficiente, esto sucede principalmente por el algoritmo de Arnold's Cat Map donde pasa la mayor parte del tiempo. Además, se comprueba que el estudio de complejidad concuerda con los resultados obtenidos ya que a medida que crece el tamaño de las imágenes, el tiempo de ejecución se incrementa de un modo mayor que lineal, sin embargo no llega a ser cuadrático.

7.5– Análisis de varianza

La varianza de una imagen es una medida estadística que indica como de uniforme se reparten los píxeles en la imagen, es decir, si tenemos una imagen donde todos sus píxeles tienen el mismo tono de gris (por ejemplo, una imagen toda blanca), su varianza es 0. Si la imagen tiene sus tonos de gris distribuidos de forma similar entre todos los tonos posibles, su varianza es muy alta. De este modo, una imagen con una varianza muy alta tendrá un histograma donde se puede comprobar que sus píxeles están distribuidos de forma uniforme. En la tabla 7.2 puede comprobarse como aumenta en gran medida el valor de la varianza tras ejecutar el algoritmo. Las rutas de las imágenes son las mismas que las del apartado 7.4. En el caso de la varianza, se obtienen resultados similares a los obtenidos en el artículo en el que se basa el criptosistema.

La forma de calcular la varianza σ^2 en una imagen $m \times n$ es la de la ecuación 7.1

$$\sigma^2(I) = \frac{\sum_{x=1}^m \sum_{y=1}^n (I(x, y) - \mu)^2}{m \times n} \quad (7.1)$$

Cuadro 7.2: Tabla de varianza para imágenes DICOM

Path	forma	varianza original	varianza cifrado
Path1	"(2965, 2729)"	1.519×10^6	3.579×10^8
Path2	"(2140, 1760)"	4.424×10^4	3.578×10^8
Path3	"(2330, 2846)"	3.534×10^7	3.577×10^8
Path4	"(819, 512)"	3.631×10^5	3.578×10^8
Path5	"(819, 512)"	3.631×10^5	3.578×10^8
Path6	"(512, 512)"	1.888×10^5	3.579×10^8
Path7	"(512, 512)"	1.677×10^5	3.594×10^8
Path8	"(2330, 2846)"	9.529×10^5	3.578×10^8
Path9	"(2330, 2604)"	8.732×10^5	3.579×10^8
Path10	"(358, 512)"	4.920×10^5	3.575×10^8
Path11	"(358, 512)"	4.920×10^5	3.575×10^8
Path12	"(2164, 2441)"	6.279×10^5	3.580×10^8
Path13	"(1760, 2140)"	4.171×10^5	3.580×10^8

siendo μ la media de intensidad de píxel como se define en 7.2, $I(x,y)$ es un píxel de la matriz que define la imagen.

$$\mu = \frac{\sum_{x=1}^m \sum_{y=1}^n (I(x, y))}{m \times n} \quad (7.2)$$

7.6– Análisis de entropía

La entropía es una medida que nos indica como de difícil es predecir el tono de gris de los píxeles de la imagen. En una imagen cuyos pixeles son todos iguales, su entropía es 0, debido a que no es nada complicado adivinar cuál será el tono de gris del siguiente pixel. Como se muestra en la tabla 7.3, el valor de la entropía puede llegar a casi duplicarse tras pasar por el algoritmo, sin embargo, en el artículo [18], los valores de la entropía suelen estar cerca de 8 una vez se encripta la imagen. Un motivo por el cual puede que suceda esto es que hayan mostrado algunos de los mejores resultados en el artículo.

Cuadro 7.3: Tabla de entropía para imágenes DICOM

Path	forma	entropía original	entropía cifrado
Path1	"(2965, 2729)"	3.0571286631790757	4.814706925257225
Path2	"(2140, 1760)"	2.7769348923833466	4.81271477994432
Path3	"(2330, 2846)"	4.303848708769495	4.814333434638513
Path4	"(819, 512)"	2.6151208667064667	4.780939574315287
Path5	"(819, 512)"	2.6151208667064667	4.780939574315287
Path6	"(512, 512)"	2.827588848065096	4.758433572378436
Path7	"(512, 512)"	2.1239382680025365	4.758825056354747
Path8	"(2330, 2846)"	3.382455984442967	4.8143226054962325
Path9	"(2330, 2604)"	3.380526475085614	4.814120468858047
Path10	"(358, 512)"	2.6493354358045007	4.730640864363006
Path11	"(358, 512)"	2.6493354358045007	4.730640864363006
Path12	"(2164, 2441)"	3.4403557365513615	4.813793404964338
Path13	"(1760, 2140)"	2.7419087489770173	4.81271058897335

