

Comparativa de algoritmos de cifrado simétrico aplicados a imágenes en escala de grises



Pablo D. Origlia

Supervisor: Dra. Andrea Rey

Instituto de Tecnología e Ingeniería
Universidad Nacional de Hurlingham

Tesina para aspirar al título de
Licenciado en Informática

Diciembre 2025

Índice general

Índice de Figuras	V
Índice de Tablas	VII
1. Introducción	5
2. Estado del Arte	9
2.1. Cifrado simétrico	11
2.1.1. Cifrado César	12
2.1.2. Cifrado de flujo	13
2.1.3. Cifrado por bloques	15
2.2. Ataques criptográficos	16
2.3. Modos de operación	18
3. Materiales y Métodos	25
3.1. Algoritmos	25
3.2. Métricas	30
3.3. Datos y preprocesamiento de las imágenes	42
3.4. Implementación experimental y entorno de pruebas	42
3.5. Flujo de procesamiento y análisis	46
4. Experimentos y Resultados	47
5. Conclusiones y Trabajo Futuro	57
Bibliografía	59

Índice de Figuras

1.1. Esquema jerárquico de la Criptología.	6
2.1. Línea de tiempo de la Criptografía.	10
2.2. Escenario básico inicial.	11
2.3. Esquema básico del cifrado de flujo.	13
2.4. Esquema básico del cifrado por bloques.	16
2.5. Esquema de cifrado (arriba) y descifrado (abajo) en modo ECB.	20
2.6. Esquema de cifrado en modo CBC.	20
2.7. Esquema de cifrado en modo CFB.	21
2.8. Esquema de cifrado en modo OFB.	22
2.9. Esquema de cifrado en modo CTR.	23
3.1. Esquema de una ronda de Feistel.	26
3.2. Esquema de una ronda de SPN.	27
3.3. Imagen original P (izquierda) e imagen encriptada C (derecha).	31
3.4. Histograma de la imagen original P (arriba) e imagen encriptada C (abajo).	32
3.5. Diferencia de histogramas: $ H_P(i) - H_C(i) $	33
3.6. Histograma de la imagen diferencia $P - C$	34
3.7. Desviación absoluta respecto de la media $H_D(i)$	35
3.8. <i>Pipeline</i> de procesamiento de imágenes, cifrado, cálculo de métricas y generación de gráficos.	45
4.1. Coeficiente de correlación por algoritmo.	48
4.2. Desviación máxima en función del tamaño de imagen y por algoritmo.	49
4.3. Desviación irregular por algoritmo.	49
4.4. Desviación respecto del histograma uniforme en función del tamaño de imagen y por algoritmo.	50
4.5. Relación señal-ruido pico por algoritmo.	51
4.6. Tasa de cambio del número de píxeles por algoritmo.	51

4.7. Intensidad promedio de cambio unificada en función del tamaño de imagen y por algoritmo.	52
4.8. Entropía de Shannon en función del tamaño de imagen y por algoritmo. . . .	52
4.9. Entropía de Shannon por algoritmo.	53
4.10. Tiempo de ejecución por algoritmo.	54
4.11. Carga de CPU por algoritmo.	54
4.12. Uso de memoria por algoritmo.	55

Índice de Tablas

2.1. Tabla de verdad de la operación XOR.	13
2.2. Diferencias entre cifrado por bloques y cifrado de flujo.	16
2.3. Comparación de modos de operación.	24

Resumen

La Criptografía es la disciplina que estudia procesos para codificar y decodificar información para que solo personas autorizadas puedan operar sobre el mensaje para transmitirlo y sean capaces de recibirlo de manera comprensible. Diversas técnicas de encriptación se han utilizado a lo largo de la historia y continúan empleándose en la actualidad, abarcando cada vez más campos de aplicación. Existe un gran número de algoritmos disponibles que varían en complejidad y seguridad, dependiendo del tipo de comunicación y la sensibilidad de la información que va a ser compartida. En este trabajo se analizan algoritmos de cifrado simétrico aplicados a imágenes en escala de grises. Se evalúan cifrados por bloques —AES, ARC2, Blowfish, CAST, DES, 3DES— y cifrados por flujo —ARC4, Salsa20 y ChaCha20—, a través de métricas como tiempo de ejecución, uso de memoria, uso de CPU, entropía, desviación máxima, desviación irregular, desviación respecto del histograma uniforme, correlación, tasa de cambio del número de píxeles, intensidad promedio de cambio unificada y relación señal-ruido pico. La metodología incluye un *pipeline* automatizado que realiza la conversión a escala de grises, recortes proporcionales, cifrado con modificación controlada de píxeles y generación de gráficas de los resultados. Se desarrollan *scripts* en Python, buscando precisión en la medición del uso de recursos computacionales y robustez criptográfica. Los resultados evidencian un compromiso entre eficiencia y seguridad en algunos algoritmos, como 3DES y ARC2 mientras que AES y ARC4 se destacan en velocidad o bajo consumo de recursos. El estudio contribuye a identificar el algoritmo óptimo según el escenario de uso.

Agradecimientos

A Natalia, mi compañera de vida y de aventuras, por acompañarme desde el primer día con su apoyo incondicional y su optimismo infinito. Por cada crítica, cada mirada atenta y cada aporte que enriqueció cada momento. Estuvo presente en cada materia, en cada clase y en cada detalle de este recorrido.

A mi hijo Santiago, que creció junto a este proceso. Deseo que este camino sea para él un pequeño ejemplo para la etapa que pronto comienza, y que pueda transitarla con dignidad y justicia.

A mis padres, Stella y Julio, por brindarme las herramientas fundamentales para formarme y desarrollarme, y por haber sostenido con paciencia y apoyo todos estos años de estudio.

A Mary y Daniel, por su acompañamiento total durante todos estos años de esfuerzo.

A mis compañeros de cursada, con quienes aprendimos a complementarnos y acompañarnos en cada materia que compartimos, brindándonos siempre lo mejor de cada uno y creciendo juntos.

A Fernando, que veló por nuestro desarrollo académico y humano y que logró construir tanto en tan poco tiempo.

A todos los integrantes del LIDEC, por abrirme las puertas de un grupo excepcional y permitirme compartir experiencias valiosas tanto en lo académico como en lo personal.

Y finalmente, a Andrea, mi directora, por tener siempre la palabra justa en el momento exacto, por su guía oportuna y por impulsarme a avanzar y concluir esta etapa de la mejor manera imaginable.

A todos, muchas gracias.

Capítulo 1

Introducción

En el contexto de la seguridad de la información, la Criptografía constituye el proceso de ocultar o codificar datos con el objetivo de que solo los destinatarios autorizados puedan interpretarlos correctamente. Herramientas criptográficas han sido utilizadas a lo largo de la historia de la humanidad, desde los jeroglíficos egipcios hasta los sofisticados algoritmos actuales, y continúan empleándose en una gran variedad de campos. Esta disciplina, que en sus inicios era considerada un arte, se ha consolidado como uno de los pilares fundamentales de la ciberseguridad moderna, siendo el componente básico de numerosos mecanismos, sistemas y protocolos de defensa. Actividades cotidianas como el uso de tarjetas bancarias, la generación de contraseñas, la navegación en la web, las operaciones financieras en línea o el teletrabajo serían inviables sin la existencia de la Criptografía.

El campo que abarca todo este estudio es la Criptología, la cual se divide en dos ramas principales: la Criptografía, encargada de diseñar y aplicar mecanismos de cifrado y descifrado, y el Criptoanálisis, centrado en el estudio de técnicas para vulnerar dichos sistemas (en la Figura 1.1 se presenta un esquema jerárquico). La Criptografía moderna se sustenta en sólidos fundamentos matemáticos, informáticos e ingenieriles, y se clasifica en tres grandes grupos: algoritmos de clave secreta o simétrica, algoritmos de clave pública o asimétrica, y funciones *hash*. Los primeros emplean una clave compartida tanto para cifrar como para descifrar, pudiendo ser de flujo (a nivel de *bit* o *byte*) o de bloque (en unidades de tamaño fijo). En cambio, la Criptografía asimétrica utiliza pares de claves matemática y computacionalmente relacionadas, como en los algoritmos Rivest Shamir Adleman (RSA), Diffie-Hellman o la Criptografía de curva elíptica [10]. Por su parte, las funciones *hash* garantizan la integridad de los datos y optimizan procesos de búsqueda en bases de datos.

A pesar de los avances en técnicas de cifrado, también proliferan los métodos de ataque y las vulnerabilidades asociadas a una gestión deficiente de claves. La debilidad, la reutilización, la falta de rotación o el almacenamiento inseguro pueden comprometer la seguridad de los sistemas. Entre las amenazas más relevantes se encuentran los ataques internos, la ausencia de auditorías y la falta de resiliencia organizacional. Para mitigar estos riesgos, se emplean sistemas

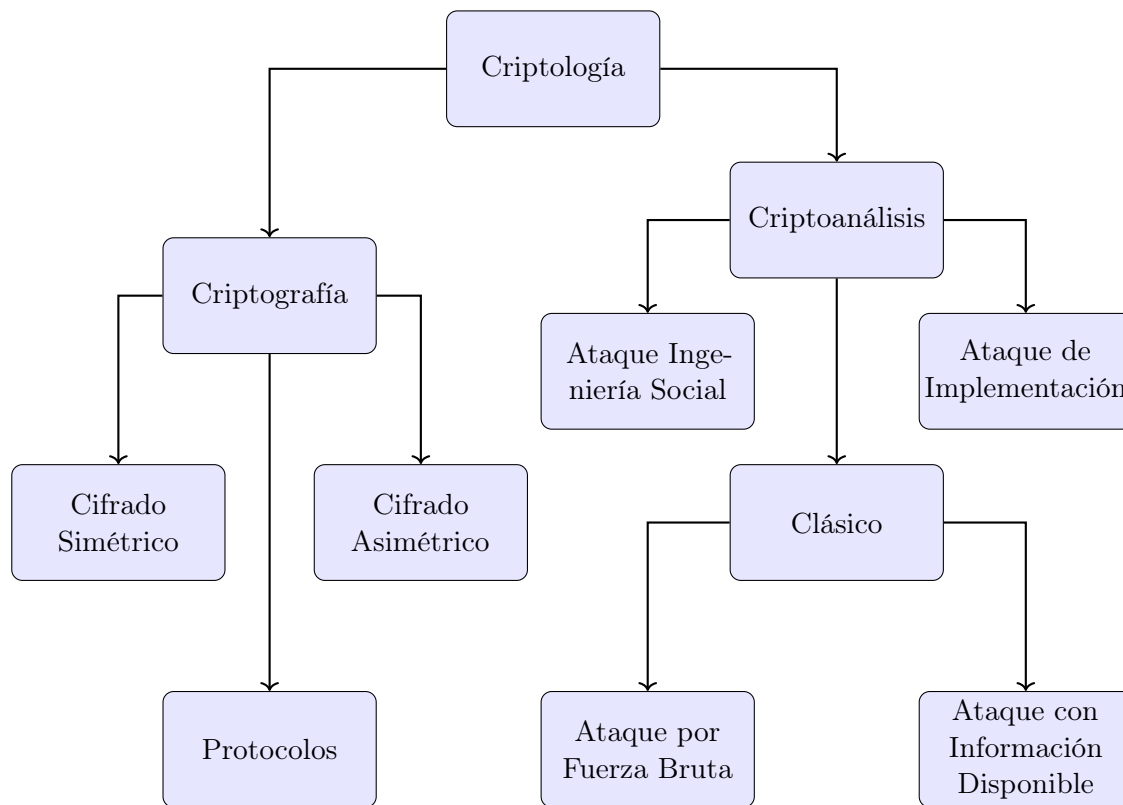


Figura 1.1: Esquema jerárquico de la Criptología.

de gestión de claves electrónicas, módulos de seguridad de *hardware* y, más recientemente, propuestas emergentes como la Criptografía cuántica, la cual ofrece propiedades de seguridad únicas aunque enfrenta desafíos que aún limitan su aplicación práctica.

Dentro de este panorama, la Criptografía aplicada a imágenes digitales constituye un área de investigación creciente. A diferencia de los textos, las imágenes poseen características particulares como alta redundancia espacial, gran tamaño y correlación entre píxeles adyacentes, lo que desafía los paradigmas tradicionales de cifrado. En este contexto, la evaluación mediante métricas específicas se convierte en un componente esencial para validar la robustez, seguridad y eficiencia de los algoritmos. Una de las razones fundamentales para utilizar métricas de evaluación es la necesidad de demostrar resistencia frente a ataques estadísticos, diferenciales y por análisis visual. Por ejemplo, la correlación entre píxeles adyacentes en imágenes cifradas debe tender a cero, lo cual evidencia la ruptura de la estructura original [3]. Asimismo, métricas como la entropía permiten cuantificar el nivel de aleatoriedad en la imagen cifrada [26], siendo deseables valores cercanos al máximo teórico para garantizar uniformidad y dificultar los ataques por patrones.

Además de los criterios de seguridad, la evaluación de esquemas de cifrado de imágenes considera la calidad visual. En este sentido, métricas como el error cuadrático medio (*Mean Square Error*, MSE) y la relación señal-ruido pico (*Peak Signal Noise Ratio*, PSNR) permiten

cuantificar el grado de alteración introducido por el cifrado [2]. Un bajo valor de PSNR indica que la imagen cifrada resulta visualmente irreconocible respecto a la original, lo que refuerza la confidencialidad frente a ataques estadísticos o inspecciones humanas. En escenarios prácticos, donde la transmisión de imágenes puede sufrir pérdidas o ruido, se emplean métricas como la tasa de cambio del número de píxeles (*Number of Changing Pixel Rate*, NPCR) y la intensidad promedio de cambio unificada (*Unified Averaged Changing Intensity*, UACI). Altos valores en ambas métricas reflejan una elevada sensibilidad del algoritmo, garantizando el efecto avalancha ante mínimas modificaciones en la imagen o en la clave [33].

Por otra parte, propiedades como la homogeneidad, el contraste y la intensidad sirven para detectar patrones visuales persistentes tras el proceso de cifrado. Un algoritmo robusto debe reducir al mínimo estas características, favoreciendo una distribución aleatoria de los valores de píxel y aumentando su resistencia frente a análisis estadísticos y diferenciales [3]. Asimismo, en entornos con restricciones de cómputo, como dispositivos embebidos o sistemas de recursos limitados, resulta crucial considerar métricas de rendimiento como el tiempo de ejecución y la complejidad computacional [3], además de los criterios de seguridad.

Finalmente, la seguridad de un esquema criptográfico no depende únicamente de su estructura matemática, sino también de su sensibilidad a la clave. Un sistema robusto debe generar criptogramas completamente distintos ante mínimas variaciones en la clave, ampliando así el espacio efectivo de claves y reduciendo la probabilidad de éxito de ataques por fuerza bruta. Tal como lo establece Shannon, alcanzar el ideal de “secreto perfecto” requiere que el número de claves posibles sea al menos igual al número de mensajes [26, 27].

La organización de este trabajo es como sigue. En el Capítulo 2 se describen distintos cifrados y ataques criptográficos. Los algoritmos empleados en este trabajo, así como las métricas de evaluación y el conjunto de datos sobre el cual se aplica la metodología propuesta se encuentran en el Capítulo 3. El Capítulo 4 incluye los experimentos realizados y los resultados obtenidos. Finalmente, en el Capítulo 5 se presentan las conclusiones y las posibles líneas de trabajo futuro.

Capítulo 2

Estado del Arte

La Criptografía ha sido utilizada por miles de años; sin embargo, fue recién en el siglo XIX cuando se comenzaron a analizar los supuestos bajo los cuales un sistema criptográfico puede considerarse suficientemente seguro para su uso en aplicaciones sensibles. Para ello, se emplea el principio de Kerckhoffs [16], que postula que **un sistema criptográfico debe ser seguro aun si el atacante conoce todos los detalles del sistema, excepto la clave utilizada para cifrar los mensajes**. Esta premisa se generalizó luego de atravesar dos guerras mundiales, período durante el cual las comunicaciones seguras eran prioritarias, aunque la implementación de los sistemas criptográficos permanecía en reserva. Esta práctica generaba enormes vulnerabilidades por no cumplir un principio básico de seguridad.

Incluso antes de la adopción formal del principio de Kerckhoffs, diversas civilizaciones desarrollaron métodos para proteger mensajes estratégicos como la escítala espartana (500 a.C.), que consistía en un bastón alrededor del cual se enrollaba una tira de pergamino con el mensaje, revelándose solo al desenrollarlo sobre otro bastón de igual diámetro. Posteriormente, el cifrado de César (50 a.C.) introdujo un desplazamiento sistemático de letras en el alfabeto, mostrando los primeros indicios de un cifrado basado en clave secreta [21]. Durante el Renacimiento, el disco cifrante de Alberti (1467) permitió variaciones polialfabéticas de manera mecánica y el cuadro de Vigenère (1553) perfeccionó estos esquemas mediante claves repetitivas, sentando las bases de la Criptografía polialfabética moderna [16].

Con los avances en la Teoría de la Computación [29] y de la Información [26], surgió la necesidad de desarrollar un marco general para abordar los problemas de la Criptografía moderna [27]. En el siglo XX, la Criptografía evolucionó hacia métodos más formales y matemáticamente fundamentados. El cifrado de un solo uso (*one-time pad*, OTP) de Vernam (1917) introdujo la noción de seguridad perfecta mediante claves tan largas como el mensaje. Más adelante, la máquina Enigma (1923) y el posterior desarrollo del Colossus (1943) representaron hitos tanto en la Criptografía mecánica como en la automatización del Criptoanálisis [21].

Con la proliferación de sistemas informáticos, surgió la necesidad de garantizar comunicaciones seguras entre los distintos componentes de las redes en desarrollo. En este contexto, a

mediados de la década de 1970, el *National Institute of Standards and Technology* (NIST) emitió un llamado para establecer un estándar de cifrado de información a nivel gubernamental, con el apoyo de la *National Security Agency* (NSA). La propuesta presentada por la *International Business Machines Corporation* (IBM) fue finalmente validada, dando lugar al estándar de cifrado de datos (*Data Encryption Standard*, DES) [18]. Este sistema fue pionero en el uso masivo de un cifrado cuyos componentes internos estaban disponibles para análisis público, permitiendo la evaluación de su seguridad de manera directa.

Finalmente, la aparición del algoritmo RSA [23] en 1978 marcó un hito al dar inicio a la Criptografía de clave pública, posibilitando el intercambio seguro de información sin necesidad de compartir previamente una clave secreta. Con el paso del tiempo y el aumento de la capacidad de cómputo, se evidenció la vulnerabilidad del DES frente a ataques de fuerza bruta, lo que impulsó la búsqueda de alternativas más robustas. En este contexto, el NIST estableció en 2001 al AES, basado en el diseño Rijndael de Daemen y Rijmen, como el nuevo estándar internacional de cifrado, consolidando así una etapa clave en la evolución hacia algoritmos más seguros y eficientes [9, 19]. Todo este desarrollo histórico se muestra en la Figura 2.1.



Figura 2.1: Línea de tiempo de la Criptografía.

Para el análisis de los diferentes escenarios donde se utilizan distintos sistemas criptográficos, se pueden plantear los elementos principales que se relacionan para el desarrollo de una comunicación segura entre los actores involucrados, esto es, en un escenario donde dos entidades desean comunicarse (Emisor y Receptor) a través de un canal inseguro donde un tercero (Oponente) puede operar para tratar de descubrir los mensajes que circulan a través del canal de comunicación entre Emisor y Receptor como se puede observar en la Figura 2.2.

Además de los actores principales, se cuenta con un mensaje en claro, sin cifrar (x) y una clave (k) para el cifrado y descifrado del mensaje encriptado (y). El supuesto principal es que el Oponente tiene acceso al mensaje cifrado y a las funciones para cifrar, $\text{enc}(\cdot)$, y para descifrar, $\text{dec}(\cdot)$, pero desconociendo la clave que es compartida solamente entre el Emisor y Receptor a través de un canal seguro previamente al establecimiento de la comunicación a analizar.

