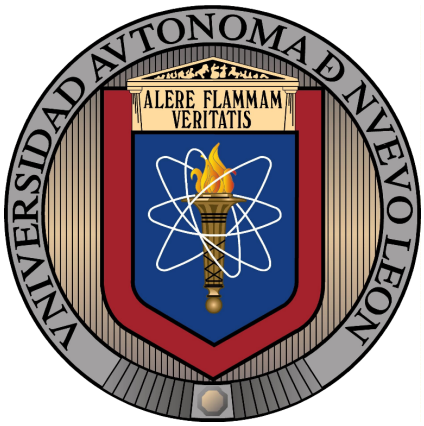
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado




# Implementación de códigos polares usando esquemas caóticos de orden fraccionario

por

Harairis Brunet César

en opción al grado de

Maestría en Ciencias

de la Ingeniería Eléctríca

San Nicolás de los Garza, Nuevo León Septiembre, 2024

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado


# Implementación de códigos polares usando esquemas caóticos de orden fraccionario

por

Harairis Brunet César

en opción al grado de

Maestría en Ciencias

de la Ingeniería Eléctríca

San Nicolás de los Garza, Nuevo León Septiembre, 2024

# UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
## Facultad de Ingeniería Mecánica y Eléctrica
## Posgrado

Los miembros del Comité de Evaluación de Tesis recomendamos que la Tesis "Implementación de códigos polares usando esquemas caóticos de orden fraccionario", realizada por la estudiante Harairis Brunet César, con número de matrícula 2173343, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestría en Ciencias en Ingeniería Eléctrica.

**El Comité de Evaluación de Tesis**

Dr. José Ramón Rodríguez Cruz
Director

Dr. Cornelio Posada Castillo
Co-director

Dr. José Antonio de la O Serna
Revisor

Dr. Miguel Ángel Platas Garza
Revisor

Dr. Ernesto Zambrano Serrano
Revisor

Vo.Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

Institución 190001

Programa 557520

Acta Núm. 4410

Ciudad Universitaria, a 20 de septiembre del 2024

# Agradecimientos

A mi mamá y a mi papá por su amor, dedicación y apoyo en todas las decisiones que tomo. Gracias a ambos he alcanzando la mayoría de los logros en mi vida.

A mi esposo, por su amor y su paciencia. Gracias por haberme seguido en mi aventura fuera de Cuba.

Al Dr. Jose Ramón Rodríguez Cruz, asesor de tesis y profesor de MCIE, por sus enseñanzas, consejos oportunos y buen estado de ánimo. Muchas Gracias por haberme guiado en este laberinto académico.

A mis compañeros de la MCIE, por haber traído un pedacito de Cuba.

Al Dr. Vázquez, coordinador de la MCIE, por habernos aceptado en el postgrado.

A al Dr. José Antonio de la O Serna, al Dr. Ernesto Zambrano Serrano y al Dr. Miguel Ángel Platas Garza por sus consejos para la confección de este documento.

A los profesores de la MCIE por compartir sus conocimientos y experiencia.

A CONAHCyT por la beca otorgada.

A todos los que contribuyeron a que pudiera llegar México a estudiar esta Maestría.

# Resumen

Harairis Brunet César.
Candidato para el grado de Maestro en Ciencias en Ingeniería Eléctrica. Universidad Autónoma de Nuevo León. Facultad de Ingeniería Mecánica y Eléctrica. Título del estudio:

Implementación de códigos polares usando esquemas caóticos de orden fraccionario

Número de páginas: 117.

Resumen: MLC NAND flash memories are crucial in all disruptive technologies due to their data access speed, energy efficiency and reliability. Providing the right price and performance for a range of applications. These devices are employed in technologies such as Artificial Intelligence, IoT, Big Data, and Cloud Computing. To enhance error correction and durability of flash memories, this research suggests the implementation of polar codes. Additionally, a fractional chaotic encryption system is applied for improving the security of stored information.

Firma del asesor: _____

Dr.José Ramón Rodríguez Cruz

# Índice general

# Índice de figuras

# Índice de tablas

Capítulo 1

# Introducción

## 1.1 Escenario

El reconocimiento facial se ha consolidado como una tecnología avanzada, fundamental en diversas aplicaciones de seguridad. Su funcionamiento se basa en la comparación de una imagen del rostro humano con otra almacenada en una base de datos, con el objetivo de confirmar la identidad de un individuo. Las bases de datos que almacenan imágenes para el reconocimiento facial son cruciales en campos como la biometría, aplicaciones financieras, seguridad de la información, cumplimiento de la ley, control de acceso en organizaciones, el entrenamiento de aplicaciones de inteligencia artificial, entre otras. Debido a su amplitud de uso y la criticidad de sus aplicaciones, estas bases de datos deben ser capaces de gestionar grandes volúmenes de datos y múltiples transacciones simultáneamente.

Para cumplir con estas demandas, estos sistemas se implementan comúnmente en SSD (Solid State Disk). Los SSD son soluciones de almacenamiento que ofrecen baja latencia de acceso aleatorio, alta eficiencia energética y una fiabilidad superior en comparación con los discos duros tradicionales. Uno de sus componentes principales de los SSD, son las memorias flash.

Actualmente las memorias flash son componentes indispensables para cualquier infraestructura. Se emplean como almacenamiento en las tecnologías disruptivas como la Inteligencia Artificial, el Internet de las cosas (IoT), el Blockchain el Big

Data. Son elementos básicos en equipamiento de automatización diseñados para la Industria 4.0 y las Ciudades Inteligentes [1].

Estos dispositivos se emplean también como el medio principal de almacenamiento para teléfonos inteligentes, computadoras portátiles, por su tamaño compacto y la constante disminución del costo por bit [2].

Existen 2 tipos de memorias flash: NAND y NOR, las cuales reciben su nombre por la forma que se hace la asignación específica de los datos ("No AND") o ("No OR"). Las memorias flash NAND tienen mayor aplicación en la industria que las NOR ya que permiten el acceso a datos en modo de página. Las memorias flash NOR son utilizadas sistemas BIOS para suministrar la funcionalidad de arranque y equipos de redes de telecomunicaciones ya que permite la recuperación de volúmenes de datos pequeños [3].

En 2023, el mercado de memoria flash se valora en USD 75.7 billones y se proyecta que durante el período 2024 - 2030, experimentará una tasa de crecimiento anual compuesta (CAGR) del 5.1 % . Para el año 2030, se espera que alcance un valor de 107.24 billones de dólares, por lo que extender el ciclo de vida útil de estos dispositivos además de los beneficios que provee, disminuiría la huella ecológica de esta tecnología [4] [5]. La figura 1.1 muestra el análisis del mercado de memoria flash en el período (2024-2030).

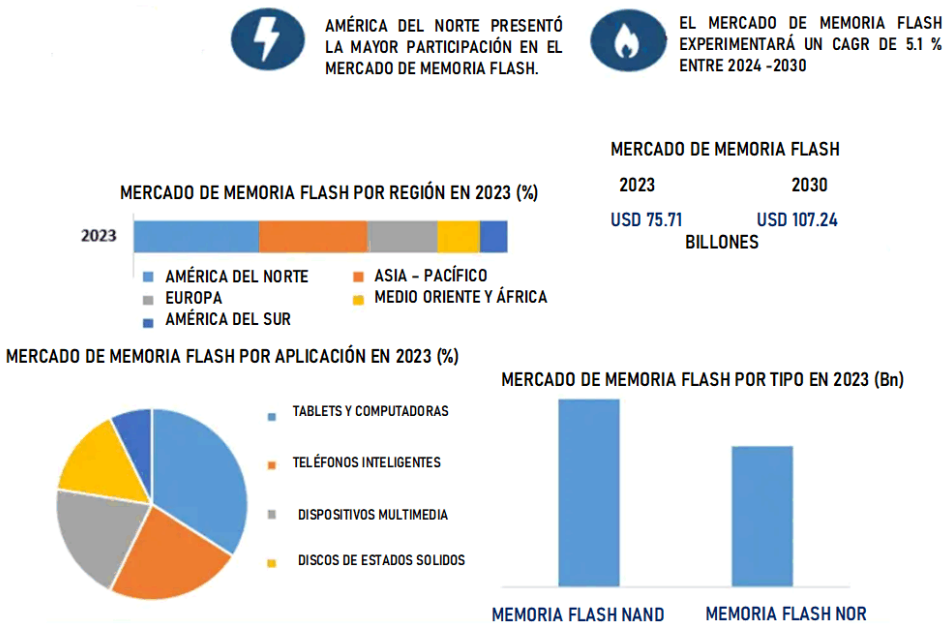



**Figura 1.1:** Análisis del mercado de memoria flash (2024-2030).

A pesar de las numerosas ventajas que ofrecen las memorias flash, estos dispositivos tiene una vida útil limitada. La utilización sistemática puede provocar la pérdida de la integridad en los datos almacenados, por lo que se necesitan procedimientos para mitigar estos efectos adversos, prolongar la durabilidad del dispositivo y garantizar la seguridad de la información que contienen.

### 1.1.1 Estructura de las memorias flash NAND

Existen varios tipos de memorias flash NAND, los cuales se diferencian por la cantidad de bits que se pueden almacenar por celda. Otras diferencias entre estos dispositivos son el costo, la capacidad y la fiabilidad [6–8]

Los tipos comunes de memoria flash NAND son SLC [1], MLC [2], TLC [3] y 3D - NAND [4]. La fiabilidad de estas memorias está determinada por el número de ciclos

[1]Las NAND SLC almacenan 1 bit de información por celda
[2]Las NAND MLC almacenan 2 bits de información por celda.
[3]Las NAND TLC almacena 3 bits de información por celda.
[4]La disposición de los transistores puede ser en dos dimensiones (2D) o de tres dimensiones (3D).

de programación/borrado (P/E) que puede sufrir una celda de flash [5] antes de que comience a desgastarse.

Las memorias flash NAND SLC tienen la fiabilidad más alta del grupo anterior, un tamaño de bloque de 64 Kbyte, una temperatura de trabajo de $-40^0C$ a $105^0C$, capacidades de hasta 512 Gbyte, un consumo de energía de lectura y escritura entre $0.002 - 0.0025$ kW*h y un P/E de 100000 ciclos [6] [10, 11] [12].

Las memorias flash NAND MLC manejan un tamaño de bloque de 128 Kbyte, una temperatura de trabajo de $-40^0C$ a $85^0C$, capacidades de hasta 2 Tbyte, un consumo de energía de lectura y escritura entre $0.005200 - 0.005700$ kW*h y un P/E de 10000 ciclos [6] [10, 11] [13].

Las memorias flash NAND TLC manejan capacidades de almacenamiento de hasta 8 Tbyte, temperatura de trabajo de entre $0^0C$ a $70^0C$, un consumo de energía de lectura y escritura entre $0.000458 - 0.000908$ kW*h y un P/E 3.000 ciclos [11] [14].

Las memorias flash NAND MLC son las memorias de mayor utilización por su capacidad para ejecutar operaciones de lectura/escritura con latencias muy pequeñas y tener el menor costo por bit del grupo anterior, en comparación con sus homólogos mecánicos [15].

La figura 1.2 muestra la relación entre los ciclos de programación/borrado, el tamaño de la memoria flash y la probabilidad de error para las memorias flash NAND MLC [2].

[5] Es un circuito electrónico que almacena información binaria y debe de ser activado para almacenar un valor lógico de 1 (nivel alto de voltaje) y reseteado para almacenar un valor lógico de 0 (nivel bajo de voltaje). El valor de la celda de memoria se mantiene hasta que sea cambiado por el proceso de programación/borrado [9].

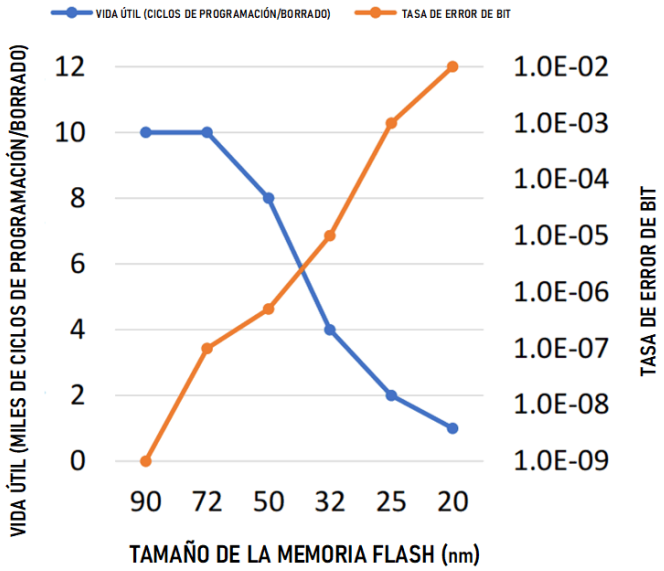



**Figura 1.2:** Relación entre los ciclos de programación/borrado, el tamaño de la memoria flash y tasa de error de bit.

Estos dispositivos se representan usando $2^2$ bits o niveles de tensión umbral (Vth) que no se superponen entre ellos. Cada celda consta de un transistor que puede almacenar carga, la cual representa un determinado valor de datos en función de la tensión umbral del transistor correspondiente [16–18]. Los transistores de este tipo de memoria, son transistores de compuerta flotante, similares a los transistores MOSFET, los cuales retienen la carga al permanecer eléctricamente aislados.

La figura 1.3, muestra los cuatro estados posibles (ER, P1, P2, P3) de una memoria flash NAND MLC, junto con sus correspondientes valores de 2 bits, el bit más significativo (MSB) y el bit menos significativo (LSB).

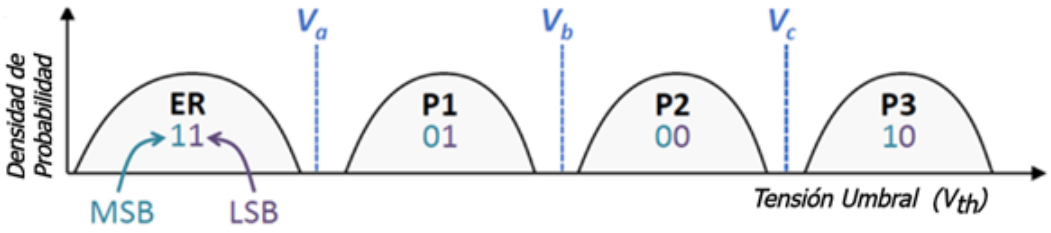



**Figura 1.3:** Distribución del voltaje de umbral para una memoria flash NAND MLC.

El estado de cada celda puede leerse aplicando uno de los tres voltajes de

referencia de lectura (Va, Vb y Vc). Antes de que cada celda pueda programarse a un nuevo estado, la celda debe borrarse (estado ER). Para leer el LSB de una celda de varios niveles, se diferencian los estados en donde el valor del LSB es 1 (ER y P1) de los estados donde el valor del LSB es 0 (P2 y P3), mediante el voltaje de referencia de lectura Vb. Para leer el MSB, se diferencian los estados en donde el valor de MSB de 1 (ER y P3) de aquellos con un valor de MSB de 0 (P1 y P2), por lo que se necesita determinar si el voltaje umbral de la celda se encuentra entre Va y Vc, por lo que se aplica cada uno de estos voltajes de referencia de lectura para determinar el MSB [16–18].

Debido a la variación en las operaciones del programación, y a la variación del proceso de fabricación, el voltaje umbral de las celdas programadas en el mismo estado sigue una distribución Normal, representada en la figura 1.3, como una curva de esta distribución para cada estado [16–18].

Muchas investigaciones refieren a la distribución Normal como modelo de canal, una vez conseguidos los datos estadísticos de un modelo, ya que en eventos aleatorios independientes y con contribuciones pequeñas, la distribución de la suma (o promedio) de esas contribuciones se aproxima a una distribución Normal, independientemente de la forma de la distribución original [19].

Una memoria flash NAND MLC está estructurada en 2 bloques básicos: un banco de celdas de memoria o *dado* y un controlador. El banco de celdas de memoria es la unión de varias celdas del mismo tipo que conforman el almacenamiento. El controlador proporciona una interfaz entre el host y el banco de celdas de memoria para administrar el directorio del sistema de archivos flash y permite almacenar, leer y borrar los datos. El controlador maneja también el $firmware$ [6] específico de los códigos correctores de errores y de seguridad, la gestión de bloques defectuosos y la nivelación ante el desgaste [20, 21].

La interacción entre el bloque de memoria flash y el controlador se puede

[6]Software que se encuentra en los dispositivos electrónicos. Se encarga de controlar el hardware y puede ser fácilmente modificado por el usuario final.

clasificar en 2 tipos principales: sin formato (Raw) y administrado (Managed). La interacción Raw tiene diferentes versiones para memorias flash NAND SLC, MLC y TLC, y requiere administración externa. La interacción Managed incorpora administración de memoria junto al banco de celdas, lo que simplifica el proceso de diseño [20, 21].

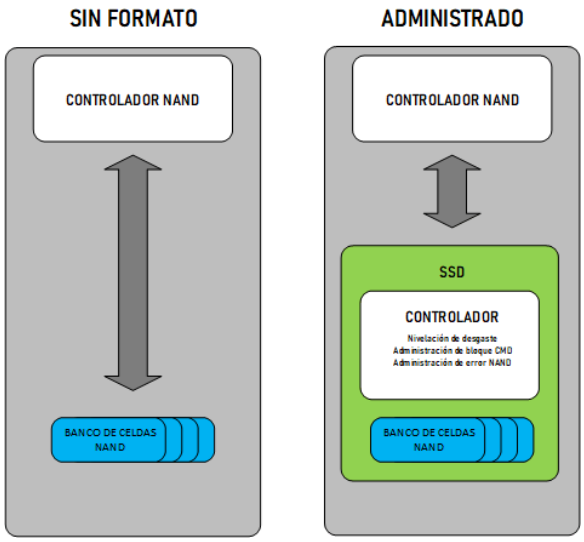



**Figura 1.4:** Estructura de una memoria flash.

Existen controladores de memorias flash NAND de diferentes arquitecturas. Entre las arquitectura más comunes de encuentran la eMMC (Embedded Multi-Media Card), UFS (Universal Flash Storage) y la ONFI (Open NAND Flash Memory) [21].

La arquitectura eMMC, maneja bloques de memoria flash NAND SLC, cuenta con 4 particiones, 4 modos de operaciones, 6 registros de diferente longitud y 13 pines. Esta arquitectura maneja solo transmisión sincrónica, acepta 64 comandos diferentes de 48 bits de longitud para su programación y códigos de redundancia cíclica [21]. La figura 1.5 muestra un diagrama en bloque de la arquitectura eMMC. En este dispositivo la señal del reloj es emitida por el host y se utiliza para sincronizar la transmisión de datos y la operación del dispositivo de almacenamiento. Durante un ciclo de reloj, las señales CMD y DAT0-7 permiten la transmisión de un bit en modo SDR (Single Data Rate). La señal DAT0-7 gestiona la transmisión entre el host y

el eMMC y puede configurarse en modo DDR (Double Data Rate), permitiendo la transmisión de dos bits por ciclo de reloj. La señal CMD es bidireccional y se emplea para que el host envíe comandos y el eMMC responda. Después del encendido o reinicio del dispositivo, solo DAT0 está habilitada, luego de la inicialización, el bus de datos puede configurarse para usar DAT0-3 o DAT0-7, permitiendo un ancho de bus de 4 o 8 bits respectivamente. Finalmente, la señal de sincronización es enviada por el eMMC al host con la misma frecuencia [21].

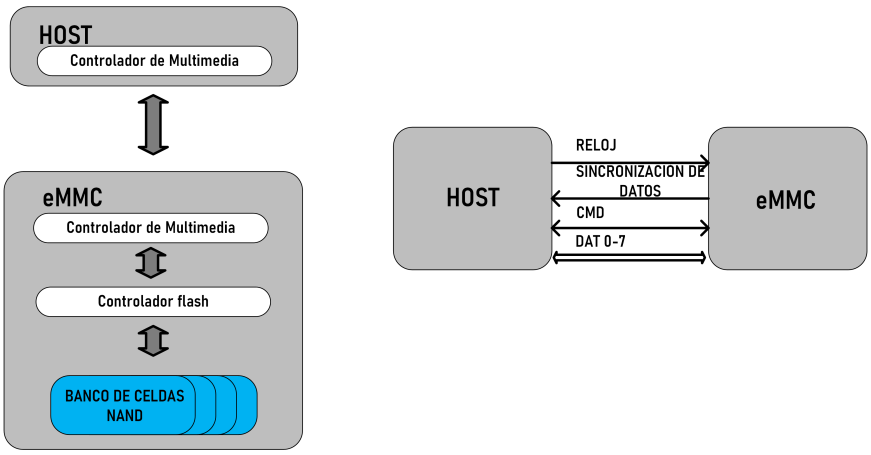



**Figura 1.5:** Diagrama en bloque de la arquitectura eMMC.

La arquitectura UFS, maneja bloques de memoria flash NAND SLC, cuenta con 8 unidades lógicas configurables, 3 particiones, 7 modos de operación, 37 registros de 32 bits de longitud y 14 pines. Maneja transmisión síncrona y asíncrona, acepta 27 comandos de 32 bits de longitud para su programación y códigos de redundancia cíclica [21].

La figura 1.6 muestra un diagrama en bloque de la arquitectura UFS. En la arquitectura UFS, las señales DOUT_C (Data Output Command/Control) y DOUT_T (Data Output True/Negative) se encargan de la salida de información desde el controlador hacia host. La señal DOUT_C transporta comandos y señales de control, mientras que la señal DOUT_T maneja la transferencia de datos reales. Por otro lado, las señales DIN_T (Data Input Negative/True) y DIN_C (Data Input Command/Control) se utilizan para la entrada de datos y comandos desde el host hacia

el controlador. La señal DIN_T transfiere los datos entrantes y DIN_C los comandos de control. La señal C/D (Command/Data) se utiliza para diferenciar entre instrucciones de control e instrucciones de datos. La señal REF_CLK (Referencia de Reloj) proporciona la sincronización temporal necesaria para las transferencias de datos y comandos, asegurando la correcta coordinación entre el host y el controlador. El bloque de lógica central se encarga de coordinar las operaciones internas en la comunicación entre el boque de entrada/salida NAND y el host. El bloque de entrada/salida NAND maneja las operaciones entre el bloque de lógica central, así como las interacciones de las señales de Control y Datos emitidas desde y hacia del bloque de memoria flash NAND [21].

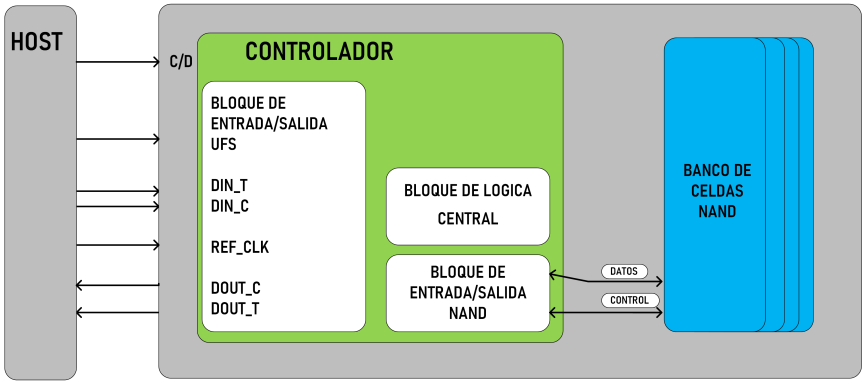



**Figura 1.6:** Diagrama en bloque de la arquitectura UFS.

La figura 1.7 muestra un diagrama en bloque de la arquitectura ONFI versión 5.0. La arquitectura ONFI maneja bloques de memoria NAND SLC y MLC, cuenta con 1 partición, 2 modos de operación y 48 pines, transmisión síncrona y asíncrona, 32 comandos de 8 o 16 bits de longitud y códigos correctores de errores [21].

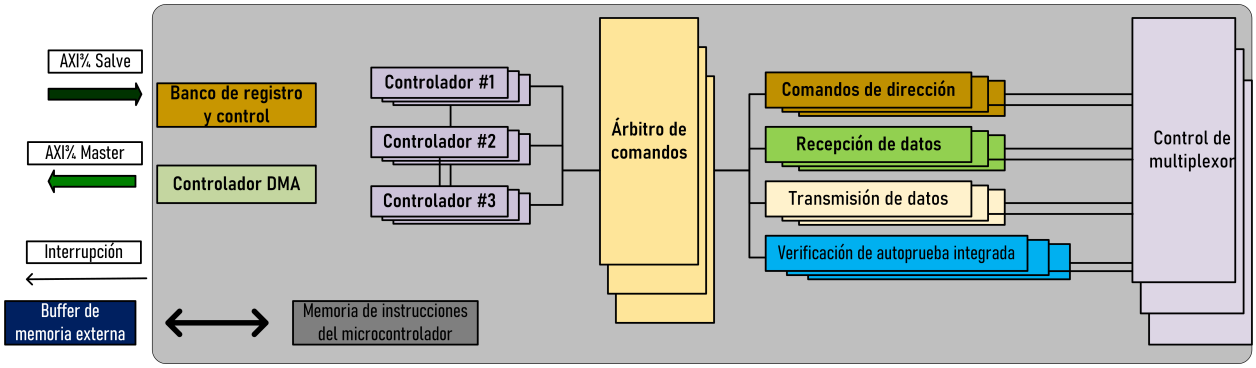



**Figura 1.7:** Diagrama en bloque de la arquitectura ONFI 5.0.

La arquitectura ONFI en su versión 5 está compuesta por interfaces y microprocesadores que permiten mejorar su funcionamiento con respecto a versiones anteriores. Admite hasta 32 microcontroladores, los cuales pueden intercambiar directamente con los bloques de memoria flash por separado. Este controlador se conecta a través de puertos AXI [7] mapeados a los bloques de memoria flash que permiten transferencias entre 32 y 1024 bits. Estos puertos pueden utilizarse para enviar instrucciones directamente a los bloques de memoria flash, programar el controlador DMA o activar manualmente copias de datos de una SRAM [8] (Static Random Access Memory) desde y hacia los bloques de memoria flash. El controlador DMA es utilizado asegurar las transferencias rápidas. El banco de memoria y control guarda el estado y los resultados de las operaciones en los bancos de memoria flash, en función de verificar que las tareas se completen exitosamente. El controlador de multiplexación administra el acceso a los bancos memoria flash. Los comandos de dirección se encargan de gestionar la ubicación de los bloques, páginas o celdas específicas donde se realiza las operaciones de lectura o escritura. Los comandos de recepción y transmisión de datos son los encargados de transferir los datos desde el host hacia el bloque de memoria flash en las operaciones de escritura [22] [23].

Existe un grupo de trabajo para arquitectura ONFI que desarrolla una interfaz estandarizada para memorias flash NAND, que permite la interoperabilidad de di-

[7] Proporciona la interfaz para los puertos de un aplicación.

[8] Memoria de acceso aleatorio. No necesita ser refrescada constantemente para mantener los datos almacenados.

ferentes fabricantes, reduce el tiempo de diseño y comercialización para una amplia gama de aplicaciones. Esta iniciativa define especificaciones de interfaz a nivel de componentes, así como especificaciones de forma de conectores y módulos para estos dispositivos, en función de simplificar la integración de estos dispositivos productos electrónicos de consumo, plataformas informáticas y cualquier otra aplicación que requiera [22].

### 1.1.2 Limitaciones de las memorias flash NAND MLC

Los problemas de fiabilidad de las memorias flash NAND MLC están causados por varios tipos de ruidos que surgen de la escritura, lectura o inactividad de la memoria. Estos ruidos desplazan la distribución del voltaje umbral, de la referencia de lectura. La figura 1.8, muestra la posición relativa de las distribuciones originales y las desplazadas. Como consecuencia de este desplazamiento, algunas celdas se leen erróneamente como si estuvieran en un estado diferente al que fueron programadas, resultando un aumento de los errores a lo largo de múltiples ciclos de operaciones de programación/borrado. Por lo anterior, la carga eléctrica almacenada en una celda flash se pierde con el tiempo, y la tasa de error aumenta durante el tiempo de inactividad o tras la aplicación del mecanismo de programación/borrado de la celda en múltiples ocasiones.

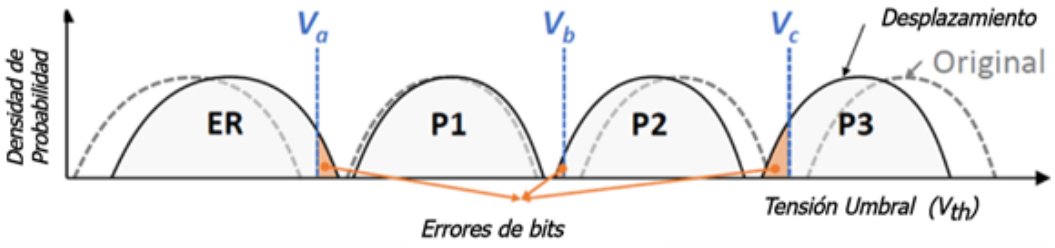



**Figura 1.8:** Distribución de la tensión umbral desplazada para la memoria flash NAND MLC.

Otros impedimentos en la comunicación que provocan estos ruidos son los errores de lectura, errores de programación, la pérdida de la carga y la interferencia entre celdas.