El cálculo de la entropía se define dado un conjunto de símbolos S (un total de 256 en 8 bits y 65536 en 16 bits) del siguiente modo 7.3

$$H(S) = \sum_{i=1}^n p_i \log\left(\frac{1}{p_i}\right) \quad (7.3)$$

siendo p_i la probabilidad de aparición de la intensidad de gris i -ésima en la imagen. Se calcula para cada $p_i \in S$.

7.7– Imágenes Homogéneas

Uno de los retos para un criptosistema para imágenes es conseguir un buen encriptado en una imagen con zonas grandes dónde el tono de los píxeles es muy similar o es el mismo. Si observamos el histograma de una imagen en la figura 7.16 podemos comprobar que se trata de una imagen con todos sus píxeles con el mismo valor, 255. Todos los píxeles de la imagen son blancos.

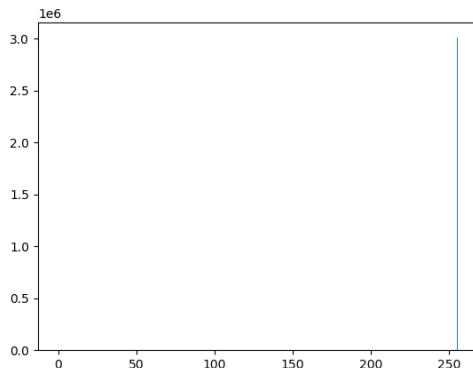


Figura 7.16: Histograma de imagen en blanco

Si ejecutamos el algoritmo con dicha imagen no podemos detectar a simple vista ninguna pista sobre el estado original de la imagen a pesar de que inicialmente era toda blanca, en la figura 7.17 puede verse el resultado, junto con su histograma en la figura 7.18.