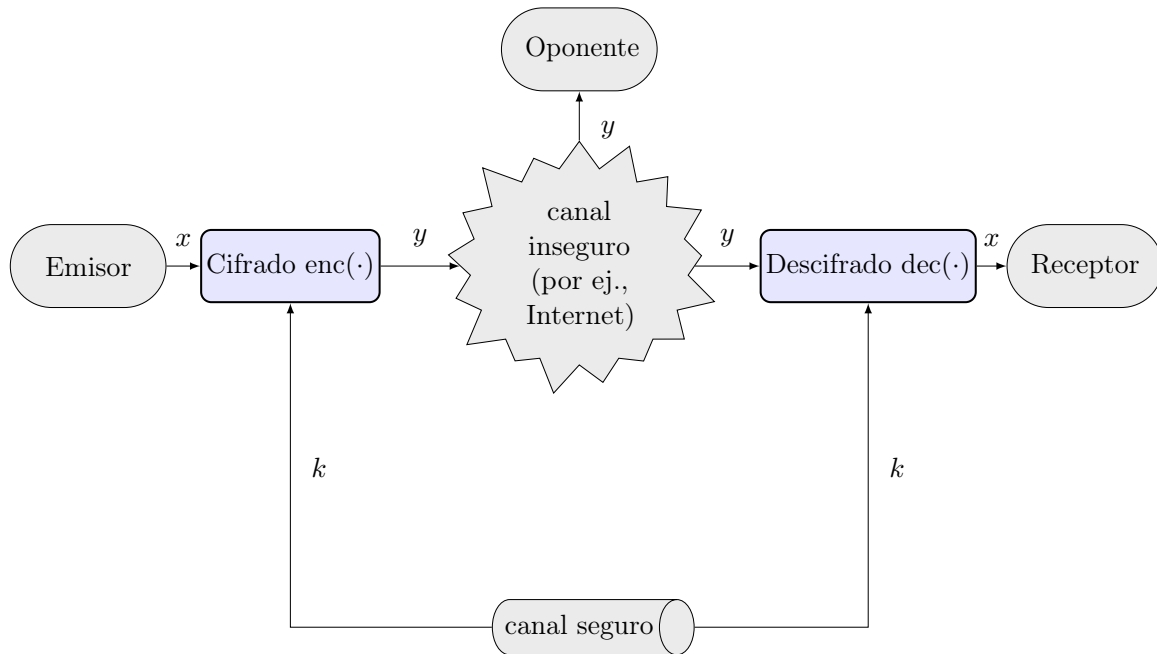


Figura 2.2: Escenario básico inicial.

2.1. Cifrado simétrico

El cifrado simétrico constituye una técnica criptográfica en la cual una misma clave es utilizada tanto para el proceso de cifrado como para el de descifrado del mensaje. En este esquema, Emisor y Receptor deben compartir previamente una clave secreta k , la cual garantiza la confidencialidad de los datos transmitidos o almacenados. La solidez de este paradigma depende de manera esencial de la protección de dicha clave, ya que una eventual filtración comprometería integralmente la seguridad del sistema [27].

Los algoritmos de cifrado simétricos se clasifican principalmente en dos categorías: cifrado por bloques y cifrado de flujo. Ambos transforman texto en claro en texto cifrado, aunque mediante enfoques distintos. El cifrado por bloques opera sobre segmentos de datos de longitud fija (por ejemplo, 64 o 128 *bits*), aplicando rondas sucesivas de sustitución y permutación, como ocurre en algoritmos representativos tales como DES o estándar de cifrado avanzado (*Advanced Encryption Standard*, AES). Y por su parte, el cifrado de flujo procesa la información de manera continua, *bit a bit* o *byte a byte*, generando un flujo de clave (*keystream*, $s(k)$) a partir de la clave k , que se combina mediante operaciones XOR con el mensaje en claro. El cifrado simétrico destaca por su eficiencia computacional, lo que lo hace especialmente adecuado para aplicaciones que requieren procesar grandes volúmenes de información en tiempos reducidos. Pero, este enfoque presenta como limitación fundamental la necesidad de un canal seguro para la distribución de claves, lo cual constituye un desafío en entornos distribuidos o de gran escala [3].

Los cifrados por bloques resultan apropiados para escenarios que demandan alto nivel de seguridad y cifrado de datos estáticos o de longitud conocida (por ejemplo, almacenamiento o archivos). En contraste, los cifrados de flujo son más adecuados cuando se requiere cifrado continuo, eficiente y de baja latencia, como en comunicaciones en tiempo real. No obstante, la reutilización de claves en cifrado de flujo puede comprometer gravemente la seguridad, mientras que los cifrados por bloques ofrecen más resiliencia bajo dicha condición, especialmente cuando se utilizan modos de operación robustos.

Desde una perspectiva histórica, el cifrado simétrico tiene raíces en la Antigüedad, con ejemplos como la escítala espartana o el cifrado César, que implementaban técnicas de transposición y sustitución, respectivamente. Con el avance de la matemática y la computación, en el siglo XX este paradigma adquirió formalización teórica gracias a los trabajos de Shannon [26, 27]. Posteriormente, en 1977, DES fue adoptado como estándar criptográfico, aunque con el tiempo se demostró vulnerable a ataques de fuerza bruta. En respuesta a esta amenaza, en 2001, el NIST estableció el AES como nuevo estándar internacional, basado en el diseño Rijndael desarrollado por Daemen y Rijmen [9, 19]. Actualmente, el cifrado simétrico continúa siendo un componente esencial en protocolos de comunicación y sistemas de seguridad de la información.

2.1.1. Cifrado César

El cifrado de César es un método histórico que se le atribuye al emperador romano Julio César [21], en el cual las letras del mensaje (x) son cifradas con la clave (k) que indica la cantidad de desplazamientos que se deben realizar desde la letra origen a la letra destino dentro del alfabeto elegido, donde m indica la cantidad de símbolos del alfabeto. Haciendo uso de la aritmética modular y de los elementos definidos anteriormente, se pueden expresar las funciones de cifrado y descifrado, respectivamente, como:

$$y_i = \text{enc}(x_i, k) = x_i + k \quad \text{mód } m, \quad (2.1)$$

$$x_i = \text{des}(y_i, k) = y_i - k \quad \text{mód } m. \quad (2.2)$$

Este tipo de cifrado en un contexto donde la mayoría de las personas no sabían leer ni escribir, suponía una dificultad extra al momento de interpretar un mensaje. Ahora bien, desde el punto de vista del Criptoanálisis, la debilidad del mismo radica en que ante un análisis de las frecuencias de ocurrencias de letras en el mensaje y , teniendo en cuenta el idioma en el que está expresado el mensaje en claro x , podrían llegarse a detectar el algoritmo y la clave utilizados para cifrar el mensaje. Siendo ésta la principal desventaja de este tipo de cifrado, ya que solo resignifica, es decir, sustituye el alfabeto con el que se expresa el mensaje cifrado pero mantiene todas las demás propiedades estructurales del mensaje en claro.

2.1.2. Cifrado de flujo

El cifrado de flujo [31], por su parte, cifra cada *bit* o *byte* individualmente mediante la operación XOR con un flujo de clave $s(k)$ generado pseudoaleatoriamente a partir de la clave k , como se observa en la Figura 2.3. Es ideal para aplicaciones en tiempo real, como transmisión continua, debido a su eficiencia y bajo coste computacional. Ejemplos de este tipo de cifrado son *Rivest Cipher 4* (RC4) [12], Salsa20 [4] y ChaCha20 [5].

El cifrado de flujo consiste principalmente en hacer uso de la operación suma exclusiva XOR, notada por $\oplus$, para producir la salida encriptada, donde el i -ésimo *bit* x_i del mensaje x , es operado con el i -ésimo *bit* del flujo de clave, denotado por s_i para $i = 1, 2, \dots, n$, donde n es el largo del mensaje x . Al utilizar la suma exclusiva se pueden dar las cuatro posibilidades que se muestran en la Tabla 2.1. Esta operación tiene características deseables para el proceso de cifrado ya que permite obtener un cambio o no del *bit* del mensaje en función del estado del *bit* del flujo de clave, con una posibilidad de invertirlo o no del 50 %.

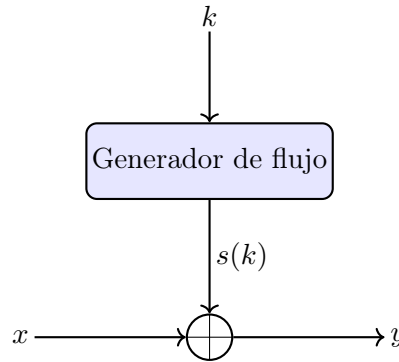


Figura 2.3: Esquema básico del cifrado de flujo.

Tabla 2.1: Tabla de verdad de la operación XOR.

x_i	s_i	$x_i \oplus s_i$
0	0	0
0	1	1
1	0	1
1	1	0

Cuando el *bit* de mensaje es 1, la salida dependerá del *bit* del flujo de clave. Esto es, si s_i es 1 la salida será un 0, lo que quiere decir que se invierte el mensaje. Mientras que si el *bit* del flujo de clave s_i es 0, la salida se mantiene en 1 y no se produce cambio. De manera análoga se opera cuando el *bit* del mensaje es 0.

Todo este proceso aplicado a un flujo de *bits* queda descrito, en forma general, por las siguientes ecuaciones de cifrado y descifrado, respectivamente:

$$y_i = \text{enc}(x_i, k) = x_i \oplus s_i, \quad (2.3)$$

$$x_i = \text{des}(y_i, k) = y_i \oplus s_i, \quad (2.4)$$

siendo k la clave a partir de la cual se genera el flujo de clave s .

Si bien este método de cifrado es sencillo, para la implementación del mismo es necesario que se produzca un flujo de clave del mismo tamaño que el mensaje que se desea cifrar. Es por esto que existen diferentes formas con las cuales se pueden generar números aleatorios para ser utilizados como s_i y cumplir algunas restricciones tales como ser del mismo largo que el mensaje y que la clave sea utilizada una única vez son las condiciones necesarias para no comprometer la seguridad del cifrado. Por lo tanto, la fortaleza en la seguridad en la mayoría de los cifrados de flujo reside en su uso único y la aleatoriedad del flujo de la clave, lo que implica que la forma en que se generan estos flujos es de suma importancia.

En Criptografía, los generadores de verdaderos números aleatorios (*True Random Number Generator*, TRNG) se utilizan para la generación de llaves de sesión que serán distribuidas entre el Emisor y el Receptor. Estos generadores aleatorios son desarrollados para producir una sucesión de ceros y unos totalmente aleatoria, a partir de algún proceso físico intrínsecamente aleatorio, por ejemplo, el lanzamiento de una moneda, el lanzamiento de un dado, el ruido de un semiconductor o el decaimiento radioactivo de un elemento.

A pesar de que la generación de números aleatorios representa el contexto ideal, en la práctica es imposible de aplicar. Es por ello que se trabaja con generadores de números pseudoaleatorios (*Pseudo Random Number Generator*, PRNG), los cuales producen secuencias que son computadas a partir de un valor semilla inicial, denotado por s_0 , de manera recursiva; es decir,

$$s_{i+1} = g(s_i), \quad (2.5)$$

para $i = 0, 1, 2, \dots$ y donde g es una función. Un ejemplo de PRNG es el generador lineal congruencial definido como

$$s_{i+1} = as_i + b \pmod{m}, \quad (2.6)$$

para $i = 0, 1, 2, \dots$, donde a , b y m son enteros positivos, pudiendo ser $b = 0$. Nótese que los PRNG no generan números aleatorios en el sentido estricto de la palabra ya que la secuencia de números puede ser recalculada, siendo de esta manera completamente determinísticos. Así, un ejemplo del *American National Standards Institute* (ANSI) está definido de la siguiente manera:

$$\begin{cases} s_0 &= 12345, \\ s_{i+1} &= 1103515245s_i + 12345 \pmod{2^{31}}, \end{cases} \quad (2.7)$$

para $i = 0, 1, 2, \dots$.

Un requerimiento típico de estos generadores es que cuenten con buenas propiedades estadísticas, o sea que se “aproximen” a una secuencia de números aleatorios. Existen muchas situaciones tales como en las simulaciones donde se necesitan datos aleatorios para pruebas. Es por ello que estas funciones suelen incluirse en las bibliotecas estándar de diferentes lenguajes.

Los generadores de números pseudoaleatorios criptográficamente seguros (*Criptografic Secure Pseudo Random Number Generator*, CSPRNG) son un tipo de PRNG a los cuales se les impone la propiedad adicional de que sean impredecibles. En la práctica, esto quiere decir que a partir de una salida del flujo de clave de n bits $s_i, s_{i+1}, \dots, s_{i+n-1}$, resulta computacionalmente inviable calcular la continuidad de la secuencias de bits $s_{i+n}, s_{i+n+1}, \dots$. En otras palabras, dados n bits consecutivos del flujo de clave, no existe un algoritmo que en tiempo polinomial pueda predecir el siguiente bit con más del 50 % de posibilidades de éxito.

Así es entonces que la combinación de un cifrado de flujo más simple y la generación criptográficamente segura de un flujo de clave que solo será utilizada una única vez para cifrar mensajes, lleva a la concepción de la “libreta de uso único” (*One Time Pad*, OTP) como un cifrado incondicionalmente seguro (técnicamente seguro) por lo que no se puede romper aun con recursos informáticos infinitos. OTP es un cifrado de flujo donde s_i proviene de un TRNG y cada s_i es utilizado solo una vez. Su desventaja reside en que el largo de la clave de cifrado debe ser del mismo largo que el mensaje a enviar y que, como su nombre lo indica, la clave debe ser utilizada una única vez.

2.1.3. Cifrado por bloques

El enfoque de cifrado por bloques constituye una de las formas más empleadas de cifrado simétrico y se caracteriza por operar sobre unidades de datos de tamaño fijo, generalmente expresadas en n múltiplos de bytes. Cada bloque es procesado en su totalidad mediante una clave criptográfica k , aplicando rondas de operaciones matemáticas estructuradas, como sustitución y permutación, que garantizan la transformación del texto en claro (x) en texto cifrado (y), como se describe en la Figura 2.4.

Cuando la información a proteger excede la longitud de un bloque, resulta necesario recurrir a diferentes **modos de operación**, tales como el libro de códigos electrónico (*Electronic Code Book*, ECB) y el cifrado de bloques encadenados (*Cipher Block Chaining*, CBC), los cuales permiten encadenar el procesamiento de bloques y mejorar la seguridad global del esquema. Entre los algoritmos más representativos de esta categoría se destacan AES, DES y Blowfish [24], ampliamente utilizados en diversos contextos de seguridad de la información.

Con el propósito de facilitar el análisis entre las dos familias principales de algoritmos simétricos, la Tabla 2.2 presenta una comparativa entre cifrados por bloques y cifrados de flujo. La comparación contempla aspectos fundamentales como la unidad de procesamiento o el impacto de los errores, así como las ventajas y limitaciones prácticas de cada enfoque. Esta

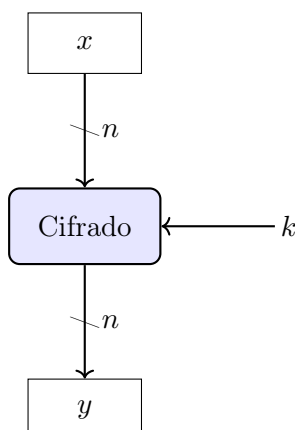


Figura 2.4: Esquema básico del cifrado por bloques.

visión sistemática permite apreciar con claridad los escenarios en los que cada tipo de cifrado resulta más adecuado, contribuyendo a una selección fundamentada de algoritmos en función de los requerimientos de seguridad, eficiencia y entorno de aplicación.

Tabla 2.2: Diferencias entre cifrado por bloques y cifrado de flujo.

Aspecto	Cifrado por bloques	Cifrado de flujo
Modo de procesamiento	Usa bloques de longitud fija (64 o 128 <i>bits</i>).	Usa <i>bits</i> o <i>bytes</i> individuales.
Velocidad / complejidad	Requiere más procesamiento. / Son algoritmos complejos.	Es más eficiente. / Tiene baja complejidad.
Principios criptográficos	Aplican difusión y confusión.	Aplican principalmente confusión.
Propagación de errores.	Un error puede afectar a todo el bloque.	Los errores suelen ser localizados.
Casos de uso típicos	Datos en reposo (por ejemplo, archivos o discos).	Transmisión en tiempo real (por ejemplo, <i>streaming</i> o comunicaciones).
Seguridad con reutilización de clave.	Más resistente a reutilización múltiple de clave.	Vulnerable si se reutiliza la misma clave.

2.2. Ataques criptográficos

El estudio de los ataques criptográficos constituye un eje central en la evaluación y diseño de algoritmos de seguridad. Estos ataques se definen como el conjunto de técnicas empleadas por un oponente para intentar vulnerar un sistema criptográfico, con el objetivo de develar información confidencial o comprometer la clave secreta. En términos generales, la solidez de

un esquema criptográfico no se mide únicamente por la complejidad de su algoritmo, sino también por su capacidad de resistir ataques prácticos y teóricos [27].

Claude Shannon, en sus trabajos fundacionales, sentó las bases para la caracterización de la seguridad al introducir los conceptos de **confusión** y **difusión**, pilares de la resistencia frente a ataques de criptoanálisis [27]. En la actualidad, los avances computacionales y el desarrollo de nuevas técnicas de análisis han dado lugar a una amplia variedad de ataques, que se clasifican según la información disponible para el adversario o la debilidad del sistema que explotan [3, 21].

Criptoanálisis clásico

El enfoque clásico busca recuperar el texto en claro o la clave secreta a partir del texto cifrado. Algunos autores distinguen entre ataques de fuerza bruta, que tratan al algoritmo como una **caja negra**, y ataques analíticos, que explotan la estructura interna del cifrado [21].

Ataques de fuerza bruta: El ataque más elemental consiste en probar todas las claves posibles hasta encontrar la correcta. Si bien su viabilidad depende del tamaño del espacio de claves, el crecimiento en la capacidad de cómputo lo hace un riesgo latente para algoritmos con claves reducidas, como ocurrió con DES de 56 *bits*. Por este motivo, estándares modernos como AES emplean longitudes de clave mucho mayores (128, 192 o 256 *bits*), consideradas seguras, hoy en día, frente a este tipo de ataques [19].

Ataques con diferentes niveles de información: La efectividad del Criptoanálisis también depende de la información disponible para el atacante, pudiendo ser:

- **Ataque de solo texto cifrado:** el adversario únicamente accede al texto cifrado y busca deducir el mensaje en claro o la clave. Un ejemplo clásico es el análisis de frecuencias aplicado al cifrado César.
- **Ataque de texto en claro conocido:** el atacante dispone de pares de texto en claro y cifrado. Esta situación permitió el desarrollo de ataques diferenciales contra DES [9].
- **Ataque de texto en claro elegido:** el adversario puede elegir textos en claro y obtener sus cifrados, lo que facilita identificar patrones en la transformación criptográfica.
- **Ataque de texto cifrado elegido:** el atacante puede introducir textos cifrados y obtener su descifrado, situación que históricamente debilitó protocolos de comunicación mal diseñados.

Ataques de implementación

Más allá del análisis matemático del algoritmo, existen ataques que explotan fugas de información en la ejecución del sistema. Entre ellos se encuentran el análisis del consumo de

energía (*power analysis*), las emisiones electromagnéticas y los tiempos de ejecución (*timing attacks*). Estos ataques de canal lateral son especialmente relevantes en dispositivos embebidos, como tarjetas inteligentes o sistemas de bajo consumo, donde el entorno físico facilita la extracción de datos sensibles [21, 3].

Ataques de ingeniería social

Los sistemas criptográficos no solo se ven amenazados por técnicas matemáticas o de implementación. El factor humano constituye a menudo el eslabón más débil. Los ataques de ingeniería social incluyen prácticas como el engaño, el soborno o la coacción para obtener claves secretas. Ejemplos incluyen llamadas falsas en las que un atacante se hace pasar por personal de soporte para obtener contraseñas, o la denominada *rubber-hose cryptanalysis*, en la cual se obliga a un usuario a revelar su clave bajo amenaza [21].

