
Universidad de Granada
Escuela Técnica Superior de Ingenierías
Informática y de Telecomunicación

# Estudio y optimización de secuencias de bits pseudoaleatorias basadas en LFSR para cifrado de flujo

**Autor:** Joaquín Sergio García Ibáñez

**Tutor:** Jesus García Miranda

14 de septiembre de 2025

# Índice general

# Capítulo 1

# Introducción

En la actualidad, la seguridad de la información se ha convertido en un pilar fundamental para el correcto funcionamiento de la sociedad digital. El aumento constante en el volumen de datos transmitidos a través de redes de comunicación, la expansión de los dispositivos conectados y la necesidad de proteger tanto la privacidad como la integridad de la información han impulsado el desarrollo de técnicas criptográficas cada vez más sofisticadas. Dentro de este amplio campo, los *cifrados de flujo* ocupan un lugar destacado debido a su eficiencia, su sencillez conceptual y a su capacidad para adaptarse a escenarios en los que la velocidad y el bajo consumo de recursos resultan esenciales.

## Contexto y motivación

El cifrado de flujo se caracteriza por combinar el texto plano con una secuencia pseudoaleatoria de bits, denominada *keystream*, generalmente mediante la operación XOR. A diferencia de los cifrados en bloque, que procesan la información en bloques de longitud fija, los cifrados de flujo trabajan de forma bit a bit, lo que permite en muchos casos un procesamiento más rápido y una implementación más ligera en dispositivos con recursos limitados.

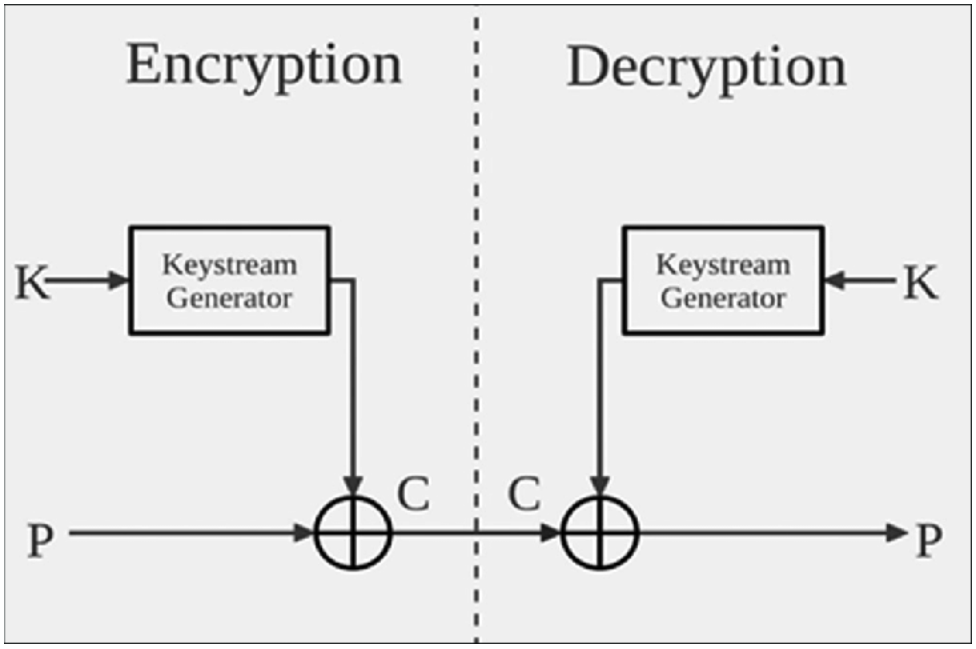


Figura 1.1: Esquema simplificado de un cifrador de flujo: el texto plano se combina con un flujo de claves generado por un LFSR o un combinador de LFSR mediante XOR.

Por estas razones, los cifrados de flujo han sido ampliamente utilizados en aplica-

ciones de comunicaciones móviles, protocolos de transmisión en tiempo real y sistemas embebidos.

| **Característica** | **Cifrado en bloque** | **Cifrado de flujo** |
|---|---|---|
| Unidad de operación | Bloques de $n$ bits | Bits individuales |
| Flexibilidad | Menor | Mayor |
| Velocidad en hardware | Alta | Muy alta |
| Ejemplos | AES, DES, Blowfish | RC4, A5/1, Salsa20 |

Cuadro 1.1: Comparación entre cifrado en bloque y cifrado de flujo.

No obstante, el gran reto que plantean estos cifrados radica en la construcción del generador de flujo de claves. Si la secuencia generada presenta patrones, redundancias o estructuras predecibles, un atacante puede explotar dichas regularidades para recuperar el mensaje original o incluso reconstruir el estado interno del generador. En consecuencia, el diseño de generadores de secuencias pseudoaleatorias con buenas propiedades estadísticas y elevada complejidad constituye un área de investigación activa dentro de la criptografía aplicada.

Además, este trabajo se enmarca como una ampliación de estudios previos, incorporando un conjunto más amplio de pruebas estadísticas y comparativas. La motivación principal es evaluar de forma más exhaustiva las secuencias generadas por LFSR y combinadores, con el fin de obtener una visión más completa de sus fortalezas y limitaciones. Con ello se busca no solo validar resultados anteriores, sino también aportar nuevos elementos de análisis que permitan profundizar en la calidad y seguridad de estos generadores.