Los errores de programación están dados por la carga acumulada en la compuerta flotante de las celdas vecinas que no están programadas en el momento de la escritura. El voltaje umbral de la celda que se está programando aumenta y puede dar lugar a un error de bit.

Los errores de lectura están dados por la carga acumulada en la compuerta flotante de celdas vecinas que no se están leyendo en el momento de la lectura. El voltaje umbral en la celda que se está leyendo aumenta y puede dar lugar a un error de bit. Este error es alarmante cuando el número de lectura entre 2 operaciones de borrado es grande.

Las pérdidas de carga en el tiempo suceden cuando un bloque de memoria con programación pierde la carga y se produce una caída del voltaje umbral, que provoca errores de bits y afecta a la capacidad de retención de datos de la memoria.

La interferencia entre celdas es producida por la capacitancia parásita de los transistores, la cual produce un aumento de la carga en las celdas programadas con niveles bajos que están rodeadas de celdas programadas con niveles altos, lo que produce errores en bits.

En el análisis de un canal de memoria flash NAND MLC, los errores anteriores no son equiprobables para los valores de 0 y 1 por lo que se puede tratar el canal de memoria flash como un Canal Binario Asimétrico [15, 24]

Los proveedores de memorias flash NAND MLC garantizan un funcionamiento fiable sólo durante un número limitado de ciclos de programación/borrado y una cantidad limitada de tiempo de retención [25–27].

La vida útil de una memoria flash NAND MLC depende del número de ciclos de programación/borrado que se hayan realizado y de la velocidad a la que se realizan estos ciclos. La mayoría de las unidades flash están diseñadas para soportar entre 10 000 y 100 000 ciclos de programación/borrado [28, 29].

Las memorias flash, como parte de indispensable la tecnología SSD para alma-

cenar datos, desempeñan un papel fundamental en la confidencialidad, la integridad y disponibilidad de la información. Una retención confiable asegura que los datos confidenciales no sean comprometidos ni accesibles por partes no autorizadas. La implementación de técnicas de cifrado en las memorias flash contribuye a garantizar la confidencialidad de los datos almacenados.

Memorias flash más confiables garantizan que los datos almacenados mantengan su integridad a lo largo del tiempo, sin corrupciones o cambios no autorizados, por tanto, algoritmos de corrección de errores y las prácticas de gestión de desgaste contribuyen a mantener la integridad.

En entornos sujetos a normativas de privacidad, la retención confiable es esencial para cumplir con requisitos específicos de disponibilidad de datos y protección de la información. Las memorias flash, al ser dispositivos de almacenamiento no volátil, aseguran la disponibilidad de los datos incluso en ausencia de alimentación eléctrica.

Lograr mayor durabilidad en términos de programación/borrado sin errores es vital para garantizar la longevidad de estos dispositivos especialmente en aplicaciones y entornos que involucran cargas de trabajo intensivas y operaciones constantes de escritura. Para tecnologías como el Blockchain, la pérdida o corrupción de datos podría comprometer la cadena de bloques y la confianza en la integridad del registro distribuido. En entornos de Cloud Computing, donde los datos se almacenan y se acceden de manera distribuida, la retención confiable es crucial para garantizar la disponibilidad y el rendimiento. En arquitecturas como el Big Data, donde se manejan grandes volúmenes de datos de manera simultánea, la confiabilidad y la velocidad son esenciales.

En [30], por ejemplo, se presentan 4 técnicas que se complementan para mitigar varias de las limitaciones de estos dispositivos: LaVAR (Layer Variation Aware Reading), LI-RAID (Layer-Interleaved Redundant Array of Independent Disks), (ReMAR) y ReNAC (Retention Model Aware Reading) y Retention Interference Aware Neighbor-Cell Assisted Correction (ReNAC). La técnica LaVAR reduce el efecto de

la variación del proceso entre capas ajustando el voltaje de referencia de lectura por separado para cada capa. La técnica LI-RAID utiliza información sobre la variación del proceso entre capas para agrupar de manera inteligente las páginas bajo la técnica de recuperación de errores RAID, reduciendo así la probabilidad de que la recuperación de un grupo falle significativamente antes que la de otros grupos. La técnica ReMAR, reduce los errores de retención al rastrear el tiempo de retención de los datos usando un nuevo modelo y adaptando el voltaje de referencia de lectura a la antigüedad de los datos. La técnica ReNAC adapta el voltaje de referencia de lectura a la cantidad de interferencia de retención que ha experimentado una página, para releer los datos después de que una operación de lectura falle.

En [31], se propone una técnica para mitigar los efectos de la temperatura, en la cual se colocan los datos de forma que se aproveche la asimetría presente en cargas de trabajo para colocar datos de usuario que se leen con frecuencia en páginas de memoria flash de baja latencia. Aunque conceptualmente simple, esta técnica es difícil de integrar en un controlador de memoria flash, su implementación es muy compleja, requiere más metadatos y está limitada por otras peculiaridades específicas de la memoria flash. Para superar estos desafíos, se propone también una nueva arquitectura de controlador de memoria flash que admite la colocación de datos en función del calor de lectura.

En [32], se expone un esquema de reintento de lectura en paralelo para aprovechar las características de reintento de lectura entre las celdas de memoria flash. Al leer múltiples páginas de datos simultáneamente, esta técnica, reduce el tiempo de respuesta de lectura al paralelizar las operaciones, de esta forma se leen simultáneamente páginas de datos que presentan características de confiabilidad virtualmente equivalentes.

### 1.1.3 Mecanismos de corrección de errores en las memorias flash NAND MLC

Una memoria flash NAND MLC puede permitir el diseño de mecanismos de corrección de errores para combatir las degradaciones descritas anteriormente. Lograr una solución eficiente para la corrección de errores es un desafío pues debe cumplir con algunas limitaciones relacionadas con el rendimiento del sistema y el consumo de recursos, por lo cual, es importante la creación de herramientas eficientes de codificación, procesamiento de señales y teoría de la información para mejorar la durabilidad y confiabilidad de estos dispositivos [33].

En la literatura científica, existen códigos correctores de errores como el LDPC (Low Density Parity Check Code), utilizados en diversas aplicaciones. Esta codificación puede tolerar una tasa de error de bits sin procesar de $10^{-6}$ [34], valores que son positivos para una memoria flash NAND MLC, pero comprometen los diseños con redundancia, latencia y sobrecarga de área [2, 15]. Un decodificador LDPC de 4 KB puede lograr un rendimiento de 19,3 Gbps libres de errores en un ŕea de $0,120\ mm^2$ para satisfacer el requisito de rendimiento de la arquitectura ONFI [35].

Otro código de corrección de errores utilizado en la literatura científica, que presenta menor complejidad computacional en comparación con los códigos LDPC, son los códigos polares [36].

En un modelo de canal para memoria flash NAND MLC descrito por 4 distribuciones Normales, donde se comparan 2 algoritmos de decodificación polar y 2 algoritmos de decodificación LDPC, se obtiene menor complejidad para los algoritmos de códigos polares [36]. La figura 1.9 muestra el resultado de esta comparación.

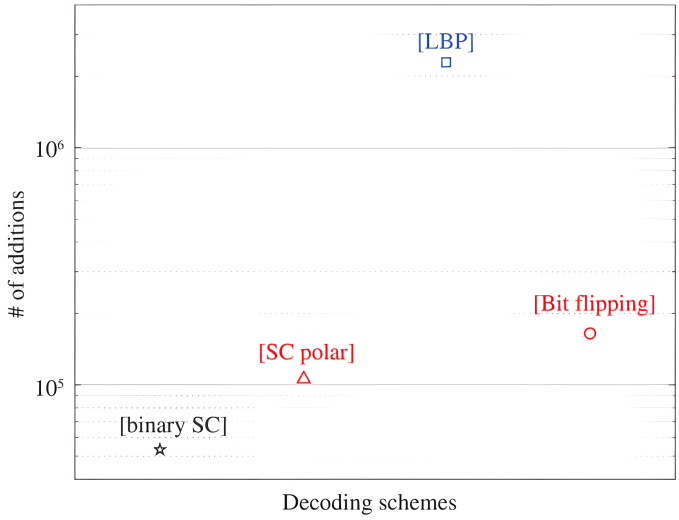



**Figura 1.9:** Comparación entre los algoritmos de decodificación PC Binary-SC, PC-SC, LDPC-LBP y LDPC Bit-flipping.

Para la transmisión de una modulación BPSK (Binary Phase Shift Keying), por un Canal de Ruido Blanco Gaussiano Aditivo (AWGN), se obtiene para los códigos polares, una Tasa de Error de Bits (BER) menor que para los códigos LDPC a partir de un 1dB de Señal a Ruido [34]. La figura 1.10 muestra el resultado de esta comparación.

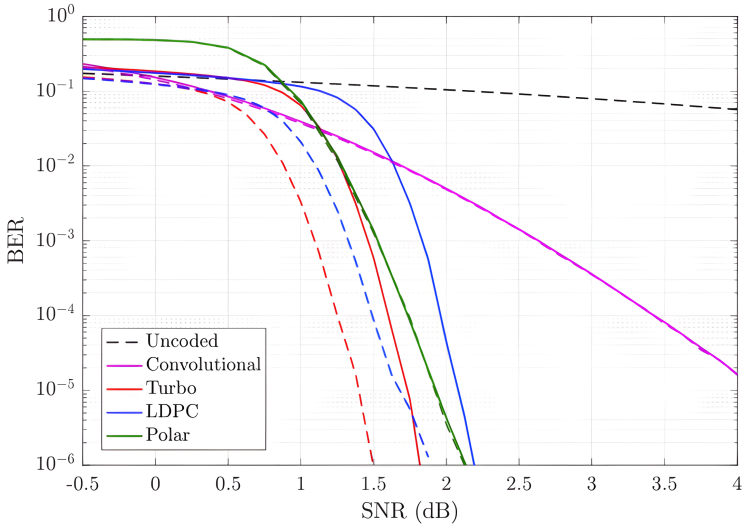



**Figura 1.10:** BER v/s. SNR para Polar Code (en verde), LDPC, Códigos Convoluciones (CC) y Turbo Códigos.

En un canal inalámbrico de alto rendimiento (Wireless High Performance) don-

de se compara la fiabilidad de Tasa de Error de Paquetes (PER) en un corto intervalo de tiempo, se refleja que se obtienen mejores resultados aplicando el algoritmo de decodificación de lista de cancelación sucesiva de códigos polares [37]. La figura 1.11 muestra el resultado de esta comparación.

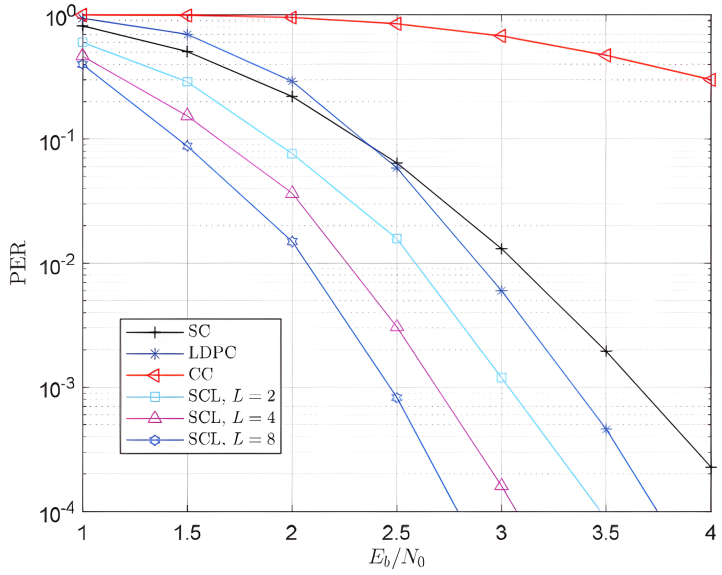



**Figura 1.11:** PER v/s. Señal a Ruido para decodificadores PC-SC, PC-SCL,CC y LPDC.

En esquemas de codificación de un canal asignado a sistemas 5G, los códigos polares logran un rendimiento deficiente en la Tasa de Error en Bloque (BLER), en comparación con los códigos LDPC y los Turbo Códigos, sin embargo, logran un alto rendimiento en URLLC (Ultra-High Reliability and Low Latency) y mMTC (Massive Machine Type Communications) [38]. La figura 1.12 muestra el resultado de esta comparación.

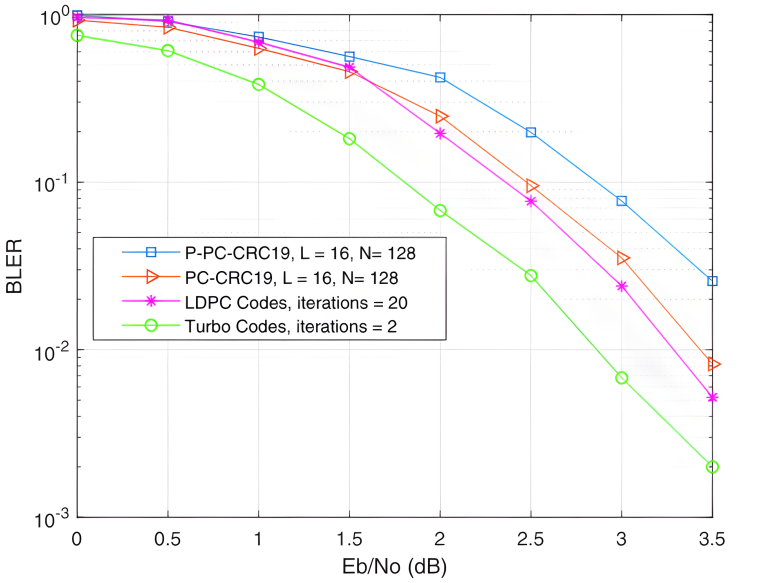



**Figura 1.12:** BLER v/s Señal a Ruido para Turbo Códigos, LDPC y PC.

Si se analiza la Señal a Ruido Pico (PSNR) sobre un canal inalámbrico con desvanecimiento de Rayleigh, durante la transmisión de una imagen, los códigos polares tienen un mejor rendimiento que los códigos LDPC, para valores mayores que 3.5 dB de Señal a Ruido. Con el aumento de la longitud del código, el rendimiento de los códigos polares mejora más rápidamente que el de los códigos LDPC en la transmisión de imágenes [39]. La figura 1.13 muestra el resultado de esta comparación.

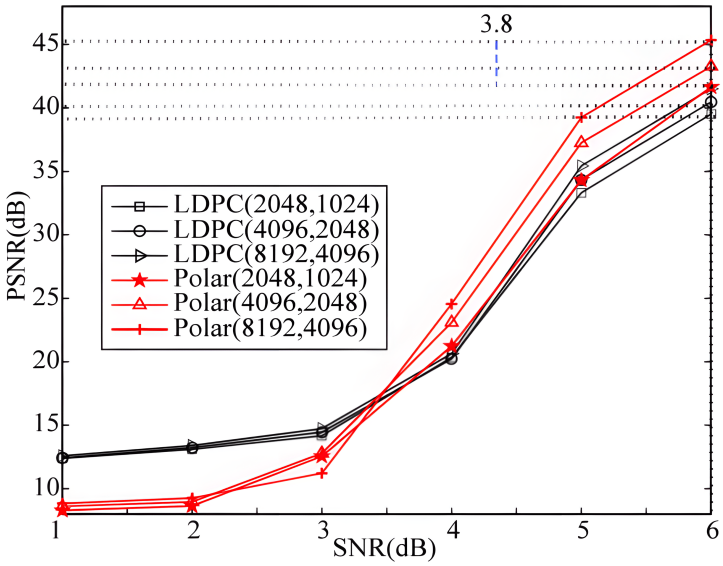



**Figura 1.13:** PSNR v/s Señal a Ruido para códigos LDPC y PC.

### 1.1.4 Modelo de canal para memoria flash NAND MLC

La corrección de errores (ECC) se puede utilizar para mejorar la confiabilidad de las memorias flash NAND MLC, sin embargo, para obtener mejores resultados, es esencial estudiar y comprender el canal de transmisión de estos dispositivos en términos de mecanismos de error [15].

Un modelo de canal para una memoria flash puede verse como una representación simplificada de su estructura física unida a los mecanismos que inducen los errores en los datos almacenados. Existen en la literatura científica, diferentes enfoques para describir un modelo de canal de memoria flash NAND MLC. Algunos autores enfocan sus investigaciones en la obtención del modelo de canal a través de la descripción de parámetros como la tasa de error de bits, la temperatura, la señal a ruido o la cantidad de ciclos de programación/borrado. Otros autores enfocan la descripción del modelo en parámetros como voltaje umbral o datos estadísticos que permiten realizar esta descripción utilizando distribuciones de probabilidad [2, 15].

Este trabajo de tesis propone la utilización del modelo de Canal Binario Asimétrico (BAC). En [15, 24] se expone al modelo de canal BAC como viable para la re-

presentación de los errores de bits que ocurren en la memoria flash NAND MLC, basándose en la asimetría que éstos presentan. Para la obtención de mejores aproximaciones con este modelo en [15], se propone utilizar la distribución de probabilidad Beta_Binomial (BBM). Esta investigación, además, permite la obtención de resultados en función de los ciclos de programación/borrado. En adelante se refiere a esta combinación como modelo de canal BAC_BBM.

### 1.1.5 Seguridad en las de memorias flash NAND MLC

Varias publicaciones científicas existentes, coinciden en que los códigos correctores pueden detectar y corregir eficientemente errores aleatorios, pero no son suficientes para proteger la información almacenada en las memorias flash NAND MLC.

En [40], se proponen los códigos Reed-Solomon (RS) para la corrección de errores y los códigos de Detección de Manipulación Algebraica (AMD), para la protección contra ataques maliciosos de inyección de fallos, que puede corregir cualquier error de 4bytes y detectar cualquier error inyectado maliciosamente con alta probabilidad, bajo un modelo de ataque fuerte.

Otra publicación describe la implementación de la seguridad en la memoria flash NAND MLC desde el hardware, aplicando detección de falsificaciones en los circuitos integrados y funciones no clonables para circuitos físicos [41].

Este trabajo de tesis, propone un sistema de cifrado que utilice mapas caóticos de orden faccionario para la protección en las memorias flash NAND MLC. Las secuencias caóticas son altamente impredecibles, incluso si se conocen las condiciones iniciales, lo que dificulta que un atacante pueda predecir o reproducir la secuencia sin conocer todos los parámetros exactos. Los generadores de secuencias caóticas pueden ser implementados de manera eficiente en hardware y software, permitiendo su uso en una amplia gama de aplicaciones de seguridad, desde sistemas embebidos hasta redes de comunicaciones.

## 1.2 Antecedentes

La búsqueda de soluciones para mejorar la tecnología de las memorias flash es un tema abordado en la literatura, aunque las investigaciones sobre la utilización de los códigos polares para la corrección de errores en el canal de memoria flash NAND MLC, no abordan la utilización del modelo de canal BAC_BBM para la caracterización del canal, ni la aplicación de esquemas caóticos fraccionarios para su protección.

En [42], se diseña un codificador y un decodificador que admite simultáneamente múltiples velocidades en un FPGA (Field Programmable Gate Array) para códigos polares que utiliza un Canal Binario Simétrico (BSC) y un Canal de Ruido Blanco Gaussiano Aditivo (AWGN), como modelos de canal para memoria flash NAND SLC. Esta investigación realiza una comparación con los códigos LDPC y los códigos polares. Se obtienen mejores aproximaciones para los códigos polares.

En [43], se estudia la aplicación de códigos polares para SSD basados en memorias flash NAND. Se detallan aspectos de codificación conjunta de canal y fuente de la codificación polar en un BSC, para la protección de datos de la memoria. Se demuestra que la información de los metadatos puede ser utilizada para mejorar la capacidad de decodificación de los códigos polares. Dicha propiedad no es funcional en los sistemas basados en códigos LDPC.

En [15], se hace una caracterización detallada de los errores que ocurren en las memorias flash NAND MLC y se desarrollan modelos de canal que reflejan mejor las características del error para la de corrección de errores (ECC). Este modelo de canal se fortalece utilizando la distribución de probabilidad Beta_Binomial (BBM) para obtener mejores resultados. En [15], se utiliza la caracterización de errores de interferencia entre celdas para evaluar el rendimiento de esquemas de codificación que mitigan este tipo de interferencia y mejoran la confiabilidad del flash NAND MLC.

En [2], se diseña un controlador de memoria flash NAND MLC inteligente basado en algoritmos para mejorar la confiabilidad de la memoria flash a bajo costo. Se propone una nueva técnica llamada WARM para mejorar la vida útil de la memoria flash. La idea clave es identificar y explotar la intensidad de escritura de la carga de trabajo en el controlador para mejorar la confiabilidad. Se propone un modelo de canal que predice la distribución de voltaje umbral, se realiza una caracterización y análisis de los errores de memoria flash 3D NAND. Esta investigación propone la distribución de probabilidad t_Student para la descripción del canal de memoria flash.

En [44], se propone un esquema de cifrado caótico y la sincronización para un sistema de O-OFDM (Orthogonal Frequency-Division Multiplexing) basado en códigos polares. En el esquema propuesto, las secuencias caóticas son generadas por un sistema de Lorenz tridimensional y se utilizan para generar la estructura que puede mejorar la seguridad de la capa física (PL) del sistema. Las secuencias caóticas también se utilizan para completar la sincronización de tiempo de símbolo de OFDM, cifrando la secuencia de entrenamiento. En comparación con la señal OFDM tradicional, puede alcanzar una ganancia de 8.2 dB con un BER después de la transmisión de 60 km en SSMF (Standar Single Mode Fiber).

En [45], se investiga un algoritmo de cifrado de capa física basado en códigos polares y secuencias caóticas para mejorar la confidencialidad de la transmisión para la tecnología 5G. Las secuencias caóticas se asignan a los bits congelados de los códigos polares, por lo tanto, la corrección de errores y el cifrado se pueden realizar simultáneamente. Dado que los $frozen\ bits$ son desconocidos para los interceptores, les resulta difícil decodificar los bits transmitidos sin conocer las secuencias caóticas. Para mejorar la seguridad, se aplican efectos encadenados de retroalimentación retardada para aumentar la complejidad de descifrado de los interceptores. El análisis teórico y los resultados de simulación de esta investigación muestran que los algoritmos propuestos pueden lograr una alta seguridad sin ninguna pérdida en el rendimiento de la tasa de error.

## 1.3 Planteamiento del problema

El problema de esta investigación es encontrar una solución implementable para mejorar la durabilidad de las memorias flash NAND MLC, utilizando códigos polares y esquemas caóticos de orden fraccionario.

## 1.4 Hipótesis

Los códigos polares aplicados a un modelo de canal BAC_BBM, unido a un sistema de cifrado que utilice esquemas caóticos de orden fraccionario puede ser implementado en la arquitectura ONFI.

## 1.5 Objetivos

### 1.5.1 Objetivo general

- Desarrollar un algoritmo que implemente códigos polares para la corrección de errores en el modelo de canal BAC_ BBM y un sistema de cifrado que incorpore esquemas caóticos fraccionarios, para la protección de la información del canal de memoria flash.

### 1.5.2 Objetivos particulares

- Estimar la probabilidad de error en función de los ciclos de programación/borrado del modelo de canal BAC_BBM.
- Calcular la capacidad del canal BAC_BBM utilizando imágenes de 512 X 512 como fuente de información.
- Estimar la relación entre la probabilidad de error y la relación señal a ruido en el sistema, considerando 10000 ciclos de programación/borrado.

- Comparar los códigos polares frente a los códigos LDPC en términos de probabilidad de error entre 2000 y 10000 ciclos de programación/borrado para el canal BAC_BBM.
- Evaluar la implementación del sistema en una arquitectura ONFI, centrándose en la relación de complejidad entre códigos polares y códigos LDPC.

## 1.6 Metodología

1. Revisar del estado del arte de los modelos de canal de memoria flash NAND MLC, códigos polares y sistemas de cifrado que utilicen esquemas caóticos de orden fraccionario.
2. Selección de un método de codificación para el modelo matemático de canal BAC_BBM.
3. Desarrollar la solución numérica que permita obtener un algoritmo que implemente códigos polares para la corrección de errores en el modelo de canal BAC_BBM y un sistema de cifrado que incorpore esquemas caóticos fraccionarios para la seguridad de la información.
4. Estimación de la probabilidad de error en función de los ciclos de programación/borrado, la capacidad de canal y la relación señal a ruido del esquema resultante.
5. Evaluación de los resultados obtenidos.
6. Conclusiones, contribuciones y trabajos a futuro.

## 1.7 Estructura del documento

Este trabajo se estructura como sigue:

En el Capítulo 2 se realiza una revisión del estado del arte de los modelos matemáticos de memoria flash NAND MLC. Se revisan las distribuciones de probabilidad t_Student, Beta, Binomial y Beta_Binomial, pues son las distribuciones con mejores resultados en la literatura científica para la descripciíon del modelo matemático del canal de transmisión de las memorias flash NAND MLC. Se describe el modelo canal BAC_BBM, así como los algoritmos que componen este modelo matemático. Se describen también los códigos correctores de errores más utilizados en la literatura científica y su relación de complejidad con los códigos polares.

El Capítulo 3 abarca los esquemas caóticos de orden fraccionario, el sistema de cifrado scrambling mejorado y los códigos polares. Se detallan las operaciones empleadas, en los procesos de cifrado de forma general y de manera específica para la implementación de esta tesis, así como el funcionamiento de cada uno de los subsistemas.

En el Capítulo 4 se muestran los resultados obtenidos. Este capítulo comprende la evaluación algoritmo propuesto. Se manejan diferentes escenarios para la evaluación de la probabilidad de error, la señal a ruido, la capacidad de canal y la comparación con los códigos LDPC.

En el Capítulo 5 se describen las conclusiones, contribuciones y los trabajos a futuro, siendo la implementación de códigos polares y esquemas caóticos de orden fraccionario una solución viable para mejorar la durabilidad de las memorias flash NAND MLC.

Capítulo 2

# La memoria flash como canal de transmisión

En este capítulo se revisan los modelos matemáticos que describen el canal de memoria flash NAND MLC y los códigos correctores de errores más referidos en la literatura científica.

En el Capítulo 1 se hace referencia al término modelo de canal de memoria flash. Pero, ¿Qué representa realmente esta definición para esta investigación?. Tradicionalmente, un canal de comunicación se define como el medio físico a través del cual se transmite la señal desde el emisor hasta el receptor. Este medio puede variar dependiendo del tipo de transmisión. En el caso de transmisiones inalámbricas, el canal se refiere al espacio libre. En transmisiones telefónicas, puede tratarse de líneas de cobre, fibra óptica u otros medios. En todos los casos, la señal enviada a través del medio físico está sujeta a diversas interferencias y perturbaciones aleatorias, como el ruido térmico aditivo, interferencias atmosféricas, descargas eléctricas durante tormentas, entre otros factores [46].

Un modelo de canal para una memoria flash NAND MLC puede entenderse como una simplificación de los mecanismos o factores que causan errores en los datos almacenados. Estas perturbaciones afectan la calidad e integridad de la información, por lo que es importante conocer su comportamiento con el fin de establecer métodos para mitigarlos.

El modelo de canal de memoria flash, representa la estructura física del *dado* de memoria flash y los errores que pueden surgir en este, debido a factores como la interferencia durante la programación, el desgaste de las celdas a lo largo de múltiples ciclos de programación/borrado, la pérdida de carga durante la retención de datos y la interferencia entre celdas [42-44].

La figura 2.1 muestra una vista de los componentes principales de un SSD. Como se explica en el Capítulo 1, estos dispositivos están compuestos por 2 partes fundamentales: un controlador y varios bancos de memoria flash NAND. El controlador opera la interfaz para el intercambio de datos y señalización con el host, los procesos de compresión, corrección de errores, y gestiona los bancos de memorias flash NAND. Este componente tiene un buffer de gestión con DRAM (Dynamic Random Access Memory)[9], que ejecuta el $firmware$ que asigna los datos del host a páginas físicas de la memoria flash NAND, realiza la recolección de basura en las páginas de flash que han sido invalidadas, aplica nivelación de desgaste para distribuir uniformemente el impacto de las escrituras en el banco de memoria flash NAND, y gestiona los bloques defectuosos. Los SSD tienen varios bancos de memorias flash NAND por unidad. Cada banco aloja uno o más *dados* flash. Cada *dado* se encuentra conectado a uno o más canales de memoria física, los cuales no se comparten entre los bancos de memoria y operan de manera independiente respecto a otros *dados*. Los *dados* contienen entre 1 y 4 planos y cada plano hospeda cientos a miles de bloques flash [48].

[9]La memoria DRAM facilita el acceso rápido a datos de uso temporal [47]. En un SSD guarda las direcciones de memoria del host que se asignan a las direcciones físicas del SSD.