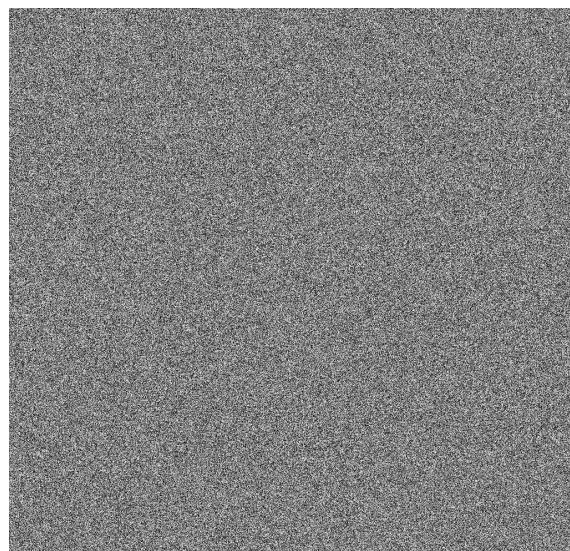


Figura 7.17: Imagen de píxeles blancos encriptada

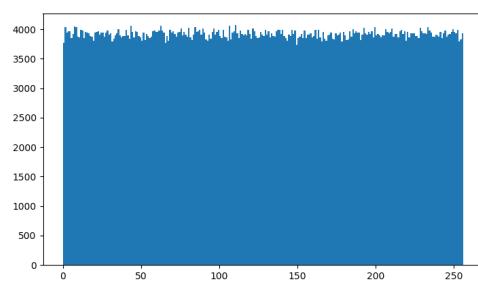


Figura 7.18: Histograma de imagen en blanco encriptada

CAPÍTULO 8

Mejoras y futuras vías de progreso

En todo proyecto de este tipo, es siempre posible realizar aún más de lo que ya se ha hecho, ampliando la capacidad de encriptar más tipos de imágenes u otro tipo de datos, mejorando el rendimiento, mejorando la seguridad por otros medios, etc. En este proyecto en concreto se ha implementado un criptosistema capaz de cifrar imágenes médicas y naturales utilizando el operador binario propuesto por un artículo científico, el cual actúa sobre un conjunto de elementos $S = \{1, 2, \dots, p-2\}$ para algún primo p . El proceso se realiza siguiendo varios pasos de mezclado y reordenamiento de píxeles, este reordenamiento se lleva a cabo utilizando el algoritmo de ACM, este algoritmo es el que conlleva una mayor carga computacional. Esto se debe en parte a que este algoritmo está definido para imágenes cuadradas, y se debe hacer una modificación del mismo para subdividir la imagen original en imágenes más pequeñas, en cada subimagen se debe ejecutar por separado dicho algoritmo, en caso de que la imagen original tenga un lado mucho mayor que el otro, puede causar que se tarde mucho en terminar con la ejecución completa del algoritmo. La mejor forma de solucionar esto es eliminando esta parte y sustituyéndola con otro algoritmo de reordenamiento de píxeles o elementos en una matriz que no esté definido solo para una matriz cuadrada.

El criptosistema implementado durante la realización de este TFG está enfocado a las matrices (imágenes), pero una posible forma de ir más allá sería tratar de hacer que este criptosistema con algunas modificaciones fuese capaz de ser un criptosistema de propósito general. Esto significaría iniciar un nuevo proyecto con el objetivo de encontrar una función biyectiva que a cualquier dato de entrada de cualquier tipo le asocie una matriz, de este modo el criptosistema sería capaz de cifrar cualquier dato de entrada. Este sería el caso ideal aunque probablemente, debido a la cantidad de tipos de dato que podríamos encontrar, las matrices serían muy grandes y sería computacionalmente inviable, sin embargo si queremos que el criptosistema encripte solo diez tipos de datos distintos, podemos tratar de encontrar las diez funciones biyectivas que asocian a cada elemento de cada tipo de dato, una matriz.

8.1– Ejemplo futuro proyecto (criptosistema universal)

Respecto a la propuesta de un futuro proyecto dónde se crea un criptosistema para múltiples tipos de datos simultáneamente se explica a continuación mediante un ejemplo simple.

El ejemplo más simple de otro tipo de dato distinto a una imagen podría ser un texto. Podemos considerar una matriz $m \times n$ donde n es el mayor número de caracteres que hay en una línea de texto con más caracteres y m es el total de líneas. Cada carácter se podría codificar con un número

real. De este modo, el primer carácter codificado correspondería con la posición (0,0) de la matriz y cada carácter se corresponde con la posición de la matriz en relación con el texto. La ausencia de un carácter en una posición se podría codificar como 0.

Si tenemos el texto:

hola, este es

un texto

de prueba.

Cuadro 8.1: Codificación caracteres ejemplo

Carácter	Código
«vacío»	0
a	1
b	2
c	3
...	...
y	24
z	25
,	26
.	27

Si codificamos cada carácter según la tabla 8.1 nuestra matriz sería la siguiente (nótese que en este caso el número primo p es 29 y podemos aplicar el operador \otimes):

$$\begin{bmatrix} 8 & 15 & 12 & 1 & 26 & 0 & 5 & 19 & 20 & 5 & 0 & 5 & 20 \\ 21 & 14 & 0 & 20 & 5 & 23 & 20 & 15 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 16 & 18 & 22 & 5 & 2 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Teniendo esta matriz, podemos aplicar de nuevo los conocimientos aplicados a criptosistemas para imágenes en esta matriz. Una vez tengamos la forma de transformar todos los tipos de datos que nos interesan en matrices, podemos codificar de igual modo cada tipo de dato con un número. Si tenemos una colección de datos de distintos tipos, por ejemplo, un documento que incluye texto e imágenes, podemos de nuevo asociar a cada tipo de dato un número, crear otra matriz de tipos de datos y encriptar dicha matriz para esconder también el tipo de dato de cada matriz (la matriz de tipos de datos debería ser reconocible de algún modo).

8.2– Posibles mejoras

El criptosistema actualmente es capaz de encriptar y desencriptar imágenes de 8 y 16 bits, con una o varias capas. Todo ello con la opción de incluir una contraseña que haga las veces de clave simétrica.