## Antecedentes históricos

El estudio de los cifradores de flujo y de los registros de desplazamiento de retroalimentación lineal (LFSR) tiene sus raíces en los primeros trabajos de Claude Shannon en la década de 1940, cuando se establecieron los fundamentos teóricos de la criptografía moderna. Shannon introdujo la noción de *confusión* y *difusión* como propiedades esenciales para la seguridad de un sistema de cifrado, lo que influyó en el diseño posterior de primitivas tanto de bloque como de flujo.

Durante los años 1950 y 1960, los LFSR comenzaron a utilizarse en sistemas de comunicaciones y en generación de códigos de corrección de errores, gracias a su simplicidad de implementación y a la posibilidad de producir secuencias de largo período conocidas como *m-secuencias*. Su bajo coste en hardware los convirtió en una opción atractiva para dispositivos de telecomunicaciones y aplicaciones militares.

En la década de 1980, los cifradores de flujo experimentaron un auge con el desarrollo de algoritmos como RC4, ampliamente adoptado en protocolos como WEP y posteriormente en SSL/TLS. Aunque con el tiempo se descubrieron debilidades críticas en su diseño, RC4 marcó un hito en la historia de la criptografía de flujo por su simplicidad y eficiencia.

Paralelamente, los sistemas de telefonía móvil GSM introdujeron el algoritmo A5/1, basado en múltiples LFSR combinados mediante funciones no lineales. Este caso mostró tanto las ventajas como las limitaciones de los LFSR: aunque permitieron implementar cifrado ligero en dispositivos con recursos muy reducidos, investigaciones posteriores demostraron vulnerabilidades que podían explotarse mediante ataques de correlación y criptoanálisis algebraico.

En la actualidad, aunque muchos sistemas modernos han migrado hacia cifradores de bloque en modos de operación que emulan flujos (como AES en CTR), los cifradores de flujo y los LFSR siguen siendo relevantes en entornos de recursos restringidos, comunicaciones en tiempo real y aplicaciones embebidas. Además, siguen constituyendo un área de investigación activa como base para la construcción de generadores pseudoaleatorios eficientes y seguros.

## Limitaciones prácticas y retos actuales

El diseño de un cifrador de flujo plantea retos que van más allá del rendimiento. En la práctica, un generador debe satisfacer simultáneamente requisitos de *robustez* (resistencia frente a ataques), *portabilidad* (facilidad de integración en plataformas heterogéneas) y *mantenibilidad* (claridad del modelo y facilidad de verificación). En sistemas embebidos e IoT, además, la seguridad compite con presupuestos estrictos de energía y memoria, lo que obliga a optar por construcciones ligeras sin sacrificar propiedades estadísticas.

Desde el punto de vista criptográfico, la dificultad reside en garantizar que el flujo de claves sea indistinguible de un proceso aleatorio ideal para cualquier adversario computacionalmente acotado. Esto excluye, por ejemplo, generadores puramente lineales: aunque produzcan secuencias de gran periodo y buena dispersión, su estructura algebraica facilita ataques que reconstruyen el estado interno a partir de un número finito de observaciones.

## Aplicaciones y escenarios de uso

Los cifradores de flujo destacan en:

- **Comunicaciones en tiempo real**: voz y vídeo donde la latencia es crítica y la granularidad bit a bit evita rellenos y modos de operación complejos.
- **Sistemas embebidos e IoT**: sensores alimentados por batería y microcontroladores de baja potencia que requieren primitivas ligeras.
- **Enlaces inalámbricos y satelitales**: donde la sencillez del hardware y la alta tasa de bits favorecen generadores simples y eficientes.

En todos estos casos, el desafío consiste en equilibrar eficiencia y seguridad, minimizando tanto los sesgos estadísticos como la predictibilidad estructural.

# Pseudoaleatoriedad en criptografía

La noción de *aleatoriedad* es fundamental en criptografía. Una secuencia verdaderamente aleatoria debería ser impredecible y carecer de todo patrón discernible. Sin embargo, la generación de aleatoriedad perfecta suele depender de fenómenos físicos (como ruido térmico, radiactividad o fluctuaciones cuánticas) que no siempre están disponibles ni son fáciles de integrar en sistemas digitales.

Por ello, se recurre habitualmente a la *pseudoaleatoriedad*, es decir, a secuencias generadas por algoritmos deterministas a partir de un estado inicial o semilla, que presentan la apariencia de ser aleatorias. Aunque en esencia son predecibles si se conoce el algoritmo y la semilla, un buen generador debe producir secuencias indistinguibles de la aleatoriedad verdadera para un observador sin esa información.

En este contexto, resultan especialmente relevantes los **postulados de Golomb**, que establecen tres propiedades básicas que deberían cumplir las secuencias pseudoaleatorias: (1) un número equilibrado de ceros y unos, (2) una distribución adecuada de rachas de bits consecutivos y (3) una autocorrelación desplazada lo más cercana posible a cero. Estas condiciones permiten evaluar rápidamente si una secuencia se asemeja a un comportamiento aleatorio ideal.

Además de los postulados de Golomb, en la práctica se emplean baterías de pruebas estadísticas desarrolladas por la NIST (Instituto Nacional de Estándares y Tecnologías), que incluyen tests de frecuencia, longitud de rachas, correlación, complejidad lineal, entre otros. Estas pruebas resultan fundamentales para determinar si un generador es apto para su uso en criptografía.

