Análisis de la Independencia Estadística de las Pruebas de Aleatoriedad NIST SP 800-22

– –

Analysis of the Statistical Independence of the NIST SP 800-22 Randomness Tests

TRABAJO DE FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2019–2020

Bittor Alaña

Directores

Luis Javier García Villalba
Elena Salomé Almaraz Luengo

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Junio de 2020

Agradecimientos

Primero de todo querría agradecer a mis directores, Luis Javier García Villalba y Elena Salomé Almaraz Luengo, por introducirme en un área tan interesante como la criptografía, y por la ayuda prestada en el camino. Además, también agradecer a los profesores Julio Hernández-Castro, de la Universidad de Kent, y Darren Hurley-Smith, de la Universidad Royal Holloway de Londres, por guiarme en vías de investigación y ayudar en el proyecto. Gracias también a Esteban y a Ana, por ayudarme con aspectos técnicos y estéticos.

También es imprescindible destacar el apoyo constante de mis padres, mi hermano, mis amigos y mis compañeras de piso, personas que me han apoyado en todo este camino, y cuyo cariño ha sido fundamental para mantener el espíritu. Y por encima de todo muchísimas gracias a Guille, porque contigo empecé este viaje a Madrid, y sé que allá donde estés estarás orgulloso de mí, como siempre lo estuve yo de ti.

Este Trabajo Fin de Grado ha sido realizado dentro del Grupo de Análisis, Seguridad y Sistemas (Grupo GASS, https://gass.ucm.es/, Grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades del proyecto de investigación THEIA (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices) con referencia FEI-EU-19-04.

Índice General

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

3GPP LTE	*3rd Generation Partnership Project Long-Term Evolution*
AES	*Advanced Encryption Standard*
BBS	*Blum Blum Shub*
CoMP	*Coordinated Multipoint*
D2D	*Device to Device*
FFT	*Fast Fourier Transform*
FIPS	*Federal Information Processing Standards*
GPS	*Global Positioning System*
HetNets	*Heterogeneous Networks*
IoT	*Internet of Things*
LFSR	*Linear Feedback Shift Register*

LPWA	*Low Power Wide Area*
LTE	*Long-Term Evolution*
LTE-A	*Long-Term Evolution (Advanced)*
M-MIMO	*Massive Multiple Input Multiple Output*
M2M	*Machine to Machine*
MIMO	*Multiple Input Multiple Output*
NIST	*National Institute of Standards and Technology*
PRNG	*Pseudo-Random Number Generator*
RAT	*Radio Access Technology*
RF	Radiofrecuencia
RFID	*Radio Frequency Identification*
SHA-1	*Secure Hash Algorithm 1*
SP	*Special Publication*
TCL	*Teorema Central del Límite*
TRNG	*True Random Number Generator*
XOR	*Exclusive OR*

Abstract

One of the most critical aspects of encrypted communications consists on properly generating random numbers, either for straight use as a cipher or else to initialise other encryption algorithms. Nonetheless, assessing randomness is no easy task. In order to tackle this problem, statistical inference is used. Vulnerabilities of randomness are sought in sequences by means of hypothesis tests, through which the hypothesis that the sequence is random can be rejected. For these tests to yield real information on the sequence, the tests cannot be redundant, and should be independent. We have carried out an experiment with a big amount of data in order to assess whether the most widely used randomness tests battery, *National Institute of Standards and Technology* (NIST) *Special Publication* (SP) 800-22, really has independent tests, as the organisation claims to have. We have analysed the data to determine which tests offer more information, and which are redundant. Besides, we have tested for other flaws in the battery.

Keywords: Correlation, Cryptography, Data Analysis, Entropy, Hypothesis Tests, NIST SP 800-22, Randomness, Statistics.