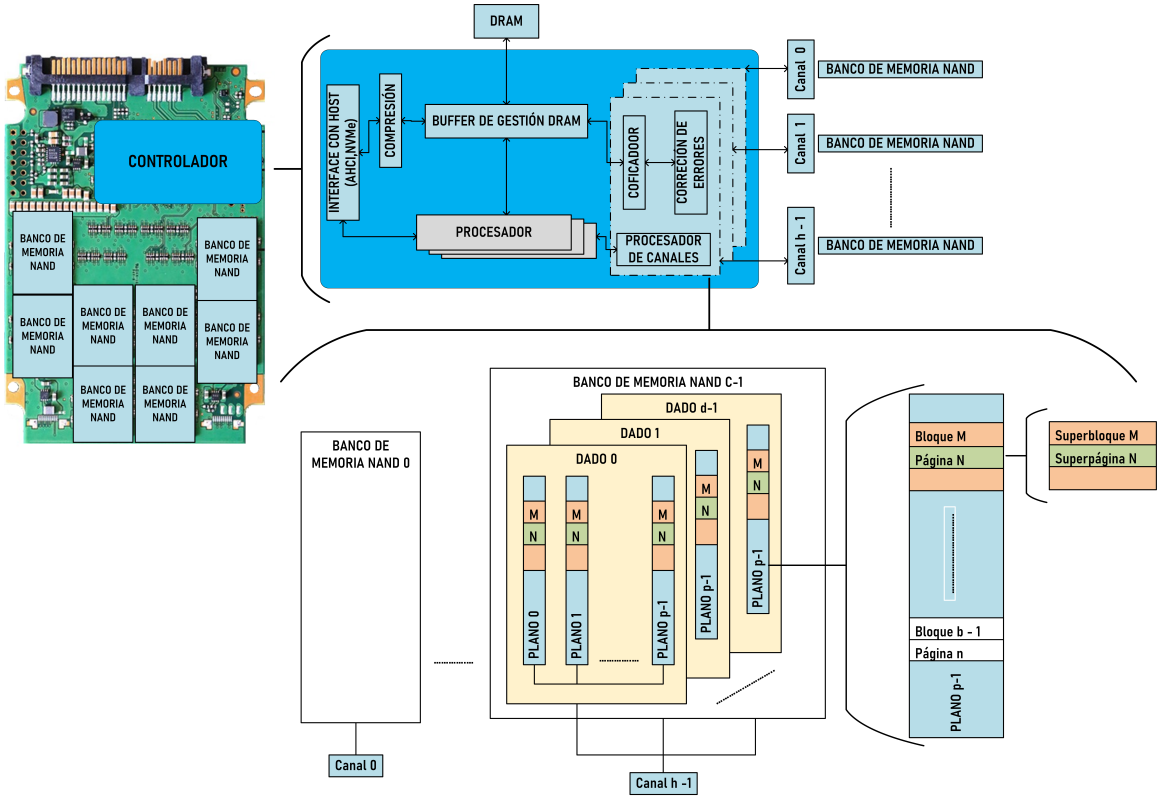



**Figura 2.1:** Vista detallada de la arquitectura de un SSD.

Un modelo de canal de memoria flash presenta utilidad para su ejecución, cuando es preciso en la descripción de los voltajes umbrales y en la caracterización de los errores de bits durante varios ciclos de programación/borrado. La información inexacta podría resultar en decisiones subóptimas y afectar las mejoras en la vida útil del dispositivo [2].

En diferentes investigaciones, se han propuesto modelos matemáticos para describir el comportamiento del canal de memoria flash NAND MLC. Estos incluyen el modelo de Canal Binario Asimétrico (BAC), el modelo de Canal Binario Simétrico (BSC), entre otros. Para aproximar correctamente el comportamiento del canal de las memorias flash NAND MLC a estos modelos, se han utilizado diversas distribuciones de probabilidad, como la distribución de probabilidad t_Student, la distribución de probabilidad Binomial y la distribución probabilidad Beta_Binomial.

Por otro lado, para abordar problemas de fiabilidad en las memorias flash

NAND MLC, se sugiere el uso de códigos correctores de errores (ECC). Entre los ECC más comunes implementados en controladores de memorias flash se encuentran el código Bose - Chaudhuri - Hocquenghem (BCH) y el LDPC. Existen aplicaciones que utilizan estos códigos de corrección de errores para los canales de memorias flash NAND MLC, que pueden ser descritos por las distribuciones de probabilidad mencionadas previamente, con el fin de garantizar su efectividad durante la ejecución.

## 2.1 Distribución de probabilidad t_Student

La distribución t_Student, es utilizada para estimar dinámicamente el cambio de distribución de voltaje umbral como función de los ciclos de programación/borrado. Este modelo permite al controlador flash adaptar su rendimiento a los cambios en el voltaje de umbral y así mejorar la confiabilidad. Estos parámetros son válidos también para predecir la vida útil restante del dispositivo flash basados en el crecimiento esperado en la tasa de error de bits, la creación de procedimientos de autorrecuperación, así como la mitigación de errores derivados de cambios de temperatura [2].

Esta distribución fue desarrollada por William Sealy Gosset, estadístico inglés en 1908 [49]. La distribucińt_Student es un modelo teórico utilizado para realizar inferencias sobre la media de una población normalmente distribuida cuando el tamaño de la muestra es pequeño y la desviación típica poblacional [10] es desconocida. Se fundamenta en la relación entre la media de una muestra y la media poblacional ajustada por el tamaño de la muestra y la varianza poblacional [11] desconocida.

En [2], se modifica la distribución t_Student y se aplica al análisis del voltaje umbral y la predicción de cambios futuros en el canal de memoria flash NAND MLC a medida que el dispositivo experimenta desgaste. Se encontró que la distribución

[10] La desviación típica poblacional es la medida de cuánta variación hay entre los datos individuales en una población. Se representa como la raíz cuadrada de la varianza y se representa como $\sigma$.

[11] La varianza poblacional es la media aritmética del cuadrado de las desviaciones. Se representa como $\sigma^2$.

t_Student presenta un error de modelado promedio del 0.68 % para este experimento.

Una variable aleatoria $t$ tiene una distribución t_Student de $n-1$ grados de libertad descrita por [50]:

$$t = \frac{\bar{x} - \mu}{\left(\frac{S}{\sqrt{n}}\right)}, \tag{2.1}$$

donde:

$t$ : es el valor de la distribución t de Student.

$\bar{x}$ : es la media de la muestra.

$\mu$ : es la media de la población.

$S$ : es la desviación estándar de la población.

$n$ : es el tamaño de la muestra.

Esta distribución introduce el término *grados de libertad*, que se determina como el número de observaciones que son libres de variar, dadas ciertas restricciones matemáticas, en una serie de valores utilizados para estimar alguna característica de la población. En el contexto de la distribución t_Student, los grados de libertad se definen como el número de observaciones menos el número de parámetros estimados [51]

$$v \ = \ n \ - 1, \ \ \forall \ n \ > 1. \tag{2.2}$$

La medida de los grados de libertad se obtienen en esta distribución mediante el cálculo de la varianza muestral $S^2$ [51]:

$$S^2 = \frac{1}{v} \sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2, \tag{2.3}$$

donde:

$x_i$ : son los valores individuales de la muestra.

$\bar{x}$ : es la media de la muestra.

$n$ : es el tamaño de la muestra.

La función de distribución acumulada t_Student no tiene una forma analítica

simple y se expresa comúnmente en téminos de la función gamma como [52]:

$$f\left(t|\nu\right)=\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}},\quad -\infty<t<\infty, \tag{2.4}$$

donde:

$t$ : es la variable aleatoria.

$v$ : son grados de libertad.

$\Gamma$ : es la función Gamma, definida como: $\Gamma\left(x\right)=\int_0^\infty t^{x-1}e^{-t}dt$.

La varianza de la distribución t_Student $\sigma^2$, para $v>2$ , se define como [49]:

$$\sigma^2=VAR\left[T\right]=E\left[\left(T-\mu\right)^2\right]=\frac{v}{v-2}. \tag{2.5}$$

La esperanza de esta distribución es $E\left[T\right]=\mu=0$, para todos los grados de libertad.

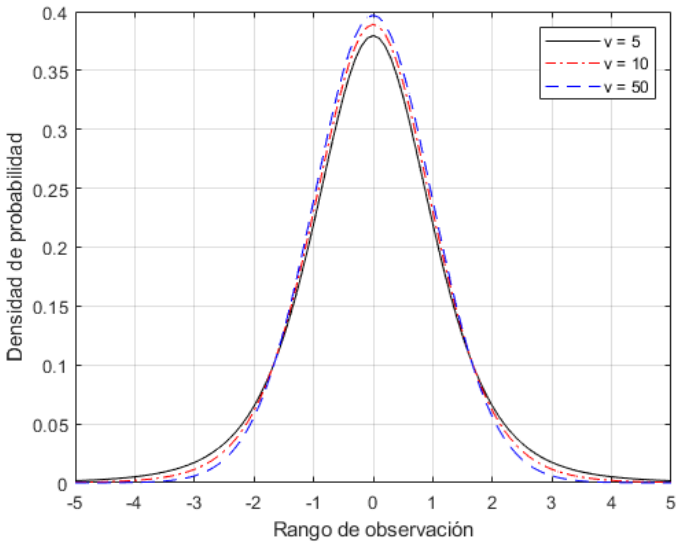



**Figura 2.2:** Función de densidad de probabilidad de la distribución t_Student para 5,10 y 50 con grados de libertad.

En comparación con la distribución Normal, la distribución t_Student introduce el parámetro adicional $v$. Cuando $v\ \rightarrow\ +\infty$ , la distribución t_Student se convierte en una distribución Normal. Cuando $v\ \rightarrow\ 0$, se convierte en una distri-

bución Normal con mayor curtosis. Para describir esta transición matemáticamente, asignamos valores diferentes de $v$, que denotamos como $\beta$, para la transición de la curva del lado izquierdo de la campana, y $\alpha$ para para la transición de la curva del lado derecho de la campana, lo que se ajusta mejor a la distribución del voltaje umbral de un canal de memoria flash NAND MLC.

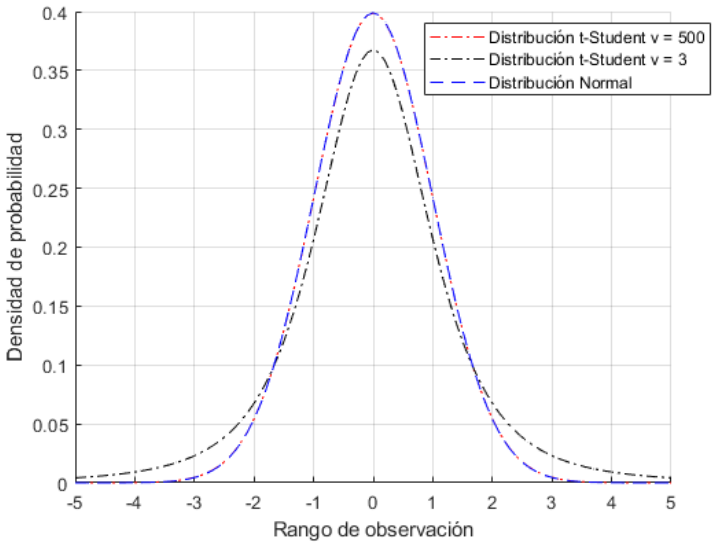



**Figura 2.3:** Función de densidad de probabilidad t_Student para v = 500, v=3 y función de densidad de probabilidad Normal.

Para reajustar de la distribución t_Student y se acerque mejor a los datos obtenidos al modelo de canal de memorias flash NAND MLC, se utiliza el parámetro puntuación estándar Z, que indica a cuántas desviaciones estńdar por encima o por debajo de la media se ubica un valor, siendo:

$$Z = \frac{v - \mu}{\sigma}. \tag{2.6}$$

De esta forma es posible utilizar la tabla de valores Z, para encontrar la probabilidad acumulada asociada con un valor específico.

Para simplificar el cálculo de la función de distribución acumulada para la distribución t_Student modificada (TCDF, por sus siglas en inglés), se utilizan los valores de $t$ precalculados en tablas para esta distribución. Cada tabla de $t$ contiene

valores de la función acumulada TCDF sobre un rango de valores de Z, para un único grado de libertad $v^3$. A continuación, se muestra el cálculo de la TCDF utilizando la tabla de la distribución $t$ precalculada [2]:

$$TDCF\left(v,\mu,\sigma,\alpha,\beta\right)=\begin{cases} t-table_{\beta}\left(Z\right) & :\upsilon\leq\mu \\ t-table_{\alpha}\left(Z\right) & :\upsilon>\mu \end{cases} \tag{2.7}$$

Para utilizar la ecuación anterior, se comparan los $v$ y $\mu$ para determinar en qué lado de la campana se encuentra $v$ . En dependencia del valor de $v$, se sustituye por $\beta$ o por $\alpha$ , para seleccionar la tabla correcta de los valores de $t$. Luego se calcula el valor de Z para buscar la función de distribución acumulada en la tabla seleccionada.

El modelo basado en la distribución t_Student estima la densidad para las celdas que deberían estar en el estado $X$, pero están incorrectamente programadas en el estado $Y$ en el canal memoria flash, cuyo voltaje umbral cae entre dos voltajes de referencia vecinos, y se denota como:

$$\begin{aligned} T_k\left(X\right)&=\left(1-\lambda_x\right)TDCF\left(v_k,\mu_X,\sigma_X,\alpha_X,\beta_X\right)+\lambda_x TDCF\left(v_k,\mu_Y,\sigma_Y,\alpha_Y,\beta_Y\right) \\ &-\left(1-\lambda_x\right)TDCF\left(v_{k-1},\mu_X,\sigma_X,\alpha_X,\beta_X\right)-\lambda_x TDCF\left(v_{k-1},\mu_Y,\sigma_Y,\alpha_Y,\beta_Y\right), \end{aligned} \tag{2.8}$$

donde $\lambda$ es el parámetro utilizado para modelar la probabilidad de que ocurran errores de programación para las celdas programadas. Los valores de $k$ y $k-1$ son utilizados para denotar dos voltajes de referencia de lectura vecinos.

El modelo matemático de canal de memoria flash NAND MLC descrito en [2] no aborda los errores de retención ni los disturbios de lectura, ya que estos son mitigados exitosamente por mecanismos de actualización de la flash. Sin embargo, el modelo puede combinarse con dichos mecanismos para proporcionar una reducción en los errores de bits sin procesar. Este modelo de canal captura exclusivamente los errores para múltiples ciclos de programación/borrado, así como los errores de programación de dos pasos, que son las fuentes de errores más comunes en la memoria

flash NAND planar MLC de 1X nm.

## 2.2 Distribución de probabilidad Binomial

La distribución Binomial es una distribución de probabilidad discreta utilizada para medir el número de éxitos en un número fijo de ensayos independientes, cada uno de ellos con una probabilidad de éxito constante. Fue desarrollada por Jacob Bernoulli y es fundamental en la teoría de la probabilidad, especialmente para variables que solo pueden tomar dos valores. La función de densidad de probabilidad de la distribución Binomial está defina por [53, 54]:

$$P\left(X = x\right) = \binom{n}{k} * \ p^k \left(1-p\right)^{n-k}, \tag{2.9}$$

donde:

$n$ : representa el número total de ensayos o eventos realizados.

$p$ : es la probabilidad de que ocurra un evento exitoso en un solo ensayo.

$1 - p$ : es la probabilidad de fracaso en un solo ensayo.

$k$ : cantidad de éxitos en $n$ ensayos.

Si una variable aleatoria discreta $X$ tiene distribución de probabilida binomial con parámetros $n \in \mathbb{N}$, y $p$ entre $0 < p < 1$, entonces se puede escribir como $X \sim Binomial(n, p)$.

La varianza de la distribución Binomial se define como:

$$\sigma^2 = VAR\left[X\right] = n * p * \left(1-p\right). \tag{2.10}$$

La media de esta distribución se define como: $E\left[X\right] = \mu = n * p$.

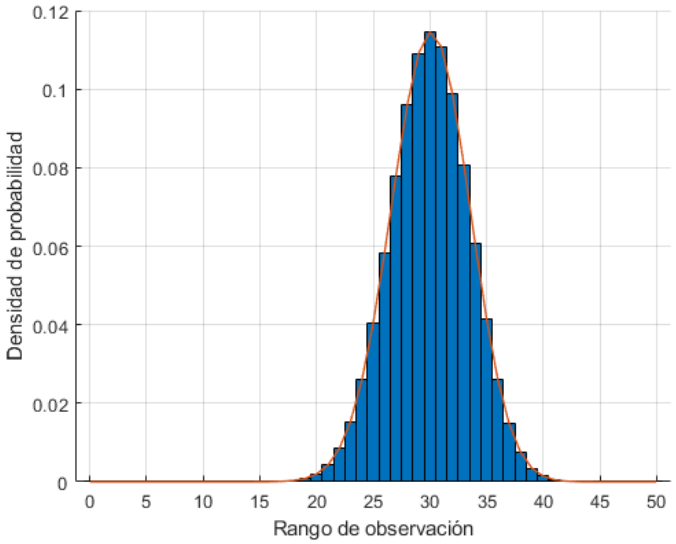



**Figura 2.4:** Distribución de probabilidad Binomial para 50 pruebas y probabilidad de éxito de 0.6.

## 2.3 Distribución de probabilidad Beta

Esta distribución tiene su origen en una investigación de Ballestero en 1973, relacionada con un método utilizado en la "Teoría General de Valoración", denominado por Ballestero y Caballero "Método de las dos distribuciones Beta". La distribución Beta es una distribución de probabilidad contínua, definida en el intervalo $[0, 1]$. Tiene 2 parámetros: $\alpha$ y $\beta$, que controlan la forma de la distribución. La función de densidad de probabilidad de la distribución Beta está dada por [25, 55]:

$$P(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, \tag{2.11}$$

donde:

$\alpha$ : representa la cantidad de eventos exitosos.

$\beta$ representa la cantidad de eventos fallidos.

$B(\alpha,\beta)$ : función beta es definida como $(\alpha,\beta) = \int_0^1 x^{(\alpha-1)}(1-x)^{\beta-1}dx$, para $0 \leq x \leq 1$

Si una variable aleatoria continua $X$ tiene distribución de probabilidad beta con parámetros $\alpha, \beta > 0$, entonces se puede escribir como $X \sim Beta\,(\alpha, \beta)$.

La varianza de la distribución Beta se define como:

$$\sigma^2 = VAR\,[X] = \frac{\alpha\beta}{(\alpha+\beta)^2\,(\alpha+\beta+1)}. \tag{2.12}$$

La media de esta distribución se define como:

$$E\,[X] = \mu = \frac{\alpha}{\alpha+\beta}. \tag{2.13}$$

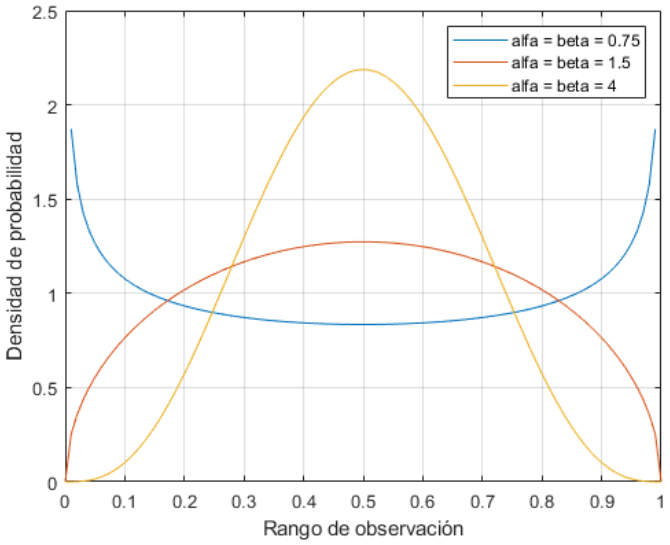



**Figura 2.5:** Distribución de probabilidad Beta para $\alpha$,$\beta$ =0.75, $\alpha$,$\beta$ =1.5 y $\alpha$,$\beta$ = 4.

## 2.4 Distribución de probabilidad Beta_Binomial

La distribución Beta_Binomial es una distribución de probabilidad que modela el número de éxitos en una secuencia de ensayos independientes, donde la probabilidad de éxito en cada ensayo sigue una distribución Beta. Combina las propiedades de la distribución Binomial y la distribución Beta. La función de masa de probabilidad de la distribución Beta_Binomial está dada por [25, 55–57].

$$P(x) = \binom{n}{k} \frac{B(k+\alpha, n-k+\beta)}{B(\alpha,\beta)}. \tag{2.14}$$

La varianza de la distribución Beta_Binomial se define como:

$$\sigma^2 = VAR[X] = \frac{n\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}. \tag{2.15}$$

La esperanza de esta distribución se define como:

$$E[X] = \mu = \frac{n\alpha}{\alpha+\beta}. \tag{2.16}$$

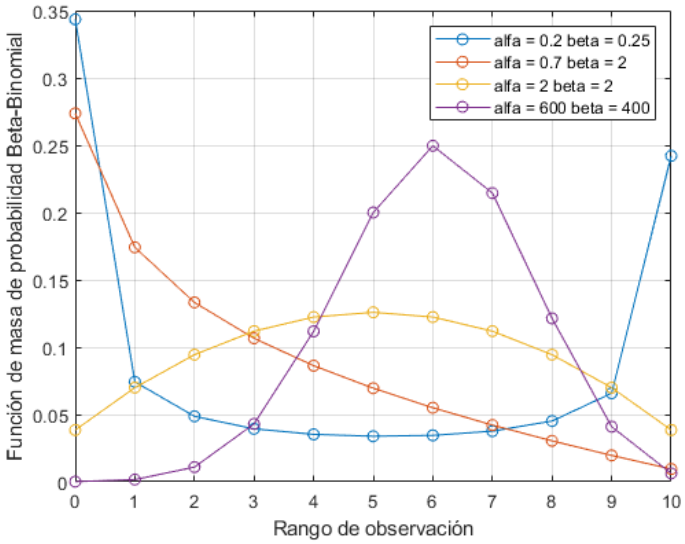



**Figura 2.6:** Función de masa de probabilidad Beta para $\alpha$ =0.2 $\beta$ =0.25, $\alpha$ =0.7 $\beta$ =2, $\alpha$ =2 $\beta$ =2,$\alpha$ =600 $\beta$ =400.

## 2.5 Modelos de canal para memorias flash NAND MLC

### 2.5.1 Modelo de Canal Binario Asimétrico (BAC) y modelo de Canal Beta_Binomial (BBM)

En un BAC, la probabilidad de error al transmitir un bit de 0 a 1 es diferente de la probabilidad de que ocurra un error al transmitir un bit de 1 a 0. Se puede representar mediante un modelo simple, como una transmisiń binaria con dos probabilidades de error diferentes [58, 59],

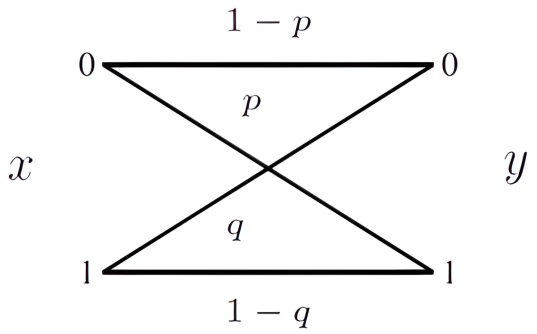



**Figura 2.7:** Representación del Canal Binario Asimétrico.

, donde:

$p$ : es la probabilidad de error de 0 a 1.

$q$ : es la probabilidad de error de 1 a 0.

En [15], se demuestra que la relación entre las distribuciones de probabilidad del número total de errores en una trama de $N$ bits y el número total de errores de bits en una trama que tiene ceros $m$ ceros y $N - m$ unos está dada por:

$$P_r\left(K = k\right) = \sum_{m=0}^{N} \frac{\binom{N}{m}}{2^N} P_r\left(K_m = k\right), \tag{2.17}$$

donde:

$K$ : es el número total de errores de bits la trama de longitud $N$.

$K\_m$ : es el número total de errores de bits en la trama de $N$ bits, donde $m$ representa los ceros y $N - m$ representa los unos.

$\frac{\binom{N}{m}}{2^N}$: representa la probabilidad de observar exactamente $m$ ceros en una trama de $N$ bits.

$K\_m$: puede ser representado como la suma del número de errores de bits cuando $0 \to 1$ y $1 \to 0$, siendo:

$$K_m = K_m^{(0)} + K_{N-m}^{(1)} \tag{2.18}$$

Tiendo en cuenta que para un Canal Binario Asimétrico $K_m^{(0)}$ y $K_{N-m}^{(1)}$ están distribuidos según la distribución binomial de probabilidad y son independientes se obtiene:

$$\begin{gathered} K_m^{(0)} \sim Binomial\,(m, p) \\ K_{N-m}^{(1)} \sim Binomial\,(N\ -\ m, p) \\ K_m^0 \perp\!\!\!\perp K_{N-m}^1, \end{gathered} \tag{2.19}$$

por tanto, la varianza de $K_m^{(0)}$ esta dada definida por:

$$\sigma^2 = VAR\left[K_m^{(0)}\right] = mp\,(1-p)\,, \tag{2.20}$$

y la media:

$$E\left[K_m^{(0)}\right] = \mu = \ mp. \tag{2.21}$$

Se define entonces la varianza de esta distribución como:

$$VAR\left[K\right] = \frac{N}{2}\left(\left(p+q\right) - qp - \frac{1}{2}\left(p^2+q^2\right)\right). \tag{2.22}$$

La media de esta distribución se reduce a:

$$E\left[K\right] = \frac{N}{2}\left(p+q\right). \tag{2.23}$$

Los parámetros $p$ y $q$ para el Canal Binario Asimétrico, se estiman como el promedio de las tasas de error de bits de $0 \rightarrow 1$ y $1 \rightarrow 0$ obtenidos a partir de datos experimentales correspondientes para ciertos ciclos de programación/borrado en la vida útil de la memoria flash. Para 8000 ciclos de programació/borrado, $N = 8192$,$p = 4.97 \cdot 10^{-3}$, y $q = 2.84 \cdot 10^{-3}$, se obtiene: $E\left[K\right] = 31.98$ y $VAR\left[K\right] = 31.86$. Al comparar la media y la varianza muestral de $K$ con datos experimentales registrados, se observó que la distribución Binomial no coincide con los valores del modelo de Canal Binario Asimétrico, por lo que este modelo no es una buena opción para la distribución de probabilidad empírica observada. Teniendo en cuenta la sobredispersión de los resultados anteriores al utilizar la distribución Binomial, se propone un modelo de canal para memorias flash MLC basado en la distribución de probabilidad Beta_Binomial [15].

La distribución de probabilidad Beta_Binomial es propuesta en varias investigaciones como la distribución de probabilidad para conteos resultantes de una distribución binomial si la probabilidad de éxito varía de acuerdo con la distribución Beta entre conjuntos de ensayos. Se ha demostrado también que esta distribución es adecuada para datos binomiales sobredispersos [60, 61].

Para el modelo de Canal Beta_Binomial se modelan las variables $K_m^{(0)}$ y $K_{N-m}^{(1)}$, y se estiman mediante la distribución Beta_Binomial.

$$
\begin{gathered}
p \sim Beta\,(a,b) \\
K_m^{(0)} \mid p \sim Binomial\,(m,p) \\
K_m^{(0)} \sim Beta - Binomial\,(m,a,b) \\
q \sim Beta\,(c,d) \\
K_{N-m}^{(1)} \mid q \sim Binomial\,(N-m,q) \\
K_{N-m}^{(1)} \sim Beta - Binomial\,(N-m,c,d) \\
K_m^0 \perp\!\!\!\perp K_{N-m}^1,
\end{gathered}
\tag{2.24}
$$

donde $a, b, c, d$ corresponden a los parámetros de la distribución de probabilidad Beta.

El modelo de canal BBM se deriva de un modelo BAC donde las probabilidades de error de bit $p$ y $p$ son variables aleatorias que cambian de una trama a otra y se distribuyen segń la distribuciń Beta.