Algunas de las posibles mejoras que puedan aplicarse al criptosistema son:

- Sustitución de ACM por otro algoritmo de difusión más rápido.
- Mejorar la fortaleza de la clave, actualmente se genera la clave de forma aleatoria o a partir de la contraseña usando funciones Hash.
- Búsqueda de un mapa aún más caótico que el ICM-2D
- Procesos de confusión y difusión entre capas de una imagen con más de una capa.

- Estudio de resultado de encriptar y, tras el añadido de cierto ruido, desencriptar. Se busca minimizar el daño causado al resultado final.
- Distintas mejoras en la interfaz gráfica.

CAPÍTULO 9

Conclusiones

Distintas organizaciones en todo el mundo tienen la necesidad de transmitir información a través de canales de comunicación no seguros. Esto hace que datos sensibles alcancen destinos indeseados por el emisor, por lo que surge la necesidad de crear criptosistemas que aseguren la información de aquellos que puedan interceptarla. En este TFG se ha implementado un criptosistema para imágenes de 8 y 16 bits diseñado en un artículo de investigación, además, se han comprobado empíricamente los resultados de dicho artículo.

9.1– Qué se ha hecho

El desarrollo de este trabajo ha conllevado una buena cantidad de horas de dedicación y esfuerzo, alcanzando así los objetivos propuestos inicialmente.

En el primer capítulo (1) se hacía un repaso a la justificación del proyecto de la necesidad de crear un criptosistema capaz de encriptar una imagen de 16 bits. Esto se debía sobre todo a la falta de software en la actualidad capaz de dicha tarea.

En el segundo (2) y tercer (3) capítulo se expone la metodología a seguir durante el proyecto, el seguimiento de una metodología es, entre otras cosas, lo que diferencia a un ingeniero de otro tipo de profesionales en el campo de la informática. Es importante conocer los estándares, el camino a seguir durante la creación, el desarrollo y la finalización de un proyecto. Seguir un camino predefinido es importante para no perder el rumbo y alcanzar la meta final. Sin embargo, esta ruta predefinida debe ser lo suficientemente abierta para permitir al proyecto reaccionar al entorno durante la ejecución. Un proyecto está vivo en el sentido de que durante el tiempo que está en marcha, sucederán eventos que modificarán el entorno del mismo, estos eventos afectarán el transcurso del proyecto en mayor o menor medida. Si diseñamos una ruta demasiado restrictiva para el proyecto es posible que acabemos asfixiándolo y el proyecto fracase, pero puede suceder lo mismo si dejamos la ruta demasiado libre, ya que podríamos perdernos en mitad del camino. También se debe señalar que gracias a seguir una metodología se puede dar una continuidad al proyecto posteriormente ya que otro ingeniero podrá usar la experiencia recogida por sus predecesores para mejorar o mantener el proyecto, o para otros proyectos de índole similar.

En los capítulos 4 y 5 se presenta la solución obtenida al realizar el proyecto. Tras dejar clara la metodología que se seguirá, presentando los posibles riesgos, planes de contingencia, alcance del proyecto y demás, se debe realizar de forma inmediata un estudio teórico profundo de las herramientas y el entorno matemático que rodea a un proyecto de este tipo. Un ingeniero informático

que se encargue de proyectos de temáticas variadas que tengan que ver con distintos campos de la ciencia se encontrará con que hay un mundo inmenso de conocimiento que es imposible abarcar por completo. Sin embargo, esto no puede ser impedimento para cumplir con los objetivos exigidos. Este estudio consumirá un gran número de horas del proyecto, aunque puede ser algo frustrante las primeras veces que se realiza un proyecto así debido a que, aparentemente, no hay avances, con el tiempo se mejora esta capacidad de estudio y además, conocer el campo sobre el que se está trabajando es un gran avance. Por otra parte, la presentación del algoritmo obtenido en forma de resumen de sus distintos componentes es también esencial para mostrar el trabajo realizado.

En algún momento se debe dejar constancia de los problemas que han ido surgiendo durante la implementación del algoritmo, y por ello se incluye el capítulo 6, si otro ingeniero pretende implementar un software similar o con las mismas herramientas, sabrá a qué capítulo dirigirse para encontrar algunas de las problemáticas con las que se puede encontrar. Por supuesto, en este capítulo se presenta la solución utilizada para resolver la situación.