Otros ataques prácticos

En un escenario realista, también deben considerarse ataques que explotan vulnerabilidades de *software* o del sistema operativo, como desbordamientos de memoria o *malware* diseñado para extraer claves de los procesos en ejecución. Estos ataques ilustran que la seguridad de un sistema no depende únicamente de la fortaleza del algoritmo criptográfico, sino de todo su ecosistema de implementación y operación.

El análisis de los ataques criptográficos resulta esencial para establecer criterios objetivos de seguridad en los algoritmos de cifrado. Comprender estas categorías no solo permite clasificar la resistencia de un esquema, sino también diseñar métricas de evaluación que contemplen escenarios realistas frente a adversarios con diferentes niveles de acceso y recursos computacionales. En este sentido, la seguridad práctica no se limita a la fortaleza matemática del algoritmo, sino también a su implementación, su resistencia a ataques de canal lateral y la robustez frente a la explotación del factor humano. Así, la consideración integral de estos ataques constituye una herramienta indispensable en la evaluación comparativa y en la selección de algoritmos apropiados para aplicaciones modernas de protección de datos [3, 21].

2.3. Modos de operación

Una vez constituido el bloque de cifrado, se debe determinar cómo actuar frente a los datos a cifrar cuyo tamaño excede el tamaño del bloque admitido. Cada una de las estrategias o modos que se analizarán tienen sus ventajas y desventajas en función de su caso de uso.

Previamente a la utilización de algún modo de operación, se debe definir un criterio para que la información a cifrar pueda ser divisible por el tamaño de bloque aceptado por el algoritmo de cifrado seleccionado. Para esto se realiza un relleno o *padding* que puede consistir en distintas estrategias. Por ejemplo, a nivel de *bits* se puede agregar un 1 seguido de la

cantidad necesaria de ceros hasta completar el bloque, estrategia conocida como *bit padding* o *ISO/IEC 9797-1 padding method 2*. Mientras que a nivel de *bytes* una opción común es calcular la cantidad de *bytes* necesarios para completar el último bloque y repetir ese valor tantas veces como sea necesario, técnica conocida como *Public Key Cryptography Standards 7 (PKCS#7) padding*.

En la literatura relacionada, es común encontrar diferentes notaciones para denotar la función de cifrado y descifrado bajo una clave k . La expresión $\text{enc}(x, k)$ representa la aplicación del algoritmo de cifrado a un bloque de datos x utilizando la clave k . De forma equivalente, se utiliza la notación $E_k(P)$, donde E indica la función de cifrado, el subíndice k explicita la dependencia de la clave y P corresponde al bloque de texto plano de entrada. Por lo tanto, $\text{enc}(P, k) \equiv E_k(P)$. Análogamente, la operación de descifrado puede expresarse como $\text{dec}(x, k)$, o bien mediante la notación $D_k(C)$, donde D representa la función de descifrado bajo la clave k y C es el bloque de texto cifrado. En consecuencia, $\text{dec}(C, k) \equiv D_k(C)$.

Modo de libro de códigos electrónico (*Electronic Code Book mode, ECB*)

El ECB es el modo de operación más básico ya que solo se dividen los datos en claro en bloques de tamaño adecuado para el cifrado y cada bloque es procesado con la misma clave k . Como ventaja de este modo se encuentra la posibilidad de paralelizar el proceso de cifrado y descifrado, por lo que su rendimiento es superior al de otros modos. Sin embargo, su principal desventaja es que es altamente determinístico: bloques idénticos de texto plano (P) producen bloques idénticos de texto cifrado (C). Esto expone patrones y hace posible un ataque por sustitución. Un ejemplo clásico sería el reemplazo de un bloque en una transferencia bancaria, alterando así información crítica como el número de cuenta destino.

En la Figura 2.5 se observa cómo cada bloque P_i se procesa de forma independiente mediante el cifrado E_k , y en el descifrado el proceso inverso D_k se aplica, de manera análoga a la encriptación, sobre cada bloque C_i para recuperar P_i , para $i = 1, 2, 3, \dots, n$, donde n representa en número total de bloques a cifrar. Lo que también se expresa de la siguiente forma:

$$C_i = E_k(P_i), \quad (2.8)$$

para $i = 1, 2, \dots, n$.

Modo de cifrado de bloques encadenados (*Cipher Block Chaining mode, CBC*)

En este modo de operación cada bloque cifrado C_i depende del bloque anterior C_{i-1} , lo cual introduce un efecto de encadenamiento. Como se visualiza en la Figura 2.6 y se expresa en la siguiente ecuación,

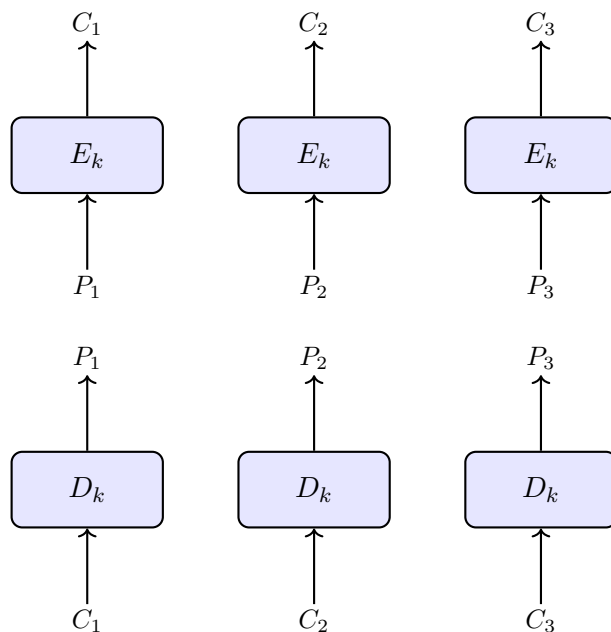


Figura 2.5: Esquema de cifrado (arriba) y descifrado (abajo) en modo ECB.

$$\begin{cases} C_0 = IV, \\ C_i = E_k(P_i \oplus C_{i-1}), \end{cases} \quad (2.9)$$

para $i = 1, 2, \dots, n$, donde IV es un vector de inicialización que asegura aleatoriedad en el primer bloque. La ventaja de CBC es que oculta patrones presentes en el texto plano, pero a costa de perder paralelización en el proceso de cifrado. Además, si se utiliza un nuevo IV por cada mensaje, el esquema se comporta como un cifrado probabilístico.

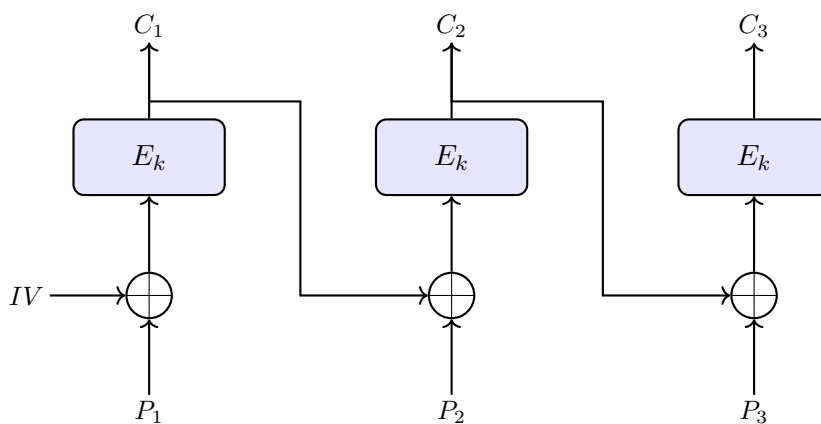


Figura 2.6: Esquema de cifrado en modo CBC.

Para recuperar el texto plano se aplican dos operaciones: primero, el descifrado del bloque cifrado con la clave $D_k(C_i)$; luego, la combinación mediante XOR con el bloque cifrado anterior C_{i-1} . En el caso del primer bloque, se utiliza el IV :

$$\begin{cases} P_1 &= IV \oplus D_k(C_1), \\ P_i &= C_{i-1} \oplus D_k(C_i), \end{cases} \quad (2.10)$$

para $i = 2, 3, \dots, n$.

Modo de cifrado retroalimentado (*Cipher Feedback mode, CFB*)

En el modo de operación de CFB, el cifrado de bloques se utiliza como generador de clave de flujo. Partiendo de un vector de inicialización IV , se aplica el cifrado E_k y el resultado se combina mediante XOR con el texto en claro P_i para producir el bloque cifrado C_i . Este último se retroalimenta como entrada al siguiente paso, es decir:

$$\begin{cases} C_0 &= IV, \\ C_i &= P_i \oplus E_k(C_{i-1}), \end{cases} \quad (2.11)$$

para $i = 1, 2, \dots, n$.

La principal ventaja de CFB es que permite cifrar datos de tamaño inferior al del bloque, como se observa en la Figura 2.7, adaptándose a transmisiones en tiempo real. Como desventaja, un error en un bloque afecta a varios posteriores.

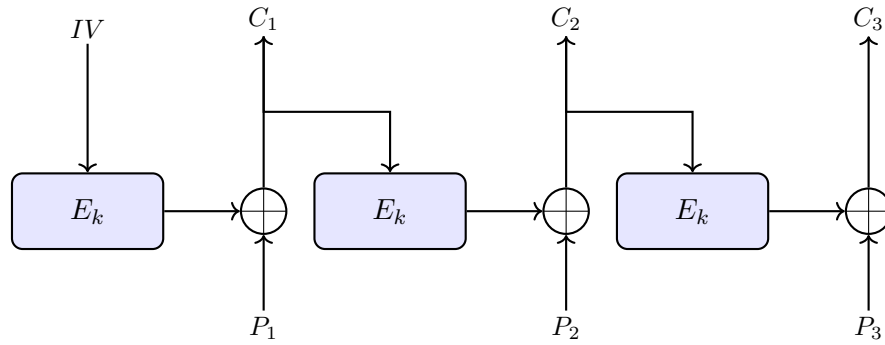


Figura 2.7: Esquema de cifrado en modo CFB.

El descifrado se realiza aplicando nuevamente E_k sobre el bloque anterior cifrado y combinando mediante XOR con el bloque cifrado recibido:

$$P_i = C_i \oplus E_k(C_{i-1}), \quad (2.12)$$

para $i = 1, 2, \dots, n$.

Modo de salida retroalimentada (*Output Feedback mode, OFB*)

En el modo OFB se genera una secuencia de claves de flujo independiente del mensaje, lo cual permite preprocesar la clave y separar el cifrado del procesamiento del mensaje. Partiendo de IV , se aplica repetidamente E_k para obtener s_i , que se combina con P_i mediante XOR para producir C_i , como ve observa en la Figura 2.8, y se calcula mediante

$$\begin{cases} s_0 = IV, \\ s_i = E_k(s_{i-1}), \\ C_i = P_i \oplus s_i, \end{cases} \quad (2.13)$$

para $i = 1, 2, \dots, n$. La ventaja es que los errores de transmisión no se propagan entre bloques, pero la sincronización entre Emisor y Receptor debe ser estricta. Dado que el flujo de clave s_i es el mismo tanto en cifrado como en descifrado, el receptor únicamente debe aplicar:

$$P_i = C_i \oplus s_i, \quad (2.14)$$

para $i = 1, 2, \dots, n$.

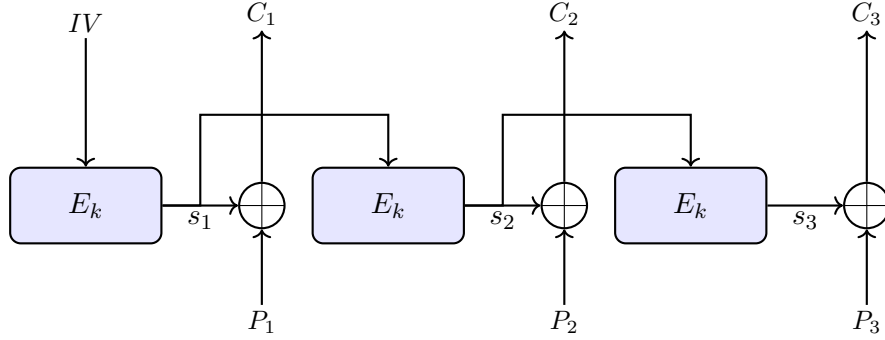


Figura 2.8: Esquema de cifrado en modo OFB.

Modo contador (*Counter mode, CTR*)

El modo contador transforma el cifrado por bloques en un cifrado de flujo utilizando un contador CTR o *nonce*¹. Para cada bloque, el valor del contador incrementado en uno se cifra con la clave k y se combina con el texto plano:

$$C_i = P_i \oplus E_k(CTR + i), \quad (2.15)$$

para $i = 1, 2, \dots, n$.

¹*Nonce (number used once)*: es un valor único que se utiliza una sola vez por clave para evitar repeticiones en el cifrado. No es un secreto, pero debe ser único y, según el algoritmo, también aleatorio.

La ventaja principal de modo contador es su alta eficiencia, ya que permite la paralelización total de operaciones, tanto en cifrado como en descifrado. Su principal requisito es que los valores de contador no se repitan nunca con la misma clave.

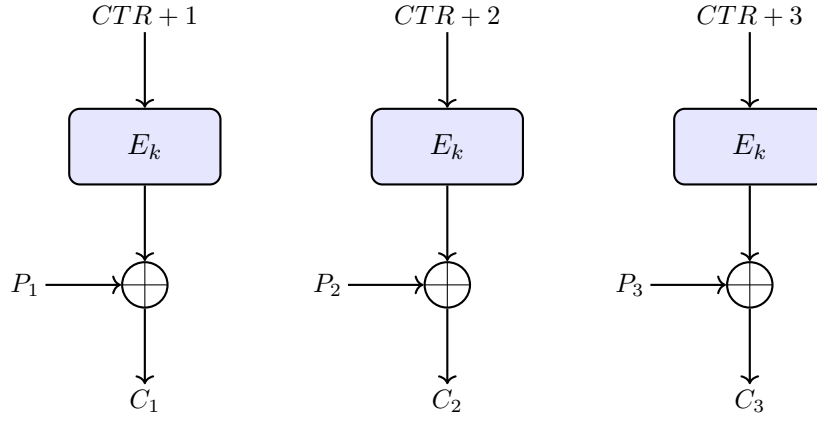


Figura 2.9: Esquema de cifrado en modo CTR.

El proceso es simétrico, dado que la secuencia $E_k(CTR + i)$ se genera de la misma manera en Emisor y Receptor:

$$P_i = C_i \oplus E_k(CTR + i), \quad (2.16)$$

para $i = 1, 2, \dots, n$.

Es importante remarcar que los modos de operación para los cifrados por bloques constituyen un elemento esencial en la seguridad práctica de los sistemas criptográficos. Cada modo responde a un conjunto particular de necesidades: algunos priorizan la simplicidad y velocidad, como ECB, mientras que otros añaden aleatoriedad y encadenamiento para garantizar confidencialidad frente a patrones repetidos, como CBC o CTR. La elección del modo de operación, por lo tanto, no es trivial y debe estar alineada con los requisitos de seguridad y eficiencia de la aplicación en cuestión.

A fin de sintetizar las características principales de cada modo de operación, se presenta la Tabla 2.3 como una comparativa, destacando aspectos clave como la forma de encadenamiento, la necesidad de vectores de inicialización, la resistencia frente a la revelación de patrones, etc. Esta visión conjunta permite apreciar de manera clara las ventajas y limitaciones relativas de cada enfoque, sirviendo como base para la selección informada de la técnica más adecuada en entornos prácticos.

Además, resulta fundamental destacar que la fortaleza de un algoritmo de cifrado no depende únicamente de la calidad matemática de su diseño, sino también del modo en que se aplica en escenarios reales. Modos inadecuados o implementaciones inseguras pueden introducir vulnerabilidades críticas que un oponente puede explotar, incluso si el cifrado subyacente es considerado robusto. De este modo, la evaluación comparativa de los diferentes

Tabla 2.3: Comparación de modos de operación.

Modo	Características	Ventajas / Desventajas
ECB	Cifra cada bloque de forma independiente.	Sencillo y rápido, pero revela patrones en los datos, inseguro para imágenes o datos estructurados.
CBC	Cada bloque de texto en claro se combina con el bloque cifrado anterior mediante XOR. Requiere un <i>IV</i> .	Más seguro que ECB, oculta patrones. Sin embargo, no permite paralelismo en cifrado y los errores se propagan.
CFB	Convierte el cifrado por bloques en un cifrado en flujo. Se usa la salida cifrada previa para realimentar.	Permite cifrar datos de longitud variable y transmisión en tiempo real.
OFB	Similar a CFB, pero realimenta la salida del cifrador en lugar del cifrado.	No propaga errores, seguro para transmisión. Sin embargo, requiere sincronización estricta.
CTR	Usa un contador cifrado para generar un flujo de claves, luego se aplica XOR con el mensaje.	Alto paralelismo, seguro y flexible. Vulnerable si se reutiliza el mismo <i>nonce</i> o contador.

modos permite establecer criterios técnicos sólidos para seleccionar la opción más apropiada en cada contexto de uso.

Capítulo 3

Materiales y Métodos

En este capítulo se presentan los algoritmos de cifrado que se van a estudiar, las imágenes sobre las cuales se van a aplicar estos algoritmos y las métricas que se emplean para evaluar la calidad de los resultados obtenidos.

3.1. Algoritmos

Los algoritmos a estudiar y su implementación se encuentran disponibles en el módulo `pycryptodome` de Python, el cual proporciona primitivas criptográficas de bajo nivel y se encuentra disponible en <https://pypi.org/project/pycryptodome/>. En esta sección, se incluye una descripción detallada de las características principales de cada uno de ellos, incluyendo ventajas y desventajas. Muchos de los algoritmos de cifrado por bloques más conocidos se construyen utilizando la estructura de red de Feistel [11]. Esta estructura matemática, introducida por Horst Feistel en la década de 1970, permite diseñar algoritmos invertibles (descifrables) a partir de funciones que no necesariamente son invertibles, facilitando la implementación eficiente y la obtención de propiedades criptográficas deseables. El cifrado DES popularizó su uso y se convirtió en el estándar más utilizado durante varias décadas.

El funcionamiento de la red de Feistel puede describirse en varias etapas claramente definidas a partir de un bloque de datos de entrada M de tamaño fijo de $2n$ bits. En primer lugar, se realiza la división del bloque inicial M en dos mitades de longitud n bits cada una: la parte izquierda L_0 y la parte derecha R_0 . Esta separación constituye el punto de partida del proceso iterativo de cifrado. Posteriormente, se llevan a cabo las rondas de transformación. En cada ronda $i = 1, 2, \dots, r$, se aplican las siguientes operaciones:

$$L_i = R_{i-1}, \tag{3.1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, s_i), \tag{3.2}$$

donde f corresponde a la función de ronda, que no necesariamente debe ser biyectiva, y s_i representa la subclave derivada de la clave principal k . Estas transformaciones se repiten de forma sucesiva a lo largo de r rondas, logrando una difusión y confusión crecientes en el mensaje.

Finalmente, en la etapa de salida, una vez completadas las r rondas, el texto cifrado se obtiene concatenando las dos mitades transformadas en el orden (R_r, L_r) . Esta estructura asegura que el proceso global de la red de Feistel sea reversible, permitiendo la correcta recuperación del texto original durante la fase de descifrado. Esto permitió el diseño de algoritmos robustos sin la limitación de requerir transformaciones elementales biyectivas. En la Figura 3.1 se muestra el esquema de una ronda de Feistel.