## Criterios de indistinguibilidad y pruebas estadísticas

El objetivo práctico de la pseudoaleatoriedad es la *indistinguibilidad*: que ningún algoritmo eficiente pueda diferenciar una secuencia generada de otra verdaderamente aleatoria con ventaja significativa. En la práctica, esta noción se aproxima mediante pruebas estadísticas que evalúan propiedades parciales (no exhaustivas) del flujo.

Entre las pruebas más utilizadas se encuentran las de la batería NIST SP 800–22. A continuación se detallan aquellas relevantes para este trabajo, junto con una breve interpretación operativa de sus estadísticos y p–valores:

**Frecuencia (monobit).** Sea $x_i \in \{0,1\}$ y $\tilde{x}_i = 2x_i - 1 \in \{-1,1\}$. El estadístico es $S_n = \sum_{i=1}^{n} \tilde{x}_i$ y el p–valor se aproxima por $p = \mathrm{erfc}\big(|S_n|/\sqrt{2n}\big)$. Valores pequeños indican sesgo global hacia 0 o 1.

**Frecuencia por bloques.** Dividiendo la secuencia en bloques de tamaño $M$, se calcula, para cada bloque $j$, la fracción $\pi_j$ de unos. El contraste $\chi^2$ sobre $\{\pi_j\}$ detecta sesgos *locales*. Un generador puede pasar monobit y fallar aquí si alterna zonas sesgadas.

**Rachas (runs).** Define $V_n = 1 + \sum_{i=1}^{n-1} \mathbf{1}_{\{x_i \neq x_{i+1}\}}$. Este test compara $V_n$ con su esperanza bajo balance. Detecta exceso o defecto de transiciones, revelando agregación o alternancia artificial.

**Sumas acumulativas (CUSUM).** Considera la caminata $S_k = \sum_{i=1}^{k} \tilde{x}_i$ y su desviación máxima $M_n = \max_{1 \leq k \leq n} |S_k|$. Un $M_n$ grande sugiere un sesgo *acumulado* no visible en un prefijo corto.

**Entropía aproximada (ApEn).** Compara la distribución de patrones de longitud $m$ y $m{+}1$ a través de sus entropías empíricas; p–valores bajos indican regularidades de corto alcance.

**Universal de Maurer.** Mide la *compresibilidad* esperada de la secuencia. Patrones repetidos o estructura redundante reducen el p–valor.

**Serial.** Evalúa la frecuencia de todas las palabras de longitud $m$ (y a menudo $m{+}1$). Detecta asimetrías o vacíos en el espectro de patrones cortos.

En todos los casos, se adopta un umbral $\alpha = 0{,}01$: si $p \geq \alpha$ se *acepta* la hipótesis de aleatoriedad para esa prueba. Fallar de forma consistente un subconjunto de tests sugiere la presencia de estructura aprovechable por un atacante.

# Registros LFSR

Uno de los mecanismos más estudiados y empleados para la generación de secuencias pseudoaleatorias es el *Linear Feedback Shift Register* (LFSR). Estos registros desplazan su estado interno en cada iteración y calculan el nuevo bit de entrada como una combinación lineal (módulo 2) de ciertos bits de la propia secuencia, definidos por un polinomio generador. Los LFSR son extremadamente eficientes en hardware y software, lo que los convierte en candidatos idóneos para sistemas de alta velocidad y entornos con recursos limitados.

Históricamente, los LFSR han sido empleados en criptografía aplicada, en sistemas de comunicaciones inalámbricas como GSM, en generación de códigos de control de errores y en simulaciones de procesos estocásticos. Su atractivo radica en su bajo coste computacional, pero su estructura lineal conlleva también importantes debilidades: dado un número suficiente de bits de salida, es posible reconstruir el polinomio de retroalimentación y el estado inicial mediante algoritmos como Berlekamp-Massey. Además, un único LFSR suele fallar en varios tests de aleatoriedad, lo que limita su uso directo en cifrado de flujo.

### Modelo algebraico y propiedades básicas

Un LFSR binario de grado $m$ viene determinado por su vector de estado $s^{(t)} = (s_0^{(t)}, \ldots, s_{m-1}^{(t)}) \in \{0,1\}^m$ y por un conjunto de *taps* $T \subseteq \{0, \ldots, m-1\}$. La actualización es lineal sobre $\mathbb{F}_2$:

$$s_0^{(t+1)} = \bigoplus_{i \in T} s_i^{(t)}, \qquad s_j^{(t+1)} = s_{j-1}^{(t)} \ \ (1 \le j \le m-1),$$

donde $\oplus$ denota la suma módulo 2. Esta dinámica equivale al polinomio de retroalimentación $f(x) = 1 + \sum_{i \in T} x^i + x^m$. Si $f(x)$ es *primitivo* sobre $\mathbb{F}_2$, el LFSR genera una *m-secuencia* de periodo máximo $2^m - 1$ (para cualquier estado inicial no nulo).

Las m-secuencias satisfacen los postulados de Golomb en el límite y presentan un espectro binario plano, pero su *complejidad lineal* es exactamente $m$: existe un LFSR de longitud $m$ (el propio) que reproduce la secuencia. Esto implica que, observando un prefijo suficientemente largo, es posible reconstruir el estado y el polinomio mediante procedimientos como Berlekamp–Massey.