Una vez se ha finalizado el desarrollo del criptosistema se deben realizar pruebas para comprobar el resultado final, esto puede verse en el capítulo 7. Se realizan pruebas que comprueban el resultado con imágenes de 8 y 16 bits. Además, se comprueban algunas medidas como los tiempos de ejecución, entropía y varianza. Respecto al tiempo de ejecución se debe comprobar si es coherente con la complejidad que tiene el algoritmo de encriptado, por ello se incluye también un estudio de la complejidad.

Por último se hace mención a los posibles caminos de expansión del proyecto y posibles nuevos proyectos que pudiesen surgir a partir de este en el capítulo 8.

9.2– Conocimiento generado

Esta sección explica el conocimiento generado y que aporta valor a la propia universidad de Sevilla.

Hay una gran cantidad de artículos que proponen criptosistemas para imágenes, sin embargo, poco se habla de criptosistemas para imágenes DICOM, en este proyecto se trata de implementar un algoritmo que encripta dicho tipo de imágenes y comprobar si los resultados son viables, dando paso así a posibles nuevos criptosistemas más robustos y polivalentes. Aunque no muchos criptosistemas se centran en imágenes DICOM, empiezan a aparecer nuevos artículos que tratan el tema. [16]

Este proyecto proporciona varias cosas. Al alumno que realiza el proyecto le interesa sobre todo el carácter didáctico de realizarlo, permite practicar sus habilidades de programación, lectura y comprensión de papers (no es tan sencillo muchas veces), gestión de proyectos, conocimiento matemático, etc. En cuanto a la universidad de Sevilla le viene bien que haya alumnos realizando trabajos de este tipo, investigación sobre papers, ya que de estos proyectos pueden surgir novedades interesantes que otorguen prestigio a la institución. A personas que quieran aportar algo nuevo al proyecto o crear el suyo propio utilizando conocimiento que se muestra en esta memoria debido a que parte de su proyecto ya habrá sido explorado y documentado.

9.3– Conclusiones personales

Personalmente este trabajo me ha hecho crecer en el campo de la investigación científica, me ha permitido aumentar mi capacidad de búsqueda de información, comprensión de artículos científicos, mejora de capacidad de programación, asentar fundamentos matemáticos, etc. Ya había hecho algunos trabajos en otras asignaturas cuya temática consistía en la comprensión de un ar-

tículo científico, sin embargo, usualmente son artículos algo más anticuados a los que se les han dado muchas vueltas, o su contenido estaba muy limitado a un solo concepto, por ejemplo, un algoritmo de búsqueda, un algoritmo de optimización, etc. Trabajar con artículos más «punteros» es un acercamiento mayor a lo que realmente supone la investigación científica, supone los primeros pasos para unir lo que otros investigadores han creado como si fuesen piezas, para crear algo nuevo y mejor.

Crear un criptosistema completo ha sido una experiencia muy placentera, leer los artículos, aprender sobre las distintas herramientas matemáticas que existen y «traducirlas» a un lenguaje de programación y comprobar que tras todo ello se obtiene algo que funciona es una experiencia muy gratificante. Sin duda lo mejor del proyecto ha sido el hecho de informarme sobre el contexto teórico para poder aplicarlo después en el proyecto. Por otro lado, también he de decir que, realizar la memoria en sí, es tal vez la parte más pesada del proyecto, sin embargo, esto probablemente se debe a la poca experiencia a la hora de realizar memorias tan largas y sin duda con algo más de práctica se me haría mucho más sencillo.

La elección de este trabajo de fin de grado fue un proceso complejo, aunque busqué distintas ofertas en varios campos de la informática, terminé por decantarme por la criptografía, algo que me llamaba mucho la atención y que hasta el segundo cuatrimestre del último curso no se ve en la carrera. Por tanto, busqué en las propuestas de TFG los trabajos disponibles y comprobé que todos estaban ofertados por mi actual tutor, así que le pregunté que trabajos podían ser más interesantes, y el resultado final ha sido muy satisfactorio.

9.4– Cierre del proyecto

Por último, queda proceder con el cierre del proyecto, solo se realiza el inicio del cierre ya que habrá partes que no sea posible realizarlas aún como son:

- **Obtener la aceptación final del producto:** Solo será posible tras finalizar la entrega de la memoria.
- **Solicitar retroalimentación del cliente sobre el proyecto:** Solo se obtendrá dicha retroalimentación del tutor, pero el resto de profesores que califican el proyecto tras la entrega no han visto aún el proyecto.