Las distribuciones de probabilidad para $K_m^{(0)}$ y $K_{N-m}^{(1)}$ serían:

$$
P\left(K_m^{(0)} = k\right) = \binom{m}{k} \frac{B\,(a+k, b+m-k)}{B\,(a,b)} \tag{2.25}
$$

$$
P\left(K_{N-m}^{(1)} = k\right) = \binom{N-m}{k} \frac{B\,(c+k, d+N-m-k)}{B\,(c,d)} \tag{2.26}
$$

La media y la varianza de estas estas distribuciones están dadas por:

$$
E\left[K_m^{(0)}\right] = \frac{ma}{a+b} \tag{2.27}
$$

$$
VAR\left[K_m^{(0)}\right] = \frac{mab\,(a+b+m)}{(a+b)^2\,(a+b+1)} \tag{2.28}
$$

$$
E\left[K_{N-m}^{(1)}\right] = \frac{(N-m)\,c}{c+d} \tag{2.29}
$$

$$VAR\left[K_{N-m}^{(1)}\right] = \frac{(N-m)\,cd\,(c+d+N-m)}{(c+d)^2\,(c+d+1)} \tag{2.30}$$

$$E\left[K\right] = \frac{N}{2}\left(\frac{a}{a+b} + \frac{c}{c+d}\right) \tag{2.31}$$

$$VAR\left[K\right] = \frac{N}{4}\left(\frac{a\,(a+b)\,(a+2b+1)+Nab}{(a+b)^2\,(a+b+1)}\right) + \frac{N}{4}\left(\frac{c\,(c+d)\,(c+2d+1)+Ncd}{(c+d)^2\,(c+d+1)}\right) - \frac{N}{4}\left(\frac{2ac}{(a+b)\,(c+d)}\right) \tag{2.32}$$

Los parámetros $a, b, c, d$ del modelo de canal BBM se utilizando el Método de Momentos $K^{(0)}$ y $K^{(1)}$, a partir de los datos experimentales de los ciclos de programación/borrado. Para la evaluación del modelo de canal BBM, se utilizan los mismos parámetros de valuación del modelo BAC. Para 8000 ciclos de programación/borrado, $N = 8192$, se obtiene: $E\left[K\right] = 32.01$ y $VAR\left[K\right] = \ 57.88$. En este caso la media coincide como los valores experimentales de referencia. La varianza obtenida sigue siendo menor que la varianza muestral; sin embargo, está claro que el modelo de canal BBM es considerablemente mejor para modelar los datos empíricos sobredispersos del número de errores de bit por trama que el modelo BAC. El modelo BBM proporciona el mejor ajuste a los datos empíricos entre todos los modelos de canal propuestos y estimaciones más precisas en el rendimiento cuando se le aplica al canal de memoria flash MLC corrección de errores.

### 2.5.2 Algoritmos del modelo canal BAC_BBM

A continuación, se muestran los algortirmos que definen el modelo de canal BAC_BBM y los valores de $a, b, c, d$ para los MSB y los LSB de unos de los chips de memorias flash NAND MLC que obtenidos en [60]:

**Algoritmo 1:** Implementación del modelo BAC

**Entrada:** Trama de entrada $x$ de longitud $N$, parámetros del modelo BAC $(p, q)$

**Salida:** Trama de datos con errores $y$

Para cada $x_i \in x$:

Generar una muestra aleatoria $u \sim Uniforme\,[0, 1]$

Si $x_i = 0$ entonces $t = p$ sino $t = q$

Si $u \leq t$ entonces $e_i = 1$ sino $e_i = 0$

$y_i = x_i \otimes \ e_i$

**Algoritmo 2:** Implementación del modelo de canal BBM

**Entrada:**Trama de entrada $x$ de longitud $N$, parámetros del modelo de canal BBM $(a, b, c, d)$

**Salida:**Trama de datos con errores $y$

$p \sim Beta\,(\ a, b\ )$ y $q \sim Beta\,(c, d)$

$y = BAC\,(x, p, q)\,[Algoritmo1]$

| Ciclos P/E | a | b | c | d |
|---|---|---|---|---|
| 2000 | 12.72 | 46368.34 | 8.05 | 42569.08 |
| 4000 | 25.95 | 20940.98 | 18.16 | 23556.92 |
| 6000 | 22.67 | 7596.71 | 22.28 | 11890.14 |
| 8000 | 20.72 | 4143.52 | 22.28 | 7821.13 |
| 10000 | 21.36 | 2819.03 | 26.12 | 5890.35 |

**Tabla 2.1:** Estimaciones de los parámetros $a, b, c, d$ para la MSB del modelo de canal BBM para un chip el proveedor A.

| Ciclos P/E | a | b | c | d |
|---|---|---|---|---|
| 2000 | 2.85 | 446831.46 | 15.31 | 24066.27 |
| 4000 | 3.57 | 315123.27 | 22.49 | 7551.62 |
| 6000 | 1.68 | 95672.63 | 18.90 | 3528.74 |
| 8000 | 2.01 | 86407.03 | 20.09 | 2682.08 |
| 10000 | 1.81 | 61326.54 | 23.79 | 2338.70 |

**Tabla 2.2:** Estimaciones de los parámetros $a, b, c, d$ para la MSB del modelo de canal BBM para un chip el proveedor B.

## 2.6 Códigos correctores de errores

### 2.6.1 Códigos BCH

Los códigos BCH son códigos correctores de errores cíclicos que se construyen utilizando polinomios sobre un campo finito, también llamado campo de Galois. Fueron creados en 1959 por el matemático francés Alexis Hocquenghem, e independientemente en 1960 por Raj Chandra Bose y D.K. Ray-Chaudhuri. El nombre surge de las iniciales de los apellidos de los creadores [62, 63]

Los códigos BCH binarios que pueden corregir múltiples errores de bits y pueden ser codificados fácilmente a través de un método algebraico conocido como decodificación de síndrome, lo que simplifica el diseño del decodificador, pudiéndose utilizar un hardware de baja potencia y tamaño reducido [63].

Un código BCH utiliza un polinomio generador para crear los bits de paridad. El polinomio es de la forma $x^n - 1$ , donde n es la longitud del código. El mensaje de entrada se representa como un polinomio de grado menor o igual que $k$, siendo $k$ la cantidad de bits de información. Para agregar bits de paridad al mensaje de entrada, se multiplica el mensaje por un polinomio de la forma $x^{k-n}$. Luego, se divide el resultado por el polinomio generador $(x)$ , utilizando la aritmética de campo finito. El mensaje codificado consiste en el mensaje original con los bits de información más los bits de paridad que se obtuvieron como resultado de la división polinómica [64].

La decodificación se realiza calculando el síndrome. El síndrome se calcula evaluando el mensaje recibido en las raíces del polinomio generador. Si el síndrome es cero, significa que no hay errores. Si es distinto de cero, indica la presencia de errores. Luego se buscan las raíces del síndrome en el campo finito. Si se encuentran raíces, se utilizan para identificar y corregir los errores [64].

Para codificar un mensaje de 4 bits utilizando un código BCH $(7, 4)$ sobre un campo finito de tamaño $2^3$, se utiliza un polinomio generador de grado 3, ya que el código tiene una longitud de 7 y 3 bits de paridad, en este caso el generador polinomial será $g(x) = x^3 + x^2 + 1$.

Si el mensaje de entrada es $m(x) = x^3 + x^2$. Para obtener el mensaje codificado, se multiplica $m(x) * x^3$, para hacer espacio para los bits de paridad, y se obtiene el residuo de la división polinómica por $g(x)$.

$$m(x) = \left(x^3 + x^2\right) * x^3 = x^6 + x^5 \tag{2.33}$$

El residuo de la división de $x^6 + x^5$ por $g(x) = x^3 + x^2 + 1$ es $x + 1$, que serían los bits de paridad. Luego se suman los bits de paridad al mensaje de entrada y se obtiene el mensaje codificado $c(x) = m(x) * x^3 = x^6 + x^5 + x + 1$.

Si recibimos $r(x) = x^6 + x^5 + x + 1$ , se calcula el síndrome $S(x)$, evaluado en $r(x)$ en las raíces del polinomio generador $g(x)$. Si el síndrome es cero no hay errores.

$$S(\alpha) = r(\alpha) \tag{2.34}$$

$$S(\alpha) = \alpha^6 + \alpha^5 + \alpha + 1 \tag{2.35}$$

Los códigos BCH se emplean para la corrección de errores en el canal de memoria flash NAND MLC, ya que son capaces de rectificar el valor del voltaje umbral. Sin embargo, esto limita la escalabilidad de dichos códigos, dado que la ventana de voltaje umbral utilizada para representar valores se reduce conforme se actualiza la

tecnología para la memoria flash NAND MLC. Debido a esta reducción, la capacidad de corrección de errores de los códigos BCH está disminuyendo para tolerar la tasa de errores de bits crudos de las celdas flash, la cual aumenta con los ciclos de programación/borrado. Por lo anterior, los diseñadores de memoria flash están examinando otros mecanismos de corrección de errores [65, 66]

### 2.6.2 Códigos LDPC

Los códigos LDPC son propuestos por R. G. Gallager en la década de 1960. Aunque inicialmente considerados demasiado complejos para aplicaciones prácticas, la representación mediante Grafos de Tanner por R. M. Tanner en la década de 1980 y el redescubrimiento de su potencial en la década de 1990 liderado por David MacKay, los posicionaron como herramientas eficaces para acercarse al límite de Shannon con alta confiabilidad y baja complejidad en los algoritmos [34, 64]

Actualmente, los códigos LDPC, son utilizados para la corrección de errores de los canales de datos de la 5G New Radio, en sustitución de los códigos convolucionales y los Turbo códigos usados por 4G LTE (Long - Term Evolution) [67].

Los códigos LDPC son códigos de bloques lineales con matrices de comprobación de paridad que contienen solo un número muy pequeño de entradas diferentes de cero. Una matriz de comprobación de paridad para un código LDPC puede ser la siguiente:

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{2.36}$$

La codificación de códigos LDPC se lleva a cabo mediante la matriz $H$, conocida como la matriz de verificación de paridad. En esta matriz, cada fila representa una ecuación de paridad, mientras que las columnas representan símbolos del bloque. Los códigos LDPC pueden ser eficazmente representados mediante los Grafos de Tanner.

En este diagrama se representan las filas y columnas de la matriz de verificación de paridad a través de los nodos de comprobación ($CN$) y nodos variables ($VN$). La figura 2.8, presenta el Grafo de Tanner de la matriz $H$, destacando las conexiones esenciales entre ($CN$) y ($VN$).

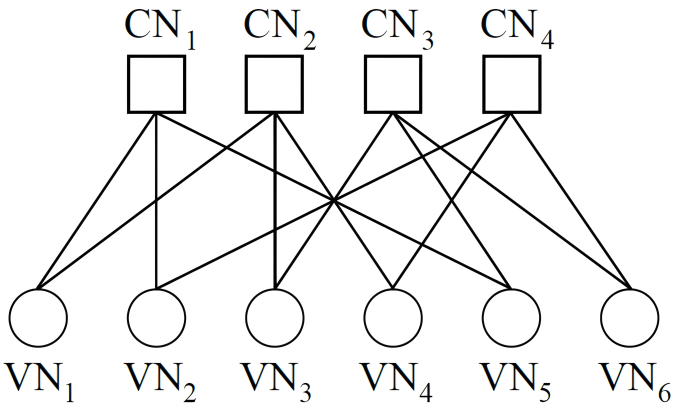



**Figura 2.8:** Grafo de Tranner de la matriz H.

La codificación puede ser escrita de la forma $c = uG$, donde $c$ representa la palabra de código de salida, $u$ denota el bloque de entrada, y $G$ es la matriz generadora. La matriz generadora se deriva a partir de la matriz de comprobación de paridad $H$ al colocar esta matriz en forma sistemática mediante la eliminación de Gauss_Jordan. La matriz $H$ se diseña para una longitud específica del bloque de entrada, y su transformación a una forma sistemática puede volverse compleja, especialmente para longitudes de bloque extensas [64].

La decodificación LDPC se realiza mediante el algoritmo Suma_Producto, que basa su funcionamiento en el envío de mensajes entre $CN_s$ y $VN_s$ en el Grafo de Tanner. Inicialmente los $VN_s$ envían las Razones de Vorisimilitud Logarítmica ($LLR$) $L_j$ a los $CN_s$, a continuación, los $CN_s$ realizan sus cálculos y los envían a los $VN_s$ conectados según [34]:

$$L_{i\to j} = 2tanh^{-1}\left[\prod_{j\prime\in N(i)-\{j\}} tanh\left(L_{j\prime\to\frac{i}{2}}\right)\right] \tag{2.37}$$

El símbolo $L_{i\to j}$ es el mensaje transmitido desde el $i-ésimo\ CN$ al $j-ésimo$ nodo de $VN$. $L_{j\to i}$, es el mensaje transmitido desde el $j-ésimo\ VN$ al $i-ésimo$ $CN$ y $N_i$ es el conjunto de $VN_s$ conectados al $i-ésimo\ CN$ [34]:

$$L_{j\to\ i} = L_j + \sum_{i'\in\ N(j)-\{i\}} L_{i'} \to\ \ j, \tag{2.38}$$

donde $N_j$ es el conjunto de $CN_s$ conectados al $j-ésimo\ CN$. En este punto, se completa una iteración, y se puede calcular el $LLR$ total como:

$$L_{j(total)} = L_j + \sum_{i\in N_j} L_i \to j \tag{2.39}$$

Los códigos LDPC son reconocidos por su capacidad para alcanzar capacidades de corrección de errores cercanas al límite de Shannon. Son robustos frente al ruido y a las variaciones en el canal de comunicación, lo que los hace adecuados para entornos donde la señal puede verse afectada por disturbios de diferentes tipos, como sucede en el canal las memorias flash NAND MLC durante múltiples ciclos de programación/borrado. Sin embargo, los códigos LDPC también presentan limitaciones y compromisos en términos de diseño. La introducción de mayor redundancia para mejorar la capacidad de corrección de errores puede resultar en un aumento de la latencia y la sobrecarga de área en el circuito. Esto implica que se requieran más recursos de hardware y tiempo de procesamiento para implementar y decodificar los códigos LDPC, lo que puede no ser viable en todas las aplicaciones, especialmente en dispositivos con restricciones de recursos [41].

Las tablas 2.3 y 2.4 resumen las caracterśticas de las distribuciones de probabilidad y de los códigos correctores de errores descritos en este capítulo.

| Distribuciones de probabilidad | Limitaciones de las memorias flash NAND MLC | Método de mitigación de errores | Deficiencia del modelo | Códigos correctores de errores | Arquitectura probada para código corrector de errores |
|---|---|---|---|---|---|
| t_Student | ▪ Errores de programación.<br>▪ Pérdida de carga en el tiempo.<br>▪ Interferencia entre celdas. | ▪ Corrección de errores asistida por celdas vecinas.<br>▪ Mecanismos de refrescamiento y reintento de lecturas<br>▪ Optimización de voltaje. | ▪ No aborda los disturbios de lectura. | ▪ LDPC<br>▪ BCH | ▪ ONFI<br>▪ SAGE |
| Binomial | ▪ Aumento de la tasa de error a lo largo de múltiples ciclos de operaciones de programación/borrado.<br>▪ Error de lectura.<br>▪ Pérdida de carga.<br>▪ Interferencia entre celdas. | ▪ Modelo de canal que describe los errores del canal estadísticamente.<br>▪ Códigos correctores de errores. | ▪ Modelo estadístico no coincide con los valores del modelo experimental BAC. | ▪ LDPC<br>▪ BCH | ▪ ONFI<br>▪ SAGE |

**Tabla 2.3:** Características de las distribuciones de probabilidad y de los códigos correctores de errores (parte 1)

| **Distribuciones de probabilidad** | **Limitaciones de las memorias flash NAND MLC** | **Método de mitigación de errores** | **Deficiencia del modelo** | **Códigos correctores de errores** | **Arquitectura probada para código corrector de errores** |
|---|---|---|---|---|---|
| Binomial | ▪ Aumento de la tasa de error a lo largo de múltiples ciclos de operaciones de programación/borrado.<br>▪ Error de lectura.<br>▪ Pérdida de carga.<br>▪ Interferencia entre celdas. | ▪ Modelo de canal que describe los errores del canal estadísticamente.<br>▪ Códigos correctores de errores. | ▪ Modelo estadístico con mejores resultados para modelar los datos del modelo BAC. | ▪ LDPC<br>▪ BCH<br>▪ Polares ∗ | ▪ ONFI<br>▪ SAGE |

∗ No existe referencia explícita que describa la implementación de códigos polares para las arquitecturas SAGE u ONFI.

**Tabla 2.4:** Características de las distribuciones de probabilidad y de los códigos correctores de errores (parte 2)

Después de revisar los modelos matemáticos que describen el canal de memoria flash NAND MLC y los códigos correctores de errores más referidos en la literatura científica, en próximo capítulo se explica la combinación del sistema de cifrado que utiliza esquemas de orden fraccionarios y los códigos polares para el diseño experimental de esta investigación.

Capítulo 3

# Cifrado y corrección de errores

El capítulo anterior abordó algunos de los principales modelos de canal para las memorias flash NAND MLC encontrados en la literatura científica, los cuales describen los errores de este medio de transmisión. Para mitigar estos errores, se propone, en este capítulo, el uso de los códigos polares como una herramienta clave para mejorar la fiabilidad del canal.

Este capítulo se describen el modelo matemático y los algoritmos del sistema caótico de orden fraccionario y de los códigos polares. Se describe también el diseño experimental.La idea subyacente es aprovechar la complejidad inherente de los sistemas caóticos fraccionarios para asegurar la información almacenada, mientras que los códigos polares garantizarán la corrección de errores.

Para esta investigación, la fuente de información está conformada por imágenes de 512 X 512 en escala de grises en formato .png, lo cual favorece la conversión a binario. Las imágenes utilizadas corresponden a rostros de personas descargadas del sitio web.`https://thispersondoesnotexist.com/`. Este sitio utiliza inteligencia artificial para crear rostros de cualquier edad, género y raza.

En la figura 3.1, se muestra el diseño experimental. Estos serían todos los estados que atraviesan las imágenes empleadas en esta investigación para la evaluación del sistema.

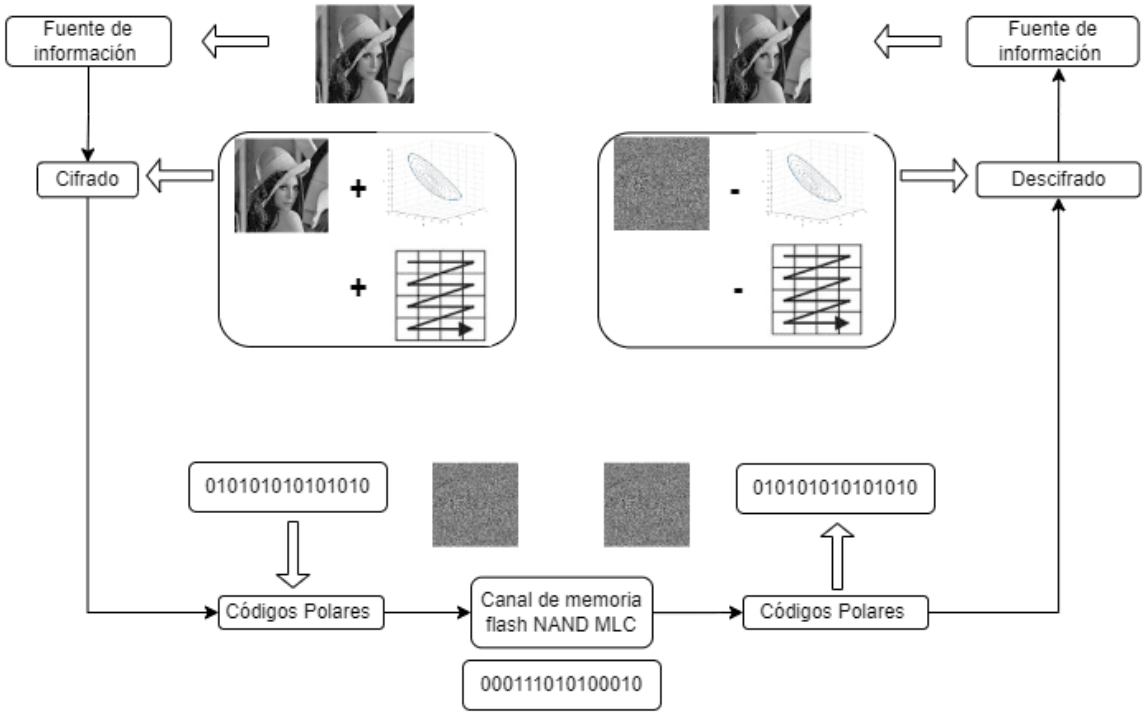



**Figura 3.1:** Diseño experimental

El diseño experimental representa la aplicación de un cifrado simétrico que incorpora los esquemas caóticos fraccionarios, la codificación polar y el envío a través del canal BAC_BBM. A la salida del canal, estas imágenes se decodifican en función de eliminar los errores introducidos, se descifran y se recupera la información. La figura 3.2 presenta una selección de 9 imágenes.

(1) (2) (3)

(4) (5) (6)

(7) (8) (9)

**Figura 3.2:** Imágenes utilizadas como fuente de información.

## 3.1 Códigos correctores de errores: códigos polares

### 3.1.1 Codificación

Una de las contribuciones de esta investigación es la implementación de los códigos polares para abordar la corrección de errores en un canal de memoria flash BAC_BBM.

Los códigos polares, creados por Arikan en 2008, son el primer código corrector de errores con una construcción explícita que logra de manera demostrable la capacidad de canal de Shannon. Esta codificación es utilizada en la corrección de errores para los canales de control de la 5G New Radio [67].

Estos códigos, pertenecen a una categoría de códigos lineales de bloque que se derivan de la transformación de polarización del canal. La polarización del canal en este caso no se refiere a la dirección del campo electromagnético, esta transformación implica la combinación y división del canal, que resulta en el ordenamiento de las posiciones de bits a medida que la longitud del código aumenta. Algunos canales se vuelven altamente confiables, mientras que otros se consideran no confiables (bits congelados o $frozen\ bits$). Durante la transmisión, los bits de información se asignan exclusivamente a los canales confiables, mientras que los canales no confiables se utilizan para transmitir bits previamente conocidos, generalmente transmitidos como ceros [68].

Si $W : X \to Y$ es un canal de comunicaciones con alfabeto entrada $X = \{0, 1\}$, alfabeto de salida $Y$, y probabilidades de transición $P(X|\ Y)$ . Se considera $I(W), Z(W)$, tal que [69, 70]:

$$I(W) \sum_{y \in Y} \sum_{x \in X} \frac{1}{2} W(y|x)\, log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)}, \tag{3.1}$$

$$Z(W)\sum_{y\in Y}\sqrt{W(y|0)\,W(y|1)}, \tag{3.2}$$

,donde $I(W) \in [0,1]$ es la capacidad de $W$, $Z(W) \in [0,1]$ es el parámetro de Bhattacharyya de $W$ y se utiliza como criterio para medir la confiabilidad.

Si se considera $W$, un canal binario de borrado ($BEC$), con probabilidad de borrado $\epsilon$, y se denota como $BEC(\epsilon)$, entonces $Z(W) = \epsilon$ , y $(W) = 1-\epsilon$, entonces $\left\{W_N^{(i)} : 1 \le i \le N\right\}$ es un conjunto de $B-DMC$ polarizados con índices $i$ , que se pueden obtener mediante polarización del canal en las $N$ copias independientes de $W$. Para cada $W_N^{(i)}$, el parámetro de Bhattacharyya es denotado como $Z\left(W_N^{(i)}\right)$ [68, 70].

$$Z\left(W_{2k}^{(i)}\right) = \begin{cases} 2Z\left(W_k^{(i)}\right) - Z^2\left(W_k^{(i)}\right) & : \ 1 \le i \le k \\ Z^2\left(W_k^{(i-k)}\right) & : \ k+1 \le i \le 2k \end{cases}, \tag{3.3}$$

siendo $N = 2^n$, $n \ge 1$, $i = 1, 2 \ldots N$, $k = 1, 2, \ldots, 2^{n-1}$.

Si $\epsilon = 0.5$, entonces $Z(W) = I(W)$, $N = 8$, $n = 3$, entonces el parámetro de Bhattacharyya sería:

$$\begin{array}{lll} & & I(W^{+++}) = 0.9961 \\ & I(W^{++}) = 0.9375 & \\ & & I(W^{++-}) = 0.8789 \\ I(W^{+}) = 0.75 & & I(W^{+-+}) = 0.8086 \\ & I(W^{+-}) = 0.5625 & \\ & & I(W^{+--+}) = 0.3164 \\ I(W) = 0.5 & & \\ & & I(W^{-++}) = 0.6836 \\ & I(W^{-+}) = 0.4375 & \\ & & I(W^{-+-}) = 0.1914 \\ I(W^{-}) = 0.25 & & \\ & & I(W^{--+}) = 0.1211 \\ & I(W^{-+}) = 0.0625 & \\ & & I(W^{-+-}) = 0.0039 \end{array}$$

**Tabla 3.1:** Parámetro de Bhattacharyya para $\epsilon = 0.5$, $N = 8$

El parámetro Bhattacharyya se ordena de menor a mayor. Los canales buenos o confiables son los de mayor probabilidad, mientras que los $frozen\ bits$ o canales no confiables son los de menor probabilidad.

Para los códigos polares, la codificación puede considerarse como una operación de multiplicación vectorial tal que $x = uG$, donde:

$u$ : es un vector fila de $N$ elementos, con $k$ elementos correspondientes al mensaje original y $N - k$ elementos correspondientes a los $frozen\ bits$.
$G$ : es la matriz generadora de $N$ x $N$ elementos, tal que $G = G^{\otimes n}$.
$x$ : es el vector fila de $N$ elementos correspondientes al vector codificado.

La matriz $G = G^{\otimes n}$ , se construye aplicando la $n - ésima$ potencia de Kronecker, tal que, $G = G^{\otimes n} = \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}$. El producto de Kronecker, en su forma general en este caso puede escribirse como:

$$G^{\otimes n} = \begin{pmatrix} G^{\otimes(n-1)} & 0 \\ G^{\otimes(n-1)} & G^{\otimes(n-1)} \end{pmatrix} \tag{3.4}$$

Si $N = 2$ entonces,

$$G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{3.5}$$

La codificación puede expresarse como $x = [u_1, u_2]\, G = [x_1, x_2]$, siendo:

$$\begin{aligned} x_1 &= u_1 \otimes\ u_2 \\ x_2 &= u_2 \end{aligned} \tag{3.6}$$

La codificación polar puede representarse como árbol binario o como recombinación de canales.

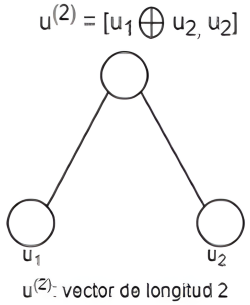



**Figura 3.3:** Representación del árbol binario para $N = 2$.

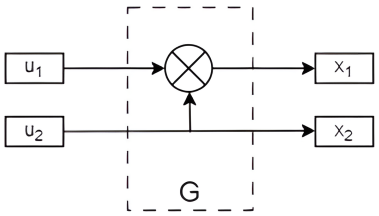



**Figura 3.4:** Representación de la combinación de canales para $N = 2$.

Si $N = 8$ entonces,

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.7}$$

El codificador puede expresarse como $x = [\, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8] G = [\, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]$, siendo:

$$
\begin{aligned}
x_1 &= u_1 \otimes u_2 \otimes u_3 \otimes u_4 \otimes u_5 \otimes u_6 \otimes u_7 \otimes u_8 \\
x_2 &= u_5 \otimes u_6 \otimes u_7 \otimes u_8 \\
x_3 &= u_3 \otimes u_4 \otimes u_7 \otimes u_8 \\
x_5 &= u_2 \otimes u_4 \otimes u_6 \otimes u_8 \\
x_6 &= u_6 \otimes u_8 \\
x_7 &= u_4 \otimes u_8 \\
x_4 &= u_7 \otimes u_8 \\
x_8 &= u_8
\end{aligned} \tag{3.8}
$$

El árbol binario para $N = 8$ está representado en la figura 3.5:

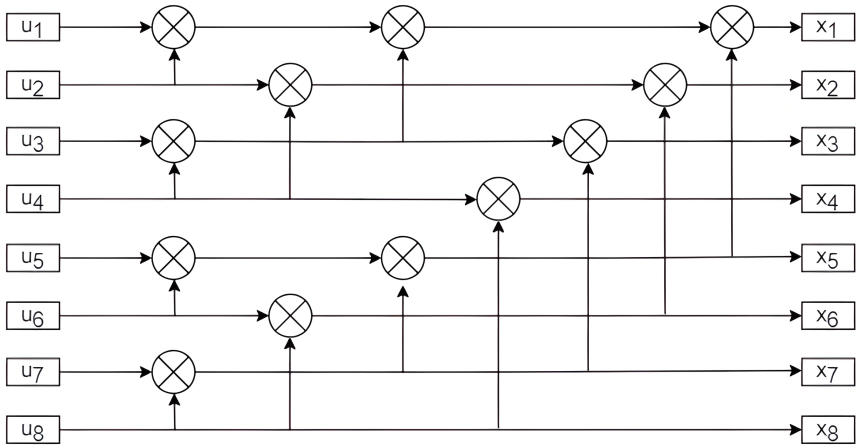



**Figura 3.5:** Representación de la combinación de canales para $N = 8$.

Para el mensaje $u = 00010011$, con $k = 4$, utlizando los valores del parámetro de Bhattacharyya de la tabla 3.1, se obtiene $x = 10100101$, siendo $x_4, x_6, x_7, x_8$ los canales buenos, y $x_1, x_2, x_3, x_5$ los *frozen bits*. La figura 3.6 describe el proceso de codificación. En rojo están señalizados los *frozen bits* y en verde los canales buenos.