Resumen

Uno de los aspectos críticos de la comunicación encriptada consiste en la buena generación de números aleatorios, tanto para usarlos directamente como cifrado como para poder inicializar otros métodos de encriptación. No obstante, medir un concepto como la aleatoriedad no es una tarea sencilla. Para tratar de solucionar este problema, se usa la inferencia estadística. Mediante test de hipótesis, se trata de buscar debilidades en una secuencia, que permitan rechazar la hipótesis de que esa secuencia es aleatoria. Para que estos test nos aporten información real sobre la secuencia, deben ser independientes entre sí. Hemos realizado un experimento con una gran cantidad de datos para determinar si la batería de test de aleatoriedad más ampliamente utilizada, la NIST SP 800-22, realmente cumple, tal como dice cumplir, con la independencia de sus test. Hemos analizado los datos para ver qué test aportan más información, y cuales pueden ser redundantes. Además, hemos puesto a prueba otros tipos de debilidades en la batería.

Palabras clave: Aleatoriedad, análisis de datos, criptografía, entropía, NIST SP 800-22, estadística, correlación, test de hipótesis.

Capítulo 1

Introducción

1.1. Motivación

Desde que existen las comunicaciones telemáticas, surgen a la par de las mismas posibles amenazas: interceptación de los mensajes, alteración, o incluso sabotaje de los mismos. Sabemos que, incluso en los suspiros finales de la República de Roma, los generales de Julio César enviaban sus cartas con un primitivo sistema de cifrado, para no revelar secretos en el caso de que fueran interceptadas.

El campo de la matemática que lidia con la protección de la confidencialidad, integridad y la autenticidad de mensajes se llama criptografía. A través de una serie de algoritmos, es posible ocultar mensajes en la vía entre el emisor y el receptor, de tal forma que el secreto de la comunicación esté asegurado.

Por este motivo, a medida que avanzan las tecnologías de comunicación, y se dan pasos agigantados hacia un mundo en el que todos los dispositivos del día a día estén interconectados, se precisa proteger comunicaciones partiendo de recursos muy limitados, ya que estos dispositivos pueden estar manejando información de alta sensibilidad. Es así como surgen en la literatura académica numerosas propuestas para poder proteger comunicaciones entre dispositivos con pocos recursos hardware. Uno de los mecanismos más básicos para este fin, y también para otros procedimientos de comunicación, es generar números aleatorios.

Un campo emergente con enorme potencial es el del conocido como Internet de las Cosas *Internet of Things* (IoT). Se plantea un escenario donde miles de millones de dispositivos muy pequeños [ASHAM17], y con recursos y consumo reducido, estén interconectados y compartiendo información, para poder mejorar nuestro día a día desde nuestras casas hasta la Industria 4.0. Se espera que los avances de las redes 5G supongan un incremento espectacular de esta tecnología, y como tal urge proteger las comunicaciones de estos aparatos, especialmente teniendo en cuenta que van a trabajar con datos muy sensibles como información de salud, secretos de empresa, estado del hogar... Y una de las formas más ligeras de ofrecer un cifrado para estas comunicaciones es la generación de números aleatorios.

Aunque pueda parecer trivial generar números que parezcan aleatorios en un ordenador o un sistema integrado, este no es para nada el caso. Por su propia naturaleza, los computadores son instrumentos deterministas que se rigen por algoritmos. De esta forma, cualquier procedimiento que una persona pueda idear para crear números aleatorios corre

el riesgo de ser fácilmente predecible.

No obstante, esta predictibilidad juega en dos direcciones. Por una parte, que una secuencia sea fácilmente predecible supone un problema si se quiere utilizar como si fuera una clave secreta. Por otra parte, si se pudiera, a partir de un secreto inicial, reproducir la secuencia de la misma manera en dos sitios diferentes, esto sí sería una propiedad deseable. Esto permitiría que dos agentes que compartieran el secreto inicial, pudieran reproducir la misma secuencia aparentemente aleatoria en dos lugares completamente separados, y utilizar esos números aleatorios que solo ellos conocieran para tener comunicaciones cifradas.