Por otra parte, el objetivo del proyecto era implementar el criptosistema propuesto, comprobar si los resultados obtenidos en el paper eran correctos y documentarlo todo de forma detallada con una memoria. Por tanto, el trabajo está hecho conforme a los requisitos. Esto puede verse en la sección 9.1.

En el apartado 9.2 se resume el conocimiento generado y lecciones aprendidas durante el proyecto. En 9.3 Se muestra el desempeño del proyecto. La documentación del capítulo 6 es también parte de la recopilación de las lecciones aprendidas.

Otra cosa que se debe comprobar es el tiempo invertido en el proyecto, en este caso se ha optado por contabilizar las horas invertidas mediante la aplicación web Clockify. [1] Esta aplicación permite hacer un seguimiento de las distintas tareas que se realizan en un proyecto determinado, estas tareas pueden contener un tag/etiqueta que permita identificar el tipo de tarea que es 9.1. Por ejemplo, en el proyecto se ha añadido una tarea llamada «Implementar funciones estadísticas» con el tag «Implementar algoritmo». Esto permitirá visualizar en un diagrama en forma de tarta el tiempo invertido en cada tarea por cada tag. 9.2

Un problema encontrado a la hora de utilizar la herramienta en su versión gratuita es que, una vez añadido un tag es posible archivarlo, sin embargo, este permanecerá en las tareas etiquetadas,



Figura 9.1: Añadir tarea en Clockify

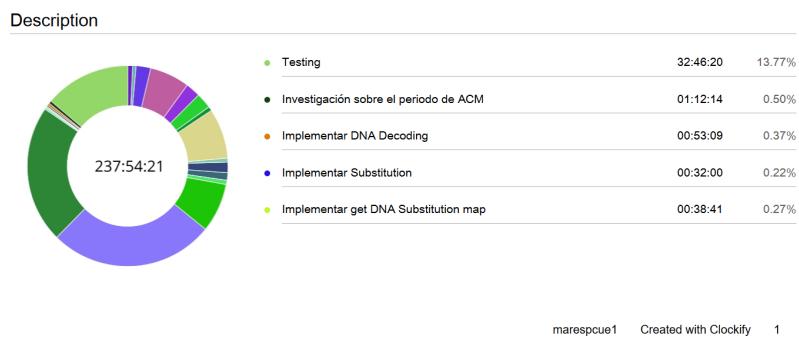


Figura 9.2: Diagrama de tarta de tareas por etiqueta

● Implementar vector inicial	01:14:19	0.43%
● Implementar Mixing	00:39:15	0.22%
● Lectura y comprensión de la documentación	52:38:24	18.21%
● Escritura documentación	76:29:07	26.47%
● Depuración	18:41:59	6.47%
● Implementar Mix rows	01:46:16	0.61%
● Implementar Infinite collapse map	02:49:11	0.98%
● Lectura y compleción de la documentación	04:04:26	1.41%
● Implementar DNA Encoding	01:21:53	0.47%
● Investigación	19:10:29	6.63%
● Implementar get Rule Map	01:49:45	0.63%
● Repaso de código	03:09:28	1.09%
● Implementar Modified ACM	05:47:56	2.01%

Figura 9.3: Etiquetas del diagrama de tarta de Clockify

● Implementar Modified ACM	05:47:56	2.01%
● Investigación sobre el operador inverso	05:36:33	1.94%
● Implementar funciones estadísticas	15:18:19	5.30%
● Implementar Descifrado	05:41:11	1.97%
● Interfaz gráfica	33:54:10	11.73%
● Ejecución de código en la nube	00:49:53	0.29%
● Unión de los algoritmos de cifrado	01:56:05	0.67%

Figura 9.4: Etiquetas del diagrama de tarta de Clockify

por tanto para eliminarlo es necesario acceder a cada tarea de forma independiente y eliminar dicha etiqueta. Por ejemplo, en lugar de crear una etiqueta para el tipo de tarea «Implementar algoritmo», pretendí crear también otras etiquetas que hicieran las veces de subetiqueta, de modo que apareciesen en el diagrama proporcionado por la aplicación. Al ver que el resultado quedaba tan complejo, decidí archivar dichas etiquetas, pero el resultado ha seguido siendo el mismo.