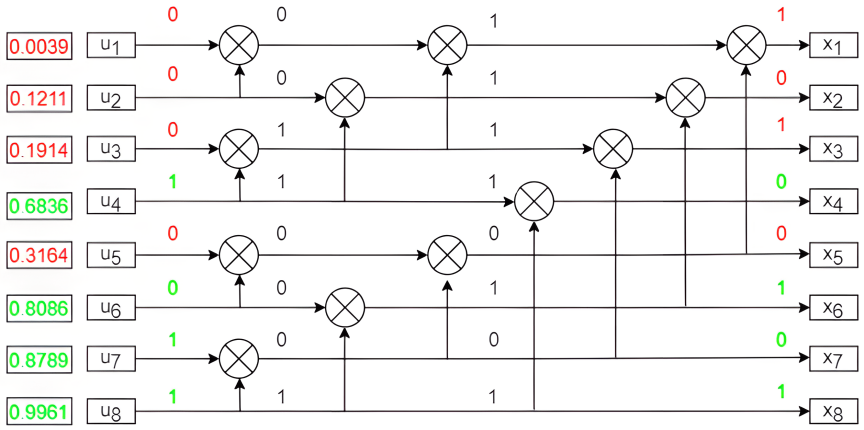



**Figura 3.6:** Representación de la recombinación de canales y el parámetro de Bhattacharyya $N = 8$, $k = 4$, $\epsilon = 0.5$

### 3.1.2 Decodificación

La figura 3.7 muestra procesos los de codificación y decodificación polar para $N = 2$ mediante la recombinación de canales. La información pasa a través del canal de memoria flash BAC_BBM, lo que introduce cierto *ruido* en la información de salida.

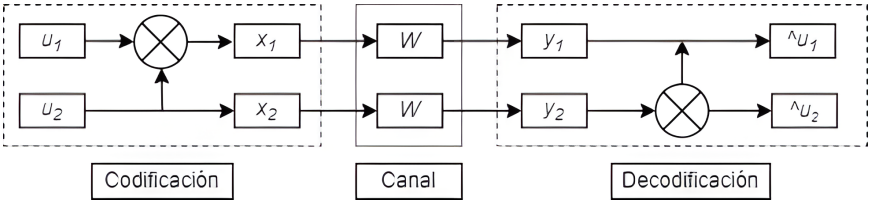



**Figura 3.7:** Representación los procesos de codificación y decodificación.

En esta figura $u_1$, $u_2$ , es la información, $x_1$,$x_2$ es la representación del vector codificado, $W$ es la representación del canal, $y_1$, $y_2$ es la información a decodificar, y $\hat{u}_1, \hat{u}_2$, es la información decodificada. En el proceso de decodificación no estamos obteniendo exactamente el vector $u_N$, debido a los *ruidos* introducidos por canal, por tanto, el vector de decodificación se denotará como $\hat{u}_N$. La decodificación se hará a través de un algoritmo de cancelación sucesiva. El decodificador toma $N$ decisiones para cada $u_i$, donde $i$ representa la posición de cada bit en la secuencia. Si $u_i$ es un $frozen\ bit$ el decodificador tomará $\hat{u}_i$ como un valor desconocido o cero. Si $u_i$ es un

bit de información, el decodificador espera a estimar todos los bits previos y luego ejecuta la razón de verosimilitud siguiente [71].

$$L_N^{(i)}\left(y^N, \hat{u}^{i-1}\right) = \frac{W_N^{(i)}\left(y^N, \hat{u}^{i-1}|0\right)}{W_N^{(i)}\left(y^N, \hat{u}^{i-1}|1\right)} \tag{3.9}$$

El decodificador establece $\hat{u}_i = 0$, si $L_N^{(i)} \geq 1$ , si no, $\hat{u}_i = 1$.

La complejidad del algoritmo de decodificación está determinada por la complejidad de calcular las razones de verosimilitud (LLR). Las relaciones recursivas para calcular los valores de LLR son:

$$L_N^{(2i-1)}\left(y^N, \hat{u}^{2i-2}\right) = \frac{1 + L_{N/2}^{(i)}\left(y^{N/2}, \hat{u}_o^{2i-2} \otimes \hat{u}_e^{2i-2}\right) L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_e^{2i-2}\right)}{L_{N/2}^{(i)}\left(y^{N/2}, \hat{u}_o^{2i-2} \otimes \hat{u}_e^{2i-2}\right) + L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_e^{2i-2}\right)} \tag{3.10}$$

$$L_N^{(2i)}\left(y^N, \hat{u}^{2i}\right) = L_{N/2}^{(i)}\left(y^{N/2}, \hat{u}_o^{2i-2} \otimes \hat{u}_e^{2i-2}\right)^{1-2\hat{u}_{2i-1}} L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_e^{2i-2}\right) \tag{3.11}$$

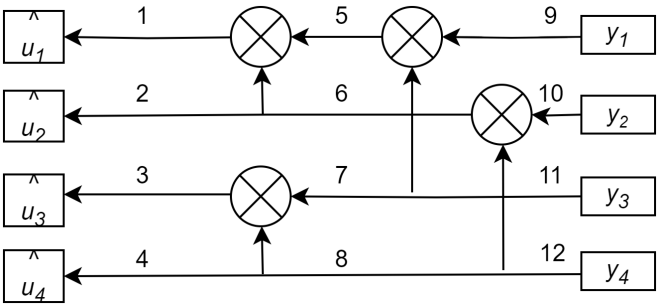



**Figura 3.8:** Decodificador para $N = 4$.

La figura 3.8 representa un decodificador polar para $N = 4$ . Si $N = 4$ el decodificador debe calcular 12 LLRs. Para la decodificación, inicialmente, se ubican en cero las posiciones de los $frozen\ bits$, ya que los valores de $k, N$ para el cálculo de estas posiciones son conocidos por decodificador. Para calcular $\hat{u}_1$, es necesario

activar el nodo 1. Para la activación del nodo 1, se necesita a su vez el cálculo de los LLRs de los nodos 5 y 6. Los nodos 5 y 6 necesitan la activación de los nodos 9,10,11 y 12. Los valores de LLR en estos nodos son conocidos ya que están en el nivel del canal. Las razones de verosimilitud en los nodos 5 y 6 son calculadas mediante la ecuación 3.10. El último LLR en calcularse en esta etapa es del nodo 1. Si $\hat{u}_1$ es un $frozen\ bits$, entonces $\hat{u}_1 = 0$.

Para el cálculo del nodo 2 se activan igualmente los nodos 5 y 6, teniendo ya calculados los LLRs de cada uno de ellos. Si $\hat{u}_1 = 0$, los LLRs de los nodos 5 y 6 se combinan mediante la ecuación 3.11. Si $\hat{u}_1 = 1$ se combina el LLR del nodo 6 con el inverso del LLR del nodo 5, mediante la ecuación 3.11. Tiendo $\hat{u}_1$ y $\hat{u}_2$, se actualizan los valores estimados para los nodos 5 y 6. Para estimar $\hat{u}_3$, el nodo 3 activa los nodos 7 y 8. Los LLRs de los nodos 7 y 8 se calculan utilizando los valores de los LLR de los nodos 9,10,11 y 12 junto con el estado de los nodos 5 y 6. El valor del nodo 5 determina si el LLR del nodo 11 debe combinarse con la razón de verosimilitud en el nodo 9 o con la inversa de la razón de verosimilitud en el nodo 9. Finalmente, desde los nodos 7 y 8 se calculan los LLR en el nodo 3. Teniendo $\hat{u}_3$, se puede calcular el nodo 4 y se completa así el procedimiento de decodificación [72].

### 3.1.3 Relación de complejidad entre códigos polares y códigos LDPC

En [73], se demuestra que las tasas de error asintóticas tanto de los códigos polares como de los códigos LDPC son funciones de la complejidad $T$ y longitud de código $N$, descritas por la función $C = 2^{-a2^{b*\frac{T}{N}}}$, donde $a, b$ son constantes que dependen del canal y el esquema de comunicación. En la figura 3.9, se muestra la relación de complejidad entre códigos polares y códigos LDPC para $T \leq O\left(N * l\ o\ g\ N\right)$, $N = 25$ y $a = b = 1$. Esta relación es determinante para valorar la implementación del sistema en la arquitectura ONFI. Este estudio demuestra que la obtención de la matriz de paridad y las iteraciones del algoritmo decodificación de pueden ser procesos complejos para los códigos LDPC , sin embargo, la recombinación de ca-

nales y la utilización de las propiedades el parámetro de Bhattacharyya facilitan la codificación polar. El algoritmo de decodificación de Cancelación Sucesiva, aunque puede ser subóptimo en determinado escenario es menos complejo en comparación con los algoritmos iterativos de decodificación LDPC.

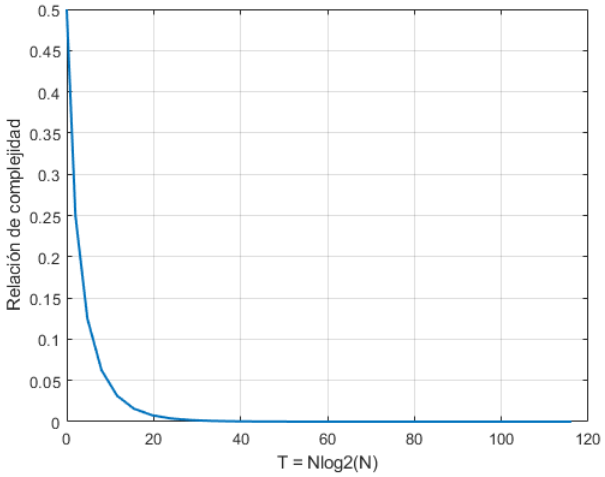



**Figura 3.9:** Relación de complejidad entre códigos LDPC y códigos polares.

## 3.2 Cifrado caótico aplicado a imágenes.

El cifrado caótico ha emergido como un área de investigación de considerable interés. Esto se debe a la ergodicidad de estos sistemas, a la sensibilidad a las condiciones iniciales, así como sus ventajas potenciales para garantizar la seguridad en la comunicación.

Claude Shannon, en 1949, fue el pionero en mencionar la viabilidad de utilizar sistemas de cifrado en la transmisión con lo cual, en el transcurso de las décadas posteriores, estos sistemas adquirieron importancia en la seguridad de las comunicaciones. En la década de 1980 se comenzó a desarrollar la teoría del cifrado caótico. La primera aplicación práctica del cifrado caótico analógico se presentén 1990 por Pecora y Carroll, quienes propusieron un método para transmitir señales y establecer un proceso de sincronización entre dos sistemas caóticos. Posteriormente, en 1998, Jiri Fridrich introdujo la primera aplicación de cifrado caótico digital, que consistía en

un cifrado simétrico utilizando mapas caóticos, avance significativo en la aplicación del caos para garantizar la seguridad en la comunicación digital [74–76]

La transformación de texto plano a texto cifrado se conoce como cifrado, mientras que el proceso inverso, se denomina descifrado. Un algoritmo de cifrado se compone de dos tipos de operaciones: difusión y confusión. En las operaciones de difusión, se dispersa la información del mensaje utilizando la señal caótica como base. Las operaciones de confusión se encargan de mezclar y enmascarar la relación entre los datos del mensaje y la clave de cifrado. Esta combinación de procesos busca fortalecer la seguridad del sistema al hacer que la información cifrada sea más difícil de interpretar para cualquier persona no autorizada [77].

Los sistemas caóticos de orden fraccionario son una generalización de los sistemas dinámicos caóticos tradicionales, donde las ecuaciones que describen el comportamiento del sistema incluyen derivadas de orden no entero. En el contexto de los sistemas dinámicos, se distinguen dos categorías principales: las ecuaciones diferenciales y los mapas iterados [78].

Las ecuaciones diferenciales modelan la evolución de sistemas continuos temporales, mientras que los mapas iterados abordan problemas donde el tiempo se considera de forma discreta. Estos sistemas pueden ser clasificados como lineales o no lineales, siendo este último caso relevante cuando se trata de estudiar fenómenos caóticos. Al incorporar sistemas de orden fraccionario en el cifrado caótico, se proporciona un nivel adicional de seguridad [79].

La teoría que refiere las derivadas e integrales de órdenes no enteros se denominan cálculo fraccionario. El operador fundamental de este cálculo es ${}_aD_t^q$ y se denota como operador integrodiferencial continuo:

$$ {}_aD_t^q = \begin{cases} \frac{\mathrm{d}^{\mathrm{q}}}{\mathrm{d}t^q}, & q > 0 \\ 1, & q = 1 \\ \int_a^t (d\tau)^q, & q < 0 \end{cases} , \tag{3.12} $$

siendo $q \in R^{+}$.

Para definir las derivadas e integrales fraccionarias, una de las representaciones más utilizada es la representación de Grünwald_Letnikov, la cual se emplea cuando se busca una solución numérica y es apropiada para ser implementada en cálculos numéricos de tipo iterativo.

Para su desarrollo se considera $f(t)$ una función continua, por tanto, su primera derivada puede expresarse como:

$$\frac{\mathrm{d}}{dt} f(t) \equiv f'(t) = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h} \tag{3.13}$$

Si se aplica la definición de la $n-ésima$ derivada de $f(t)$, se obtiene la formula general:

$$\frac{\mathrm{d^n}}{\mathrm{d}t^n} f^{(n)}(t) \equiv \ f^{(n)}(t) = \lim_{h \to 0} \frac{1}{h^n} \sum_{j=0}^{n} (-1)^j \binom{n}{j} f(t - jh) \tag{3.14}$$

Si se sustituye $n$ por $q$, se considera $n = \frac{t-a}{h}$ , y se tienen en cuenta las ecuaciones 3.13 y 3.14, se puede escribir la derivada de orden fraccionario para $q > 0$ de Grünwald_Letnikov como:

$${}_aD_t^q f^n(t) \equiv f^n(t) = \lim_{h \to 0} \frac{1}{h^q} \sum_{j=0}^{\frac{t-a}{h}}, (-1)^j \binom{q}{j} f(t - jh) \tag{3.15}$$

donde $q(t)$, es una función acotada en el intervalo $(0, 1)$.

En [80], se estudia el desarrollo numérico de ecuaciones diferenciales de orden fraccionario, fundamentado en la definición de Grünwald_Letnikov, utilizando esquemas de diferencias finitas para aproximar la solución de derivadas de orden fraccionario. La formulación numérica para resolver la ecuación diferencial de orden fraccionario se puede expresar explícitamente como:

$$y_{n+1} = c_1^q y_n + c_2^q y_{n-1} + \ldots + c_{n+1}^q y_0 + r_{n+1}^q y_0 + h^q f\left(y_n\right), \tag{3.16}$$

donde $h$ es el paso de integración y $0 < q \leq 1$.

$C_n^\alpha$, es el coeficiente binomial, el cual se define como:

$$C_n^\alpha = \left(1 - \frac{\alpha + 1}{n}\right) C_{n-1}^\alpha \tag{3.17}$$

Y $r_n^\alpha$ se sería el termino de corrección [80]:

$$r_n^\alpha = \frac{t^{n-\alpha}}{\Gamma\left(n + 1 - \alpha\right)} \tag{3.18}$$

Los sistemas caóticos de orden fraccionario se clasifican en dos categorías: conmensurados e inconmensurados. Se consideran sistemas de orden conmensurado cuando el valor de $q$ es idéntico en todas las ecuaciones que componen el sistema. Por otro lado, se denominan sistemas de orden inconmensurado cuando el valor de $q$ varía entre las diferentes ecuaciones del sistema.

En [79], se proponen varios sistemas caóticos de orden fraccionario para el cifrado de imágenes. Uno de los sistemas más estudiados en la literatura científica, es el sistema de Lorenz, reconocido por su paradójica naturaleza caótica, ya que describe $efecto\ mariposa$, término utilizado para representar la sensibilidad a las condiciones iniciales en la teoría del caos. El $efecto\ mariposa$ implica que pequeñas variaciones en las condiciones iniciales pueden generar resultados completamente diferentes en el sistema, lo cual lo convierte en una herramienta valiosa para el cifrado de imágenes debido a su naturaleza impredecible.

Este sistema está descrito por las siguientes ecuaciones:

$$\begin{cases} {}_0D_t^{q_1} = \sigma\left(y\left(t\right) - x\left(t\right)\right) \\ {}_0D_t^{q_2} = x\left(t\right)\left(\rho - y\left(t\right)\right) - z\left(t\right) \\ {}_0D_t^{q_3} = y\left(t\right)x\left(t\right) - \beta z\left(t\right) \end{cases}, \tag{3.19}$$

donde $\sigma$ es número de Prandtl [12], $\rho$ es el número de Rayleigh [13], y $\beta$ el tamaño de la región espacial aproximada por el sistema. A continuación, se muestra el sistema de Lorenz para $(\sigma, \rho, \beta) = (10, 28, 8/3)$, paso de muestreo $h = 0.005$, orden conmensurado $q_1$, $q_2$, $q_3 = 0.995$ y condiciones iniciales $(x\left(0\right), y\left(0\right), z\left(0\right))$=$(0.1, 0.1, 0.1)$.

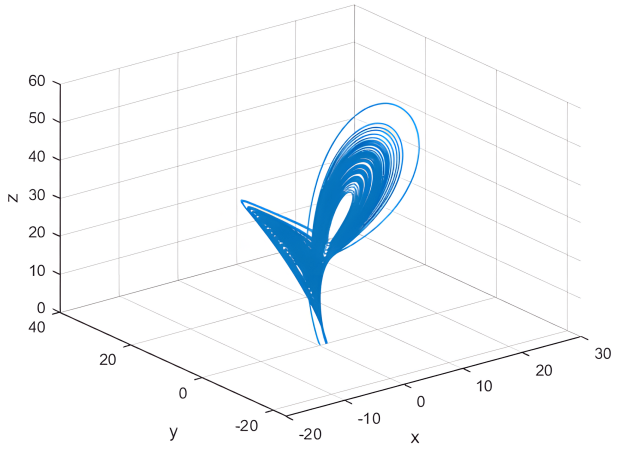



**Figura 3.10:** Sistema caótico de orden fraccionario de Lorenz para un tiempo de simulación de 100 segundos y muestreo $h = 0.005$, $q_1 = q_2 = q_3 = 0.995$, $(x\left(0\right), y\left(0\right), z\left(0\right)) = (0.1, 0.1, 0.1)$, $(\sigma, \rho, \beta) = (10, 28, 8/3)$

El sistema caótico de orden fraccionario de Lorenz, es utilizado para el cifrado de la información, y depende de la llave del sistema de cifrado.

Un sistema de cifrado se define especificando el mensaje junto a tres algoritmos distintos, cada uno con funcionalidades particulares:

1. Algoritmo generador de llaves aleatorias: Este algoritmo produce una llave que se emplea en el proceso de cifrado. La llave se genera de forma aleatoria dentro de un rango predefinido. Cuando coincide la llave en el cifrado, así como en el

[12] Razón entre la viscosidad cinemática y la difusividad.
[13] Caracteriza la convección térmica.

descifrado, se considera un cifrado simétrico. Cuando se utilizan llaves distintas para el cifrado y el descifrado, se considera un cifrado asimétrico.

2. Algoritmo de cifrado: Este proceso utiliza la llave y el mensaje como entradas, y produce un texto cifrado como salida.

3. Algoritmo de descifrado: Utilizando la llave y el texto cifrado como entradas, este algoritmo genera el mensaje como salida. Se considera el algoritmo inverso al algoritmo de cifrado.


**Figura 3.11:** Cifrado simétrico

### 3.2.1 Generación de llaves aleatorias

El tamaño de la llave se selecciona en dependencia de la cantidad total de combinaciones posibles que pueden ser utilizadas en el proceso de cifrado. En este caso, la llave proporciona al sistema de cifrado los valores iniciales y el orden para el sistema caótico empleado. Los bits de la clave se dividen en partes, y cada una de las partes corresponden a un parámetro específico del sistema de cifrado.

En métodos de cifrado como AES (Advanced Encryption Standard), se considera que un espacio de llave de 128 bits es muy seguro. Una llave de $n$ bits equivale

a $2^n$ posibles combinaciones.

La llave de 128 bits, se convierte a hexadecimal y se obtienen 32 caracteres entre 0 y F. Las 8 primeras posiciones corresponden con el valor inicial de $x$. Las posiciones del 9 al 16 corresponden con el valor inicial de $y$. Las posiciones del 17 al 24 corresponden con el valor inicial de $z$. Las posiciones de 25 al 32 corresponden al orden conmensurado. Los valores se introducen en el algoritmo 2, que devuelve los valores de $x, y$ y $z$ de la 3.3. La tabla 3.2, a continuación, describe los pasos anteriores.

| | | | | |
|---|---|---|---|---|
| Paso 1 | 1 2 3 4...31 32<br>Secuencia de 32 bits | 33 34 35...63 64<br>Secuencia de 32 bits | 65 66 67...95 96<br>Secuencia de 32 bits | 97 98 99...127 128<br>Secuencia de 32 bits |
| Paso 2 | Conversión a HEX | Conversión a HEX | Conversión a HEX | Conversión a HEX |
| Paso 3 | 8 caracteres entre 0 y F | 8 caracteres entre 0 y F | 8 caracteres entre 0 y F | 8 caracteres entre 0 y F |
| Paso 4 | Algoritmo de conversión<br>$0 \leq x(0) \leq 1$ | Algoritmo de conversión<br>$0 \leq y(0) \leq 1$ | Algoritmo de conversión<br>$0 \leq z(0) \leq 1$ | Algoritmo de conversión<br>$0.9941 \leq q \leq 1$ |
| | Condición inicial $x(0)$ | Condición inicial $y(0)$ | Condición inicial $z(0)$ | Orden conmensurado |

**Tabla 3.2:** Obtención de las condiciones iniciales del sistema a partir de la generación la llave aleatoria.

**Algoritmo 3:** Obtención de las condiciones iniciales y orden fraccionario para el sistema Lorenz

**Entrada:** Número hexadecimal convertido a cadena de caracteres.

**Salida:** Número flotante entre 0 y 1.

1. Convertir de hexadecimal a decimal la cadena de caracteres.
2. Normalizar aplicando decimal/ (16*8 - 1).
3. Fin.

En [79] se llevá cabo un análisis de la cuenca de atracción de Lorenz mediante el cálculo de los exponentes de Lyapunov. Este análisis garantiza la selección de un espacio de llaves adecuado y proporciona valores iniciales que generan un comportamiento caótico en el sistema. A continuación, se muestran los rangos de las condiciones iniciales y orden fraccionario considerados para el espacio de llave para el sistema de Lorenz.

| Condiciones iniciales |
|---|
| $0 \leq x(0) \leq 1$ |
| $0 \leq y(0) \leq 1$ |
| $0 \leq z(0) \leq 1$ |
| $0.9941 \leq q \leq 1$ |

**Tabla 3.3:** Rangos de condiciones iniciales y orden fraccionario considerados para el espacio de llave para el sistema de Lorenz.

La figura 3.4 muestra un ejemplo de obtención de las condiciones iniciales del sistema a partir de la llave de 128 bits 11111010111110101111101011111010 |1101100101 000011101001000110 |11110001001000100101011110011 10 |11111111101000101111000111110010.

| Conversión HEX | FAFAFAFA | 1B2C3A46 | 78912BCE | FFA2F1F2 |
|---|---|---|---|---|
| Condiciones iniciales | $x(0) = 0.9803$ | $y(0) = 0.106$ | $z\,(0) = 0.4709$ | $q = 0.9986$ |

**Tabla 3.4:** Ejemplo de obtención de las condiciones iniciales del sistema a partir de la llave.

De este modo, las condiciones iniciales para producir la trayectoria del sistema son definidas a través de la llave. Las trayectorias de cada eje de la señal caótica se generan según el tamaño de los datos del mensaje y en relación con la llave para el proceso de cifrado.

A continuación, se muestra el sistema de Lorenz y trayectoria de $x(t)$ para las condiciones iniciales $(x(0), y(0), z(0)) = (0.9803, 0.106, 0.4709)$ , orden conmensurado $q_1 = \ q_2 = \ q_3 \ = \ 0.9986$ , un paso de muestreo $h = 0.005$ y un tiempo de simulación de 1400 segundos.

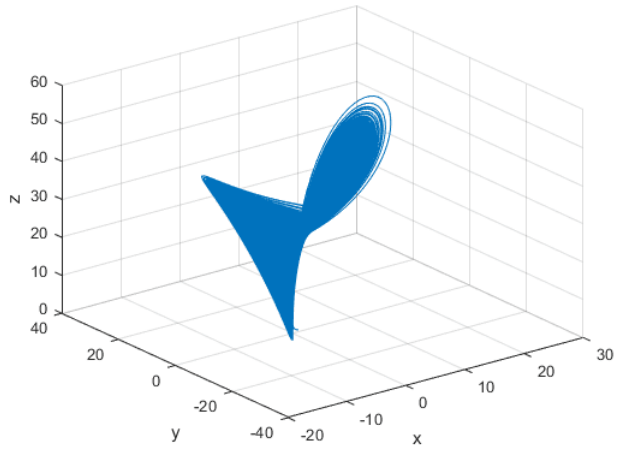



**Figura 3.12:** Sistema caótico de orden fraccionario de Lorenz para un tiempo de simulación de 1400 segundos, paso muestreo $h = 0.005$, $q_1 = q_2 = q_3 = 0.9986$,$(x\,(0)\,, y\,(0)\,, z\,(0)) = \ (0.9803, 0.106, 0.4709)$ y $(\sigma, \rho, \beta\ \ ) = (10,\ 28,\ 8/3)$.

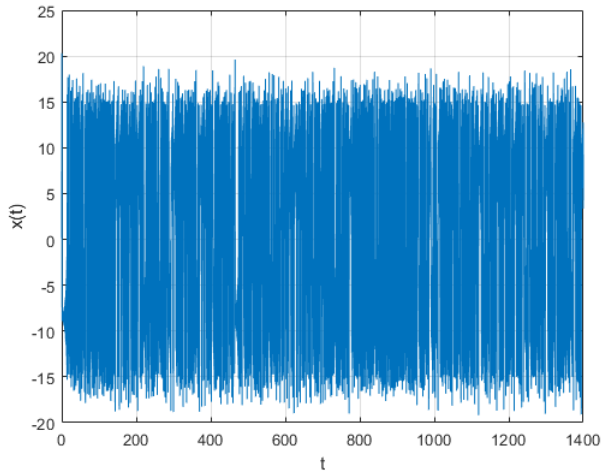



**Figura 3.13:** Trayectoria del estado x(t) del sistema caótico de orden fraccionario de Lorenz para un tiempo de simulación de 1400 segundos, paso muestreo $h = 0.005$,
$q_1 = q_2 = q_3 = 0.9986$,$(x\,(0)\,, y\,(0)\,, z\,(0)) = \ (0.9803, 0.106, 0.4709)$ y $(\sigma, \rho, \beta\ ) = (10,\ 28,\ 8/3)$.

En la trayectoria caótica de cada uno de los estados $x, y$ y $z$, se buscan los elementos de menor y mayor valor. Una vez identificados estos valores, se escalan para que estén dentro de un rango de 0 a 255, lo que permite que cada punto pueda ser representado por 8 bits. Luego se organizan los datos en una matriz $T$ con dimensiones en proporción a las imágenes que se cifraran. Una vez formada esta matriz, se inician las operaciones para el cifrado.

Como se menciona previamente, el algoritmo de cifrado se compone de operaciones de difusión, que incorporan la información de la señal caótica, y operaciones de confusión, que desordenan los datos del mensaje. Esta investigación, al aplicar estos procesos de cifrado a imágenes de 512 X 512 píxeles en escala de grises, las operaciones de difusión influirán directamente en los píxeles de las imágenes, mientras que las operaciones de confusión, al alterar la posición del mensaje, se manifestarán como un reordenamiento de los píxeles siguiendo un patrón específico.

En este contexto, se propone la utilización de un sistema de cifrado de scrambling mejorado. Este esquema de cifrado ha sido propuesto y evaluado en [79]. Los resultados en función la razón de cambio de píxeles, proporción máxima de señal a ruido, error cuadrático medio, análisis de la entropía, entre otros, son superiores en comparación con otros esquemas evaluados en ese trabajo de investigación.

El sistema de cifrado de scrambling mejorado está compuesto por la operación de confusión de scrambling_pixel y las operaciones de difusión scrambling_bit y baker_bit. Estas 3 operaciones se explican a continuación.

### 3.2.2 Operación scrambling_bit.

Todos los algoritmos de esta investigación están diseñados para trabajar con imágenes de rostros de personas de 512 x 512 píxeles. Al leer cada una de estas imágenes, tiene en total una matriz de 262,144 elementos y cada uno de estos elementos serán de valor decimal entre 0 y 255. En adelante se refiere a esta matriz de elementos de la imagen como matriz $M$.