Tanto obtener números verdaderamente aleatorios como pseudoaleatorios, que es como nos referimos a aquellos que se obtienen de una forma determinista a partir de un secreto inicial, es una tarea ardua. Una parte fundamental es medir si, una vez realizado un generador, su salida se comporta realmente de una manera aleatoria. En términos estadísticos, se pretende dar una métrica de cuán robustas son las secuencias generadas, para que generen secuencias que no se pueden predecir sin conocer el secreto inicial, que cualquier número tenga la misma probabilidad de aparecer en la secuencia, y un largo et cétera.

Esta es la tarea de las baterías de pruebas de aleatoriedad. Deciden si ciertas secuencias se comportan de manera aleatoria, o no. Utilizan lo que llamamos estadísticos, que son sencillamente funciones que toman como entrada la secuencia de un generador de números, y obtienen un valor a partir de la secuencia. Una vez se tienen estos estadísticos, como se pueden demostrar teóricamente los valores que se esperan obtener para esos estadísticos en secuencias aleatorias, se puede cuantificar la probabilidad de que la secuencia observada se esté comportando efectivamente como una secuencia aleatoria.

Por lo tanto, cada vez que se hace una de estas pruebas de aleatoriedad, con un estadístico específico, el estadístico está preparado para estudiar una propiedad particular. Puede ser la cantidad de ceros y unos que tiene la secuencia, el ritmo en el que oscilan los ceros y los unos, la media de la secuencia tratándola como cadena de bytes sin signo... Por lo tanto, si cada prueba busca una vulnerabilidad, pasa lo siguiente: por una parte, pasar todas las pruebas satisfactoriamente quiere decir que no se le han encontrado vulnerabilidades a la secuencia, con lo cual podremos decir que no tiene debilidad conocida (pero ello no quiere decir que sea necesariamente perfectamente aleatoria), y por otra parte, es fundamental que las debilidades que detecten las pruebas no se solapen.

Las pruebas deben ser independientes. Si tuviéramos una batería de cuatro pruebas, pero tres de ellas fueran altamente interdependientes, y midieran esencialmente la misma debilidad, aprobar las tres baterías daría una falsa sensación de seguridad, ya que realmente solo un tipo de debilidad sería la evitada. Recíprocamente, fallar las tres parecería catastrófico, pero realmente solo apuntaría a un tipo de fallo, esencialmente.

Por ello nos proponemos estudiar una de las baterías de pruebas de aleatoriedad más utilizadas actualmente, con un enfoque de uso criptográfico, la SP 800-22 de NIST. Tras estudiar a fondo el estado del arte, hemos realizado una investigación experimental, tratando una gran cantidad de datos, y análizandolos desde una perspectiva de inferencia estadística, para poder obtener resultados estadísticamente significativos de la información real que aportan dichas pruebas.

1.2. Contexto

Este Trabajo Fin de Grado ha sido realizado dentro del Grupo de Análisis, Seguridad y Sistemas (Grupo GASS, https://gass.ucm.es/, Grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades del proyecto de investigación THEIA (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices) con referencia FEI-EU-19-04.

1.3. Objeto de la Investigación

La SP 800-22 de NIST es la batería de pruebas aleatorias más establecida y sometida a mayor escrutinio en criptografía. Como tal, al ser probablemente la mayor referencia a la hora de medir la aleatoriedad, es de vital importancia que la información que aporte sea significativa. Esto quiere decir que las pruebas deberían ser independientes, y en [BRS+10] se sostiene que este es el caso.

No obstante, ha habido varias investigaciones poniendo a prueba la supuesta independencia de los test, arrojando diversos resultados, no necesariamente consistentes entre unas publicaciones y otras. Nosotros nos hemos propuesto, partiendo de las investigaciones realizadas a día de hoy, poder entenderlas, replicarlas, y avanzar en la interpretación estadística de resultados.