Aunque gran parte del proyecto ha consistido en trabajar con un PC, no siempre se han contabilizado las horas de forma absolutamente estricta. Por ejemplo, ha habido momentos que se ha dedicado a leer información relacionada con el TFG sin preocuparme por activar el contador, no se han contado reuniones con expertos como podría ser el tutor del TFG, y hay momentos en los que surge la inspiración y se utiliza un bolígrafo y un papel para empezar a desarrollar ideas para implementar el algoritmo. A fecha de 20 de Junio, los resultados mostrados por la herramienta pueden comprobarse en 9.5.



Figura 9.5: Tiempo invertido mensualmente hasta 20/06/2022

Hasta el momento (20/06/2022), las horas recogidas por clockify son las que se muestran en la tabla 9.1.

Algo que si se puede apreciar observando la estimación inicial del cuadro 3.2 es que la implementación del código fuente ha sido mucho más corta finalmente, esto ha sucedido debido a que se ha invertido mayor tiempo a comprender en profundidad la teoría que hay detrás de los algoritmos, y aunque se han tenido algunos problemas durante la implementación que ha habido que resolver, se esperaba encontrar muchos más. Por otro lado, la escritura de la documentación ha sido mucho más costosa en tiempo. En la carrera se han escrito una gran cantidad de memorias, sin embargo, esta puede haber costado algo más debido a que se realiza en Latex [5] y al cuidado que hay que dedicar a una memoria de TFG. Algo que sería muy útil en un proyecto de la misma

Cuadro 9.1: Horas según Clockify a fecha de 20/06/2022

Description	Time (h)
Escrutura documentación	76:29:07
Investigación	19:10:29
Implementar Mix rows	01:46:16
Repaso de código	03:09:28
Lectura y compleción de la documentación	04:04:26
Implementar Infinite collapse map	02:49:11
Implementar DNA Encoding	01:21:53
Implementar get Rule Map	01:49:45
Interfaz gráfica	33:54:10
Ejecución de código en la nube	00:49:53
Unión de los algoritmos de cifrado	01:56:05
Implementar Modified ACM	05:47:56
Implementar Descifrado	05:41:11
Implementar funciones estadísticas	15:18:19
Investigación sobre el operador inverso	05:36:33
Implementar get DNA Substitution map	00:38:41
Implementar Mixing	00:39:15
Lectura y comprensión de la documentación	52:38:24
Investigación sobre el periodo de ACM	01:12:14
Testing	32:46:20
Implementar Substitution	00:32:00
Implementar DNA Decoding	00:53:09
Implementar vector inicial	01:14:19
Depuración	18:41:59
Total	289:01:03

temática es contar con dos personas implicadas en el desarrollo directo del proyecto, una dedicada en mayor medida a la implementación y otra dedicada en mayor medida a la documentación y la memoria.

Otra cosa a tener en cuenta es que por varios motivos personales que no podían preverse a la hora de identificar los riesgos, no se ha podido seguir por completo con el cronograma inicial, esto ha causado un retraso en la ruta crítica del proyecto, sin embargo, el cronograma inicial proponía terminar el proyecto en abril, por tanto el retraso no ha casado la cancelación del proyecto.

Como última apreciación, se esperaba inicialmente terminar el TFG en 307 horas, a pesar de algunos cambios en el cronograma durante la ejecución del proyecto, este se ha terminado con éxito cumpliendo los objetivos requeridos y contabilizando 289 horas en la herramienta Clockify. Aún falta por contabilizar el tiempo que va a ser invertido en la preparación de la defensa, últimos repasos a todo el contenido de la memoria y tratar de hacer disponible la interfaz en la nube para que pueda ser utilizada durante la defensa. Con ello y las horas no contabilizadas en la herramienta, el tiempo invertido en el proyecto superará las 300 horas.

CAPÍTULO 10

Apéndices

.1– Acta de adjudicación de TFG

Como modelo de Acta de Constitución de este trabajo, se incluye a continuación un acta de adjudicación de Trabajo Fin de Grado.