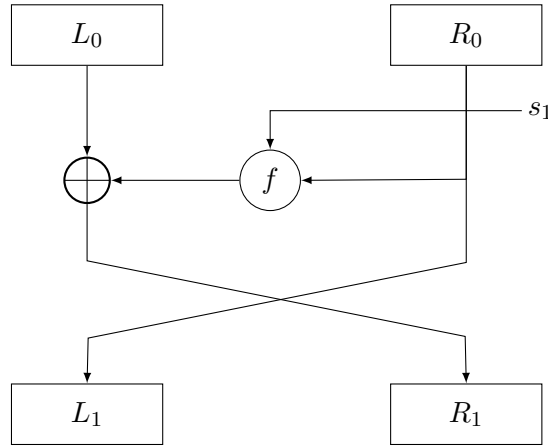


Figura 3.1: Esquema de una ronda de Feistel.

La invertibilidad de la red se realiza como sigue. Dado un par (L_i, R_i) y la subclave s_i , es posible recuperar (L_{i-1}, R_{i-1}) de la siguiente manera, a partir de (3.1):

$$L_i = R_{i-1} \implies R_{i-1} = L_i, \quad (3.3)$$

y sustituyendo en (3.2),

$$R_i = L_{i-1} \oplus f(R_{i-1}, s_i) = L_{i-1} \oplus f(L_i, s_i) \implies L_{i-1} = R_i \oplus f(L_i, s_i). \quad (3.4)$$

De esta manera, se recupera el par (L_{i-1}, R_{i-1}) de forma determinista. El hecho de que f no sea invertible no afecta el proceso, ya que la estructura matemática de la red de Feistel garantiza la reversibilidad de cada ronda y, en consecuencia, de todo el cifrado.

De manera análoga las redes de sustitución-permutación, SPN por su equivalente en inglés *substitution-permutation network*, constituyen otro de los modelos fundamentales en el diseño de cifrados por bloques modernos. Entre los algoritmos más relevantes basados en esta estructura se encuentra el AES. Una SPN se construye a partir de la repetición de varias rondas, en las cuales se alternan transformaciones bien definidas, es decir, sustituciones y

permutaciones determinadas. El objetivo de esta estructura es garantizar tanto la confusión como la difusión, propiedades fundamentales para la seguridad de un cifrado por bloques.

En cada ronda, como se observa en la Figura 3.2, la primera operación consiste en la aplicación de cajas de sustitución o *S-boxes*. Estas funciones no lineales introducen complejidad en el sistema y dificultan la predicción de la salida a partir de la entrada, lo que constituye un elemento esencial para contrarrestar ataques basados en aproximaciones lineales o diferenciales. A diferencia de transformaciones lineales, que pueden ser descritas mediante combinaciones algebraicas simples, la no linealidad de las *S-boxes* impide que un adversario modele el comportamiento del cifrado mediante relaciones matemáticas de baja complejidad. Además, un buen diseño de *S-box* asegura propiedades como una alta no linealidad, una distribución uniforme de las salidas y un bajo grado de correlación entre *bits* de entrada y salida, características que incrementan significativamente la resistencia frente a técnicas avanzadas de criptanálisis.

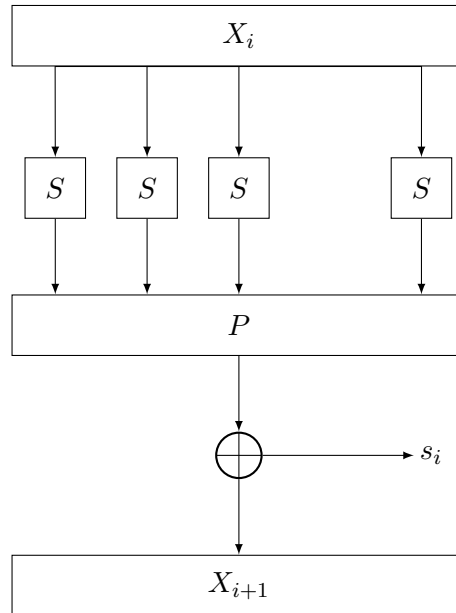


Figura 3.2: Esquema de una ronda de SPN.

Posteriormente, se lleva a cabo una etapa de permutación en P , en la que los *bits* o palabras del estado intermedio son reordenados. Esta difusión asegura que las modificaciones locales en la entrada se propaguen ampliamente a lo largo del texto cifrado después de varias rondas. Finalmente, se realiza la adición de la subclave s_i , que consiste en la combinación del estado con una porción de la clave maestra previamente expandida. Este paso vincula de manera directa la seguridad del esquema con la clave secreta y refuerza la dependencia del texto cifrado respecto de ella.

En el caso del algoritmo AES, se aplica una estructura de SPN con las siguientes operaciones principales en cada ronda:

- *SubBytes*: sustitución no lineal *byte* a *byte*.
- *ShiftRows*: desplazamiento cíclico de filas.
- *MixColumns*: mezcla lineal de columnas.
- *AddRoundKey*: combinación XOR con la subclave de ronda.

Las SPN presentan diversas ventajas que explican su amplia adopción en el diseño de cifrados modernos. Su estructura es conceptualmente clara y relativamente sencilla de implementar, lo que facilita tanto el análisis teórico como la optimización práctica. Además, proporcionan una resistencia adecuada frente a ataques clásicos, en particular aquellos basados en análisis estadístico. Otro aspecto relevante es su eficiencia, ya que pueden implementarse de manera eficaz tanto en *software* como en *hardware*. Ofrecen flexibilidad para adaptarse a diferentes longitudes de clave, lo que permite equilibrar entre seguridad y rendimiento según las necesidades del sistema.

No obstante, las SPN también presentan ciertas limitaciones que deben considerarse en su diseño e implementación. En primer lugar, si las cajas de sustitución no cumplen criterios rigurosos de construcción, el esquema puede ser vulnerable a ataques diferenciales y lineales, dos de las técnicas más efectivas en Criptoanálisis. A su vez, como ocurre con la mayoría de los cifrados por bloques, las implementaciones prácticas de SPN están expuestas a ataques por canales laterales, que explotan información filtrada a través del tiempo de ejecución, el consumo de energía o incluso fallos inducidos en el dispositivo. Estas debilidades evidencian la necesidad de acompañar el diseño algorítmico con contramedidas robustas en el plano de la implementación.

La implementación de algoritmos basados en SPN, como AES, exige la consideración de diversos aspectos técnicos que influyen directamente en su seguridad y eficiencia como, por ejemplo, la expansión de clave, que debe realizarse de manera segura para garantizar que cada ronda utilice subclaves adecuadamente derivadas de la clave maestra. A este proceso se suman distintas optimizaciones prácticas, entre las cuales destacan el uso de tablas precomputadas y las implementaciones en forma de *bitslicing*, ambas orientadas a reducir el tiempo de ejecución y aprovechar mejor los recursos disponibles.

Por lo tanto, cualquier implementación debe someterse a métodos rigurosos de prueba estadística y validación de seguridad. Estas evaluaciones no solo verifican la corrección funcional del algoritmo, sino que también aseguran que la salida cifrada presente propiedades aleatorias suficientes para resistir intentos de criptanálisis.

A continuación, se describen los algoritmos de encriptación utilizados en el presente trabajo.

Advanced Encryption Standard, AES: Algoritmo de cifrado por bloques basado en redes de sustitución-permutación con bloques de 128 *bits* y claves de 128, 192 o 256 *bits* [19], que

derivan en una implementación con 10, 12 y 14 rondas, respectivamente. AES es ampliamente adoptado debido a su seguridad comprobada contra ataques conocidos como diferencial y lineal [9], así como su alta eficiencia en *software* y *hardware*. Entre sus ventajas se destacan la resistencia a ataques criptoanalíticos, la flexibilidad en el tamaño de clave y la disponibilidad de soporte *hardware*. La principal desventaja es el alto consumo computacional en entornos de *hardware* muy limitado o sistemas embebidos antiguos.

Alleged Rivest Cipher 2, ARC2: Cifrado por bloques de 64 *bits* con clave variable de hasta 128 *bits* [15]. Fue diseñado como reemplazo de DES y se caracteriza por ser eficiente en *software*. ARC2 permite variaciones en el número de rondas, lo que ofrece cierta flexibilidad en seguridad y rendimiento. Hace uso de redes de Feistel desequilibradas que pueden llegar a ser de 18 rondas combinadas de diferentes formas de permutación. Sin embargo, es vulnerable a Criptoanálisis moderno [25] y no se recomienda para nuevas implementaciones, siendo su principal limitación la seguridad insuficiente frente a algoritmos, como por ejemplo el AES.

Alleged Rivest Cipher 4, ARC4: Cifrado de flujo con clave variable [12], conocido por su velocidad y simplicidad. Fue ampliamente usado en SSL (*Secure Sockets Layer*), TLS (*Transport Layer Security*) y WEP (*Wired Equivalent Privacy*). Entre sus ventajas se encuentra la rapidez en *software* y la implementación sencilla. Sin embargo, presenta debilidades críticas, como sesgos en la secuencia de salida y vulnerabilidad a ataques estadísticos [22], por lo que actualmente está desaconsejado.

Blowfish: Cifrado por bloques de 64 *bits* con clave de hasta 448 *bits* [25]. Blowfish es rápido en *software* y ofrece buena resistencia frente a ataques conocidos. Su diseño incluye una red Feistel con sustituciones y permutaciones que proporcionan difusión eficiente en sus 16 rondas. Entre sus ventajas están la flexibilidad de longitud de clave y la ausencia de patentes. Su principal desventaja es el tamaño de bloque limitado, que puede ser inseguro en aplicaciones modernas con grandes volúmenes de datos [24].

Carlisle Adams and Stafford Tavares, CAST: Cifrado por bloques basado en la red de Feistel, con variantes CAST-128 (bloques de 64 *bits* con 12 o 16 rondas, claves de 40–128 *bits*) y CAST-256 (bloques de 128 *bits* con 48 rondas, claves de hasta 256 *bits*) [17, 1]. CAST es eficiente, seguro y libre de regalías, con resistencia a ataques diferenciales y lineales. Sus ventajas incluyen flexibilidad en tamaño de clave y diseño robusto. Su desventaja principal es que su seguridad es menor, comparada con AES, y su adopción global es limitada.

Chacha20: Cifrado de flujo moderno, derivado de Salsa20, que utiliza clave de 256 *bits* y *nonce* de 96 *bits* [5]. Chacha20 es rápido y seguro, con alta resistencia a ataques y diseñado para dispositivos móviles. Entre sus ventajas, se destacan la rapidez en *software*, la resistencia

a ataques criptoanalíticos y la simplicidad en el diseño. Las desventajas incluyen menor soporte en *hardware* comparado con AES y menor difusión en sistemas antiguos (*legacy*).

Data Encryption Standard, DES: Cifrado por bloques de 64 *bits* con clave de 56 *bits* [18], basado en redes de Feistel de 16 rondas. Fue el estándar durante varias décadas y es rápido en *hardware*. Sus ventajas son la simplicidad del diseño y la compatibilidad con sistemas antiguos. Sus desventajas son la vulnerabilidad a ataques de fuerza bruta y ataques diferenciales [6], lo que lo hace obsoleto frente a AES.

Salsa20: Cifrado de flujo rápido y seguro diseñado por Daniel J. Bernstein, con clave de 128 o 256 *bits* y *nonce* de 64 *bits* [4]. Salsa20 es eficiente, resistente a ataques y de diseño simple. Sus ventajas incluyen velocidad elevada y facilidad de implementación en *software*. La desventaja es la menor difusión comparada con cifrados por bloques como AES, lo que puede requerir un uso cuidadoso en contextos que demanden fuerte propagación de cambios en los datos.

Triple DES, 3DES: Aplicación de DES tres veces con claves independientes [25], aumentando la seguridad frente a DES pero incrementando el costo computacional. Es compatible con sistemas DES y más seguro que DES, aunque presenta vulnerabilidad a ataques como *meet-in-the-middle* [20]. La principal desventaja es la baja velocidad y eficiencia comparada con AES.

3.2. Métricas de evaluación

En esta sección se describen las métricas utilizadas en este estudio para evaluar la calidad de encriptación de imágenes en escala de grises. Estas métricas ayudan a cuantificar el grado de diferencia entre una imagen original P y su versión cifrada C , bajo diferentes criterios, lo cual es crucial para garantizar la seguridad y aleatoriedad visual del proceso criptográfico.

En adelante, P y C denotarán imágenes de tamaño $M \times N$, donde M indica el alto y N el ancho de la imagen. De esta manera, $P(x, y)$ y $C(x, y)$ denotan el valor del píxel en la posición (x, y) de las imágenes P y C , respectivamente. A modo ilustrativo, en la Figura 3.3, se muestra una imagen (izquierda) junto con la imagen encriptada aplicando el método AES (derecha).

Coefficiente de Correlación

El coeficiente de correlación entre dos imágenes evalúa cuánto se parecen los valores del mismo píxel en la imagen original comparado con su correspondiente en la imagen encriptada. En imágenes similares suele ser alto, mientras que en imágenes cifradas debe acercarse a 0. Asimismo, esta métrica también se interpreta en función de la dependencia espacial: una alta



Figura 3.3: Imagen original P (izquierda) e imagen encriptada C (derecha).

correlación en la imagen cifrada revela que el algoritmo no logró romper las dependencias estructurales presentes en el original, lo que facilita ataques estadísticos o de solo texto cifrado. En contraposición, valores bajos de correlación indican que el cifrado dispersa efectivamente la información, dificultando inferencias basadas en la estructura de la imagen. El coeficiente de correlación está definido como

$$CC = \frac{\text{Cov}(P, C)}{\sigma_P \sigma_C}, \quad (3.5)$$

donde σ_P y σ_C son los desvíos estándar de los valores de los píxeles de P y C , respectivamente, y la covarianza se define como

$$\text{Cov}(P, C) = \frac{1}{MN} \sum_{x=1}^M \sum_{y=1}^N (P(x, y) - \bar{P})(C(x, y) - \bar{C}), \quad (3.6)$$

donde $\bar{P}$ y $\bar{C}$ representan las medias de los valores de los píxeles en las imágenes P y C , respectivamente. Si el valor de CC es próximo a cero, el esquema logra romper la estructura visual de la imagen, dificultando ataques estadísticos.

En el contexto de cifrado, un algoritmo ideal debe producir imágenes cifradas C estadísticamente independientes de P . Bajo este supuesto, la covarianza entre ambas imágenes es nula y, en consecuencia, el valor esperado del coeficiente de correlación es

$$\mathbb{E}[CC(P, C)] \approx 0. \quad (3.7)$$

Lo mismo ocurre cuando se calcula la correlación entre dos imágenes cifradas obtenidas con claves diferentes. Valores experimentales próximos a cero indican ausencia de relación lineal, lo cual es deseable para prevenir ataques basados en correlación [2, 3].

Desviación máxima

La idea de la métrica de desviación máxima es evaluar el grado de alteración que produce el proceso de cifrado en la distribución de intensidades de la imagen. Se basa en la comparación entre el histograma de la imagen original H_P y el histograma de la imagen cifrada H_C . Como ejemplo, en la Figura 3.4, se muestra el histograma de una imagen original (arriba) junto con el histograma de la imagen encriptada aplicando el método AES (abajo) de las imágenes representadas en la Figura 3.3.

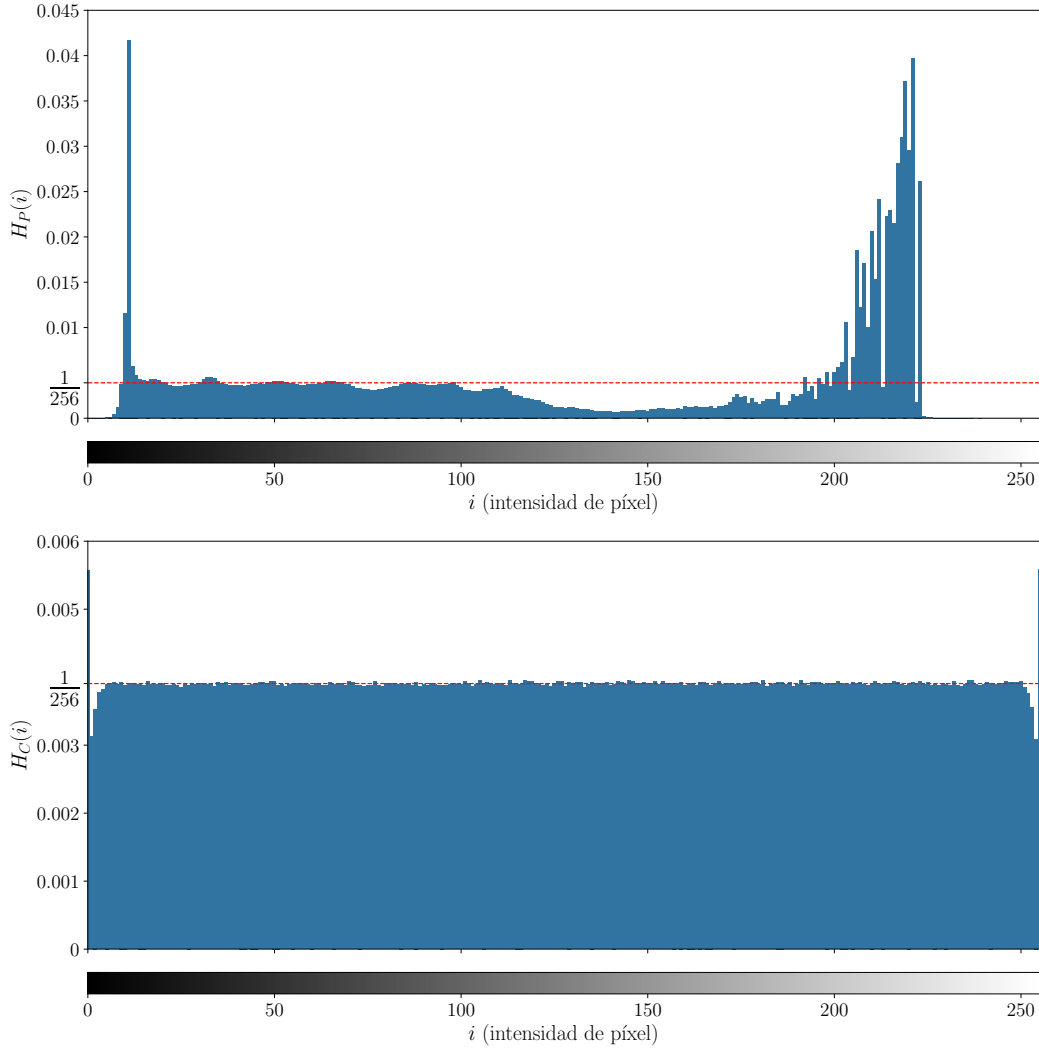


Figura 3.4: Histograma de la imagen original P (arriba) e imagen encriptada C (abajo).

El histograma representa la frecuencia relativa de cada nivel de gris en el intervalo $[0, 255]$, y un cifrado eficaz debería modificarlo sustancialmente. La diferencia se define como

$$d_i = |H_P(i) - H_C(i)|, \quad (3.8)$$

para $i = 0, 1, \dots, 255$, donde $H_P(i)$ y $H_C(i)$ son las frecuencias normalizadas para el nivel de intensidad i en la imagen original y la cifrada, respectivamente (esto se ilustra en la Figura 3.5). A partir de estas diferencias, se obtiene una medida global de la métrica dada por

$$D = \frac{d_0 + d_{255}}{2} + \sum_{i=1}^{254} d_i. \quad (3.9)$$

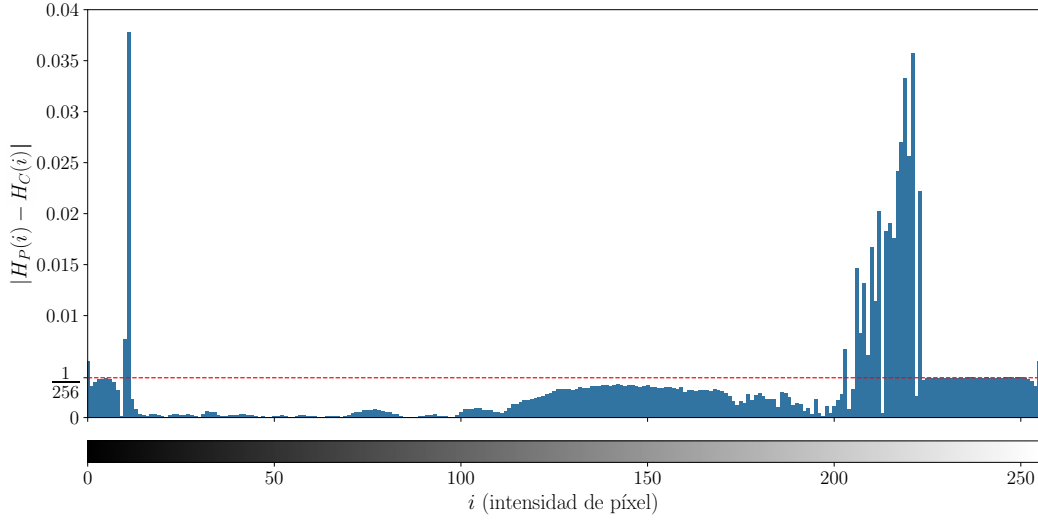


Figura 3.5: Diferencia de histogramas: $|H_P(i) - H_C(i)|$.