Desde un punto de vista de inferencia estadística, nos hemos propuesto cuantificar en qué niveles de confianza podemos decidir si los test son o no independientes, hemos ofrecido una explicación razonada de por qué se dan las dependencias observadas, y hemos propuesto qué decisiones tomar al respecto, y por qué. Además, hemos mirado más a fondo en las variantes de los test, y prestado atención al coste computacional que tienen las pruebas.

1.4. Plan de Trabajo

El desarrollo de este trabajo se ha realizado en tres fases:

1. **Investigación**: al inicio, nos propusimos estudiar diferentes implementaciones generadores de números pseudoaleatorios, especialmente para sistemas de *Radio Frequency Identification* (RFID). Después, y tras ver qué métricas se utilizaban para juzgar a estos generadores, nos centramos en el estudio de los test estadísticos de aleatoriedad, a sugerencia del profesor Julio Hernández-Castro. Nos centramos especialmente en la batería NIST SP 800-22, por ser la más utilizada en criptografía, dado su alto grado de exigencia, y por estar respaldada por una institución de estándares de gran reputación.

2. **Desarrollo**: entonces pasamos a generar datos para el experimento. Este proceso fue llevado a cabo en un ordenador de la Facultad de Ciencias Matemáticas. A raíz del confinamiento hubo que pasar a hacerlo en nuestros ordenadores personales, lo cual nos atrasó notablemente. Se generaron scripts en C y Python para generar datos, poder ejecutar los test automáticamente sobre los archivos, tratándolo como secuencias de distintas longitudes, y para almacenar esos datos en tablas para su

posterior análisis estadístico. También se probaron dos fuentes de aleatoriedad, para poder contrastar resultados.

3. **Análisis estadístico e interpretación**: durante los últimos meses, se realizó la interpretación de todos los datos obtenidos, viendo qué dependencias se obtenían entre los test de la batería, y qué informaciones redundantes se aportaban.

La Tabla 1.1 muestra las actividades realizadas durante el desarrollo del proyecto.

Tabla 1.1: Actividades del Plan de Trabajo

Mes	**Actividad**
Octubre	Estudio y simulación de generadores de números aleatorios para RFID.
Noviembre	
Diciembre	
Enero	Generación de los datos para el test. Scripting y procesamiento.
Febrero	
Marzo	Análisis de los datos obtenidos y tratamiento estadístico.
Abril	
Mayo	
Junio	

1.5. Estructura del Trabajo

De aquí en adelante, el trabajo está organizado en 7 capítulos:

El Capítulo 2 establece el marco conceptual en el que se sitúa el trabajo. Es decir, una vez expuestas la motivación y los objetivos del trabajo, nos proponemos detallar algunos conceptos básicos de criptografía e IoT. Introducimos algunos de los aspectos físicos que demanda la tecnología IoT, algunas soluciones que se dan al problema de la encriptación ligera, y conceptos básicos sobre generadores aleatorios y pseudoaleatorios, con uno de los ejemplos más relevantes en el mundo de las telecomunicaciones, como son los *Linear Feedback Shift Register* (LFSR).

El Capítulo 3 está dedicado a explicar cómo se mide la aleatoriedad en términos estadísticos, y en particular cómo lo hace la batería NIST SP 800-22. Introducimos todos los test, tanto con una explicación informal de sus objetivos –para transmitir al lector qué debilidades busca cada test–, como una formulación matemática de la toma de estadísticos, y la obtención de probabilidades a partir de estos.

El Capítulo 4 detalla las investigaciones realizadas sobre la dependencia de los test de la batería. Explicamos qué aportan las investigaciones, y qué conclusiones permiten sacar en claro. Además, explicamos la evolución que ha tenido la batería a lo largo del tiempo, desde sus inicios en 2001 hasta hoy día. A partir de los aciertos de los otros estudios, y las

áreas que dejan sin cubrir, vemos dónde podemos avanzar en nuestra propia investigación, y qué cosas nuevas podemos aportar.