La operación de difusión scrambling_ bit crea 2 vectores de 512 elementos para definir las posiciones de las filas y las columnas de la matriz $M$. Esta operación se aplica a nivel de bit y divide la matriz $M$ en 8 secciones como muestra la figura 3.14:

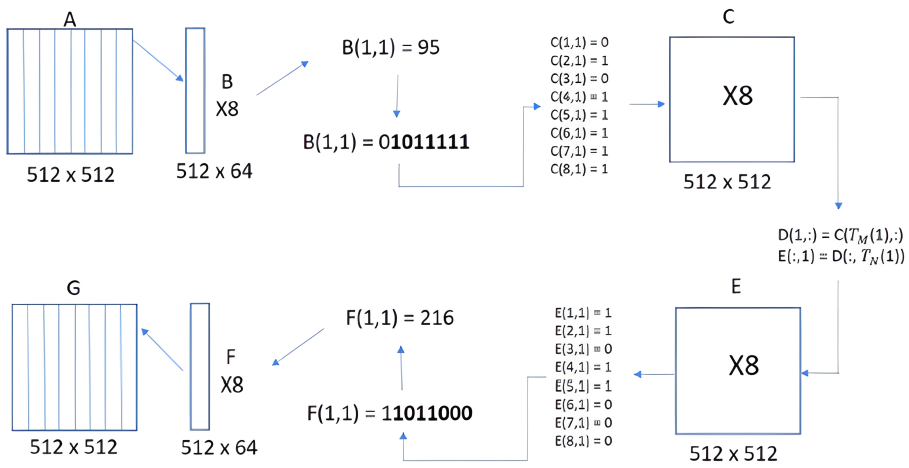



**Figura 3.14:** Diagrama de la operación scrambling_bit.

Los procedimientos de esta operación se detallan a continuación [79]:

1. Creación de los vectores de permutación $T_M$ y $T_N$, resultado de la cuantificación en 8 bits de las 512 muestras resultantes de la integración numérica del oscilador caótico de orden fraccionario.

2. Se obtiene la matriz A, con los elementos de la matriz $M$, provenientes de la imagen a escala de grises, que se redimensiona a $M$ x $N$.

3. La matriz A se divide en 8 matrices de dimensiones $M$ x $N/8$.

4. Los píxeles de cada matriz B se transforman a numeración binaria y cada bit ocupa una nueva posición, formando 8 matrices $C$ con elementos booleanos.

5. Se aplica el vector $T_M$, proveniente de la señal caótica a cada una de las 8 matrices $C$ y se obtienen 8 matrices D de dimensión $M$ x $N$.

6. Se aplica el vector de permutación $T_N$ a las 8 matrices $D$ y se obtienen 8 matrices E de dimensión $M$ x $N$.

7. Las 8 matrices E se convierten de binario a decimal mediante la operación inversa a la descrita en el paso 4 y así se obtienen 8 matrices F de dimensión $M$ x $N/8$.

8. Se obtiene la matriz $G$, uniendo los bloques de matrices las 8 matrices $F$.

El algoritmo para generación de vectores de permutación $T_M$ y $T_N$ se encuentra en la sección 3.2.4. La figura 3.15 muestra el rostro 8 de la base de datos con la operación scrambling_bit aplicada:

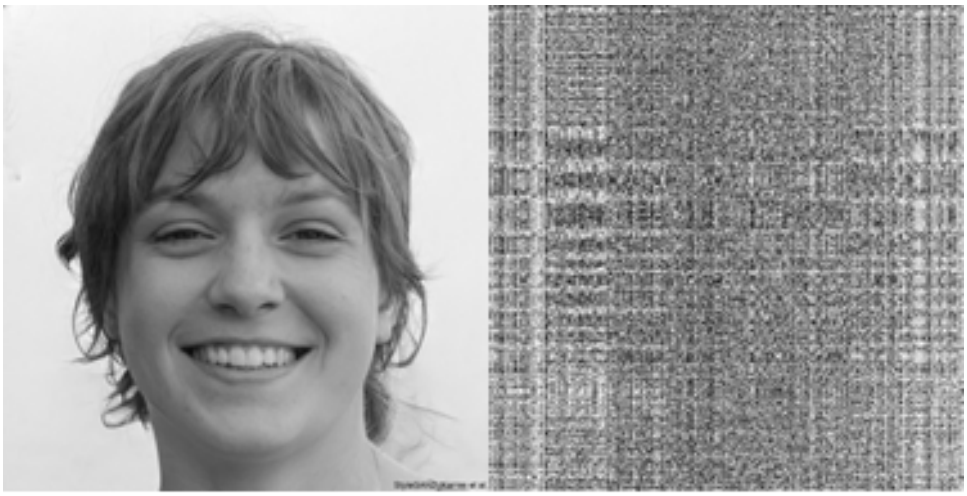


**Figura 3.15:** Rostro 8 cifrado aplicando operación scrambling_bit.

### 3.2.3 Operación baker_bit

Esta operación es descrita en [81]. La idea es que el nuevo valor para cada bit dependa del valor del píxel anterior y del elemento correspondiente de la matriz $T$ proveniente de la señal caótica. La ecuación que define esta operación es:

$$C(k) = \phi(k) \otimes [M(k) + \phi(k)] \ \ mod256 \ C(k-1), \tag{3.20}$$

donde:

$C(k)$ : es el valor de píxel cifrado.

$\phi(k)$ : corresponde al elemento $k$ de la señal caótica.

$M(k)$ : píxel de la imagen a cifrar.

$C(k-1)$ : es el valor de píxel anterior cifrado. $M(1,1)$ se utiliza como valor inicial de $C(0)$.

La figura 3.16 muestra el rostro 9 de la base de datos con la operación baker_bit aplicada:

**Figura 3.16:** Rostro 9 cifrado aplicando la operación baker_bit

### 3.2.4 Operación scrambling_píxel

Esta operación esta descrita en [82]. Aplica la misma secuencia de pasos que la operación scrambling_bit, pero no aplica el paso 4, por tanto, la operación se queda a nivel de píxeles. Empleando la misma línea que la operación scrambling_bit se crean los vectores de permutación $T_M$ y $T_N$ , según el Algoritmo 3. Estos vectores modifican los píxeles de la matriz $C$ en el paso 5 de la secuencia, donde se aplica el cambio en las filas utilizando $T_M$.Posteriormente se aplica el cambio en las columnas aplicado $T_N$ a las matrices $D$.

El algoritmo 4 muestra el pseudocódigo para generación de vectores de permutación $T_M$ y $T_N$ [80].

**Algoritmo 4:** Pseudocódigo de algoritmo para generación de vectores de permutación $T_M$ y $T_N$

1. Inicio;
2. i=1; j=1; k=1;
3. mientras (i < 513) hacer;
4. si n1(k) en vector $T_M$ = sí) entonces;
5. Ir a paso 10;
6. sino (Se encuentra n1(k) en vector $T_M$ = no);
7. M [i] = n1(k);
8. i=i+1;
9. terminar sí;
10. k=k+1;
11. terminar mientras;
12. k=1;
13. mientras (j < 513) hacer;
14. si n2(k) en vector $T_N$ = sí) entonces;
15. Ir a paso 20;
16. sino (Se encuentra n2(k) en vector $T_N$ = no);
17. $T_N$ [i] = n2(k);
18. j=j+1;
19. terminar sí;
20. Fin;

La figura 3.17 muestra el rostro 1 de la base de datos con la operación scrambling_pixel aplicada:

**Figura 3.17:** Rostro 1 cifrado mediante la operación scrambling_pixel

Con los resultados de estas las 3 operaciones anteriores, es posible conformar el sistema de scrambling mejorado.

### 3.2.5 Sistema de scrambling mejorado

El esquema de cifrado scrambling mejorado es una de las principales aportaciones de [79]. Es la combinación de los 3 esquemas de cifrado anteriormente explicados y al aplicarlo a las imágenes se obtiene la fuente de información completamente distorsionada.

**Figura 3.18:** Rostro 3 cifrado mediante el esquema de cifrado scrambling mejorado para 2 rondas de operaciones.

## 3.3 Implementación de códigos polares usando esquemas caóticos de orden fraccionario.

La figura 3.19 relaciona y describre los procesos que forman parte del diseño experimental: el canal de memoria flash BAC_BBM, la codificación polar y el sistema de cifrado scrambling mejorado que incorpora los mapas caóticos fraccionarios. Este diagrama representa una versión más detallada del diseño experimental mostrado en la figura 3.1 al inicio de este capítulo.

Este sistema tiene 2 entradas: las imágenes de 512 X 512 píxeles en escala de grises (fuente de información) y la llave generada de forma aleatoria. La trayectoria de la fuente de información a través de los diferentes procesos del sistema, se representa mediante una línea azul gruesa. Las líneas más delgadas representan la aplicación directa o inversa de los algoritmos asociados con las secuencias caóticas. Inicialmente la fuente de información es convertida a una matriz de números entre 0 y 255. A la llave generada da forma aleatoria se le aplican los procedimientos referidos en la sección 3.2.1 para obtener la secuencia caótica.

La secuencia caótica se ordena mediante un proceso de división y escalamiento referido en la sección 3.2.1, y se obtiene una matriz de 512 X 512 de números enteros entre 0 y 255, la cual se utilizará en unión a la matriz de la fuente de información en el algoritmo scrambling_píxel. Los valores de la matriz de números obtenida se convierten a binario y se introducen en los algoritmos de scrambling_bit y baker_bit, junto a la matriz de la secuencia caótica convertida a binario. La matriz de la secuencia caótica convertida a binario también se utilizarán para los *frozen bits* de los códigos polares.

Los códigos polares se aplican a la secuencia binaria que proviene del cifrado donde se aplican los algoritmos de scramblig_bit y baker_bit. Esta secuencia es ordenada en vectores filas de 1536 bits de información para la codificación y tránsito por el canal. Los otros 512 bits necesarios para completar una secuencia de 2048 bits

para una tasa de código $R = k/N = 0.75$, son los $frozen\ bits$, los cuales no serán transmitidos en cero o vacíos como define la concepción de los códigos polares. En los $frozen\ bits$ se transmitirá parte la secuencia caótica convertida a binario con el fin de enmascarar la información que transita por el canal.

Después de transmitida toda la información a través del canal, para la decodificación, se tendrá presente la posición de los $frozen\ bits$, los cuales se establecerán en cero, pues como se explica en la sección 3.1.2, es determinante establecer en cero estas posiciones para decodificar. Al terminar la decodificación, los bits se colocarán en una matriz binaria con las dimensiones correctas para la extracción de las secuencias caóticas convertidas a binario mediante el proceso inverso a las operaciones de scrambling_bit y baker_bit. Al terminar estos procesos de decifrado, los valores de la matriz obtenida se convierten de binario a entero, teniendo como resultado de una matriz de números enteros de 512 X 512.A esta matriz se le aplica el proceso inverso a la operación scrambling_pixel para la extracción de las secuencias caóticas y la recuperación de la imagen.

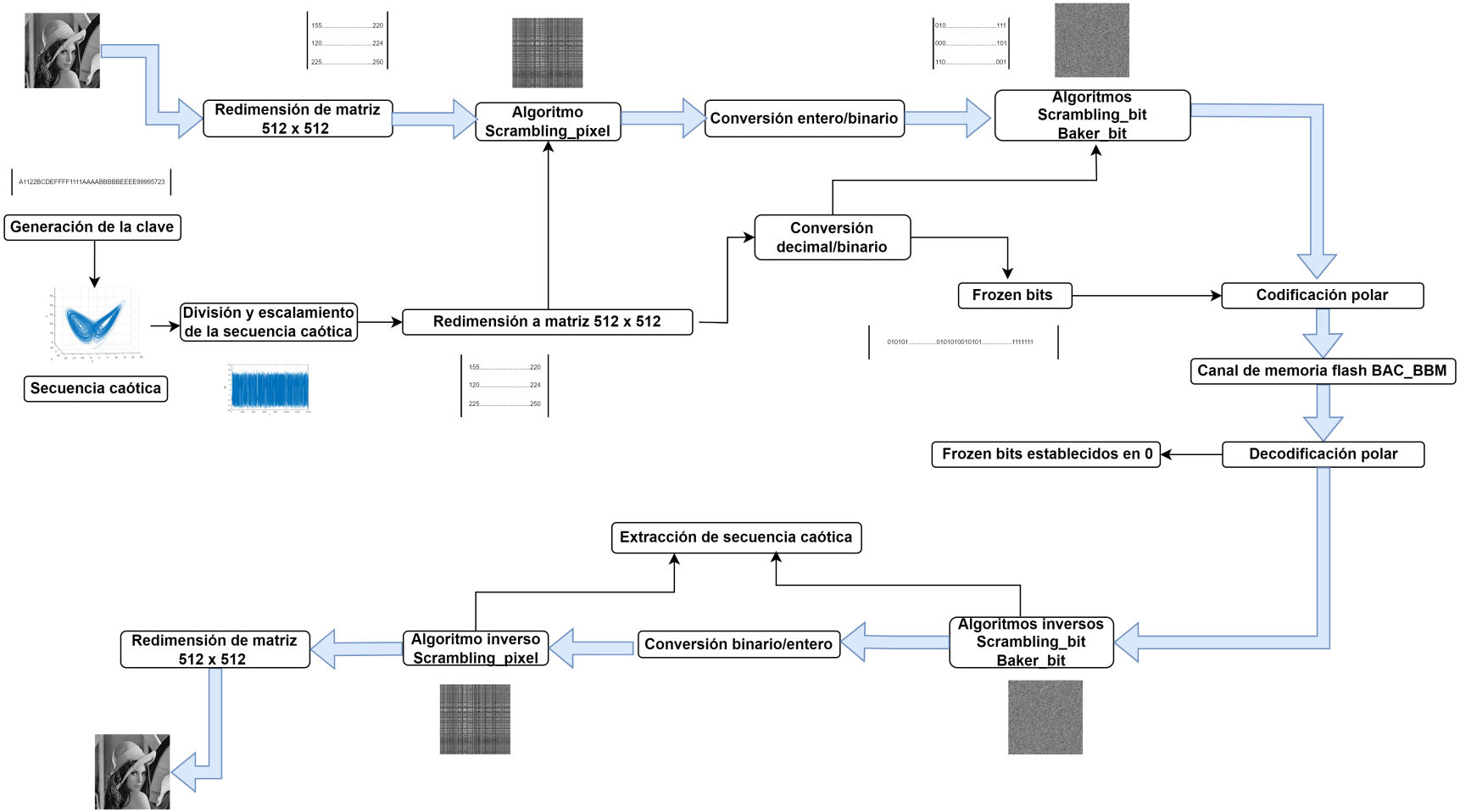



**Figura 3.19:** Diseño experimental experimental detallado.

En el próximo capítulo se evaluarán los elementos correspondientes al modelo canal BAC_BBM, el cifrado y la codificación. Se realiza una evaluación del sistema aplicando AWGN en el canal y por último se realiza una comparación entre códigos polares y códigos LDPC.

Capítulo 4

# Códigos polares, cifrado y canal de memoria flash NAND MLC

En los capítulos anteriores, se detallan las partes que conforman el sistema: el modelo de canal BAC_BBM, los códigos polares y el sistema de cifrado que utiliza el esquema caótico de orden fraccionario de Lorenz. En este capítulo se presentan los resultados de la evaluación individual de cada subsistema, así como la combinación de estos en un solo marco de trabajo.

Los experimentos en este capítulo se enfocan en comprobar la hipótesis, y resolver el problema de esta investigación. Se realiza también una comparación entre los códigos polares y los códigos LDPC para la misma fuente de información, se analiza la aplicación de AWGN al modelo de canal BAC_BBM y se evalúa la complejidad del sistema propuesto.

Para comprender con plenitud la evaluación de las partes del sistema, así como las diferentes pruebas realizadas, se tendrá como referencia la figura 4.1. Esta figura es la representación simplificada de las figuras 3.1 y 3.19 del Capítulo 3.

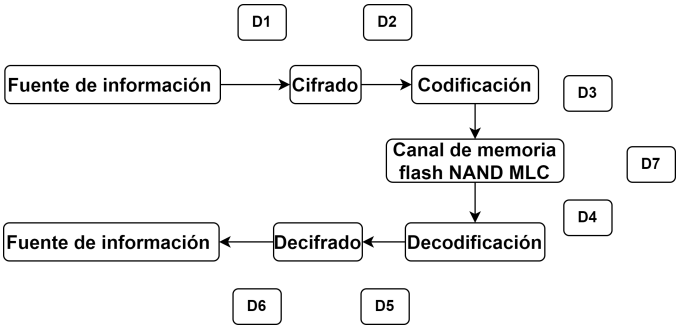



**Figura 4.1:** Diseño experimental con referencias para la evaluación del sistema.

En la figura 4.1, D1, D2, D3, D4, D5, D6 y D7 son las referencias utilizadas para la obtención de los parámetros a evaluar después de la aplicación de cada algoritmo. La referencia D7 representa al algoritmo del modelo de canal BAC_BBM más la señal de AWGN.

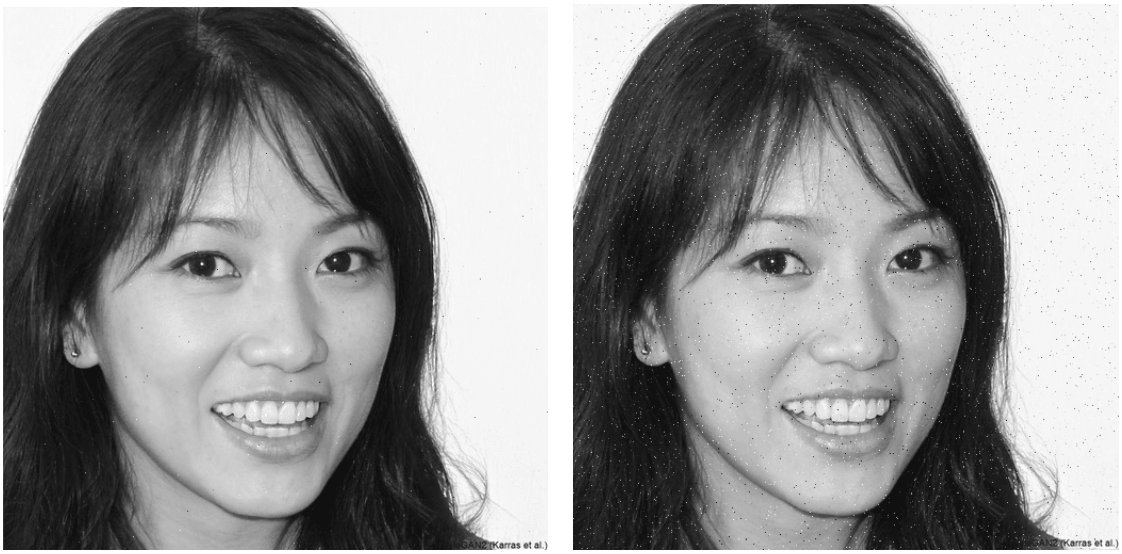
## 4.1 Evaluación del modelo de canal BAC_BBM

Como se describe al principio del Capítulo 2, un modelo de canal para memoria flash debe incorporar los elementos que causan errores en los datos almacenados. En este caso, la descripción experimental del modelo de canal BAC_BBM contempla los errores generados en el rango de 2000 a 10000 ciclos de programación/borrado.

En esta sección se comparan los datos de las referencias D3 y D4 de la figura 4.1. Para la representación de la información, las imágenes de 512 X 512 píxeles se convierten a un vector de 1 x 262144 y se aplican los algoritmos 1 y 2 para todos los parámetros de las tablas 2.1 y 2.2 de la sección 2.5.2. Los parámetros $a, b, c, d$ corresponden a la distribución Beta_Binomial y cada conjunto de valores coincide con un número específico de ciclos de programación/borrado.

La figura 4.2 muestra la influencia de los ciclos de programación/borrado en la información almacenada en una memoria flash NAND MLC. El impacto de los impedimentos del canal en estos dispositivos, puede verse en la corrupción de la información transmitida al aumentar los ciclos de programación/borrado. La conse-

cuencia directa de los errores del canal es la pérdida de información, lo que impone la necesidad del uso de códigos correctores de errores, los cuales son efectivos para mantener la integridad de la información, pero reducen la tasa de transferencia efectiva ya que utilizan parte de la información transmitida como redundancia.



**(a)** **(b)**

**Figura 4.2:** Rostro 2 evaluado en los algoritmos del modelo de canal BAC_BBM para 2000 ciclos de programación/borrado (a), y 10000 ciclos de programación/borrado (b).

La figura 4.3 muestra la probabilidad de error en función de los ciclos de programación/borrado para los MSB y los LSB del modelo de canal BAC_BBM. Se puede apreciar el crecimiento de la probabilidad de error con el aumento de los ciclos de programación/borrado, siendo los LSB los de mayor probabilidad. Para la obtención de la probabilidad de error, se utiliza un algoritmo que compara el vector de la imagen de tamaño 1 x 262144 antes y después de la ejecución de los algoritmos del modelo matemático del canal BAC_BBM. La comparación se utiliza para hacer un conteo de los errores o diferencias entre estos vectores. El resultado se divide entre la longitud del vector de la imagen, en este caso 252144. La probabilidad de error se calcula para cada uno de los valores que refiere a los ciclos de programación/borrado como:

$$P_e = Número\ de\ errores/262144 \tag{4.1}$$

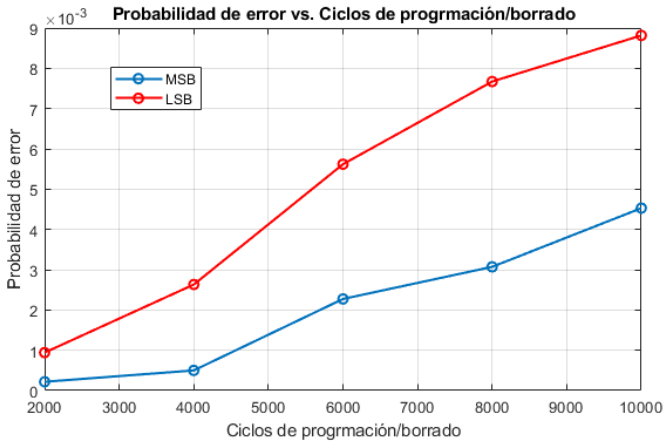



**Figura 4.3:** Probabilidad de error vs. Ciclos de programación/borrado para el canal BAC_BBM.

En este caso, la probabilidad de error para 10000 ciclos de programación/borrado es el peor de los casos. Por ello, la mayoría de las pruebas mostradas en este capítulo están realizadas para 10000 ciclos de programación/borrado.

## 4.2 Capacidad del canal BAC_BBM

En función de obtener una medida de la cantidad de información que se puede transmitir con fiabilidad a través de este canal para la fuente de información escogida, se calcula la capacidad de canal. Al inicio del Capítulo 3, se explica que las imágenes escogidas para esta investigación son de 512 X 512 píxeles y formato .jpeg. Cada píxel es un número entre 0 y 255, lo que facilita la conversión a binario. La capacidad de canal se obtiene introduciendo todas las combinaciones posibles los algoritmos 1 y 2 del Capítulo 2. El proceso se repite 5000 veces para provocar que el sistema devuelva errores suficientes, en función de obtener las matrices de transición para los LSB y MSB.

Las matrices obtenidas son matrices estables ya que el valor absoluto de cada elemento en la diagonal principal es mayor o igual a la suma de los valores absolutos de los elementos en su respectiva fila. Con los valores obtenidos se calcula la capacidad de canal como:

$$C = I_{max}\left(X;Y\right) = H\left(X\right) - H\left(X|\ Y\right), \tag{4.2}$$

donde:

$I_{max}(X;Y)$ es la información mutua.

$H\left(X\right) = -\sum_x\ p(x)logp(x)$ es la entropía de entrada.

$H\left(X|Y\right) = \sum_x \sum_y\ p(x,y)log_2\frac{1}{p(x\backslash y)}$ es la entropá condicional.

La figura 4.4 muestra la capacidad de canal en función de los ciclos de programación/borrado.

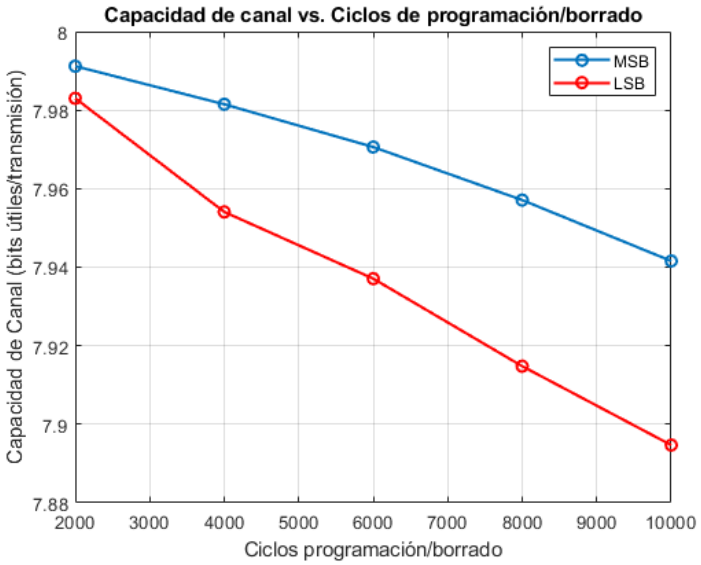



**Figura 4.4:** Capacidad de canal vs. Ciclos de programación/borrado.

La figura 4.4 muestra que con el aumento de los ciclos de programación/borrado, LSB tendrán menos bits útiles en la transmisión que los MSB, no obstante, los valores obtenidos para la capacidad de canal son cercanos a la cantidad de bits necesarios para representar la fuente de información, por lo cual este canal no necesita alta redundancia para la transmisión de la información. Por ello, la tasa de código utilizada en el deseño experimental que se presenta es $R\ = 0.75$.

## 4.3 Corrección de errores en el canal BAC_BBM

En el Capítulo 3 se explica el funcionamiento de los códigos polares en el sistema propuesto. A continuación, se muestran los resultados obtenidos luego de aplicar a las imágenes los algoritmos del modelo de canal BAC_BBM y el algoritmo que corresponde a los códigos polares.

La figura 4.5, muestra el rostro 2 luego de ser codificado, evaluado por el modelo de canal BAC_BBM y decodificado. La codificación polar se implementa con una tasa de código $R = 0.75$ y épsilon $\epsilon = 0.5$ y datos de $a, b, c, d$ para 10000 ciclos de programación/borrado.

Para obtener la diferencia entre el rostro 2 y la imagen procesada, se comparan las matrices de 512 X 512 píxeles correspondientes a la referencia D2 y la referencia D5 de la figura 4.1, sin aplicar el algoritmo de la referencia D7, luego se aplica la fórmula para la probabilidad de error expuesta en la sección 4.1. La imagen procesada tiene 50 píxeles diferentes con respecto a la imagen original del rostro 2. Esta diferencia equivale a una probabilidad de error en píxel de $1.907 \times 10^{-4}$. Los códigos polares aportan robustez al sistema ya que son capaces de mejorar la probabilidad de error del canal BAC_BBM para 10000 ciclos de programación/borrado.

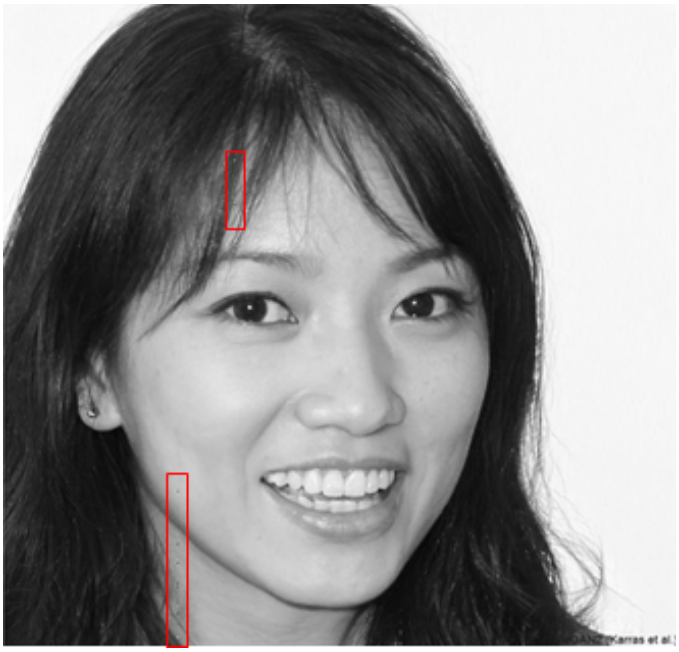


**Figura 4.5:** Rostro 2 codificado, evaluado por el algoritmo del modelo de canal BAC_BBM y decodificado. En rojo se representan los errores obtenidos.

## 4.4 Canal BAC_BBM y scrambling mejorado

La figura 4.6 muestra la aplicaciń del sistema de cifrado scrambling mejorado en el rostro 2. En este caso, a la fuente de informaciń, primero se aplican las operaciones de cifrado scrambling_pixel, scrambling_bit y baker_bit. Luego se emplean los algoritmos del modelo de canal BAC_BBM y por último se descifra aplicando las operaciones inversas de los algoritmos de cifrado. En este caso, se comparan las referencias D1 y D6, de la figura 40, sin aplicar los procesos de codificación y decodificación. La imagen obtenida difiere de la imagen original en 117350 píxeles. Esta diferencia equivale a una probabilidad de error de 0.4. Sin la aplicación de la codificación polar, el sistema de cifrado junto al algoritmo al modelo de canal BAC_BBM degradan completamente la fuente de información.

**Figura 4.6:** Rostro 2 cifrado por el algoritmo de scrambling mejorado, evaluado por modelo de canal BAC_BBM y descifrado por las operaciones inversas del sistema scrambling mejorado.