Entonces, si la imagen cifrada conserva una distribución similar a la original, el valor de D será bajo, lo cual revela un cifrado débil o predecible. En contraste, si el cifrado es eficaz, el histograma de la imagen cifrada se aproximará a una distribución uniforme, generando una gran discrepancia respecto de H_P , y en consecuencia, un valor alto de D .

En el caso ideal, un cifrado perfecto produce un histograma cifrado cercano a

$$U(i) = \frac{MN}{256}, \quad (3.10)$$

siendo $i = 0, 1, \dots, 255$, donde $U(i)$ representa la distribución uniforme de niveles de gris en una imagen de tamaño $M \times N$. En este escenario, la desviación máxima alcanza su valor de referencia esperado máximo, ya que

$$\mathbb{E}[D] = \frac{|H_P(0) - U(0)| + |H_P(255) - U(255)|}{2} + \sum_{i=1}^{254} |H_P(i) - U(i)|. \quad (3.11)$$

Por lo tanto, el criterio de referencia para evaluar esta métrica es que cuanto más se acerque el histograma H_C a la distribución uniforme, mayor será el valor observado de D , lo cual

constituye un indicador de un cifrado exitoso al transformar de manera significativa las estadísticas de la imagen original [3].

Cabe destacar que una desviación reducida sugiere que el cifrado no introduce suficiente aleatoriedad, lo que podría exponer el esquema frente a ataques de Criptoanálisis clásicos basados en distribuciones de frecuencias. En cambio, una alta desviación refleja que el algoritmo logra ocultar patrones explotables por adversarios, fortaleciendo así la resistencia frente a este tipo de ataques.

Desviación irregular

En el caso de la desviación irregular, esta métrica analiza el histograma de la imagen diferencia $P - C$, calculado píxel a píxel entre la imagen original P y la imagen cifrada C . El histograma resultante se denota por $(h_0, h_1, \dots, h_{255})$, donde h_i representa la frecuencia relativa del nivel de gris i en la imagen diferencia, para $i = 0, 1, \dots, 255$, tal cual se exhibe en la Figura 3.6 a modo de ejemplo.

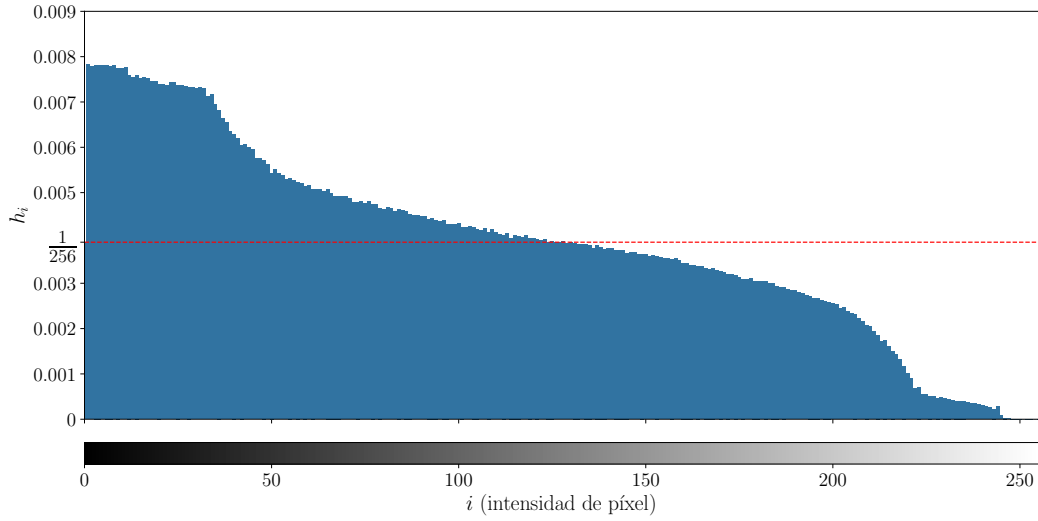


Figura 3.6: Histograma de la imagen diferencia $P - C$.

Esta métrica mide la desviación absoluta de cada valor del histograma respecto de su media aritmética (ilustrado en la Figura 3.7), definida como

$$H_D(i) = |h_i - \bar{H}_D|, \quad (3.12)$$

donde

$$\bar{H}_D = \frac{1}{256} \sum_{i=0}^{255} h_i. \quad (3.13)$$

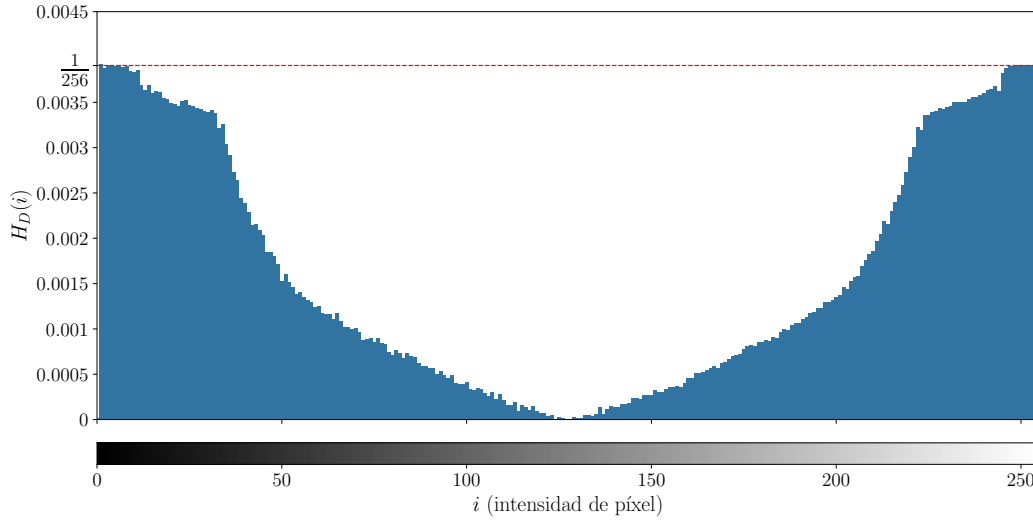


Figura 3.7: Desviación absoluta respecto de la media $H_D(i)$.

El valor global de la métrica se obtiene como la suma de las desviaciones individuales:

$$I_D = \sum_{i=0}^{255} H_D(i). \quad (3.14)$$

Cuanto menor sea el valor de I_D , más uniforme será la distribución del histograma de diferencias, lo cual es deseable ya que implica ausencia de patrones deterministas. Una desviación reducida puede sugerir que el cifrado no introduce suficiente aleatoriedad, exponiendo el esquema a ataques de Criptoanálisis clásico basados en distribuciones de frecuencias. En cambio, una desviación elevada refleja que el algoritmo logra ocultar patrones estadísticos explotables por adversarios, reforzando la resistencia del cifrado frente a ataques de solo texto cifrado.

En el caso ideal de un cifrado perfecto, el histograma de la diferencia entre las imágenes se aproxima a una distribución uniforme, es decir, $h_i \approx 1/256$ para todo $i = 0, 1, \dots, 255$. En consecuencia, la desviación respecto de la media tiende a ser nula, por lo que el valor esperado de la métrica es

$$\mathbb{E}[I_D] \approx 0. \quad (3.15)$$

Por lo tanto, un valor de I_D cercano a cero indica que el algoritmo de cifrado genera diferencias con comportamiento pseudoaleatorio, garantizando que la imagen cifrada no revela información estadística explotable sobre la imagen original [2].

Desviación respecto del histograma uniforme

La desviación respecto del histograma uniforme evalúa qué tan uniforme es el histograma de la imagen cifrada considerada de manera independiente, sin referencia a la imagen original.

En un cifrado ideal, la distribución de los niveles de gris debería aproximarse a una distribución completamente uniforme, lo cual elimina redundancias estadísticas visibles y maximiza la entropía como se describe en el análisis de la métrica Entropía de Shannon, presentado a continuación en la Sección 3.2. Alejamientos significativos de esta uniformidad constituyen debilidades explotables mediante ataques estadísticos o de texto en claro conocido, ya que permiten que un atacante relacione frecuencias de píxeles cifrados con estructuras presentes en la imagen original. Por ello, la uniformidad del histograma se interpreta como una medida de resistencia frente a este tipo de análisis.

Para calcular esta medida, se define primero el histograma idealmente uniforme, que depende del tamaño total de la imagen, como:

$$H_U(i) = \begin{cases} \frac{MN}{256} & 0 \leq i \leq 255, \\ 0 & \text{en otro caso.} \end{cases} \quad (3.16)$$

Luego, se comparan los valores observados del histograma de la imagen cifrada, H_C , con los valores uniformes esperados H_U . La desviación total se obtiene sumando las diferencias absolutas y normalizando por el número total de píxeles:

$$D_p = \sum_{i=0}^{255} \frac{|H_C(i) - H_U(i)|}{MN}. \quad (3.17)$$

Un valor bajo de D_p indica que el histograma de la imagen cifrada se aproxima a la distribución uniforme. Esta condición es deseable porque implica que la imagen cifrada carece de patrones estadísticos explotables, incrementando así la robustez del esquema de cifrado.

En un escenario de cifrado ideal, la probabilidad de cada nivel de gris tienden en promedio al valor uniforme de $1/256$, de modo que se cumple:

$$\mathbb{E}[D_p] \approx 0. \quad (3.18)$$

Por lo tanto, esta métrica cuantifica la discrepancia entre el histograma empírico y la distribución uniforme teórica. Valores de D_p próximos a cero reflejan un alto grado de aleatoriedad en la imagen cifrada, lo que reduce de manera significativa la vulnerabilidad frente a ataques de análisis de histograma [3] y otros enfoques basados en frecuencias estadísticas.

Relación señal-ruido pico

La relación señal-ruido pico es una métrica muy conocida en procesamiento de imágenes puesto que mide la relación entre la señal (imagen original) y el ruido (diferencia introducida por el cifrado), aunque tradicionalmente se usa para evaluar compresión con pérdida, también puede aplicarse para estimar qué tanto difiere una imagen cifrada de la original. La PSNR se

basa en el cálculo del error cuadrático medio entre las dos imágenes. A partir de este error, se estima cuántos decibelios de diferencia existen entre ambas imágenes. Cuanto mayor es el error, menor es la PSNR, lo que indica una diferencia mayor.

En el contexto del cifrado de imágenes, un valor de PSNR bajo es deseable, ya que significa que la imagen cifrada es muy distinta de la original. Valores altos de PSNR implican que aún persisten similitudes perceptibles entre la imagen original y la cifrada, lo que reduce la seguridad frente a ataques de solo texto cifrado. Valores bajos son indicativos de que la imagen cifrada se asemeja a ruido aleatorio, reforzando la resistencia ante análisis visuales o estadísticos.

La PSNR se emplea habitualmente para cuantificar la distorsión entre una imagen original y una procesada. Su definición parte del error cuadrático medio, definido como:

$$\text{MSE} = \frac{1}{MN} \sum_{x=1}^M \sum_{y=1}^N (P(x, y) - C(x, y))^2. \quad (3.19)$$

A partir de este valor se obtiene la PSNR como:

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right), \quad (3.20)$$

donde MAX_I es el valor máximo posible del píxel ($\text{MAX}_I = 255$ en imágenes de 8 *bits*).

Para el caso de imágenes cifradas, se asume que los valores de $C(x, y)$ son variables aleatorias independientes y uniformemente distribuidas en el rango $[0, 255]$, sin relación con los valores originales de $P(x, y)$. Por lo tanto, la diferencia $P(x, y) - C(x, y)$ es una variable aleatoria uniformemente distribuida con media cero y varianza aproximada $\sigma^2 \approx 255^2/6$ ya que la varianza de una resta de variables aleatorias es la suma de las varianzas de cada una de las variables, además la varianza de una distribución uniforme esta dada por $\sigma^2 = (b - a)^2/12$ donde a y b en este caso toman valores de 0 y 255 respectivamente.

En consecuencia, el valor esperado del MSE tiende a

$$\mathbb{E}[\text{MSE}] \approx \frac{255^2}{6} \quad (3.21)$$

y, por lo tanto, el valor esperado de la PSNR es:

$$\mathbb{E}[\text{PSNR}] \approx 10 \cdot \log_{10}(6) \approx 7.78 \text{ dB}. \quad (3.22)$$

El resultado implica que, en un escenario de cifrado ideal, la PSNR entre la imagen original y la imagen cifrada toma valores muy bajos (en torno a 8 dB), lo cual refleja una diferencia visual significativa entre ambas. En este sentido, un valor de PSNR cercano al valor esperado confirma que la imagen cifrada no conserva información visual de la original y, por ende, satisface el criterio de confidencialidad [21].

Tasa de cambio del número de píxeles

La tasa de cambio del número de píxeles, NPCR por su equivalente en inglés *Number of Pixels Change Rate*, es un indicador que se utiliza en Criptografía de imágenes para medir la sensibilidad del algoritmo de cifrado ante pequeñas variaciones en la imagen original. En particular, se evalúa el efecto de modificar mínimamente la entrada por ejemplo, alterando un único *bit* de la imagen original P , y se cuantifica el porcentaje de píxeles que cambian entre dos imágenes cifradas C_1 y C_2 , obtenidas a partir de P y de su versión modificada.

Formalmente, la NPCR se define como

$$\text{NPCR} = \frac{\sum_{x=1}^M \sum_{y=1}^N \Delta(x, y)}{MN} \cdot 100\%, \quad (3.23)$$

donde $\Delta(x, y)$ es una función indicadora definida como

$$\Delta(x, y) = \begin{cases} 1 & \text{si } C_1(x, y) \neq C_2(x, y), \\ 0 & \text{en caso contrario.} \end{cases} \quad (3.24)$$

Un valor de NPCR cercano al 100% indica que prácticamente todos los píxeles del cifrado cambian al modificar mínimamente la imagen original. Esto representa una alta difusión, característica esencial en los sistemas de cifrado seguros, ya que garantiza resistencia frente a ataques diferenciales [3, 2].

Si un cambio mínimo en la imagen original no se traduce en modificaciones significativas en la imagen cifrada, el esquema puede ser vulnerable a ataques diferenciales. Por ello, un alto valor de NPCR refleja robustez frente a este tipo de ataques, asegurando que el algoritmo de cifrado cumple con el efecto de avalancha esperado.

Se considera que C_1 y C_2 son dos imágenes cifradas generadas a partir de entradas que difieren en un único *bit*, y que el algoritmo de cifrado produce salidas estadísticamente independientes y distribuidas uniformemente en el rango de intensidades $[0, 255]$. Bajo este supuesto, para cada píxel (x, y) se cumple que la probabilidad de que $C_1(x, y) = C_2(x, y)$ es

$$\Pr[C_1(x, y) = C_2(x, y)] = \frac{1}{256}, \quad (3.25)$$

ya que ambos píxeles son independientes y cada uno puede tomar cualquiera de los 256 niveles de gris con igual probabilidad. La probabilidad de que $C_1(x, y) \neq C_2(x, y)$ es entonces

$$\Pr[C_1(x, y) \neq C_2(x, y)] = 1 - \frac{1}{256} = \frac{255}{256}. \quad (3.26)$$

Dado que la métrica NPCR mide el porcentaje de píxeles diferentes, su valor esperado es

$$\mathbb{E}[\text{NPCR}] = \frac{255}{256} \cdot 100\% \approx 99.61\%. \quad (3.27)$$

El valor ideal esperado de la métrica NPCR en imágenes de 8 *bits* por píxel es aproximadamente 99.61 %. Este resultado establece un criterio de referencia: cuanto más se acerque el valor medido al ideal, mayor será la capacidad del cifrado para generar un efecto de avalancha y resistir ataques diferenciales, lo cual es una propiedad fundamental de los esquemas de cifrado robustos.

Intensidad promedio de cambio unificada

La intensidad promedio de cambio unificada, UACI por su equivalente en inglés *Unified Average Changing Intensity*, es una métrica que cuantifica el cambio promedio en la intensidad de los píxeles entre dos imágenes cifradas derivadas de entradas ligeramente distintas, como puede ser la modificación de un único *bit* en la imagen original. Mientras que la NPCR evalúa la proporción de píxeles que cambian, la UACI mide la magnitud del cambio en dichos píxeles.

Formalmente, esta métrica se define como

$$\text{UACI} = \frac{1}{MN} \left[\sum_{x=1}^M \sum_{y=1}^N \frac{|C_1(x, y) - C_2(x, y)|}{256} \right] \cdot 100 \%, \quad (3.28)$$

donde C_1 y C_2 son dos versiones cifradas de la misma imagen P , que difieren ya sea en la clave empleada o en un pequeño cambio en la entrada.

El valor de UACI refleja cuán intensamente cambian los píxeles entre dos cifrados. Un valor bajo puede indicar que el algoritmo no altera suficientemente la imagen y, por lo tanto, puede ser predecible. Si un cambio mínimo en la imagen original no se traduce en modificaciones significativas en la imagen cifrada, el esquema puede ser vulnerable a ataques diferenciales [2, 3]. Por ello, un valor alto de UACI refleja robustez frente a este tipo de ataques, asegurando que el algoritmo de cifrado cumple con el efecto de avalancha esperado.

Bajo el supuesto de un cifrado ideal, cada píxel de las imágenes C_1 y C_2 se modela como una variable aleatoria independiente y uniforme en el rango $[0, 255]$. En este escenario, el valor esperado de la diferencia absoluta entre dos píxeles independientes se calcula como

$$\mathbb{E}[|C_1 - C_2|] = \frac{1}{256^2} \sum_{i=0}^{255} \sum_{j=0}^{255} |i - j| = \frac{255}{3}. \quad (3.29)$$

Sustituyendo este valor en (3.28), se obtiene

$$\mathbb{E}[\text{UACI}] = \frac{100}{255} \cdot \frac{255}{3} \approx 33.46 \%. \quad (3.30)$$

Entonces, el valor ideal esperado de la métrica UACI en imágenes de 8 *bits* por píxel es aproximadamente 33.46 %. Esto significa que, en un sistema de cifrado estadísticamente perfecto, el cambio promedio de intensidad entre dos imágenes cifradas se sitúa en torno a un

tercio de la escala de grises. Valores experimentales cercanos a este valor ideal constituyen una evidencia de alta sensibilidad del cifrado, propiedad deseable para resistir ataques diferenciales y garantizar seguridad en el procesamiento de imágenes.

Entropía de Shannon

El concepto de entropía introducido por Shannon en [26] puede emplearse para cuantificar el nivel de aleatoriedad en los valores de píxeles de una imagen. Es una métrica fundamental para estimar la impredecibilidad de una imagen cifrada, y se obtiene como

$$S(\mathcal{P}) = - \sum_{i=0}^{255} p_i \log_2(p_i), \quad (3.31)$$

donde $\mathcal{P} = \{p_1, p_2, \dots, p_{255}\}$, indicando la probabilidad de aparición del valor de gris i por p_i . Si todos los valores tienen la misma probabilidad de ocurrencia; es decir, $p_i = 1/256$ para $i = 0, 1, \dots, 255$, la entropía alcanza su valor máximo.