El Capítulo 5 detalla nuestro estudio, desde la parte experimental al análisis estadístico. Explicamos brevemente qué datos hemos generado y estudiado, y después nos centramos en los resultados, y en una interpretación del significado de los mismos. Explicamos qué tratamiento hemos dado a los datos, el porqué de las transformaciones a las que se han sometido, y relacionamos los resultados obtenidos con los otros resultados presentes en la literatura.

El Capítulo 6 muestra las principales conclusiones de este trabajo, así como las posibles vías de investigación en el futuro para ahondar en la temática.

Los Capítulos 7 y 8 son las traducciones al inglés de la Introducción y de las Conclusiones.

Capítulo 2

Marco Conceptual

2.1. Dispositivos de IoT

El concepto de IoT consiste en conectar una gran cantidad de pequeños aparatos heterogéneos y diversos en una misma arquitectura de red. A partir de esta idea se pueden plantear ciudades inteligentes, redes eléctricas inteligentes, hogares inteligentes, sistemas agronómicos automonitorizados, sistemas de cuidado de salud y control de constantes. A continuación se detallan algunas de las aplicaciones más importantes que puede tener esta tecnología.

2.1.1. Smart Homes

La idea de la domótica inteligente o Smart Home es un hogar que dispone de dispositivos conectados a Internet que pueden realizar decisiones autónomas, basándose en información recibida por sensores. Todo esto permite dar facilidades a mejorar la vida diaria del usuario facilitando el control de los electrodomésticos y demás sistemas [ASHAM17]. Esto genera la necesidad de conectar muchos aparatos en un entorno no muy grande, y con poca exigencia de latencia. Sin embargo, es deseable un gasto energético pequeño, para que aparatos muy pequeños alimentados con baterías puedan durar largos periodos.

2.1.2. Sistemas de transporte inteligente

Los sistemas de transporte inteligente tienen por objetivo asegurar la monitorización y control de las redes de transporte. Utilizando una red compuesta por subsistemas de vehículos (como el *Global Positioning System* (GPS)), identificación por radiofrecuencia RFID, y subsistemas estacionarios como máquinas en carretera, se busca garantizar la seguridad, fiabilidad, disponibilidad y eficiencia de la red de transportes. También puede servir para asegurar la cadena de frío del transporte de alimentos y medicinas. El auge de los coches autónomos tiene el potencial de catalizar este uso particular de IoT.

2.1.3. Smart City

La gestión de la situación ambiental, de tráfico, de la red eléctrica, del agua, de la gestión de residuos, y de muchos otros fenómenos en tiempo real puede ayudar a crear ciudades inteligentes. Esto requiere de conexiones a una distancia considerable. No obstante, la latencia no es especialmente crítica para estos aspectos.

2.1.4. Industria

Para aplicaciones industriales, es fundamental la fiabilidad de los datos. Además, hay limitaciones de tiempo mayores, exigiendo latencias extremadamente bajas. Dos de los mayores impedimentos al IoT industrial son la incapacidad de las redes inalámbricas de bajo consumo de ofrecer fiabilidad, y que el protocolo IP para conexiones *end-to-end* no está adaptado para dispositivos hoja limitados.

2.1.5. Atención sanitaria inteligente

La tecnología IoT puede ser de gran ayuda a la hora de tomar constantes vitales, tomar la temperatura, presión arterial, nivel de colesterol, ritmo cardíaco, etc. Si se logra automatizar esta toma de datos y comunicarlos y procesarlos de forma adecuada, se da pie a un sistema de fácil detección de anomalías, pudiendo evaluar la evolución de los pacientes de una forma mucho más automática. Además, esta toma de datos aporta mucha información de cara a algoritmos de inteligencia artificial y aprendizaje automático.