## 4.5 Códigos polares y el sistema de cifrado scrambling mejorado.

La figura 4.7 muestra el resultado de aplicar códigos polares al sistema de scramblig mejorado sin tener en cuenta los algoritmos referentes al canal BAC_BBM. La codificación polar se implementa con una tasa de código $R = 0.75$ y épsilon $\epsilon = 0.5$. Para obtener la diferencia entre el rostro 2 y la imagen procesada se comparan las matrices de 512 X 512 píxeles correspondientes a las referencias D1 y D6 de la figura 4.1 sin aplicar los procesos D3, D4 o D7 referentes a los algoritmos del canal. La imagen obtenida es igual a la imagen del rostro 2.


**Figura 4.7:** Rostro 2 cifrado por el algoritmo de scrambling mejorado, codificado, decodificado y descifrado por las operaciones inversas del sistema scrambling mejorado.

## 4.6 Códigos polares y el sistema de cifrado scrambling mejorado para un canal de memoria flash NAND MLC

La figura 4.8 muestra el resultado de implementar códigos polares y el sistema de cifrado scrambling mejorado para un canal de memoria flash NAND MLC. La codificación polar se implementa con una tasa de código $R = 0.75$ y épsilon $\epsilon = 0.5$ y los datos de $a, b, c, d$ para 10000 ciclos de programación/borrado. Para obtener la diferencia entre el rostro 2 y la imagen procesada se comparan las matrices de 512 X 512 píxeles correspondientes a las referencias D1 y D6 sin tener en cuenta la referencia D7. La imagen obtenida difiere de la imagen original del rostro 2 en 299 píxeles. Esta diferencia equivale a una probabilidad de error de $1.14 \times 10^{-3}$. Este resultado esta en el rango admisible de probabilidad de error de un según [2], y menor que los valores obtenidos en la evaluación de la misma imagen en el canal

BAC_BBM para 10000 ciclos de programación/borrado.


**Figura 4.8:** Códigos polares y sistema de cifrado que utiliza esquemas caóticos de orden fraccionario para un canal de memoria flash NAND MLC.

Con esta prueba se verifica la hipótesis propuesta: los códigos polares aplicados a un modelo de canal BAC_BBM, unido a un sistema de cifrado basado en esquemas caóticos de orden fraccionario pueden ser utilizado para mejorar la vida útil de las memorias flash NAND MLC. El resultado de esta prueba presenta menor probabilidad de error que las pruebas referentes al modelo de canal BAC_BBM para 10000 ciclos de programación/borrado de la figura 4.1. Con esta prueba se resuelve también el problema de investigación. La tabla 4.1, a continuación, resume la probabilidad de error para cada una de las pruebas anteriores.

| **Experimento** | **Probabilidad de error para 10000 ciclos de programación/borrado** |
|---|---|
| Canal BAC_BBM | $4.035 \times 10^{-2}$ |
| Códigos polares + Canal BAC_BBM | $1.907 \times 10^{-4}$ |
| Cifrado + Canal BAC_BBM | $4.477 \times 10^{-1}$ |
| Cifrado + Códigos polares + Canal BAC_BBM | $1.141 \times 10^{-3}$ |

**Tabla 4.1:** Probabilidad de error para 10000 ciclos de programación/borrado.

## 4.7 Señal a Ruido en el Canal BAC_BBM

La razón Señal a Ruido (SNR) es una medida de la información útil transmitida. Para hacer la evaluación de este parámetro se agrega Ruido Blanco Gaussiano Aditivo (AWGN) en el algoritmo del canal BAC_BBM para los valores de $a, b, c, d$ correspondientes a 10000 ciclos de programación/borrado. El AWGN puede considerarse la representación de otros de los impedimentos que afectan al canal de memoria flash NAND MLC como la temperatura o la pérdida de retención a lo largo del tiempo, los cuales no son considerados en la descripción experimental del modelo de canal BAC_BBM.

La figura 4.9 muestra la aplicación del AWGN en el diseño experimental. La señal de AWGN se agrega al vector $u$ del algoritmo 1 de la sección 2.5.2. La función de AWGN supone que la potencia de la señal $u$ es $1W$ $(0\ dBW)$. Para el cálculo de la probabilidad de error se utiliza la ecuación, y se realizan 2 operaciones. Ambas operaciones utilizan una tasa de código $R = 0.9378$ y épsilon $\epsilon = 0.75$.

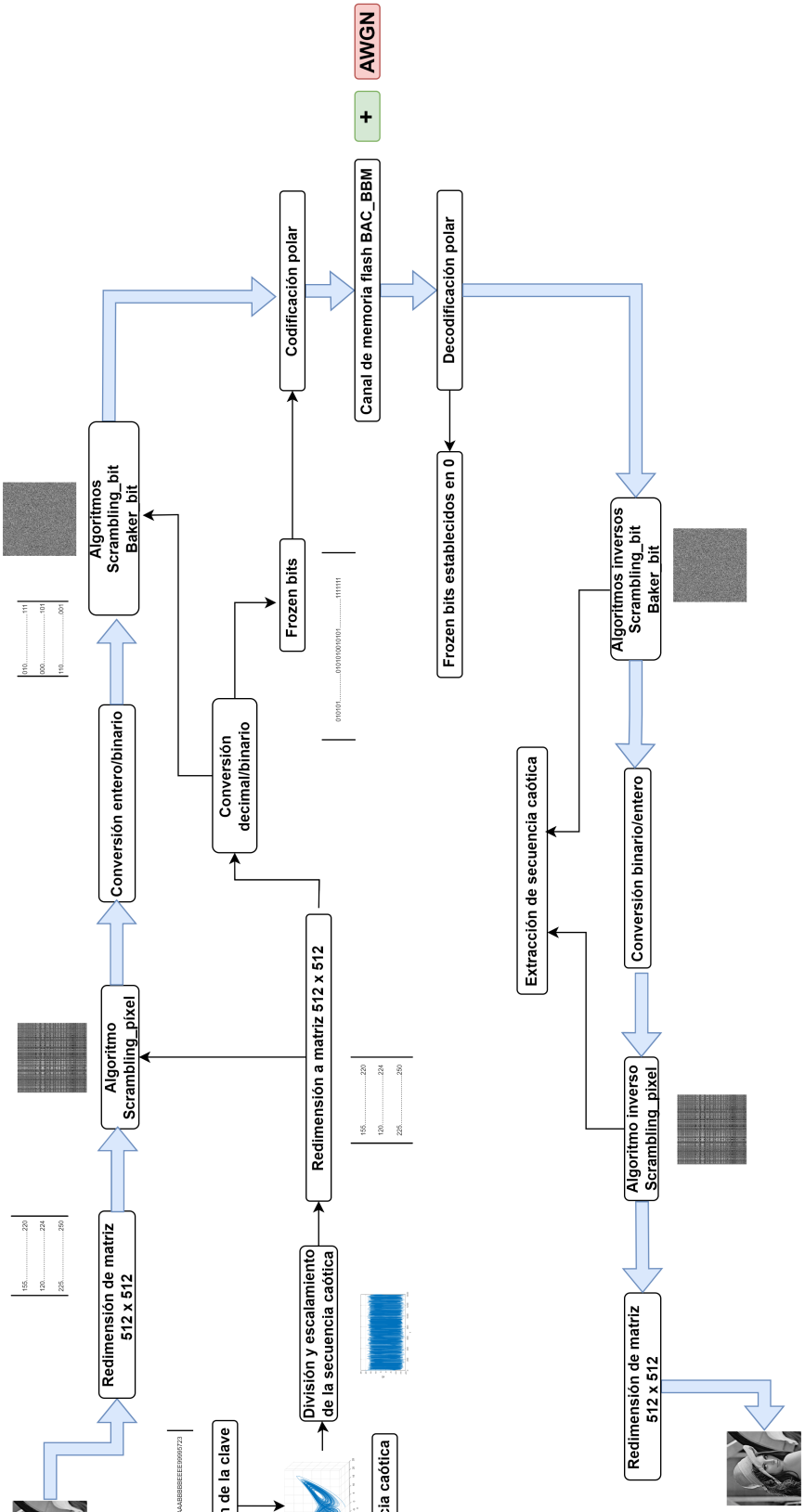



**Figura 4.9:** Aplicación de AWGN al diseño experimental.

La primera operación compara la secuencia de bits de la fuente de información codificada antes de ser introducida al canal, con la secuencia de bits después de ser evaluada en el algoritmo del modelo de canal BAC_BBM que suma AWGN, mientras varía la SNR de $-100\ dBW$ a $100\ dBW$. En este caso se comparan las referencias D3 y D7 de la figura 4.1. El resultado de esta operación se muestra en la figura 4.10 mediante la curva azul.

La segunda operación compara la secuencia de bits de la fuente de información codificada antes de ser introducida al canal con la secuencia de bits después de ser aplicado el algoritmo del modelo de canal BAC_BBM que suma AWGN decodificada, mientras varía la SNR de $-100\ dBW$ a $100\ dBW$. En este caso se comparan las referencias D2 y D7 decodificada de la figura 4.1. El resultado de esta operación se muestra en la figura 4.10 mediante la curva roja.

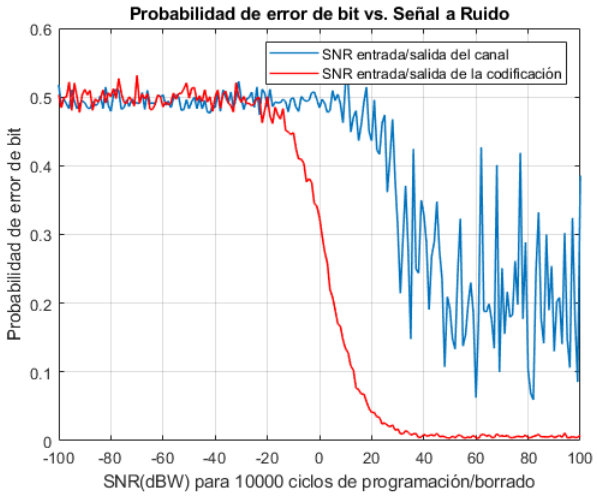


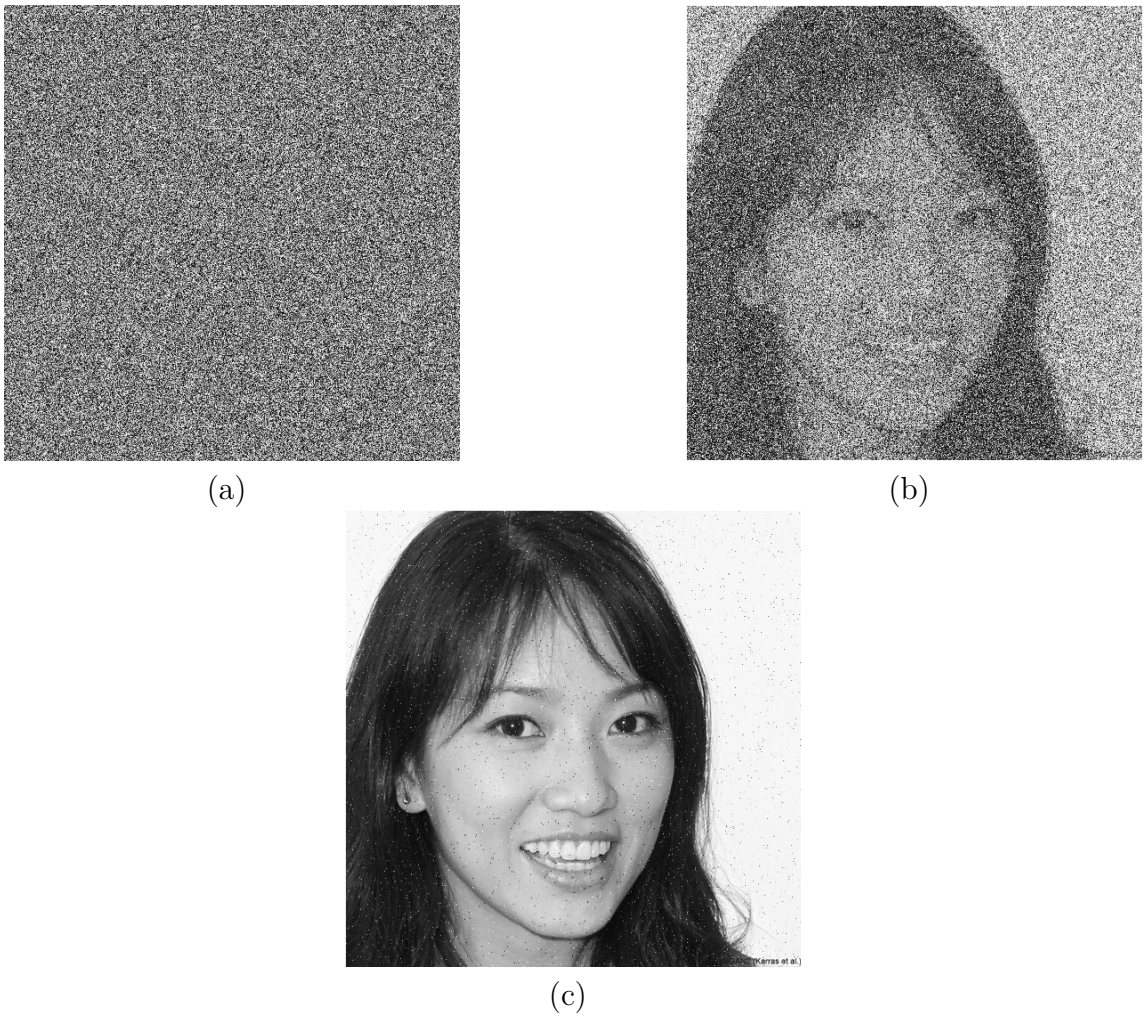

**Figura 4.10:** Probabilidad de error vs. Señal a Ruido para 10000 ciclos de programación/borrado.

Para verificar el comportamiento de la probabilidad de error para 10000 ciclos de programación/borrado y confirmar que disminuye con el aumento de la razón Señal a Ruido se realizan las siguientes pruebas: primero se calcula la probabilidad de error para el canal BA_BBM con AWGN. Luego se calcula la probabilidad de error del canal BAC_BBM con AWGN, utilizando códigos polares para la corrección de errores. Por último, se calcula la probabilidad de error utilizando la información

cifrada en canal BAC_BBM con AWGN, utilizando códigos polares para la corrección de errores.

En estos casos, se utiliza como referencia la figura 4.1, pero se sustituye la referencia D4 por la D7. En la primera prueba se comparan las referencias D3 y D7, en la segunda prueba se comparan las referencias D2 y D5, y en la tercera prueba se comparan las referencias D1 y D6. Las figuras a continuación muestran las pruebas se realizadas para $-100\ dBW$, $0\ dBW$ y $100\ dBW$.

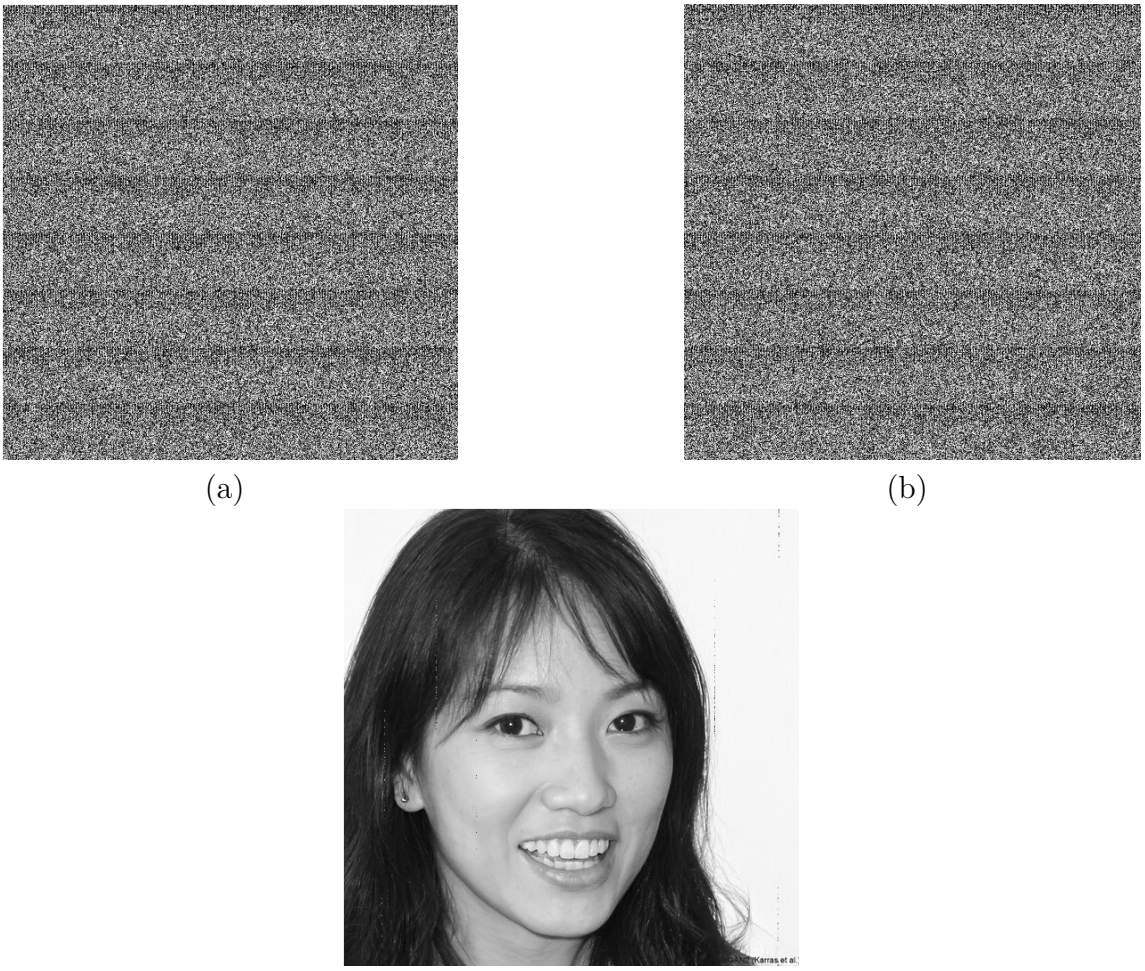
La figura 4.11 muestra el rostro 2 luego de ser evaluado en el canal BAC_BBM con AWGN. Como se puede observar, el AWGN deteriora la fuente de información, sin embargo, la imagen mejora con el aumento de la razón Señal a Ruido.



(a) (b) (c)

**Figura 4.11:** Rostro 2 evaluado en los algoritmos el modelo de canal BAC_BBM con AWGN, para $-100\ dBW$ (a), $0\ dBW$ (b) y $100\ dBW$ (c) para 10000 ciclos de programación/borrado.

La figura 4.12 muestra el rostro 2 evaluado en los algoritmos el modelo de canal BAC_BBM con AWGN. En esta prueba se aplica corrección de errores para eliminar los ruidos provenientes del canal. Los resultados coinciden con la información de la gráfica de la figura 4.10, a pesar de no utilizarse los mismos valores de $R$ y $\epsilon$ . Si la Señal a Ruido es cercana a 100 $dBW$ en el canal BAC_BBM con AWGN y corrección de errores, la probabilidad de error es mucho menor.



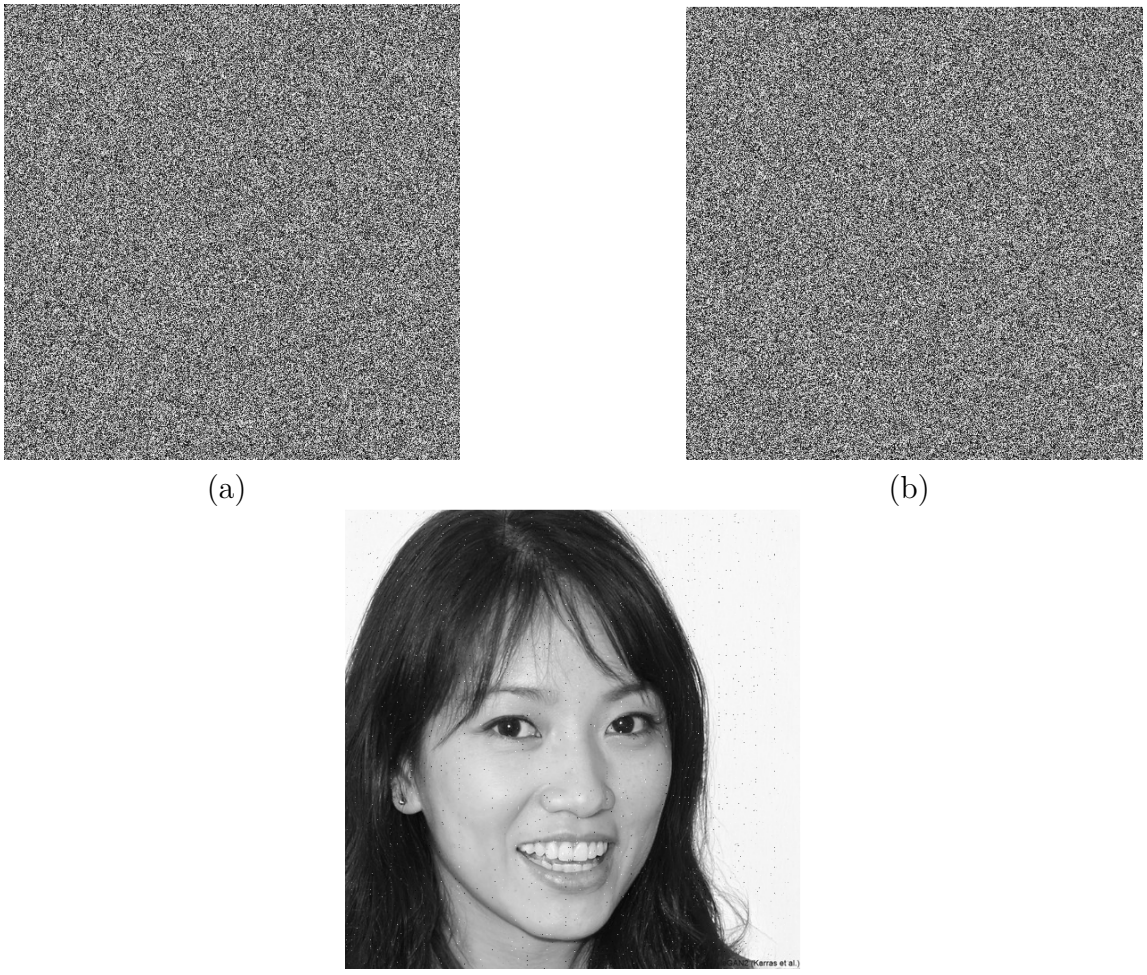
(a) (b)

(c)

**Figura 4.12:** Rostro 2 evaluado en los algoritmos el modelo de canal BAC_BBM con AWGN y corrección de errores, para $-100$ $dBW$ (a), 0 $dBW$ (b) y 100 $dBW$ (c) para 10000 ciclos de programación/borrado.

La figura 4.13 muestra el rostro 2 cifrado, evaluado en los algoritmos el modelo de canal BAC_BBM con AWGN. Al igual que en la prueba anterior, se utiliza corrección de errores. En este caso la probabilidad de error es mayor a la obtenida en la prueba anterior, pues el sistema de cifrado al igual que el AWGN reducen la

robustez de la fuente de información.



(a) (b)

(c)

**Figura 4.13:** Rostro 2 cifrado, evaluado en los algoritmos el modelo de canal BAC_BBM con AWGN y corrección de errores, para $-100$ $dBW$ (a), 0 $dBW$ (b) y 100 $dBW$ (c) para 10000 ciclos de programación/borrado.

| **Experimento** | **Probabilidad de error** $-100$ **dBW** | **Probabilidad de error** 0 **dBW** | **Probabilidad de error** 100 **dBW** |
|---|---|---|---|
| Canal BAC_BBM + AWGN | $9.961 \times 10^{-1}$ | $9.530 \times 10^{-1}$ | $4.609 \times 10^{-2}$ |
| Canal BAC_BBM + AWGN + códigos polares | $9.962 \times 10^{-1}$ | $9.958 \times 10^{-1}$ | $1.072 \times 10^{-3}$ |
| Cifrado + Canal BAC_BBM + AWGN+ códigos polares | $9.961 \times 10^{-1}$ | $9.958 \times 10^{-1}$ | $1.514 \times 10^{-3}$ |

**Tabla 4.2:** Probabilidad de error para diferentes valores de razón de Señal a Ruido.

En este punto es necesario mencionar algunos elementos. El Capítulo 3 explica la procedencia comercial de las imágenes de esta investigación. Si estas fuesen imáge-

nes médicas la percepción de los errores obtenidos son graves ya que puede existir riesgo de diagnósticos incorrectos, dado que la falta de claridad o distorsión posibilita la generación de interpretaciones erróneas. Por ello, las imágenes de tomografías o resonancias magnéticas se estudian en vóxeles en vez de píxeles. La palabra *vóxel* es una contracción de las palabras inglesas *volumetric* y *pixel*. Los vóxeles permiten una representación detallada de la estructura interna del cuerpo en tres dimensiones, facilitando diagnósticos más precisos y planificación de tratamientos [83].

Asimismo, si se estuviese trabajando con imágenes meteorológicas la percepción de los errores obtenidos puede implicar pronósticos incorrectos, una mala planificación operativa, o una errónea gestión de desastre. Por ello, la obtención de los píxeles que componen este tipo de imágenes se hace a través de sensores especializados utilizados para detectar la radiación infrarroja. Estos dispositivos miden los cambios en la resistencia eléctrica que ocurren como resultado de los cambios en la temperatura. Estos cambios son procesados para crear una imagen que muestra un mapa de temperatura de la escena observada, por lo que el proceso de creación de estas imágenes es más complejo para evitar errores y tiene además una constante actualización. Para disminuir la presencia de errores en imágenes meteorológicas, actualmente se utilizan técnicas de Big Data, las cuales hacen coincidir datos obtenidos con imágenes anteriores, referencias históricas y estadística del fenómeno analizado antes de exponer una imagen definitiva [84].

Aunque cada tecnología tiene sus especificaciones definidas para la creación, manejo y restauración de las imágenes, existen técnicas para mejorar la recuperación de imágenes comerciales utilizando el conocimiento a priori del fenómeno de degradación. Estas técnicas están orientadas hacia la modelización de la degradación y la aplicación del proceso inverso para recuperar la imagen original. La restauración se puede realizar en presencia de ruido mediante filtros espaciales, los cuales utilizan la información estadística. Otras técnicas de restauración de imágenes utilizan la convolución ya que la función que describe a la imagen degradada más el ruido en el dominio de la frecuencia puede ser expresada como el producto de las transformadas

de la imagen y de la degradación, seguido por la adición de la transformada del ruido. Por otro lado, también se utilizan filtros inversos los cuales intentan recuperar la imagen original aplicando la operación inversa a la degradación sufrida o filtros de error cuadrático los cuales intentan encontrar el error entre la imagen restaurada y la imagen original [85].

Igualmente, existen metodologías que proponen la utilización de otras técnicas como Deep Learning para la restauración de imágenes comerciales. Estas técnicas utilizan múltiples fotos dañadas como entrada, las cuales se utilizan como entrenamiento para construir un modelo de reconocimiento para la información faltante. Entre los métodos más investigados se encuentra el método *image inpainting*, el cual introduce una simplificación de procesos y una mejora de las herramientas para el procesamiento de imágenes [86].

Otro aspecto a valorar en la obtención de los resultados de estas pruebas es la compresión del estándar .jpg. Si las imágenes utilizadas fueran en formato .bmp la probabilidad de error obtenida sería menor ya que en los procesos de cifrado se pierde la correlación de la imagen por lo que puede considerarse como un tipo de ataque estadístico. El formato .bmp, a diferencia del formato .jpg tiene mejor resistencia a ataques estadísticos y mayor robustez contra manipulación [87].

Aunque las imágenes obtenidas de las pruebas realizadas presentan errores, son válidas para un sistema de reconocimiento facial, ya que estos sistemas basan su funcionamiento en la geometrá del rosto.

Para la identificación de una persona en los sistemas de reconociendo facial utilizan algoritmos que detectan medidas del rostro humano como la separación entre los ojos, la hondura de las órbitas oculares, la longitud desde la frente hasta la barbilla, la estructura de los pómulos y el perfil de los labios, las orejas y el mentón [88].

## 4.8 Comparación entre los códigos polares y códigos LDPC para el canal BAC_BBM

Uno de los objetivos de esta investigación es la comparación entre los códigos LDPC y códigos polares. En el Capítulo 3, se define la relación de complejidad existente entre ambas codificaciones. La figura 4.14 muestra las curvas de probabilidad de error de bit promedio en funcińd de los ciclos de programación/borrado de ambas codificaciones, entre 2000 y 10000 ciclos para el canal BAC_BBM.