En una imagen cifrada ideal, todos los valores entre 0 y 255 aparecen con igual probabilidad. Por lo tanto, sustituyendo en (3.31),

$$S_{\text{ideal}} = - \sum_{i=0}^{255} \frac{1}{256} \cdot \log_2 \left(\frac{1}{256} \right) = \log_2(256) = 8 \text{ bits/píxel}. \quad (3.32)$$

El valor S_{ideal} corresponde al máximo teórico posible en imágenes de 8 *bits*. Luego, una entropía próxima a 8 indica que la imagen cifrada presenta una distribución uniforme de intensidades, similar a un ruido blanco, lo que dificulta la extracción de información estadística útil para un ataque. En contraste, una entropía significativamente menor implica que un adversario puede aprovechar redundancias mediante Criptoanálisis estadístico, reduciendo la resistencia frente a ataques de solo texto cifrado o de texto en claro conocido. Por ello, una entropía cercana al valor ideal se interpreta como un indicador de alta resistencia criptográfica [26].

Tiempo de ejecución

En el contexto de evaluación de algoritmos de cifrado, el tiempo de ejecución representa una métrica fundamental que mide la eficiencia temporal del algoritmo, registrando la duración total del proceso desde el inicio hasta la finalización. Esta métrica resulta significativa en aplicaciones de tiempo real o en sistemas con recursos limitados, donde se requiere minimizar la latencia. Para su cálculo, se emplearon módulos estándar de Python que permiten capturar con exactitud las marcas de tiempo antes y después de la ejecución del algoritmo; la diferencia entre ambas constituye el tiempo requerido para procesar cada imagen. Un menor tiempo de ejecución se asocia con una mayor eficiencia computacional, aunque debe analizarse en

conjunto con las métricas de seguridad y con la exposición potencial a ataques de canal lateral basados en tiempos de ejecución.

Uso de memoria

El uso de memoria es un indicador que refleja los recursos de almacenamiento temporal requeridos durante la ejecución del algoritmo. Se cuantificó el consumo de memoria en todo el ciclo de ejecución, registrando tanto el uso actual como el pico máximo alcanzado. Esta métrica es especialmente relevante en entornos de *hardware* restringido, como dispositivos embebidos o móviles, donde la memoria disponible puede ser limitada. Un consumo reducido indica mayor adaptabilidad del algoritmo a dichos entornos, mientras que un consumo excesivo podría incrementar la exposición a ataques de canal lateral, por ejemplo mediante análisis de consumo energético o de memoria.

Carga de CPU

La carga de CPU implica la medición del tiempo de procesamiento efectivamente consumido por el algoritmo de cifrado, excluyendo pausas debidas a operaciones de entrada/salida, bloqueos o inactividad del sistema. Este enfoque permite evaluar el costo computacional puro, proporcionando una métrica precisa y consistente para comparar la eficiencia entre algoritmos. Al centrarse en el uso exclusivo del procesador, la carga de CPU elimina el ruido introducido por condiciones externas de ejecución, lo que facilita análisis reproducibles y comparaciones justas entre diferentes implementaciones. Una carga elevada puede indicar una mayor complejidad algorítmica, mientras que valores moderados sugieren un mejor balance entre eficiencia y seguridad, considerando también la minimización de riesgos frente a ataques de implementación.

El análisis de estas métricas de rendimiento permite vincular la eficiencia computacional con posibles ataques criptográficos de implementación. Mientras que medidas como correlación, histograma y entropía se relacionan principalmente con la resistencia frente a criptoanálisis estadístico, otras métricas como NPCR y UACI aportan evidencia sobre la fortaleza ante ataques diferenciales. Por su parte, los parámetros de tiempo de ejecución, uso de memoria y carga de CPU permiten estimar la exposición a ataques de canal lateral y otros ataques de implementación en entornos de *hardware* restringido.

Por lo tanto, comprender estas métricas y su vínculo con posibles ataques resulta indispensable para una evaluación integral de esquemas criptográficos aplicados a imágenes. El análisis comparativo basado en ellas no solo cuantifica la robustez teórica de un algoritmo, sino también su idoneidad práctica en escenarios reales, donde adversarios con diferentes recursos pueden explotar tanto debilidades matemáticas como vulnerabilidades de implementación [21, 3].

3.3. Datos y preprocesamiento de las imágenes

Para la realización de los experimentos se empleó un conjunto de 145 imágenes originales en formato JPG, obtenidas de la plataforma <https://www.pexels.com/>. Estas imágenes presentan dimensiones del orden de 6000×4000 píxeles y abarcan tanto fotografías en color como en escala de grises.

Con el objetivo de garantizar la homogeneidad en el análisis y permitir la aplicación uniforme de los algoritmos criptográficos, todas las imágenes fueron convertidas a escala de grises. En aquellos casos en los que las imágenes ya se encontraban en escala de grises, éstas fueron preservadas sin modificaciones.

Posteriormente, y tomando como referencia la imagen de menor tamaño (5984×4000 píxeles), todas las imágenes fueron recortadas para estandarizar sus dimensiones. Se impuso además, que el número total de píxeles fuera múltiplo de 32, evitando así operaciones de *padding* y *unpadding*. Esta condición asegura que los algoritmos de cifrado puedan procesar las imágenes sin ajustes adicionales y permite comparaciones más precisas entre los distintos métodos.

A partir del conjunto inicial, se generaron recortes centrados en seis escalas relativas al tamaño común: 1:1 (100 %), 3:4 (75 %), 2:3 (66 %), 1:2 (50 %), 1:3 (33 %) y 1:4 (25 %). Todos los recortes se realizaron tomando como referencia el centro de la imagen, de manera que el contenido principal permaneciera en las versiones generadas. Se garantizó además que las dimensiones finales de cada recorte fueran múltiplo de 32 píxeles en ambos ejes.

Adicionalmente, para cada recorte se generó una variante en la que se modificó aleatoriamente el valor de un único píxel. Este procedimiento permite evaluar la sensibilidad de los algoritmos frente a perturbaciones mínimas, midiendo cómo un cambio insignificante en la entrada puede afectar la salida cifrada. De esta manera, por cada imagen original se obtuvieron 12 instancias: seis recortes y seis variantes perturbadas. Considerando los nueve algoritmos de cifrado seleccionados, el total de instancias procesadas ascendió a

$$(6 + 6) \times 9 \times 145 = 15\,660. \quad (3.33)$$

Los resultados consolidados constituyen la base para el análisis comparativo presentado en el capítulo siguiente, permitiendo contrastar la robustez, eficiencia y fidelidad de los distintos algoritmos de cifrado sobre imágenes de diferentes tamaños y con perturbaciones mínimas.

3.4. Implementación experimental y entorno de pruebas

Las pruebas se ejecutaron en un equipo portátil con las siguientes características: procesador Intel Core i7-8550U (octava generación), 8 GB de memoria RAM DDR4, almacenamiento SSD M.2 NVMe de 256 GB y sistema operativo Windows 11. El procesamiento se realizó en

un entorno virtualizado con soporte para bibliotecas de manejo de imágenes y estructuras numéricas. Todas las etapas del flujo de trabajo fueron implementadas en Python y coordinadas mediante un **Makefile** que organiza la ejecución secuencial y asegura reproducibilidad.

El desarrollo del *pipeline* de procesamiento de imágenes, cifrado y análisis se apoyó en diversas bibliotecas de Python, cada una seleccionada por sus capacidades específicas en manejo de imágenes, análisis numérico, visualización de datos y procesamiento de archivos. A continuación se describen las principales bibliotecas empleadas y su función dentro del flujo experimental.

OpenCV [7] es una biblioteca de código abierto ampliamente utilizada para el procesamiento de imágenes y visión por computadora. En este estudio, su aplicación resultó fundamental en diversas etapas del flujo de trabajo.

En primer lugar, se empleó para la lectura y escritura de imágenes en formato **JPG**, así como para la conversión de éstas a escala de grises, requisito necesario para uniformar los experimentos. Asimismo, se utilizó para el cálculo de histogramas, lo que permitió analizar la distribución de niveles de gris en las imágenes originales y cifradas.

Otra funcionalidad clave fue el cálculo de diferencias entre imágenes mediante la operación **absdiff**. Esta herramienta resultó esencial para obtener métricas de seguridad como I_D , D_p , NPCR y UACI, que permiten evaluar la robustez del proceso de cifrado frente a pequeñas variaciones en los datos.

Finalmente, **OpenCV** se utilizó para la manipulación de matrices de píxeles con el fin de generar recortes centrados y versiones perturbadas de las imágenes, lo cual fue necesario para los experimentos comparativos realizados en este trabajo.

NumPy [13] es una biblioteca fundamental para el cálculo numérico en Python. En este trabajo, su uso fue clave para realizar operaciones matemáticas de manera eficiente mediante arreglos y cálculos vectorizados.

En primer lugar, se utilizó para la conversión de imágenes a estructuras de tipo **ndarray**, lo que permitió aprovechar la computación vectorizada propia de la biblioteca. Esto resultó esencial para acelerar el procesamiento de grandes volúmenes de datos y facilitar la aplicación de operaciones sobre los píxeles.

Además, **NumPy** se empleó en el cálculo de métricas estadísticas, tales como la media, la desviación estándar y la covarianza, necesarias para la obtención del coeficiente de correlación. También fue ampliamente utilizada en operaciones aritméticas y lógicas requeridas en métricas de evaluación de seguridad como PSNR, NPCR, UACI y otras medidas criptográficas.

Por último, la biblioteca permitió la normalización de histogramas y el cálculo de desviaciones respecto a distribuciones uniformes, lo que posibilitó un análisis detallado de la aleatoriedad introducida por los algoritmos de cifrado.

Pandas [28] es una biblioteca especializada en la manipulación y análisis de datos estructurados. En este trabajo, desempeñó un papel central en la gestión de la información producida durante el proceso de cifrado y evaluación de métricas. Su utilización incluyó la

lectura y escritura de archivos en formato **CSV**, los cuales fueron generados por los *scripts* de cifrado y de cálculo de métricas. A partir de estos registros, **Pandas** permitió la consolidación de la información mediante la operación **merge**, integrando datos de los *logs* de encriptación junto con las métricas en un único archivo estructurado, coherente y ordenado.

Asimismo, la biblioteca resultó fundamental en la preparación de los datos para su posterior visualización. Esto incluyó tareas de filtrado y ordenamiento en función del nombre del archivo, el algoritmo utilizado y el tamaño de la imagen, lo que facilitó tanto el análisis comparativo como la elaboración de gráficos estadísticos.

Matplotlib [14] y **Seaborn** [32] fueron las bibliotecas utilizadas para la visualización de resultados. Su uso permitió transformar los datos obtenidos en representaciones gráficas claras y comparables.

En primer lugar, **Matplotlib** proporcionó la infraestructura base para la generación de gráficos en dos dimensiones, sirviendo como soporte esencial para la construcción de las figuras. Sobre esta base, **Seaborn** aportó herramientas de mayor nivel para la creación de gráficos estadísticos, entre ellos *boxplots* y *lineplots*, que facilitaron la comparación de métricas entre distintos algoritmos y tamaños de imagen.

Finalmente, ambas bibliotecas se configuraron de manera cuidadosa en lo referido a estilos visuales, tamaños de fuente, rotación de etiquetas y diseño de leyendas. Estas personalizaciones garantizaron la claridad y la consistencia en la presentación de los resultados obtenidos, mejorando la interpretación de los gráficos en el análisis comparativo.

Tqdm [8] es una biblioteca ligera orientada a mejorar la interacción con procesos de ejecución prolongada. En este trabajo, se empleó para brindar retroalimentación visual durante la ejecución de los *scripts* que procesan grandes volúmenes de imágenes.

Su principal aporte fue la incorporación de barras de progreso, las cuales permitieron monitorear en tiempo real el estado de las operaciones. Esto incluyó tanto el cálculo de métricas como la generación de gráficos, ofreciendo al usuario una visión clara del avance de cada tarea. De esta manera, **tqdm** contribuyó a mejorar la trazabilidad y la usabilidad del *pipeline*, haciendo más eficiente y comprensible el seguimiento de las etapas de procesamiento.

Argparse [30] es la biblioteca estándar de Python para la definición de interfaces de línea de comando. En este trabajo, se utilizó para parametrizar la ejecución de los distintos *scripts* sin necesidad de modificar directamente el código fuente.

Gracias a su integración, fue posible establecer argumentos que definieron directorios de entrada y salida, nombres de archivos **CSV** y diversas opciones de configuración. Esto otorgó flexibilidad al usuario y permitió adaptar el proceso a diferentes escenarios de experimentación. Asimismo, **argparse** facilitó la incorporación de los *scripts* en un flujo de trabajo automatizado mediante **Makefile**, contribuyendo a la modularidad y reproducibilidad del *pipeline*.

Otras bibliotecas estándar. Además de las herramientas principales, se emplearon varias bibliotecas incluidas en la biblioteca estándar de Python que resultaron esenciales para complementar el flujo de trabajo.

Las bibliotecas `os` y `sys` se utilizaron para la gestión de rutas de archivos, la verificación de la existencia de directorios y documentos, así como para el control de la salida estándar en situaciones de error. Por su parte, `datetime` permitió registrar marcas temporales en los *logs* de procesamiento, garantizando un seguimiento preciso de cada etapa de ejecución. Finalmente, la librería `csv` facilitó la escritura de resultados en archivos de este formato, siendo especialmente útil para el almacenamiento de los *logs* generados durante el cálculo de métricas.

En conjunto, estas bibliotecas permitieron implementar un *pipeline* reproducible, combinando procesamiento de imágenes, análisis estadístico y generación de visualizaciones, garantizando así la evaluación objetiva de los algoritmos criptográficos estudiados.

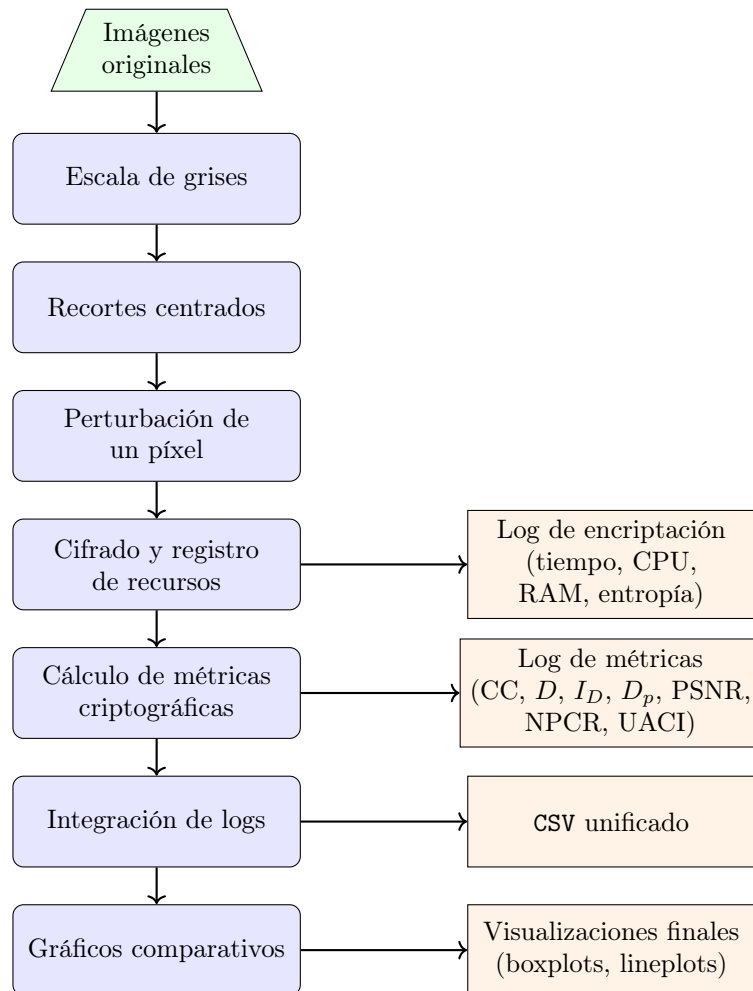


Figura 3.8: *Pipeline* de procesamiento de imágenes, cifrado, cálculo de métricas y generación de gráficos.

3.5. Flujo de procesamiento y análisis

El procesamiento de imágenes y la evaluación de los algoritmos de cifrado se organizaron en un *pipeline* secuencial, representado en la Figura 3.8, dividido en las siguientes siete etapas principales, cada una implementada mediante un *script* específico en Python.

1. **Conversión a escala de grises:** convierte todas las imágenes a escala de grises para garantizar uniformidad. Aquellas imágenes ya en escala de grises se conservan sin modificación.
2. **Recorte y generación de versiones:** realiza recortes centrados en escalas relativas al tamaño original, preservando la estructura visual central de la escena. Las dimensiones finales se aseguran como múltiplo de 32 píxeles, evitando *padding* y simplificando la comparación entre algoritmos.
3. **Generación de perturbaciones puntuales:** introduce variaciones mínimas en cada recorte, modificando aleatoriamente un píxel. Este procedimiento permite evaluar la robustez de los algoritmos ante cambios sutiles en la entrada.
4. **Cifrado de las imágenes:** aplica los nueve algoritmos de cifrado presentados en la Sección 3.1 a todas las instancias generadas, almacenando las imágenes cifradas junto con metadatos relevantes (algoritmo, modo de operación, parámetros utilizados). Además, registra métricas de rendimiento como tiempos de ejecución, uso de CPU y memoria, y entropía en `log_encryption.csv`. El orden de procesamiento de los algoritmos se asigna aleatoriamente para evitar optimizaciones implícitas del intérprete de Python.
5. **Cálculo de métricas de seguridad y fidelidad:** compara imágenes originales, cifradas y perturbadas, calculando las métricas como CC, D , I_D , D_p , PSNR, NPCR y UACI. Los resultados se registran en `log_metrics.csv`.
6. **Integración de resultados:** unifica los archivos de *log* de encriptación y métricas en un único CSV consolidado (`log_merge.csv`), ordenado por nombre de archivo, tamaño y algoritmo. Esto facilita los análisis comparativos posteriores.
7. **Visualización de resultados:** genera gráficos comparativos en formato *boxplot* (agrupados por algoritmo) y *lineplot* (en función del tamaño de archivo), permitiendo identificar tendencias, evaluar desempeño criptográfico y eficiencia computacional.

Capítulo 4

Experimentos y Resultados

En este capítulo se presentan los resultados obtenidos, los cuales se muestran mediante dos tipos de representaciones gráficas. Para llevar a cabo el análisis comparativo, se empleó un enfoque basado en la distribución de los datos representadas gráficamente mediante *boxplots*. Esta técnica permite observar de forma conjunta medidas de tendencia central (mediana), dispersión (rango intercuartílico e intervalos adyacentes), asimetría y presencia de valores atípicos o *outliers*.

En primer lugar, se recopilaron los valores de cada métrica obtenida en las pruebas experimentales de cifrado del conjunto de imágenes, según lo expuesto en la Sección 3.2. Posteriormente, se agruparon los resultados según el algoritmo empleado y se construyeron *boxplots* individuales y combinados para cada métrica. Con ello se visualiza la distribución estadística de los datos y se facilita la comparación entre algoritmos. De esta manera, el análisis metodológico no se limita únicamente a valores promedios, sino que contempla la variabilidad, la consistencia y los posibles comportamientos anómalos en cada métrica, lo que otorga una visión más completa del desempeño de los algoritmos evaluados. En segundo lugar, se utilizaron gráficos de líneas que muestran la variación de las métricas en función del tamaño de la imagen procesada, agrupando también por algoritmo.

En la Figura 4.1, se presentan los resultados correspondientes al coeficiente de correlación, las medianas se mantienen cercanas a cero en todos los algoritmos, indicando la eliminación de correlaciones entre píxeles adyacentes. La dispersión es baja ($IQR \approx 0.00037$), aunque se detectan *outliers* entre 7 y 26 según el algoritmo. AES y ARC2 presentan mayor número de valores extremos, mientras que Blowfish y Salsa20 muestran menor variabilidad. En general, todos los cifrados cumplen con la falta de correlación esperada (0, indicada en una línea de trazos en rojo), logrando una adecuada difusión de la estructura visual de la imagen original, cumpliendo satisfactoriamente con el objetivo de eliminar patrones perceptibles en la misma.