2.2. Redes 5G

Hasta la actualidad se han utilizado tecnologías de redes móviles como 3G, 4G para interconectar pequeños dispositivos. Se trabaja, sobre todo, sobre el estándar *3rd Generation Partnership Project Long-Term Evolution* (3GPP LTE) para acometer las necesidades de conectividad de la IoT. Se están introduciendo mejoras en la comunicación máquina a máquina *Machine to Machine* (M2M), y desarrollándose la tecnología NarrowBand-IoT. Para poder habilitar los casos de uso con utilización masiva y crítica de IoT, se debe mejorar en el concepto *Low Power Wide Area* (LPWA), además de ofrecer gran fiabilidad, siendo capaz de soportar latencia extremadamente baja y rendimiento alto [ASHAM17].

En la capa física, las redes 5G utilizan algunas de las propiedades estandarizadas en la tecnología *Long-Term Evolution* (LTE) y *Long-Term Evolution (Advanced)* (LTE-A). Se trata de los siguientes conceptos [SKS+20]:

- Agregación de canales. Fue introducida en 4G LTE-A, y agrega hasta cinco canales de 20 MHz para generar un ancho de banda total de hasta 100 MHz.
- *Multiple Input Multiple Output* (MIMO) masivo (*Massive Multiple Input Multiple Output* (M-MIMO)). Esta tecnología consiste en multiplexación espacial. Flujos de datos son multiplexados en varias antenas, siendo transmitidos a través de canales separados. M-MIMO consigue bajar la latencia de la señal, aumentar capacidad, y proteger frente a interferencias [SKS+20].

- Procesamiento multipunto coordinado *Coordinated Multipoint* (CoMP). Se trata de utilizar MIMO de forma distribuida en diferentes antenas, de tal forma que se aumenta el rendimiento del dispositivo final, se reduce la interferencia y se aumenta la calidad de la señal recibida.
- Redes heterogéneas *Heterogeneous Networks* (HetNets). Las redes estarán compuestas de diferentes capas de células y tecnologías de acceso por radio *Radio Access Technology* (RAT).
- Comunicaciones dispositivo a dispositivo *Device to Device* (D2D). Si bien HetNets coordina estaciones base de alta y baja potencia, no es eficiente para comunicaciones de corto alcance. Por ello, se busca desarrollar una comunicación directa dispositivo a dispositivo, para comunicaciones de menos de 200 metros. Sin embargo, esto puede suponer problemas de seguridad, que es a lo que nos referimos a continuación.

2.3. El impacto de las redes 5G y los dispositivos IoT en la seguridad

Los dispositivos IoT y la comunicación dispositivo a dispositivo D2D plantean un reto para la seguridad: se estarán comunicando una cantidad enorme de datos potencialmente sensibles, o confidenciales, siendo vulnerables a ser interceptados, modificados, o saboteados.

Al igual que los dispositivos de identificación por radiofrecuencia RFID, será de gran importancia que los equipos de IoT se puedan comunicar con garantías de autenticidad, confidencialidad, integridad y seguridad.

Para obtener estas propiedades, una de las opciones de encriptación elegidas es la del cifrado de flujo.

2.3.1. Cifrado de flujo

El cifrado de flujo es un sistema criptográfico de clave simétrica. Esto es, la misma clave sirve para encriptar y desencriptar la información [Was]. Los sistemas simétricos tienen velocidades significativamente mayores a los asimétricos.

Hay dos tipos principales de sistemas simétricos de cifrado:

- Cifrados de bloque. Tratan la información en bloques, y la función de encriptación actúa sobre un bloque entero. La función encriptado/desencriptado transforma bloques en bloques.
- Cifrados de flujo. La información se trata a nivel de bit. Estos cifrados mantienen un cierto estado, que va cambiando el algoritmo de cifrado. Se puede modelizar con dos funciones:

$$\sigma_{i+1} = f(\sigma_i, b_i, k)$$
$$c_i = g(\sigma_i, b_i, k).$$

Tenemos que el estado del cifrado σ_{i+1} cambia según el bit b_i y la clave k, y para cifrar un bit se utiliza el estado anteriormente obtenido y la clave.