### Complejidad lineal y recuperación de estado

La complejidad lineal $L$ de una secuencia binaria es la longitud del LFSR más corto capaz de generarla. El algoritmo de Berlekamp–Massey estima $L$ a partir de un prefijo $x_1, \ldots, x_n$ en tiempo $O(n^2)$, devolviendo además un polinomio de conexión candidato. En términos criptográficos, $L$ acota el esfuerzo necesario para reconstruir el generador por medios puramente lineales; valores bajos de $L$ hacen a la secuencia predecible con pocos bits observados.

Estas limitaciones motivan el uso de *combinadores no lineales* de varios LFSR, buscando aumentar $L$ y romper correlaciones explotables, sin abandonar la eficiencia estructural de los registros.

## Estado del arte y generadores combinados

Con el fin de superar estas limitaciones, la literatura ha propuesto múltiples estrategias basadas en la combinación de varios LFSR mediante funciones no lineales. Estos esquemas incrementan la complejidad del sistema y dificultan los intentos de predicción del flujo generado. Entre los ejemplos más representativos se encuentran:

- El **generador Shrinking**, que utiliza un LFSR de control para decidir qué bits de otro LFSR se incluyen en la secuencia final.
- El **generador de Geffe**, que combina tres LFSR mediante una función booleana no lineal que incrementa la complejidad de la secuencia producida.

- El **generador de Mayoría**, en el que la salida depende del valor mayoritario de varios LFSR en cada instante.

Estos generadores, aunque aumentan la resistencia frente a ciertos ataques, requieren un análisis detallado mediante métricas estadísticas y estructurales para determinar si la mejora es realmente significativa respecto al uso de un LFSR individual.

### Ataques clásicos a generadores basados en LFSR

La linealidad facilita varios vectores de ataque:

- **Ataques de correlación**: cuando la salida combinada conserva correlación estadística con la salida de algún LFSR interno, es posible "desacoplar" el sistema probando estados parciales y maximizando verosimilitudes.
- **Ataques algebraicos**: expresando la salida como polinomios booleanos del estado, se plantean sistemas de ecuaciones sobre $\mathbb{F}_2$ que, bajo ciertas condiciones, admiten resolución más rápida que la búsqueda exhaustiva.
- **Time–memory tradeoff**: técnicas que precalculan grandes tablas (por ejemplo, cadenas de estados) para acelerar la recuperación del estado durante un ataque en línea.

Estos ataques justifican dos líneas de defensa complementarias: (i) aumentar la *complejidad lineal* efectiva de la salida y (ii) reducir correlaciones explotables entre la salida y los estados internos, tanto en promedio como en subconjuntos de bits.

## Objetivos del trabajo

El objetivo general de este Trabajo Fin de Grado es **estudiar y optimizar secuencias de bits pseudoaleatorias basadas en LFSR para su aplicación en cifrado de flujo**. Para alcanzar este propósito se plantean los siguientes objetivos específicos:

1. Implementar en Python generadores basados en LFSR y sus variantes combinadas (Shrinking, Geffe y Mayorías).
2. Generar secuencias de prueba de longitud suficiente para un análisis estadístico riguroso.
3. Evaluar la calidad pseudoaleatoria de dichas secuencias mediante un conjunto de tests seleccionados de la batería por la NIST (Instituto Nacional de Estándares y Tecnologías), así como los postulados de Golomb.
4. Calcular la complejidad lineal de las secuencias generadas empleando el algoritmo de Berlekamp-Massey.

5. Comparar los resultados obtenidos entre diferentes configuraciones de LFSR y combinadores, analizando ventajas y limitaciones de cada caso.

6. Ampliar el conjunto de pruebas estadísticas respecto a trabajos anteriores, incluyendo no solo los tests clásicos de la batería por la NIST (Instituto Nacional de Estándares y Tecnologías), sino también métricas adicionales que permitan una caracterización más precisa de las secuencias.

7. Analizar críticamente las diferencias obtenidas respecto a estudios previos, valorando en qué medida los nuevos resultados confirman, refuerzan o contradicen conclusiones anteriores.

## Hipótesis y alcance

La hipótesis de partida es que los LFSR individuales, aunque eficientes en términos de implementación, presentan limitaciones tanto en complejidad lineal como en propiedades pseudoaleatorias, lo que los hace vulnerables a ataques de predicción o reconstrucción. No obstante, al combinar varios registros mediante funciones no lineales, se espera obtener secuencias con mayor calidad estadística y una complejidad superior, incrementando así la resistencia frente a ataques criptoanalíticos.

El alcance de este trabajo se centra en el análisis de generadores en un entorno software, sin abordar implementaciones hardware ni optimizaciones específicas para plataformas embebidas. El estudio se focaliza en tres combinadores representativos de la literatura —Shrinking, Geffe y Mayorías— y en un conjunto de pruebas estadísticas seleccionadas que permitan valorar de forma fiable la aleatoriedad y la complejidad de las secuencias producidas. A diferencia de estudios previos, este trabajo no se limita a replicar experimentos, sino que los amplía incorporando un mayor número de pruebas comparativas y métricas adicionales, con el objetivo de ofrecer una caracterización más completa del comportamiento de los generadores basados en LFSR. De este modo, el presente TFG se plantea como una continuación y mejora de investigaciones anteriores en la materia.