En la Figura 4.2 se muestra la variación de la desviación máxima entre las imágenes originales y las cifradas, observando una relación directa entre el tamaño de la imagen y el valor de esta métrica, alcanzando magnitudes del orden de 10^7 , lo que sugiere una elevada

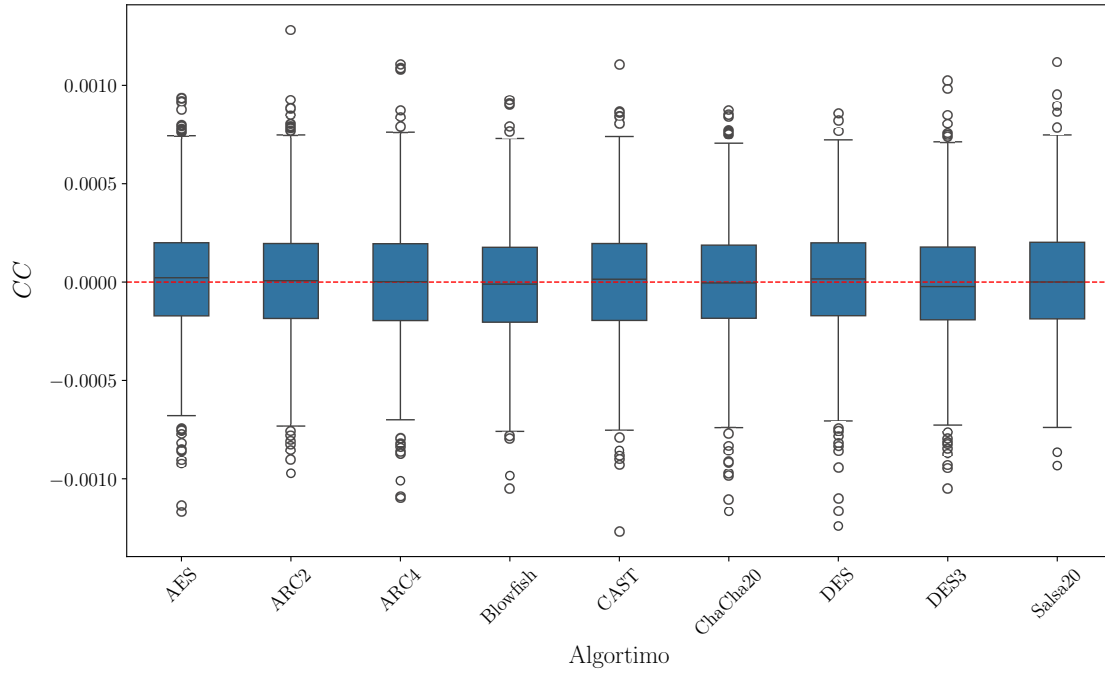


Figura 4.1: Coeficiente de correlación por algoritmo.

discrepancia entre la imagen original y la cifrada. Asimismo, independientemente del algoritmo de cifrado empleado, no se evidencia una superioridad significativa de ninguno en particular, dado que las curvas correspondientes a cada uno de los algoritmos evaluados se encuentran prácticamente superpuestas.

En el caso de la desviación irregular, cuyos valores se muestran en la Figura 4.3, las medianas se encuentran en torno a 0.52, con dispersión leve ($IQR \approx 0.08$). Al igual que en la desviación máxima, el comportamiento es homogéneo, por lo tanto, no resulta discriminante para identificar un algoritmo con un desempeño superior respecto a los demás.

Cuando se analiza la desviación respecto del histograma uniforme, tal cual se muestra en la Figura 4.4, al igual que lo observado para la Figura 4.2, las curvas se encuentran casi superpuestas no evidenciando una preferencia en alguno de los algoritmos, pero con la particularidad de que a medida que aumenta el tamaño de la imagen analizada se logra una mejora en esta métrica ya que la misma tiende a valores mínimos. Se podría decir, entonces, que a medida que aumenta el tamaño de la imagen, la información visual se distribuye más uniformemente en toda la imagen cifrada, tendiendo al valor ideal 0, indicado en una línea de trazos en rojo.

En el análisis de los valores para PSNR, que se exhiben en la Figura 4.5, se concluye que no existe una variabilidad apreciable para cada uno de los algoritmos que se ponen a prueba en este trabajo de investigación. En la Figura 4.5, los valores de PSNR se sitúan en

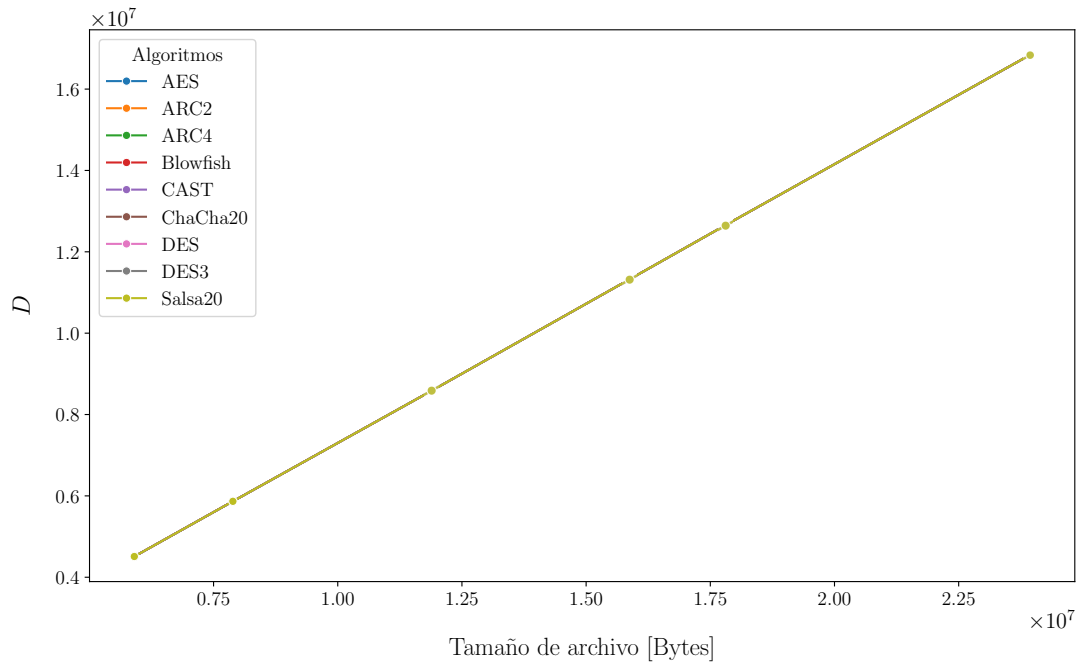


Figura 4.2: Desviación máxima en función del tamaño de imagen y por algoritmo.

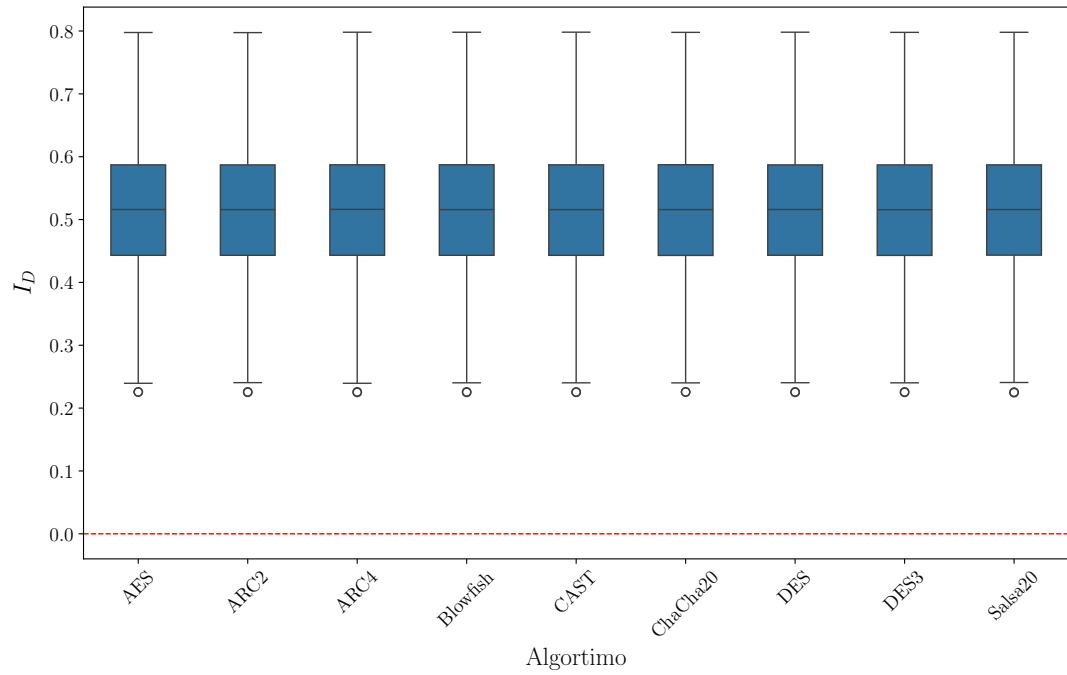


Figura 4.3: Desviación irregular por algoritmo.

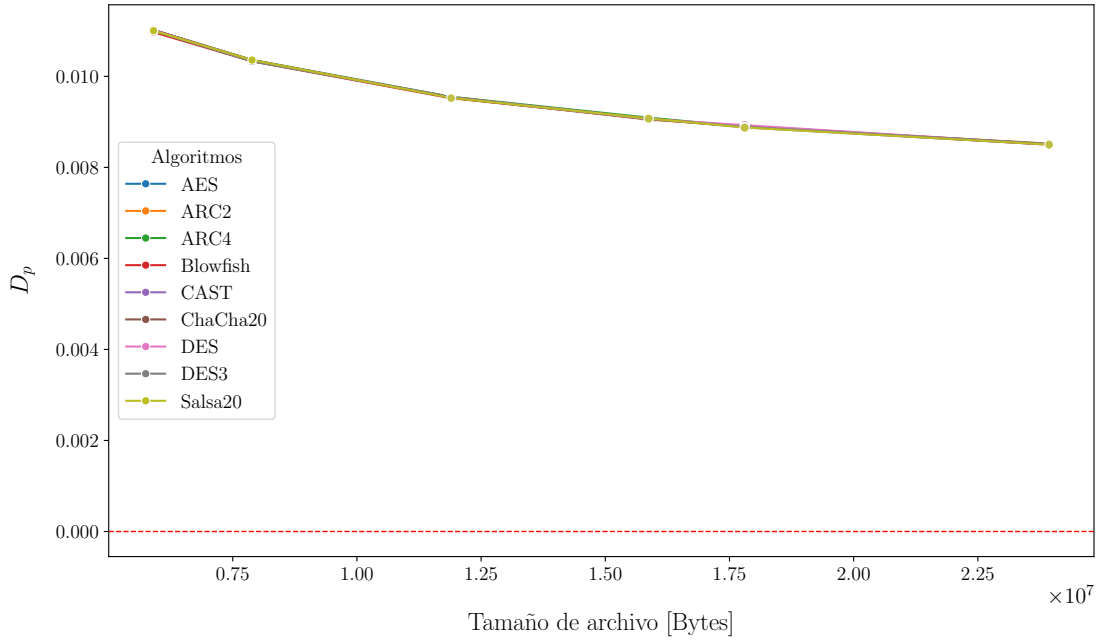


Figura 4.4: Desviación respecto del histograma uniforme en función del tamaño de imagen y por algoritmo.

torno al valor ideal de 7.78dB para todos los algoritmos (indicado en una línea de trazos en rojo), sin variabilidad significativa. Este nivel corresponde al esperado al comparar imágenes estadísticamente independientes, confirmando que todos los algoritmos eliminan efectivamente la estructura visual original, aunque esta métrica no discrimina diferencias de desempeño entre ellos.

El análisis de la tasa de cambio del número de píxeles, representada en la Figura 4.6, muestra que los resultados obtenidos son uniformes e independientes del algoritmo empleado. Esto indica que todos los diseños evaluados están orientados a que una pequeña perturbación en la imagen original genere diferencias significativas en la imagen cifrada, presentando una variación inferior al 0.014 %. Con una mediana en todos los algoritmos muy próxima al valor esperado de 99.61 % indicado en una línea de trazos en rojo.

Adicionalmente, en la Figura 4.7, se observa que el valor de UACI varía levemente con el tamaño de la imagen, en torno a un 0.25 % entre los diferentes tamaños evaluados, permaneciendo todos cercanos al 33.46 %, indicado en una línea de trazos en rojo. Este valor es considerado óptimo para imágenes en escala de grises de 8 *bits*, ya que garantiza que un pequeño cambio en la imagen original provoca una alteración significativa en la imagen cifrada.

De manera similar, el análisis de la entropía de Shannon a partir de la Figura 4.8, muestra que el valor de esta métrica varía únicamente de forma marginal con el tamaño de la

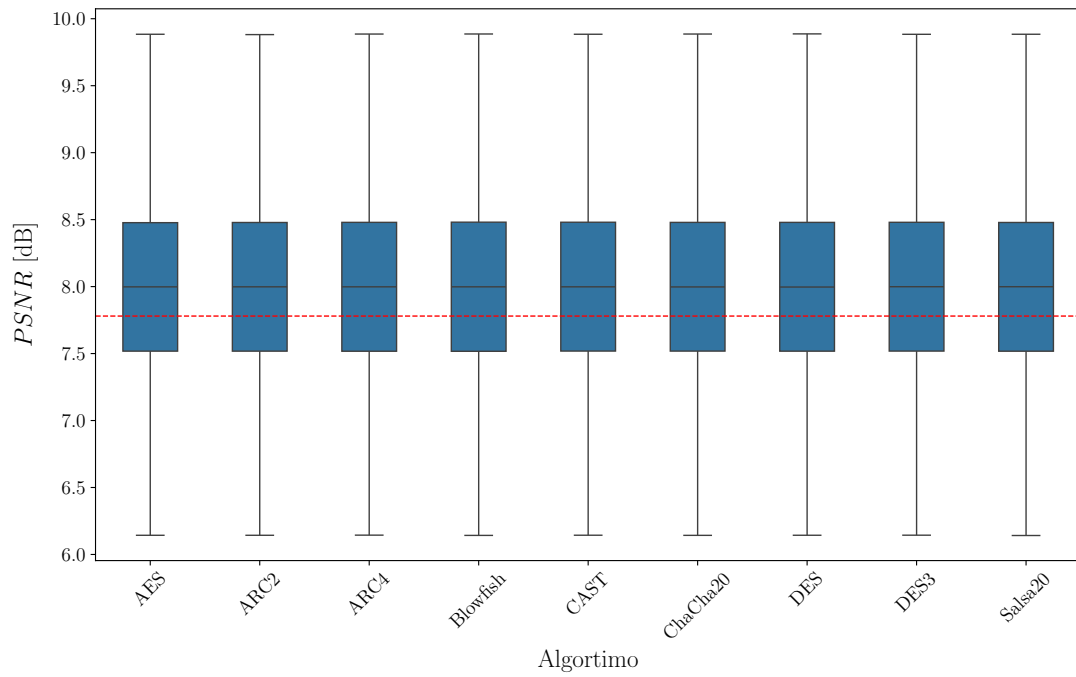


Figura 4.5: Relación señal-ruido pico por algoritmo.

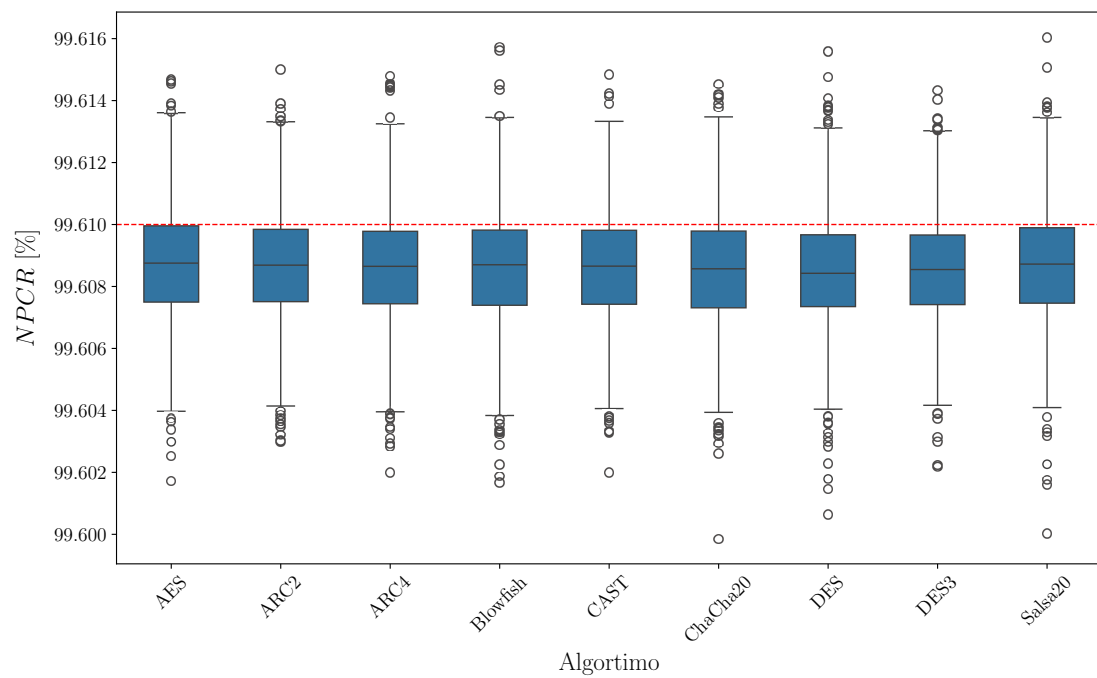


Figura 4.6: Tasa de cambio del número de píxeles por algoritmo.

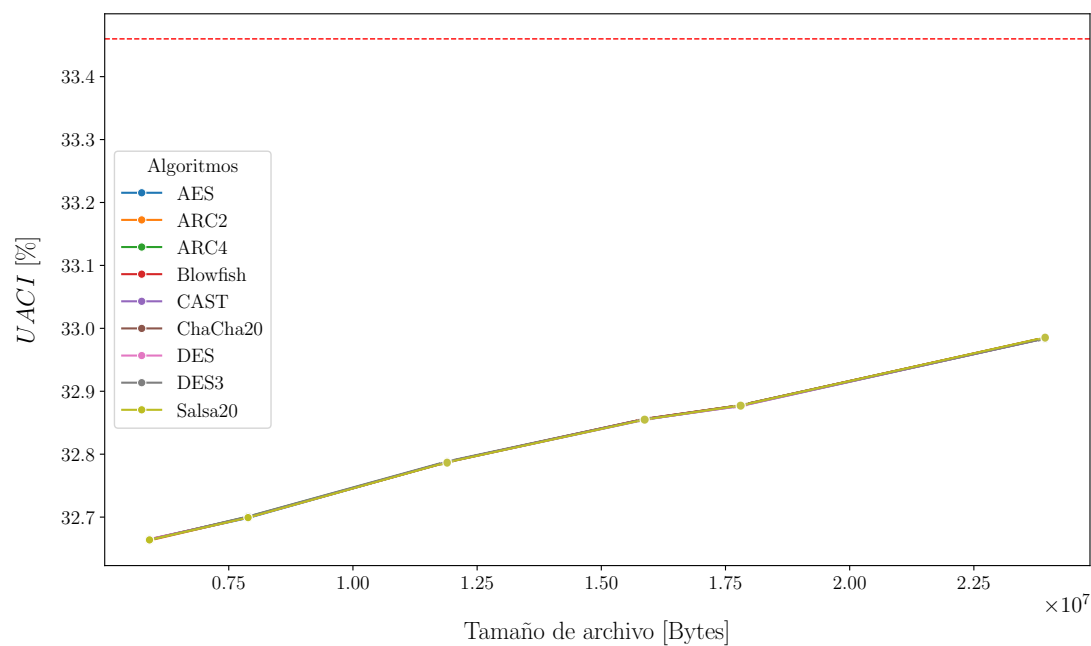


Figura 4.7: Intensidad promedio de cambio unificada en función del tamaño de imagen y por algoritmo.