En la práctica, una de las formas más comúnmente utilizadas y más rápidas de aplicar cifrados de flujo es realizar operaciones de disyunción exclusiva *Exclusive OR* (XOR) sobre los bits, operación que denotaremos $\oplus$. Nótese que, si tenemos una serie de bits b_i y una clave k_i, $i = 0, ..., n-1$, entonces podemos encriptar y desencriptar cada bit aplicando un XOR.

A las funciones encriptar y desencriptar un bit b_i dado el bit k_i de la clave las denotamos, respectivamente, $\mathcal{E}$ y $\mathcal{D}$, y actúan de la siguiente manera: $\mathcal{E}(b_i) = b_i \oplus k_i$ y $\mathcal{D}(c_i) = c_i \oplus k_i$. Observamos que, efectivamente, descifrar el mensaje cifrado devuelve el mensaje original:

$$\mathcal{D}(\mathcal{E}(b_i)) = (b_i \oplus k_i) \oplus k_i = b_i \oplus (k_i \oplus k_i) = b_i \oplus 0 = b_i$$

Un ejemplo sencillo:

Mensaje m	010111
Clave k	011010
$c = \mathcal{E}(m)$	001101
$\mathcal{D}(c)$	010111

Por lo tanto, un cifrado de flujo puede ir generando la clave y realizando la disyunción exclusiva de forma simultánea, y muy rápidamente, ya que esta operación es extremadamente barata computacionalmente. Es por ello que, para sistemas de capacidad hardware restringida, como dispositivos IoT y etiquetas de RFID, resulta una opción adecuada.

Para generar las claves de encriptación al instante, y también para poder realizar protocolos de autenticación, estos dispositivos pueden equiparse con generadores de números pseudoaleatorios.

2.4. Generadores de números verdaderamente aleatorios

A la hora de generar números aleatorios, es difícil encontrar fuentes de aleatoriedad auténtica. Un ordenador puede generar secuencias mediante algoritmos, pero estos son deterministas, y no proceden de fenómenos intrínsecamente aleatorios. Es por ello que se suelen usar como fuentes de aleatoriedad fenómenos físicos. Por ejemplo, en el proyecto *RANDOM.ORG* [Haa98] lo hacen con la aleatoriedad de la meteorología. En cambio, la mayoría de los ordenadores utilizan el *jitter*, que es el ruido de la periodicidad del reloj, o también se usa el ruido térmico de los elementos electrónicos.

Definición 2.4.1 *Llamamos True Random Number Generator (TRNG) a un dispositivo o método de extraer números aleatorios uniformemente distribuidos a partir de fenómenos con aleatoriedad intrínseca*

2.5. Generadores de números pseudoaleatorios

El mayor problema con los TRNG es que suele ser caro obtener bits aleatorios de estas fuentes, y para muchas aplicaciones se requiere generar una gran cantidad de bits en un tiempo reducido, o con recursos hardware muy limitados. Por lo tanto, a partir de una cierta cantidad de información aleatoria, se busca extender esta información y generar secuencias muy largas de números aleatorios de alguna forma.

Esa es la función de los generadores de números pseudoaleatorios o *Pseudo-Random Number Generator* (PRNG). Reciben como entrada una **semilla**, que suele provenir de un TRNG, y se generan, algorítmicamente, secuencias muy largas de números en un tiempo reducido, mediante algoritmos. Las salidas de los PRNG suelen ser funciones determinstas respecto a la semilla.