## Terminología y notación

## Metodología y reproducibilidad

El estudio sigue una metodología empírico–comparativa con implementación en Python y control de semillas. El proceso se organiza en: (i) definición formal de generadores (LFSR y combinados), (ii) generación de secuencias con longitudes prefijadas, (iii) evaluación estadística (subset NIST) y estimación de complejidad lineal, y (iv) síntesis de resultados mediante medias, desviaciones estándar y tasas de aceptación para $\alpha = 0.01$.

Se garantiza reproducibilidad mediante control de semillas, registro de parámetros de

| Símbolo | Descripción |
|---|---|
| $\mathbb{F}_2$ | Campo binario $\{0,1\}$ con suma XOR. |
| $x_1, \ldots, x_n$ | Secuencia binaria de longitud $n$. |
| $\tilde{x}_i$ | Transformación $2x_i - 1 \in \{-1, 1\}$. |
| $L$ | Complejidad lineal de una secuencia. |
| $f(x)$ | Polinomio de realimentación del LFSR. |
| $T$ | Conjunto de taps del LFSR. |
| $S_n$ | Estadístico monobit $\sum_{i=1}^{n} \tilde{x}_i$. |
| $M$ | Tamaño de bloque en frecuencia por bloques. |
| $\alpha$ | Nivel de significación (aquí $\alpha = 0.01$). |
| $p$ | p–valor de una prueba estadística. |

Cuadro 1.2: Glosario mínimo de símbolos y términos utilizados.

ejecución, separación de código/datos/resultados, exportación de tablas en CSV/LaTeX y documentación de dependencias.

## Criterios de evaluación y significación

La comparación entre configuraciones se basa en:

- Tasas de aceptación por prueba (umbral $\alpha = 0.01$) agrupadas sobre múltiples semillas.
- Estadísticos agregados (media y desviación estándar) de p–valores por prueba y configuración.
- Complejidad lineal estimada por Berlekamp–Massey como indicador operativo de predictibilidad.
- Longitud efectiva de salida en combinadores con descarte (p. ej., Shrinking).

Se considera que una configuración mejora a otra cuando incrementa las tasas de aceptación sin sacrificar complejidad lineal ni longitud efectiva de forma significativa.

## Casos de uso y escenarios de aplicación

Se consideran tres dominios representativos:

- Sensórica IoT con restricciones de energía y memoria.
- Streaming en tiempo real de voz/vídeo con baja latencia.
- Enlaces inalámbricos de alto caudal con hardware sencillo.

En estos contextos, combinadores basados en LFSR ofrecen un compromiso entre coste computacional y calidad estadística siempre que superen pruebas básicas y presenten alta complejidad lineal.

## Limitaciones y amenazas a la validez

- Selección parcial de pruebas NIST; algunas debilidades podrían no detectarse.
- Dependencia del tamaño muestral y del número de repeticiones.
- Resultados condicionados a polinomios, semillas y longitudes escogidas.
- Ausencia de criptoanálisis activo más allá de medidas estadísticas y de complejidad lineal.

Estas amenazas se mitigan mediante multi–semilla, reporte de dispersión y documentación de parámetros para facilitar replicación.

## Objetivos extendidos y aportaciones medibles

- Cobertura: ejecutar $r \geq 30$ repeticiones por configuración y semillas variadas.
- Longitudes: evaluar $n \in \{10^5, 5 \cdot 10^5, 10^6\}$ para estudiar estabilidad de p–valores.
- Complejidad lineal: reportar la distribución empírica de $L$ por configuración.
- Reproducibilidad: publicar parámetros, semillas y resultados tabulados desde las ejecuciones.

## Resumen del capítulo

Se ha establecido el contexto y la motivación del uso de LFSR y combinadores en cifrado de flujo, junto con objetivos, alcance, metodología y criterios de evaluación. El Capítulo 2 desarrolla la pseudoaleatoriedad y las métricas empleadas; el Capítulo 3 aborda LFSR, complejidad lineal y generadores combinados; el Capítulo 4 presenta el experimento y los resultados; el Capítulo 5 ofrece conclusiones y líneas futuras.

# Capítulo 2

# Pseudoaleatoriedad y propiedades estadísticas

En criptografía, la *aleatoriedad* es un requisito fundamental. Una secuencia de bits que actúe como clave o como flujo cifrante debe ser impredecible y no mostrar patrones repetitivos que puedan ser explotados por un adversario. Sin embargo, obtener aleatoriedad verdadera no es trivial: requiere fuentes físicas como ruido electrónico, fluctuaciones cuánticas o fenómenos atmosféricos. Estos métodos, aunque útiles en algunos contextos, no siempre son viables en sistemas informáticos convencionales debido a su coste o complejidad.

Por ello, en la práctica se utilizan **generadores de números pseudoaleatorios** (PRNG, por sus siglas en inglés). Estos algoritmos producen secuencias deterministas, pero diseñadas para que sus propiedades estadísticas sean indistinguibles de una secuencia verdaderamente aleatoria. En criptografía, cuando además ofrecen resistencia frente a ataques de predicción, se habla de **CSPRNG** (Cryptographically Secure PRNG).

## Aleatoriedad verdadera vs. pseudoaleatoriedad

Una secuencia aleatoria ideal cumple dos propiedades:

1. Cada bit es independiente de los anteriores.

2. La probabilidad de obtener 0 o 1 es exactamente 1/2.

En contraste, un generador pseudoaleatorio produce secuencias totalmente deterministas a partir de una semilla inicial. La clave está en que, si el algoritmo es robusto, el resultado debe ser *indistinguible* de una secuencia aleatoria para un adversario con capacidad de cómputo acotada.