**ASIGNATURA TRABAJO FIN DE GRADO****SOLICITUD DE ADJUDICACIÓN**

Apellidos y nombre del alumno: Espejo Cuenca, Marcel

D.N.I.: 29550049V

Fecha de solicitud: 10-11-2021

Correo electrónico: marespcue1@alum.us.es

Teléfono: 665880976

Titulación: II-Tecnologías Informáticas

Créditos que restan para acabar la carrera: 60

Título del proyecto: Un criptosistema para imágenes médicas basado en un nuevo operador

Tutor: José Andrés Armario Sampalo

Departamento: Matemática Aplicada I

Objetivos del proyecto: En este TFG se debe implementar un criptosistema para imágenes médicas basado en un nuevo operador binario. El criptosistema utiliza diferentes herramientas matemáticas (secuencias caóticas, computación basada en ADN, etc) que deben ser bien documentadas. El punto de partida es un artículo internacional reciente. Se espera corroborar experimentalmente los resultados que aparecen en el documento.

El Tutor,

El Alumno,

Fdo.: José Andrés Armario Sampalo

Fdo.: Marcel Espejo Cuenca

Figura 1: Acta de constitución

Bibliografía

- [1] Varios Autores. *Clockify*. fecha de consulta: 09 de Junio de 2022. 2022. URL: <https://app.clockify.me/>.
- [2] Varios Autores. *Dataset DICOM*. fecha de consulta: 08 de Junio de 2022. 2022. URL: <https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=70226443>.
- [3] Varios Autores. *dicomstandard*. fecha de consulta: 08 de Junio de 2022. 2022. URL: <https://www.dicomstandard.org/about-home>.
- [4] Varios Autores. *Free Online Gantt Chart Software*. fecha de consulta: 04 de Noviembre de 2021. 2021. URL: <https://www.onlinegantt.com/>.
- [5] Varios Autores. *Latex*. fecha de consulta: 09 de Junio de 2022. 2022. URL: <https://www.latex-project.org/>.
- [6] Varios Autores. *matplotlib*. fecha de consulta: 17 de Junio de 2022. 2022. URL: <https://matplotlib.org/>.
- [7] Varios Autores. *numpy*. fecha de consulta: 17 de Junio de 2022. 2022. URL: <https://numpy.org/about/>.
- [8] Varios Autores. *opencv*. fecha de consulta: 17 de Junio de 2022. 2022. URL: <https://pypi.org/project/opencv-python/>.
- [9] Varios Autores. *pydicom*. fecha de consulta: 17 de Junio de 2022. 2022. URL: <https://pypi.org/project/pydicom/>.
- [10] Varios Autores. *sympy*. fecha de consulta: 17 de Junio de 2022. 2022. URL: <https://www.sympy.org/en/index.html>.
- [11] Varios Autores. *The Cancer Imaging Archive*. fecha de consulta: 08 de Junio de 2022. 2022. URL: <https://www.cancerimagingarchive.net/>.
- [12] M. S. Baptista. «Cryptography with chaos». En: *PHYSICS LETTERS A* 1 (1998), pág. 5.
- [13] Marcel Espejo Cuenca. *GitHub marespcue1*. fecha de consulta: 20 de Junio de 2022. 2022. URL: <https://github.com/marespcue1/CriptosistemaOperadorBinarioTFG>.
- [14] Saber N. Elaydi. *Discrete Chaos*. Chapman Hall/CRC, 2007.
- [15] Qigui Yang Jianghong Bao. «Period of the discrete Arnold cat map and general cat map». En: *Nonlinear Dynamics* 1 (2012), pág. 11.
- [16] P. Murali Joan S. Muthu. «A novel DICOM image encryption with JSMP map». En: *Optik* 1 (2022), pág. 18.
- [17] T. L. Carroll L. M. Pecora. «Synchronization in Chaotic Systems». En: *Physical Review Letters* 1 (1990), pág. 6.
- [18] C. Bhaya P. Mishra et al. «A novel binary operator for designing medical and natural image cryptosystems». En: *Signal Processing: Image Communication* 1 (2021), pág. 13.

-
- [19] Gabriel Peterson. «Arnold's Cat Map». En: *Math 45* 1 (1997), pág. 7.
 - [20] PMI. *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)*. Project Management Institute, Inc, 2012.
 - [21] Y. Zhou W. Caoa Y. Mao. «Designing a 2D infinite collapse map for image encryption». En: *Signal Processing: Image Communication* 1 (2020), pág. 13.