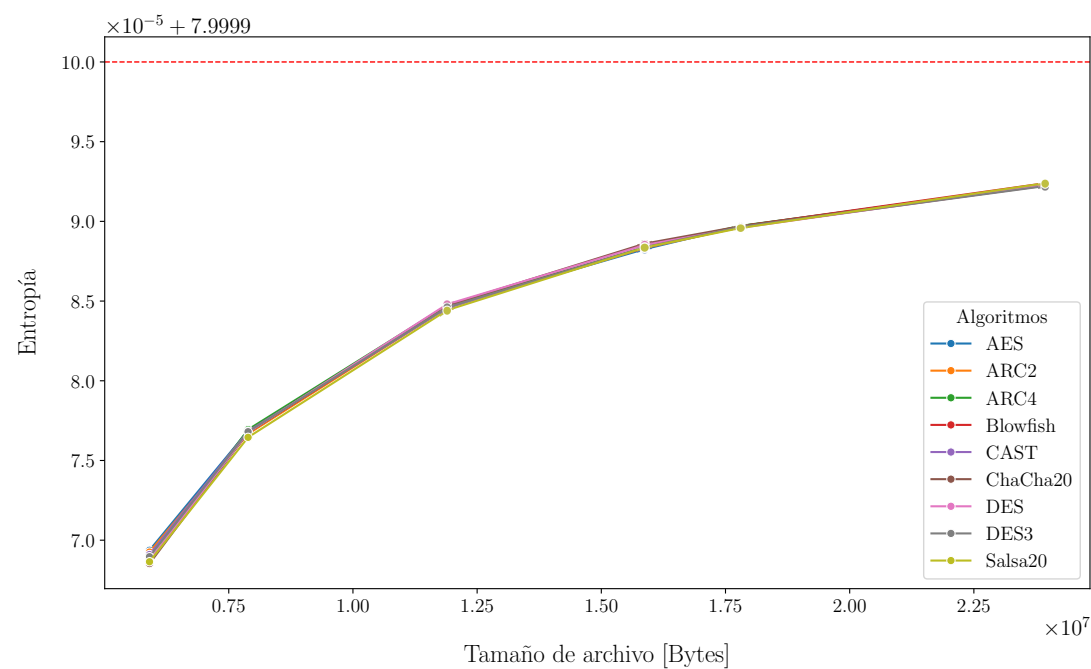


Figura 4.8: Entropía de Shannon en función del tamaño de imagen y por algoritmo.

imagen, mientras que no presenta diferencias significativas entre los algoritmos evaluados, ya que las curvas correspondientes se encuentran completamente superpuestas. Analizando la Figura 4.9, todos los algoritmos alcanzan valores cercanos al máximo teórico de 8 *bits* por píxel, indicado en una línea de trazos en rojo, con una mediana ≈ 7.999987 y una dispersión mínima ($IQR \approx 0.000013$) y no se observan *outliers*. La asimetría es ligeramente negativa y homogénea en todos los casos. Esto confirma que los cifrados generan distribuciones altamente aleatorias e indistinguibles de ruido, sin diferencias significativas entre algoritmos.

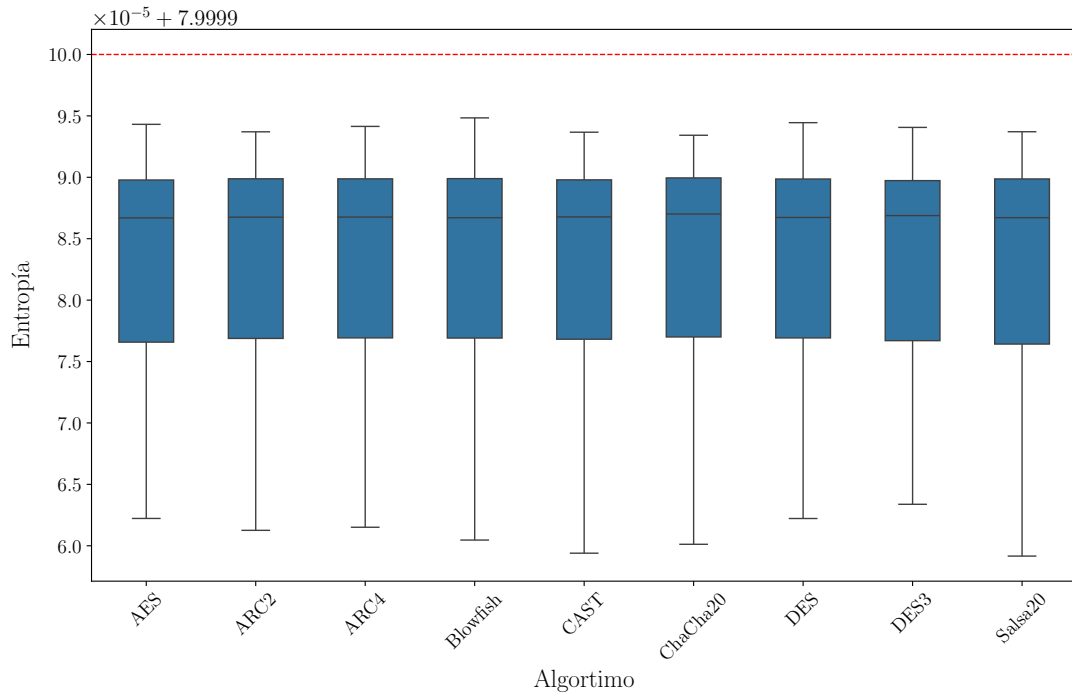


Figura 4.9: Entropía de Shannon por algoritmo.

El análisis conjunto de las métricas de tiempo de ejecución y carga de CPU, presentadas en las Figuras 4.10 y 4.11, respectivamente, evidencia una correlación directa entre ambas, ya que el uso de CPU y el tiempo de procesamiento están intrínsecamente relacionados en un entorno computacional. A partir de estas representaciones, se observa que los algoritmos con mayor demanda de recursos, tanto en tiempo como en carga de CPU, son 3DES y ARC2, cuyos valores se sitúan por encima de la media del conjunto evaluado. En contraste, los algoritmos con menor tiempo de ejecución fueron AES y ARC4, seguidos por ChaCha20 y Salsa20. Indicado en líneas de trazos en rojo en la figura el valor ideal deseado.

Por último, en la Figura 4.12, que representa el uso de memoria del sistema, se observa que no existe variabilidad apreciable entre los diferentes algoritmos evaluados, independientemente del tamaño de las imágenes empleadas en el ensayo. Todos los algoritmos presentan un consumo de memoria prácticamente idéntico (mediana ≈ 13.5 MB, $IQR = 9.68$ MB), con

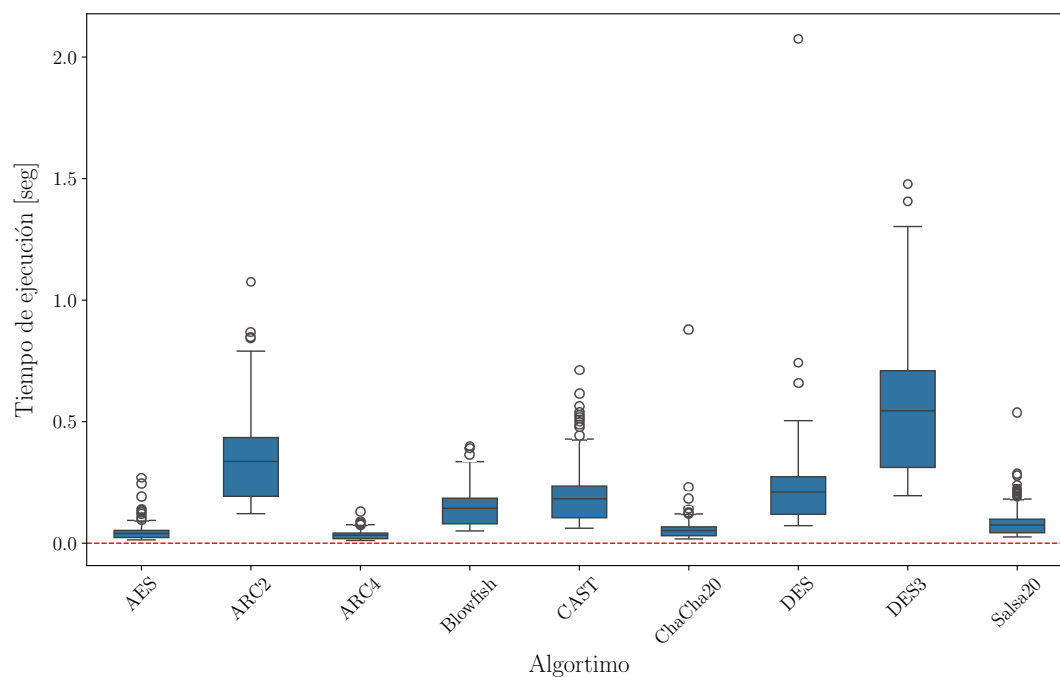


Figura 4.10: Tiempo de ejecución por algoritmo.

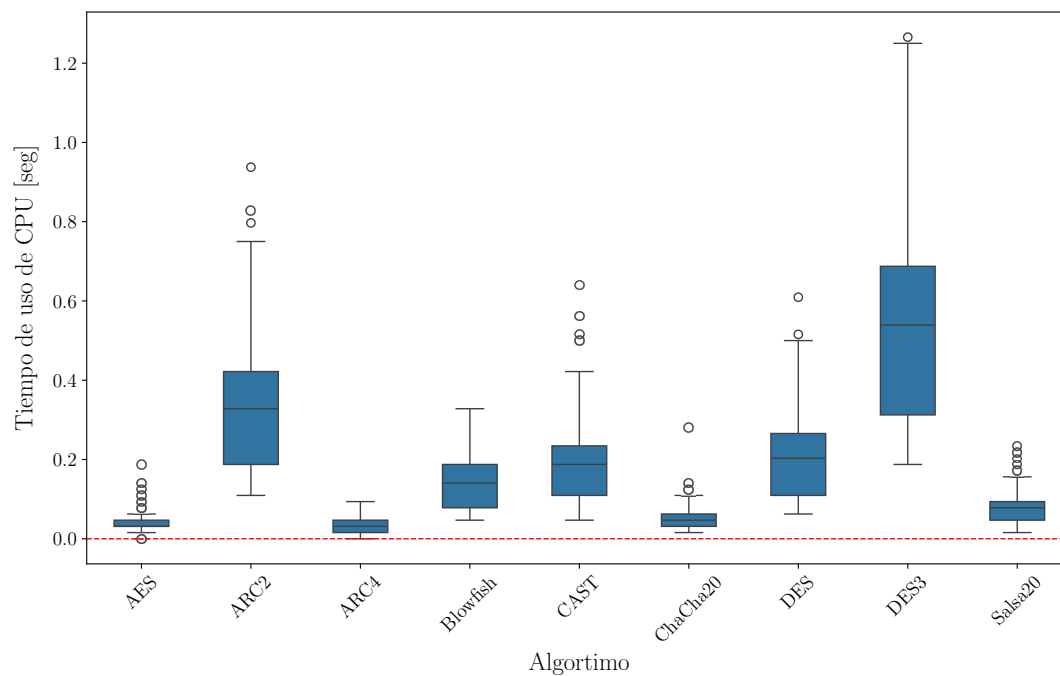


Figura 4.11: Carga de CPU por algoritmo.

asimetría leve positiva y sin *outliers*. Esto indica que el factor diferenciador principal entre algoritmos no radica en la memoria, sino en el tiempo de ejecución.

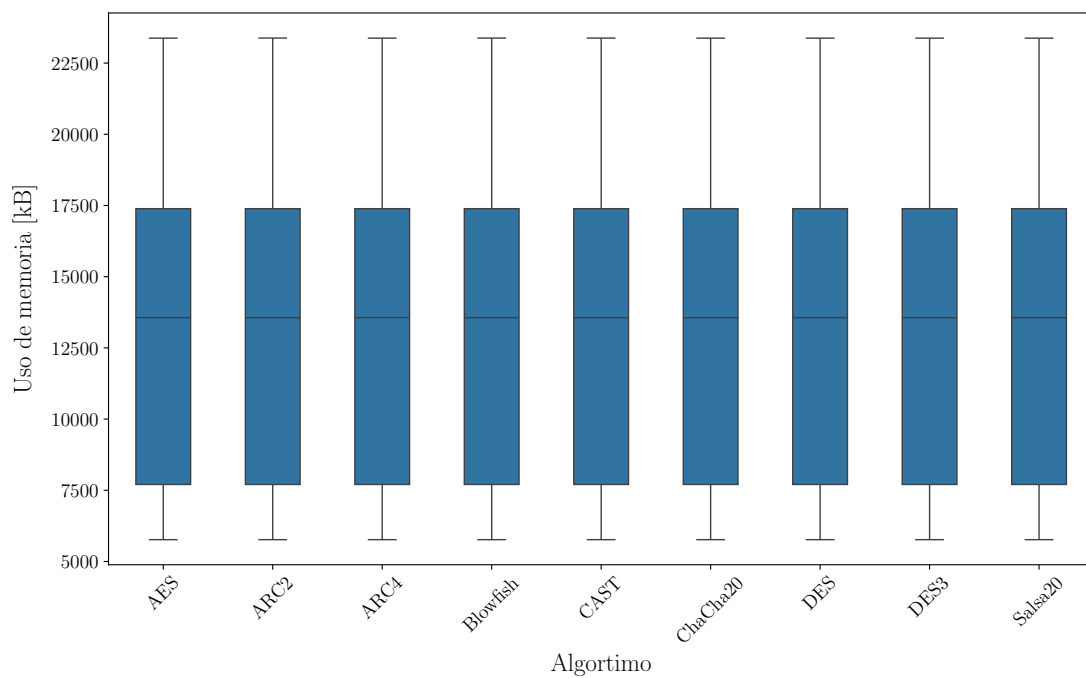


Figura 4.12: Uso de memoria por algoritmo.

Capítulo 5

Conclusiones y Trabajo Futuro

En este trabajo se estudió el rendimiento de nueve algoritmos criptográficos aplicados a imágenes en escala de grises. En base al análisis de ciertas métricas de calidad y rendimiento generadas a partir de las mediciones realizadas y evaluando el desempeño de cada cifrado a partir de cada una de las métricas propuestas en la metodología, el análisis integral muestra lo siguiente.

- En términos de difusión (entropía, coeficiente de correlación, tasa de cambio del número de píxeles, intensidad promedio de cambio unificada), todos los algoritmos presentan un desempeño excelente, con valores muy próximos y diferencias mínimas.
- En métricas de calidad (relación señal-ruido pico, desviación respecto del histograma uniforme, desviación máxima, desviación irregular), el comportamiento es igualmente homogéneo y consistente entre algoritmos.
- El factor diferenciador principal se encuentra en el rendimiento temporal: ARC4 y AES son los más rápidos, seguidos por ChaCha20 y Salsa20. Blowfish y CAST tienen tiempos intermedios, mientras que DES y 3DES resultan los más lentos.
- El uso de memoria es uniforme en todos los casos, por lo que no constituye un criterio de selección relevante.

Se concluye que, en el caso del cifrado de imágenes en escala de grises de 8 *bits*, los algoritmos seleccionados cumplen los objetivos requeridos para su uso. En particular, permiten una transformación que distribuye efectivamente la estructura visual de la imagen original a través de toda la imagen cifrada resultante. En consecuencia, la elección del algoritmo debería priorizar el tiempo de ejecución cuando la eficiencia es crítica (favoreciendo a ARC4 o AES), dado que las métricas de difusión y calidad no muestran diferencias significativas entre ellos.

El código desarrollado para la implementación de los algoritmos de encriptación, de las métricas de calidad descriptos en este trabajo, los *scripts* de pre o posprocesamiento, así como el conjunto completo de imágenes originales utilizadas en las pruebas, se encuentra

disponible en un repositorio de GitHub, accesible desde <https://github.com/pablo-origlia/tesina-licenciatura>.

Bibliografía

- [1] Adams, C. (1997). The CAST-128 encryption algorithm. RFC 2144.
- [2] Ahmad, J. and Ahmed, F. (2012). Efficiency analysis and security evaluation of image encryption schemes. *International Journal of Video & Image Processing and Network Security (IJVIPNS-IJENS)*, 12(4):18–29.
- [3] Alghamdi, Y. and Munir, A. (2024). Image encryption algorithms: A survey of design and evaluation metrics. *Journal of Cybersecurity and Privacy*, 4:126–152.
- [4] Bernstein, D. J. (2008). The Salsa20 family of stream ciphers. In *New stream cipher designs: the eSTREAM finalists*, pages 84–97. Springer.
- [5] Bernstein, D. J. et al. (2008). ChaCha, a variant of Salsa20. In *Workshop record of SASC*, volume 8, pages 3–5. Lausanne, Switzerland.
- [6] Biham, E. and Shamir, A. (1991). Differential cryptanalysis of des-like cryptosystems. In *Advances in Cryptology – CRYPTO’90*, pages 2–21. Springer.
- [7] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*.
- [8] da Costa-Luis, C., Larroque, S. K., Altendorf, K., Mary, H., richardsheridan, Korobov, M., Yorav-Raphael, N., Ivanov, I., Bargull, M., Rodrigues, N., Shawn, Dektyarev, M., Górný, M., mjstevens777, Pagel, M. D., Zugnoni, M., JC, CrazyPython, Newey, C., Lee, A., pgajdos, Todd, Malmgren, S., redbug312, Desh, O., Nechaev, N., Boyle, M., Nordlund, M., MapleCCC, and McCracken, J. (2024). tqdm: A fast, extensible progress bar for Python and cli.
- [9] Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES–The Advanced Encryption Standard*. Springer.
- [10] Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.
- [11] Feistel, H. (1973). Cryptography and computer privacy. *Scientific American*, 228(5):15–23.
- [12] Fluhrer, S., Mantin, I., and Shamir, A. (2001). Weaknesses in the key scheduling algorithm of RC4. In *Selected Areas in Cryptography (SAC 2001)*, pages 1–24.
- [13] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

- [14] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95.
- [15] Kaliski, B. (1998). RC2 encryption algorithm. RFC 2268.
- [16] Kerckhoffs, A. (1883). La cryptographie militaire. *Journal des Sciences Militaires*.
- [17] Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (1997). *Handbook of Applied Cryptography*. CRC Press.
- [18] NIST (1999). Data encryption standard (DES). *FIPS 46-3*.
- [19] NIST (2001). Advanced encryption standard (AES). *FIPS 197*, 197(1).
- [20] NIST (2017). Transitioning the use of cryptographic algorithms and key lengths. Technical Report NIST Special Publication 800-131A Rev. 2, U.S. Department of Commerce.
- [21] Paar, C. and Pelzl, J. (2009). *Understanding cryptography: A textbook for students and practitioners*. Springer Science & Business Media.
- [22] Popov, A. (2015). Prohibiting RC4 cipher suites. RFC 7465.
- [23] Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.
- [24] Schneier, B. (1993). Description of a new variable-length key, 64-bit block cipher (Blowfish). *Fast Software Encryption*, 809:191–204.
- [25] Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 2nd edition.
- [26] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.
- [27] Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715.
- [28] The pandas development team (2020). pandas-dev/pandas: Pandas.
- [29] Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Journal of Math*, 58(345-363):5.
- [30] Van Rossum, G. and Drake Jr, F. L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.
- [31] Vernam, G. S. (1926). Cipher printing telegraph systems: For secret wire and radio telegraphic communications. *Journal of the AIEE*, 45(2):109–115.
- [32] Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.
- [33] Wu, Y. et al. (2011). NPCR and UACI randomness tests for image encryption. *Cyber Journal of Multidisciplinary Sciences*, 1(2):31–38.