*Observación práctica.* En la práctica, ninguna fuente física ni generador determinista cumple a la perfección estas propiedades. Los generadores de hardware pueden verse afectados por ruido ambiental, fallos de calibración o incluso manipulación adversaria. Los generadores puramente software heredan limitaciones matemáticas (ciclos finitos, correlaciones residuales). La pseudoaleatoriedad moderna se entiende, por tanto, como un compromiso: producir secuencias que, sin ser aleatorias en sentido estricto, resistan cualquier intento eficiente de distinción por parte de un adversario.

# Clasificación de generadores pseudoaleatorios

Existen distintas familias de generadores, cada una con características y aplicaciones específicas:

- **Generadores lineales congruenciales (LCG)**: definidos por la recurrencia $X_{n+1} = (aX_n + c)$ mód $m$. Son muy eficientes, pero su calidad estadística es insuficiente para aplicaciones criptográficas.
- **Generadores basados en registros de desplazamiento**: como los LFSR, ampliamente estudiados en este trabajo. Ofrecen periodos largos y gran velocidad, pero presentan debilidades frente a ataques de reconstrucción de estado.
- **Generadores basados en criptografía de bloque**: por ejemplo, el uso de AES en modo contador (AES-CTR). Proporcionan alta seguridad, pero a costa de mayor consumo computacional.
- **Generadores basados en funciones hash**: se apoyan en la resistencia de funciones hash como SHA–256 para producir secuencias indistinguibles de aleatorias.
- **Generadores híbridos**: combinan distintas técnicas (por ejemplo, ruido físico como semilla más un PRNG software) para equilibrar seguridad y rendimiento.

*Periodo y eficiencia.* Una característica clave en esta clasificación es el *periodo*, es decir, la longitud máxima antes de que una secuencia empiece a repetirse. Mientras que los LCG pueden presentar periodos del orden de $10^6$ o $10^9$, los generadores basados en LFSR alcanzan valores exponenciales en el grado del polinomio. En contraste, esquemas como AES–CTR o los basados en SHA no presentan un límite práctico observable, ya que su ciclo efectivo está más allá de cualquier capacidad computacional realista. Este contraste entre eficiencia y robustez estadística guía la selección de un PRNG en función de la aplicación.

La elección del generador depende del equilibrio entre coste computacional, requisitos de seguridad y disponibilidad de recursos.

## Importancia de la aleatoriedad en criptografía

La seguridad de numerosos protocolos criptográficos depende de la calidad de los números aleatorios empleados. Por ejemplo, en el intercambio de claves mediante el protocolo Diffie–Hellman, la elección de parámetros y exponentes secretos debe ser impredecible: un generador deficiente podría permitir a un adversario anticipar claves y romper el sistema. De manera similar, en firmas digitales como RSA o ECDSA, el uso de valores aleatorios sesgados ha conducido en el pasado a la recuperación de claves privadas a partir de firmas observadas.

La aleatoriedad también resulta crítica en esquemas de inicialización de cifrados por bloques (IVs), en la construcción de sal para el almacenamiento seguro de contraseñas, y en sistemas de autenticación. En todos estos contextos, un generador pseudoaleatorio débil constituye un eslabón vulnerable que compromete la seguridad completa del sistema, independientemente de la robustez matemática de la primitiva criptográfica principal.

# Postulados de Golomb

Golomb propuso en 1967 tres propiedades estadísticas que sirven como primera aproximación para evaluar secuencias binarias pseudoaleatorias:

1. **Balance**: en una secuencia de longitud $N$, el número de unos y ceros debe diferir a lo sumo en una unidad:
$$|\#1 - \#0| \leq 1.$$

2. **Distribución de rachas**: el número de rachas de longitud $k$ debe aproximarse a la mitad del número de rachas de longitud $k-1$. Esto implica que las rachas largas son poco frecuentes y las cortas abundan.

3. **Autocorrelación**: el corrimiento de la secuencia $k$ posiciones respecto a sí misma debe producir aproximadamente el mismo número de coincidencias que discrepancias. La autocorrelación se define como:
$$C(k) = \sum_{i=1}^{N} (-1)^{x_i \oplus x_{i+k}},$$
y debe ser cercana a cero para todo $k \neq 0$.

## Limitaciones de los postulados de Golomb

Aunque útiles como primer criterio, los postulados de Golomb presentan limitaciones importantes:

- No detectan correlaciones de largo alcance ni dependencias no lineales.
- Secuencias que los cumplen pueden seguir siendo predecibles mediante algoritmos algebraicos.
- Son sensibles al tamaño de muestra: secuencias cortas tienden a violarlos, aunque el generador sea adecuado.

Por estas razones, los postulados se emplean como una comprobación preliminar, que debe complementarse con baterías estadísticas más completas como las del NIST o TestU01.

### Casos históricos de fallos por mala aleatoriedad

Existen múltiples ejemplos documentados donde la falta de una adecuada pseudoaleatoriedad ha tenido consecuencias graves:

- **Debilidad en Netscape SSL (1995)**: la generación de claves dependía del reloj del sistema y de procesos predecibles, lo que permitió a atacantes reconstruir claves de sesión.
- **Vulnerabilidades en Debian OpenSSL (2006–2008)**: una modificación en el código redujo drásticamente el espacio de claves posibles, haciendo que certificados y claves SSH fueran fácilmente predecibles.
- **Ataques a GSM/A5/1**: el uso de LFSR con combinaciones débiles mostró que, pese a producir secuencias de largo periodo, se podía explotar la correlación entre registros para recuperar claves en tiempo práctico.
- **Previsibilidad en generadores de hardware defectuosos**: en ciertos dispositivos IoT se han documentado PRNG que reutilizan semillas o carecen de suficiente entropía inicial, permitiendo ataques remotos para predecir tokens de autenticación.