La comparación entre los códigos LDPC y los códigos polares para el modelo de canal BAC_BBM, se realiza con una longitud de código de 1024 bits para los códigos polares, divididos en 128 paquetes de 8 bits y Para los códigos LDPC la longitud de código es de 1360 bits divididos en 170 paquetes de 8 bits. Ambas codificaciones se implementan con una tasa de error del 30 % y la misma fuente de información. Se realizan 100 iteraciones para el cálculo de la probabilidad de error promedio de bit en el modelo de canal BAC_BBM considerando y todos los ciclos de programación/borrado de las tablas 2.1 y 2.2 de la sección 2.5.2.

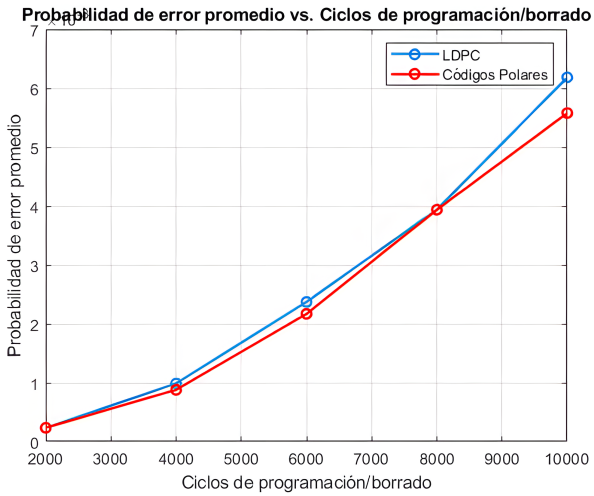



**Figura 4.14:** Comparación de la probabilidad de error promedio vs. Ciclos de programación/borrado para LDPC y códigos polares.

Para los códigos LDPC (curva azul), se obtiene una probabilidad de error

promedio de $2.33 \times 10^{-4}$ después de 2000 ciclos de programación/borrado. En comparación, los códigos polares (curva roja), muestran una probabilidad de error de bit de $2.34 \times 10^{-4}$. Al aumentar el número de ciclos de programación/borrado a 10000, los códigos LDPC presentan una probabilidad de error de bit de $6.2 \times 10^{-3}$, mientras que los códigos polares muestran este parmetro en un valor de $5.6 \times 10^{-3}$.Ambas codificaciones experimentan un aumento en la probabilidad de error de bit a medida que se incrementan los ciclos de programación/borrado.

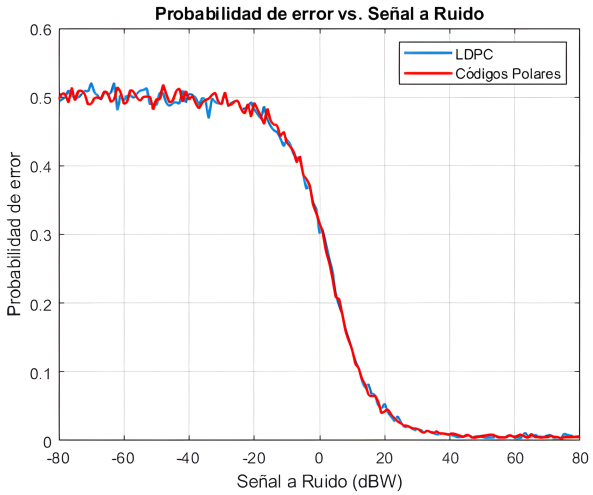



**Figura 4.15:** Comparación de la probabilidad de error promedio vs. Señal a Ruido para LDPC y códigos polares.

En la figura 4.15, los resultados indican que en el contexto específico del modelo de canal BAC_BBM, tanto los códigos LDPC como los códigos polares muestran rendimientos similares en correspondencia con la Señal a Ruido, con ligeras variaciones dependiendo de los parámetros específicos de la función Beta_Binomial y la cantidad de ciclos de programación/borrado considerados. Al analizar la probabilidad de error promedio para 2000 ciclos de programación/borrado, se observa una mínima disparidad entre ambas codificaciones. Sin embargo, al ampliar la evaluación a 10000 ciclos de programación/borrado, se nota una disminución en el valor para los códigos polares. Los resultados de la evaluación de la capacidad del canal y la probabilidad de error promedio, concuerdan con investigaciones que resaltan la superioridad de los códigos polares en implementaciones que manejan mensajes cortos.

## 4.9 Evaluación de la complejidad del sistema

Para la evaluación de la complejidad del sistema se medirá el tiempo de ejecución en Matlab R2021 de una cada de las pruebas principales realizadas en esta investigación, y descritas en este capítulo en una computadora con un procesador Intel(R) Core (TM) i3-5005U CPU @ 2.00GHz, 12 GB de memoria RAM DDR3 y SSD de 512GByte. La tabla 4.3 muestra los tiempos de ejecución.

| **Experimento** | **Tiempo de ejecución** |
|---|---|
| Canal BAC_BBM | 43.161 s |
| Canal BAC_BBM + códigos polares | 189.891 s |
| Canal BAC_BBM + cifrado | 201.932 s |
| Cifrado + Canal BAC_BBM + códigos polares | 397.631 s |
| Canal BAC_BBM + AWGN | 120.031 s |
| Canal BAC_BBM + AWGN + códigos polares | 312.763 s |
| Cifrado + Canal BAC_BBM + AWGN+ códigos polares | 583.839 s |

**Tabla 4.3:** Tiempos de ejecución.

Como se puede apreciar en la tabla 4.3, los tiempos de ejecución de los algoritmos que incorporan el cifrado o el AWGN son mayores que el resto de los algoritmos, por lo que se asume que tiene mayor complejidad. La generaciń de AWGN es aleatoria, por lo que se tienen pocas maneras de predecir su comportamiento. El proceso de cifrado caótico al igual que el AWGN genera secuencias que son difÃciles de predecir y replicar sin conocer las condiciones exactas iniciales y parḿetros del sistema, sin embargo, el proceso de cifrado se puede acondicionar para acotar el tiempo de ejecución.

Como se explica en el Capítulo 3, esta investigación utiliza cifrado simétrico por lo que necesitamos la misma secuencia para cifrar y descifrar. Por otro lado, la creación de la señal caótica depende del algoritmo que genera la llave. Lo ideal es que

cada vez que se cifre la información se generen estas secuencias, las cuales pueden ser guardadas en un sistema en la nube en lo que dure la ejecución de los algoritmos y así disminuir el tiempo, ya que se generarán una sola vez. Otra vía de disminuir la complejidad del sistema es trabajar con el HASH de las secuencias ya que estos algoritmos producen un mensaje de salida de longitud fija sin importar la longitud de la entrada. La desventaja de esta propuesta es que habría que incorporar otro algoritmo al sistema.

Por los tiempos de ejecución se considera que los sistemas implementados no son lo suficientemente simples para que su ejecución se realice en menor tiempo en una arquitectura ONFI. Como se muestra en la tabla 4.3, el algoritmo de cifrado es uno de los que más complica la ejecución del sistema. En el Capítulo 3, se hace referencia la relación de complejidad entre el código LDPC y el código polar, siendo este último de menor complejidad. Por otro lado, existe la literatura científica evidencia que un código LDPC puede ser ejecutado en la arquitectura ONFI, por lo que se considera un código polar sin otro algoritmo que haga compleja se ejecución, puede ser ejecutado sin grandes complicaciones en la interfaz ONFI.

Capítulo 5

# Conclusiones, contribución y trabajos a futuro

En este apartado final, se presentan las conclusiones derivadas de la investigación realizada en relación a la implementación de códigos polares usando un sistema de cifrado que utiliza esquemas caóticos fraccionarios para un canal de memoria flash NAND MLC.

Teniendo en cuenta la investigación realizada, se obtienen:

1. Los algoritmos que permiten la combinación de los códigos polares y con un sistema de cifrado que utiliza esquemas caóticos de orden fraccionario variable para un canal de memoria flash NAND MLC.

2. La probabilidad de error en función de los ciclos de programación/borrado para el modelo de canal BAC_BBM. La información obtenida describe el comportamiento de los LSB y los MSB entre 2000 y 10000 ciclos de programación/borrado.

3. Una aproximación de la capacidad del canal BAC_BBM utilizando imágenes de 512 X 512 como fuente de información.

4. La probabilidad de error en función de la señal a ruido del sistema en el rango de 100 dBW a - 100 dBW para 10000 ciclos de programación/borrado en diferentes escenarios.

5. La comparación de la probabilidad de error entre 2000 y 10000 de programación/borrado de los códigos polares y los códigos LDPC para el canal BAC_BBM.

6. La evaluación de la implementación del sistema en una arquitectura ONFI utilizando como referencia la relación de complejidad entre códigos polares y los códigos LDPC y el retardo en la ejecución.

Los resultados del presente estudio corroboran los objetivos propuestos y validan la hipótesis planteada, por tanto, se puede concluir que la implementación de códigos polares usando esquemas de orden fraccionario es una solución válida para mejorar la durabilidad de las memorias flash NAND MLC.

Las principales constribuciones de este trabajo de tesis son:

- La implementación de los códigos polares como sistema corrector de errores para un canal BAC_BBM para mejorar la durabilidad de las memorias flash NAND MLC.
- La utilización de un sistema de cifrado que utiliza esquemas caóticos fraccionarios para la protección de la información codificada en un canal de memoria flash NAND MLC.

Se publicó, además, en la revista de divulgación Ingenierías editada por la Universidad Autónoma de Nuevo León, a través de la Facultad de Ingeniería Mecánica y Eléctrica el artículo “Comparación de códigos correctores de errores LDPC y códigos polares en un canal de memoria flash NAND MLC.”

- Implementación en hardware del algoritmo de códigos polares usando esquemas caóticos de orden fraccionario.
- Implementación de un algoritmo que utilice códigos polares usando esquemas caóticos de orden fraccionario variable para incrementar la durabilidad de las memorias flash NAND MLC.

- Implementación de un algoritmo que utilice códigos polares usando esquemas caóticos de orden fraccionario discreto para incrementar la durabilidad de las memorias flash NAND MLC.

- Implementación de un algoritmo que utilice códigos polares usando esquemas caóticos de orden fraccionario discreto variable para incrementar la durabilidad de las memorias flash NAND MLC

- Estudio comparativo en cuanto a complejidad y probabilidad de errores en la implementación de los códigos polares de todas las variantes anteriores.

- Optimización del tiempo computacional para la simulación de los sistemas caóticos de orden fraccionario.

- Aplicar el diseño de codificación/cifrado propuesto a otras fuentes de información.

# Bibliografía


[1] K. Ishimaru, M. Fujiwara, H. Miyagawa, and Y. Aiba, "Flash memory and its manufacturing technology for sustainable world," *IEEE Journal of the Electron Devices Society*, vol. 10, pp. 737–743, 2021.

[2] Y. Luo, "Architectural techniques for improving nand flash memory reliability," 2018. [Online]. Available: http://reports-archive.adm.cs.cmu.edu/anon/2018/CMU-CS-18-101.pdf

[3] "Guía de memoria flash." [Online]. Available: https://media.kingston.com/pdfs/MKF_283.2_Flash_Memory_Guide_ES.pdf

[4] "Flash memory market: Global industry analysis and forecast (2024-2030)." [Online]. Available: https://www.maximizemarketresearch.com/market-report/global-flash-memory-market/54019/?utm_source=whatech&utm_medium=refferal&utm_campaign=shorturl&utm_content=whatech-com-832907

[5] "Flash memory market is expected to grow at a cagr of 5.1 % from 2024 to 2030 as revealed in new report." [Online]. Available: https://www.whatech.com/og/markets-research/energy/832907%flash%memory%market%is%expected%togrow-at-a%cagr%of%5%1%from%2024%to%2030%as%revealed%in%new%repor

[6] "Diferencias entre slc, mlc, tlc y 3d nand en dispositivos flash usb, ssds y tarjetas de memoria." [Online]. Available: https://www.kingston.com/latam/blog/pc-performance/difference-between-slc-mlc-tlc-3d-nand

[7] "The inner workings of solid state flash:slc versus mlc." [Online]. Available: https://www.techbriefs.com/component/content/article/tb/supplements/et/features/articles/6610

[8] S. Green, "Flash ssd endurance and reliability: Four influential factors," 2023. [Online]. Available: https://www.electronicdesign.com/technologies/embedded/article/21267454/phison-electronics-flashmemory-ssd-endurance-and-reliability-four-influential-factors

[9] "Celda de memoria." [Online]. Available: https://es.wikipedia.org/wiki/Celda_de_memoria

[10] S. Karmakar, "Quantum dot gate non-volatile memory as single level cell (slc), multi-level cell (mlc) and triple level cell (tlc)," 2018.

[11] H. Giménez, "Diferencias entre las memorias flash ssd nand: Qlc / slc / mlc / tlc," 2020. [Online]. Available: https://www.reneelab.es/diferencias-entre-qlc-slc-mlc-tlc.html

[12] "Mlc nand flash memory." [Online]. Available: https://www.micron.com/products/nand-flash/mlc-nand

[13] "Slc nand flash memory." [Online]. Available: https://www.micron.com/products/nand-flash/slc-nand

[14] "Tlc nand flash memory." [Online]. Available: https://www.micron.com/products/nand-flash/tlc-nand

[15] V. Taranalli, "Error characterization, channel modeling and coding for flash memories," 2017. [Online]. Available: https://escholarship.org/content/qt9r03n1tk/qt9r03n1tk.pdf?t=ontyn7

[16] Y. Cai, E. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in mlc nand flash memory: Characterization, analysis, and modeling," *2013 Design, Automation  Test in Europe Conference  Exhibition*

*(DATE)*, 2013. [Online]. Available: https://users.ece.cmu.edu/~omutlu/pub/flash-memory-voltage-characterization_date13.pdf

[17] ——, "Error patterns in mlc nand flash memory: Measurement, characterization, and analysis," *2012 Design, Automation Test in Europe Conference and Exhibition (DATE)*, 2012. [Online]. Available: https://people.inf.ethz.ch/omutlu/pub/flash-error-patterns_date12.pdf

[18] Y. Cai, Y. Luo, E. Haratsch, K. Mai, and O. Mutlu, "Data retention in mlc nand flash memory: Characterization, optimization, and recovery," *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015. [Online]. Available: https://people.inf.ethz.ch/~omutlu/pub/flash-memory-data-retention_hpca15.pdf

[19] E. Seneta, "A tricentenary history of the law of large numbers," 2013. [Online]. Available: https://arxiv.org/pdf/1309.6488

[20] "Nand flash memory types." [Online]. Available: https://www.micron.com/products/nand-flash/choosing-the-right-nand

[21] K. Khalifa, H. Fawzy, S. El-Ashry, and K. Salah, "Memory controller architectures: A comparative study," *2013 8th IEEE Design and Test Symposium*, 2013.

[22] "Discover the advantages of an onfi world." [Online]. Available: https://onfi.org/

[23] "Onfi 5.0 nand flash controller ip compliant to jedec." [Online]. Available: https://www.arasan.com/wp-content/uploads/2021/08/ONFI_5.1.pdf

[24] E. E. Gad, Y. Li, J. Kliewer, M. Langberg, A. A. Jiang, and J. Bruck, "Asymmetric error correction and flash-memory rewriting using polar codes," *IEEE Transactions on Information Theory*, vol. 62, no. 7, pp. 4024–4038, 2016.

[25] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit error rate in nand flash memories," *IEEE International Reliability Physics Symposium*, 2008.

[26] Y. Luo, S. Ghos, Y. Cai, E. F. Haratsch, and O. Mutlu, "Enabling accurate and practical online flash channel modeling for modern mlc nand flash memory," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 9, pp. 2294–2311, 2016.

[27] N. Papandreou, T. Parnell, H. Pozidis, C. Camp, T. Griffin, G. Tressler, and A. Walls, "Using adaptive read voltage thresholds to enhance the reliability of mlc nand flash memory systems," pp. 2294–2311, 2014.

[28] "Nand flash endurance testing." [Online]. Available: https://www.swissbit.com/en/support/application-notes/nand-flash-endurance-testing/

[29] Y. Li and K. N. Quader, "Nand flash memory: Challenges and opportunities," *Computer*, vol. 46, no. 8, pp. 23–29, 2013.

[30] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "Improving 3d nand flash memory lifetime by tolerating early retention loss and process variation," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2018.

[31] R. Pletka, N. Papandreou, R. Stoica, H. Pozidis, N. Ioannou, T. Fisher, A. Fry, and K. Ingram, "Improving nand flash performance with read heat separation," *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2020.

[32] J. Cui, Z. Zeng, J. Huang, W. Yuan, and L. T. Yang, "Improving 3-d nand ssd read performance by parallelizing read-retry," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems)*, vol. 42, no. 3, 2023.

[33] M. Saadoon, S. H. A. Hamid, H. Sofian, H. H. Altarturi, Z. H. Azizul, and N. Nasuha, "Fault tolerance in big data storage and processing systems: A review on challenges and solutions," *Ain Shams Engineering Journal*, vol. 13, no. 2, 2021. [Online]. Available: https://fsktm.um.edu.my/fsktm/images/RESEARCH/PUBLICATION/2022/

Scopus/6-HZ_Fault%20tolerance%20in%20big%20data%20storage%20and%20processing%20systems_%20A%20review%20on%20challenges%20and%20solutions.pdf

[34] B. Tahir, S. Schwarzy, and M. Rupp, "Ber comparison between convolutional, turbo, ldpc, and polar codes," *2017 24th International Conference on Telecommunications (ICT)*, 2017. [Online]. Available: https://publik.tuwien.ac.at/files/publik_262129.pdf

[35] L.-W. Liu, Y.-C. Liao, and H. C. Chang, "Up-gdbf: A 19.3 gbps error floor free 4kb ldpc decoder for nand flash applications," *IEEE Open Journal of Circuits and Systems*, vol. 3, pp. 228–236, 2022.

[36] H. Song, J.-C. FU, S. J. Zeng, J. Sha, Z. ZhangG, X. You, and C. Zhang, "Polar-coded forward error correction for mlc nand flash memory," *2016 8th International Conference on Wireless Communications Signal Processing (WCSP)*, 2018.

[37] C. Yang, M. Zhan, Y. Deng, M. Wang, X. H. Luo, and J. Zeng, "Error-correcting performance comparison for polar codes, ldpc codes and convolutional codes in high-performance wireless," *2019 6th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, 2019.

[38] M. DHUHEIR and S. ÃZTÃRK, "Polar codes analysis of 5g systems," *2018 6th International Conference on Control Engineering and Information Technology (CEIT)*, 2018.

[39] P. Shi, W. Tang, S. Zhao, and B. Wang, "Performance of polar codes on wireless communication channels," 2012.

[40] P. Luo and Z. Wang, "Secure nand flash architecture resilient to strong fault-injection attacks using algebraic manipulation detection code," 2013. [Online]. Available: https://sites.bu.edu/mark/files/2018/02/231.pdf

[41] H. Gordon, J. Edmonds, S. Ghandali, W. Yan, N. Karimian, and F. Tehranipoor, "Flash-based security primitives: Evolution, challenges and future directions," *MDPI Open Access Journals*, vol. 5, no. 1, 2021.

[42] Y. Zhong, C. Zhang, C. Xiong, and Z. Yany, "Multi-rate polar codes for solid state drives," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[43] T. Xie, Y. Y. Tai, and J. Zhu, "Polar codes for nand-based ssd systems: A joint source channel coding perspective," *2017 IEEE Information Theory Workshop (ITW)*, 2017.

[44] Y. Xiao, C. Long, J. Cao, Y. Chen, Y. Liu, J. He, and Z. Xiao, "Chaotic pilot and synchronization encryption of optical ofdm system based on polar code," *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019.

[45] X. Lu, J. Lei, W. Li, K. Lai, and Z. Pan, "Physical layer encryption algorithm based on polar codes and chaotic sequences," *IEEE Access*, vol. 7, pp. 4380–4390, 2018.

[46] J. G. Proakis and M. Salehi, *Digital Communications*. Mcgraw Hill International, 1996.

[47] "La memoria dram." [Online]. Available: https://www.crucial.es/support/what-is-computer-memory-dram

[48] J. J. Mesa, "Large scale studies of memory, storage, and network failures in a modern data center," 2018. [Online]. Available: https://kratos4.ethz.ch/wp-content/uploads/2019/01/justin-phd-thesis-December_25_2018.pdf

[49] C. Walck, *Hand-book on Statistical Distributions for experimentalists*, University of Stockholm, 2007.

[50] M. F. Triola, *Estadística. Décima edición.* ISBN: 978-970-26-1287-2, 2009. [Online]. Available: https://www.uv.mx/rmipe/files/2015/09/Estadistica.pdf

[51] R. D. Rivero and D. Monasterio, *Probabilidad y estadística. Aplicaciones a la ingeniería.* UNEXPO, 2013. [Online]. Available: https://gc.scalahed.com/recursos/files/r161r/w24762w/9D/ProbabilidadEstadIsticaaplicacionesIngenierIa.pdf

[52] *Distribuciones de probabilidad. Estadística Aplicada*, 2021. [Online]. Available: https://negocios.ucab.edu.ve/wp-content/uploads/2021/10/Distribuciones-de-Probabilidad-eBook.pdf

[53] M. M. B. Mónica Martínez Gómez, *La distribución binomial.* Estadística,Investigación Operativa Aplicadas y Calidad. Universidad Politécnica de Valencia. [Online]. Available: https://riunet.upv.es/bitstream/handle/10251/7936/Distribucion%20binomial.pdf

[54] J. I. G. Ruíz and A. M. R. Mendoza, *Distribuciones de Variables Aleatorias Discretas.* Editorial Lectorum, 2016. [Online]. Available: https://www.cicata.ipn.mx/assets/files/cicata/ProME/docs/Archivos/ProgramaEditorial/ApoME/Aportaciones%20de%20la%20Matematica%20Educativa%20No1%20Distribuciones%20de%20variables%20aleatorias%20discretas%20en%20libros%20de%20texto.pdf

[55] M. A. F. Caldera, L. A. Caldito, N. C. Dios, and J. P. Mayo, *Estimación y predicción bayesiana. El modelo Beta_Binomial.* Universidad de Extremadura. [Online]. Available: https://www.asepelt.org/ficheros/File/Anales/2003%20-%20Almeria/asepeltPDF/147.PDF

[56] R. E. Walpole, R. H. Myers, R. H. Myers, and R. H. Myers, *Probabilidad y estadística para ingeniería y ciencias.* PEARSON EDUCACIÓN, 2012. [Online]. Available: https://vereniciafunez94hotmail.files.wordpress.com/2014/08/8va-probabilidad-y-estadistica-para-ingenier-walpole_8.pdf

[57] J. G. Skellam, "A probability distribution derived from the binomial distribution by regarding the probability of success as variable between the sets of trials," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 10, no. 10, pp. 257–261.

[58] *Error Probability Analysis of Binary Asymmetric Channels*. Information Theory Laboratory Department of Electrical Engineering National Chiao Tung University, 2010. [Online]. Available: https://moser-isi.ethz.ch/docs/papers/smos-2010-2.pdf

[59] S. D. Constantin, *On the Theory of Binary Asymmetric Error Correcting Codes*. Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas, 1979. [Online]. Available: https://core.ac.uk/download/pdf/82505206.pdf

[60] V. Taranalli, H. Uchikawa, and P. H. Siegel, "Error analysis and inter-cell interference mitigation in multi-level cell flash memories," *Error Analysis and Inter-Cell Interference Mitigation in Multi-Level Cell Flash Memories*, 2015.

[61] ——, "Channel models for multi-level cell flash memories based on empirical error analysis," *IEEE Transactions on Communications*, vol. 64, no. 8, pp. 3169–3181, 2016.

[62] A. Marelliand and R. Micheloni, "Bch and ldpc error correction codes for nand flash memories," pp. 281–320, 2016. [Online]. Available: https://picture.iczhiku.com/resource/eetop/WHieEDHGPoWIQCnm.pdf

[63] R. E.Blahut, "Algebraic codes for data transmission," 2003. [Online]. Available: https://catdir.loc.gov/catdir/samples/cam033/2001043802.pdf

[64] B. Varsha Regulapati, "Error correction codes in nand flash memory," 2015. [Online]. Available: https://repositories.lib.utexas.edu/server/api/core/bitstreams/b6236ec2-8df3-4be9-93ca-2c7a52912d31/content

[65] M. Zhang, F. Wu, Y. Du, W. Liu, and C. Xie, "Pair-bit errors aware ldpc decoding in mlc nand flash memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2312–2320, 2019.

[66] K. Zhaoand, W. Zhao, H. Sun, T. Zhang, X. Zhang, and N. Zheng, "Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives," *USENIX Conference on File and Storage Technologies*, 2013. [Online]. Available: https://www.usenix.org/system/files/conference/fast13/fast13-final125.pdf

[67] J. H. Bae, A. Abotabl, H.-P. Lin, K.-B. Song, and J. Lee, "An overview of channel coding for 5g nr cellular communications," 2019. [Online]. Available: https://www.cambridge.org/core/services/aop-cambridge-core/content/view/CF52C26874AF5E00883E00B6E1F907C7/S2048770319000106a.pdf/an-overview-of-channel-coding-for-5g-nr-cellular-communications.pdf

[68] S. Buzaglo, "Permuted successive cancellation decoder for polar codes," *2014 International Symposium on Information Theory and its Applications*, 2017.

[69] X. Li, "Evaluation of channel coding methods for next generation mobile communication standards," 2021. [Online]. Available: https://ebuah.uah.es/dspace/handle/10017/49528?locale-attribute=es

[70] R. Hooshmand, M. R. Aref, and T. Eghlidos, "Physical layer encryption scheme using finite-length polar codes," *IET Communications*, vol. 9, no. 15, pp. 857–1866, 2015. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-com.2014.0933

[71] R. E. Amrani, "Grado en ingeniería en tecnologías de telecomunicación. estudio y verificación de los códigos polares en canales ruidosos," 2022. [Online]. Available: https://ebuah.uah.es/dspace/bitstream/handle/10017/52129/TFG_El_Amrani_2022.pdf?sequence=1&isAllowed=y

[72] R. Pedarsani, "Master project polar codes: Construction and performance analysis," 2011. [Online]. Available: https://web.ece.ucsb.edu/~ramtin/MSThesis.pdf

[73] Q. Peng, D. Yin, D. Chang, Y. Li, H. Zhang, G. Yan, and G. Wang, "On the performance of low-complexity decoders of ldpc and polar codes," 2024. [Online]. Available: https://arxiv.org/pdf/2403.19266

[74] C. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[75] Z. Li, S. Li, G. Alvarez, and W. Halang, "Analog chaos-based secure communications and cryptanalysis: A brief survey," *3rd INTERNATIONAL CONFERENCE "PHYSICS AND CONTROL"(PhysCon 2007)*, 2007.

[76] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," pp. 1259–1284, 1998.

[77] A. R. M. Angulo, "Cifrado simÃ©trico basado en la teorÃ-a del caos para imÃ¡genes digitales," 2015. [Online]. Available: https://repositorioinstitucional.buap.mx/server/api/core/bitstreams/84350301-9044-48c6-a6e3-516c6e9da69b/content

[78] S. H. Strogatz, *Nonlinear Dynamics and Chaos with Student Solutions Manual With Applications to Physics, Biology, Chemistry, and Engineering*, 2018.

[79] J. E. T. n. Ortiz, "Evaluación del desempeño de técnicas decifrado caótico en imágenes," 2020. [Online]. Available: http://eprints.uanl.mx/id/eprint/24032

[80] R. Scherer, S. L. Kalla, Y. Tang, and J. Huang, "The grünwald_letnikov method for fractional differential equations," vol. 62, no. 3, pp. 902–917, 2011. [Online]. Available: https://core.ac.uk/download/pdf/82546931.pdf

[81] S. C. Koduru and V. Chandrasekaran, "Integrated confusion-diffusion mechanisms for chaos based image encryption," 2008.

[82] G. Ye, "Image scrambling encryption algorithm of pixel bit based on chaos map," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 347–354, 2010.

[83] J. Beutel, H. L. Kundel, and R. L. V. Metter., *Handbook of Medical Imaging*. SPIEâThe International Society for Optical Engineering Bellingham, Washington USA, 2009, vol. 2.

[84] E. Hussein, R. Sadiki, Y. Jaftaand, M. MujahidSungay, O. Ajayi, and A. Bagula, "Big data processing using hadoop and spark:the case of meteorology data," *Social Informatics and Telecommunications Engineering*, vol. 311, no. 180–185, 2020.

[85] R. Gonzalez and R. Woods, *Digital Image Processing, Fourth Edition*. Pearson Education, 2018. [Online]. Available: https://dl.icdst.org/pdfs/files4/01c56e081202b62bd7d3b4f8545775fb.pdf

[86] R. K. Cho, K. Sood, and C. S. C. Channapragada, "Image repair and restoration using deep learning," *2022 4th International Conference on Artificial Intelligence and Speech Technology (AIST)*, 2023.

[87] E. E. A. b. Elgabar, "Comparison of lsb steganography in bmp and jpeg images," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 3, no. 5, 2013. [Online]. Available: https://www.ijsce.org/wp-content/uploads/papers/v3i5/E1900113513.pdf

[88] "Reconocimiento facial: definición y explicación." [Online]. Available: https://latam.kaspersky.com/resource-center/definitions/what-is-facial-recognition