Estos casos ilustran que el problema de la aleatoriedad no es teórico, sino un factor determinante en la seguridad de sistemas reales.

### Ejemplo ilustrativo

Consideremos la secuencia $S = 11001010111100011101$ de longitud 20. El análisis es el siguiente:

| | |
|---|---|
| Número de unos | 11 |
| Número de ceros | 9 |
| Diferencia | 2 |
| Rachas de 1 bit | 5 |
| Rachas de 2 bits | 3 |
| Rachas de 3 bits | 1 |

Cuadro 2.1: Ejemplo de análisis de balance y rachas en una secuencia corta.

En este ejemplo, el balance se aproxima al ideal, las rachas siguen la proporción esperada y la autocorrelación en desplazamientos pequeños no revela patrones fuertes. Sin embargo, por ser una secuencia corta no se cumplen de forma exacta los postulados.

## Pruebas estadísticas del NIST

Para un análisis más exhaustivo, el NIST (National Institute of Standards and Technology) definió en el documento SP 800–22 una batería de quince pruebas diferentes.

Estas evalúan aspectos como equilibrio, distribución de patrones, entropía y ausencia de estructuras repetitivas. En este trabajo se seleccionan las siguientes, por su relevancia y relación directa con las debilidades de los LFSR:

- **Monobit**: comprueba que la proporción de unos y ceros es cercana al 50 %. El estadístico se define como

$$S_n = \sum_{i=1}^{n}(2x_i - 1), \quad p = \operatorname{erfc}\left(\frac{|S_n|}{\sqrt{2n}}\right).$$

- **Frecuencia por bloques**: divide la secuencia en bloques de tamaño $M$ y mide la proporción de unos en cada bloque. Detecta sesgos locales invisibles al test monobit global.

- **Rachas**: evalúa la cantidad y longitud de las rachas de bits iguales consecutivos. Una secuencia aleatoria debería presentar una distribución predecible de rachas cortas y largas.

- **Sumas acumulativas (CUSUM)**: interpreta la secuencia como una caminata aleatoria y mide la desviación máxima de la suma parcial respecto a cero. Indica si hay un sesgo acumulado hacia 0s o 1s.

- **Entropía aproximada**: compara la frecuencia de patrones de longitud $m$ y $m+1$. El p–valor bajo indica que algunos patrones aparecen con más frecuencia de lo esperado.

- **Maurer**: mide la compresibilidad de la secuencia. Una cadena con patrones repetidos puede comprimirse, mientras que una secuencia aleatoria no.

- **Serial**: analiza la distribución de todos los patrones de longitud $m$. Si ciertos patrones son anormalmente frecuentes o infrecuentes, la prueba falla.

*Cobertura de la batería.* Además de estas pruebas, el NIST incluye otras como el test de transformada discreta de Fourier (DFT), coincidencias en plantilla y complejidad lineal. Aunque en este trabajo no se aplican en su totalidad por razones prácticas, cada prueba está orientada a detectar patrones específicos que podrían pasar inadvertidos en otras. En consecuencia, la combinación de varios tests proporciona una visión más completa de la calidad pseudoaleatoria de un generador.

## Formalización matemática de algunas pruebas

**Entropía de Shannon.** La entropía mide la incertidumbre media de una variable aleatoria. Para una secuencia binaria $X$ con probabilidades $p_0$ y $p_1$ de aparición de 0 y 1,

respectivamente, la entropía se define como:

$$H(X) = -\sum_{i=0}^{1} p_i \log_2 p_i.$$

El valor máximo es $H(X) = 1$, alcanzado cuando $p_0 = p_1 = 0{,}5$. Desviaciones importantes de este valor reflejan sesgos en el generador.

**Contraste $\chi^2$ para frecuencia por bloques.** Sea una secuencia dividida en $N$ bloques de tamaño $M$. Para cada bloque $j$ se calcula $\pi_j$, la fracción de unos. El estadístico es:

$$\chi^2 = 4M \sum_{j=1}^{N} \left(\pi_j - \tfrac{1}{2}\right)^2,$$

que bajo la hipótesis nula de aleatoriedad sigue aproximadamente una distribución $\chi^2$ con $N$ grados de libertad. Un valor grande de $\chi^2$ indica desviaciones significativas en la frecuencia local de unos.

**Autocorrelación.** La función de autocorrelación para un desfase $k$ se define como:

$$R(k) = \frac{1}{N-k} \sum_{i=1}^{N-k} (2x_i - 1)(2x_{i+k} - 1).$$

Valores cercanos a cero sugieren independencia entre bits separados $k$ posiciones. Autocorrelaciones elevadas revelan estructura periódica o dependencia.

| Prueba | Parámetro | Rango recomendado | Condición de validez |
|---|---|---|---|
| Frecuencia por bloques | $M$ | 128, 512, 1024 | $N/M \geq 100$ |
| Entropía aproximada | $m$ | 4–6 | $N \gg 2^{m+1}$ |
| Serial | $m$ | 2–5 | $N \gg 2^m$ |
| CUSUM | – | – | $N$ grande (leyes límite) |

Cuadro 2.2: Guía práctica de parámetros típicos en pruebas NIST seleccionadas.

# Importancia de las pruebas estadísticas

El uso de baterías estadísticas no garantiza la seguridad criptográfica de un generador, pero sí actúa como un filtro necesario. Una secuencia que no supera estos tests presenta irregularidades explotables en ataques prácticos. Por ello:

- Superar los tests es una **condición necesaria** pero no suficiente para la seguridad.

- Fallar repetidamente en uno o varios tests indica que el generador **no debe utilizarse** en aplicaciones críticas.
- El análisis estadístico debe complementarse con métricas estructurales, como la complejidad lineal y el estudio de correlaciones.

*Contexto normativo.* En el ámbito académico, superar estas pruebas se considera un requisito mínimo para proponer un nuevo generador. En la práctica industrial, estándares como FIPS 140–3 exigen baterías adicionales (p. ej., TestU01, Diehard) e incluso pruebas específicas de dispositivo. Este marco regulatorio evidencia que la pseudoaleatoriedad es también un criterio de cumplimiento normativo.

En este TFG, las pruebas estadísticas sirven como primer paso para descartar configuraciones de LFSR con deficiencias evidentes antes de aplicar análisis más detallados.

## Interpretación de los p–valores

Cada prueba devuelve un **p–valor** en el rango $[0, 1]$. Si $p \geq \alpha$ (con $\alpha = 0{,}01$ en este trabajo), se considera que la secuencia *pasa* la prueba. Valores muy bajos ($p < 0{,}01$) indican que la secuencia muestra irregularidades estadísticamente significativas.

Por ejemplo, si una secuencia de 1000 bits contiene 700 unos y 300 ceros, el test monobit produce un p–valor prácticamente cero, lo que invalida el generador como fuente pseudoaleatoria.

## Tamaño muestral, potencia y comparaciones múltiples

La fiabilidad de los p–valores depende del tamaño muestral $N$. Con $N$ pequeño, los tests carecen de potencia para detectar sesgos sutiles; con $N$ grande, incluso desviaciones mínimas resultan "significativas". En este trabajo se fijan longitudes $N$ en el rango $[10^5, 10^6]$ para equilibrar coste computacional y sensibilidad.

**Potencia estadística.** La *potencia* es la probabilidad de rechazar la hipótesis nula cuando es falsa. Aumenta con $N$ y con el tamaño del efecto (p. ej., un sesgo $|p_1 - 0{,}5|$ mayor). Para el test monobit, desviaciones del orden $O(1/\sqrt{N})$ son detectables; por ejemplo, con $N = 10^6$, un sesgo de 0,001 ya puede generar p–valores muy bajos.

**Elección de parámetros por prueba.** En pruebas con parámetros (p. ej., frecuencia por bloques) se recomienda:

- **Frecuencia por bloques:** tamaños $M \in \{128, 512, 1024\}$, con $N/M \geq 100$ bloques para asegurar validez asintótica del contraste.
- **Entropía aproximada:** longitudes $m \in \{4, 5, 6\}$ según $N$; valores grandes de $m$ requieren $N$ muy superiores para evitar celdas con conteos casi nulos.

- **Serial:** mismo criterio que ApEn; aumentar $m$ solo si $N$ lo permite.

**Comparaciones múltiples.** Al aplicar varias pruebas (y repetidas semillas/configuraciones) se inflan los falsos positivos. Dos estrategias habituales:

- **Bonferroni:** usar un umbral $\alpha' = \alpha/K$ si se realizan $K$ tests independientes (conservador).
- **FDR (Benjamini–Hochberg):** controla la tasa de falsos descubrimientos y es menos conservador cuando $K$ es grande.

En este TFG se reportan tasas de aceptación por prueba y configuración, y se discuten los resultados en conjunto para mitigar conclusiones espurias.

**Repetición por semillas.** Se emplean múltiples semillas por configuración (p. ej., $r \geq 30$) y se reportan medias y desviaciones estándar de p–valores y tasas de aceptación. Este promediado reduce la varianza muestral y permite comparar configuraciones con mayor robustez.

# Resumen del capítulo

Se han introducido las nociones de aleatoriedad y pseudoaleatoriedad, clasificado los PRNG más comunes y discutido los postulados de Golomb y la batería NIST. También se formalizaron métricas básicas (entropía, $\chi^2$, autocorrelación) y se revisaron casos históricos donde una mala aleatoriedad comprometió sistemas reales. Estos elementos proporcionan el marco con el que, en capítulos posteriores, se evaluarán LFSR y combinadores.

# Relevancia para este trabajo

Los postulados de Golomb y los tests NIST seleccionados ofrecen un marco sólido para evaluar la calidad pseudoaleatoria de secuencias binarias. En este TFG se aplicarán a:

- Secuencias generadas por LFSR individuales.
- Secuencias producidas por combinadores clásicos: Shrinking, Geffe y Mayoría.

El objetivo es comprobar si los generadores combinados mejoran de forma significativa las propiedades pseudoaleatorias respecto a los LFSR simples, y si logran superar las limitaciones inherentes a su estructura lineal.

# Capítulo 3

# Optimización de la complejidad lineal

# Capítulo 4

# Análisis de resultados

# Capítulo 5

# Conclusiones