Definición 2.5.1 *Llamamos PRNG a dispositivos o algoritmos que, dados una entrada llamada* ***semilla****, generan largas secuencias de números de apariencia aleatoria.*

2.5.1. Linear Feedback Shift Register

Un ejemplo básico de PRNG es el conocido como registro de desplazamiento con retroalimentación lineal, LFSR. Un LFSR es un registro de desplazamiento donde, para suplir al bit que se pierde en cada ciclo, se toma como bit de entrada la suma módulo dos de un cierto conjunto de bits. Cada n ciclos, se produce una salida de un elemento de n bits. Así funciona la transición de ciclos: sea

$$s = (s_0, ..., s_{n-1})$$

el estado del registro. Entonces, en el siguiente ciclo se tiene

$$s' = (c, s_0, ..., s_{n-2}) \qquad c = \sum_{i=0}^{n-1} c_{n-i-1} s_i \mod 2 \qquad c_j \in \{0, 1\} \ \forall j.$$

Nótese que sumar bits módulo 2 equivale a realizar la operación XOR o disyunción exclusiva entre ellos.

Al polinomio $p(x) = \sum_{i=0}^{n-1} c_i x^i + x^n$ con coeficientes en el cuerpo $\mathbb{F}_2$, es decir los enteros módulo 2, se le llama polinomio característico del LFSR. Está demostrado [ANSS15] que si p es un polinomio primitivo, y el estado inicial, es decir la semilla, es no nulo, entonces este generador produce una secuencia de periodo $2^n - 1$, es decir, produce $2^n - 1$ números antes de empezar a repetir la salida.

Esto hace que sea un generador con poca exigencia de hardware, muy rápido y de coste extremadamente reducido, para poder usarse en sistemas embebidos o circuitos integrados. Sin embargo, al retroalimentarse y generar salidas con una relación lineal, se presentan debilidades estadísticas. Su carácter lineal hace que se puedan inferir los coeficientes del polinomio a través de un sistema de ecuaciones.

Sin embargo, a partir de esta construcción sencilla se pueden producir generadores más complejos y más seguros.

2.5.2. Combinación de TRNGs y PRNGs para proteger las comunicaciones

El proceso de creación de seguridad criptográfica es complejo. Uno de los mayores retos en el proceso es conseguir una fuente de aleatoriedad verdadera [AJ]. Al ser la seguridad de un dispositivo de IoT toda una cadena de varios componentes, basta con que un eslabón de la misma sea debilitado para poner en compromiso toda la seguridad. La criptografía es, por tanto, un elemento fundamental de la seguridad, aunque no el único.

La encriptación consiste esencialmente en guardar un secreto. Proteger este secreto es fundamental para la seguridad del sistema. De estos secretos requerimos las siguientes características:

- **Impredictibilidad**. Que no se pueda conocer el secreto de antemano.
- **Distribución uniforme**. Todos los números tienen la misma probabilidad de aparecer.
- **Independencia estadística**. El secreto actualmente no depende de los anteriores secretos, y estos no pueden aportar información que ayude a revelar el actual.

En el sistema dual TRNG/PRNG que hemos presentado, el secreto viene dado por un TRNG, que aporta un número verdaderamente aleatorio, y es este el que se utiliza como semilla para el PRNG. Se debe cambiar frecuentemente de semilla, y el patrón de semillas debe ser irreconocible. Quien conozca la semilla, que es el secreto, puede reproducir la secuencia que se genere, ya que los PRNG son, esencialmente, deterministas. Por lo tanto, es vital que tanto el TRNG como el PRNG cumplan estrictos requisitos de seguridad. Si el PRNG fuera vulnerable, uno podría hallar el secreto, conocer su estado interior y predecir la salida. Si en cambio, el TRNG tuviera problemas, sería posible tratar de averiguar las semillas, y a partir de ellas se podría reproducir la secuencia de números que devuelve el PRNG.

Para la apariencia de impredictibilidad y la uniformidad, tenemos la batería que nos ocupa [BRS+10]. Para la independencia estadística del secreto, y cómo obtenerlo a partir de fuentes verdaderas de aleatoriedad, se da una serie de directrices en otra publicación de NIST, en [TBK+18].

Hay diversas maneras de atacar a los TRNG. Algunos de estos ataques se realizan desde el plano físico. Desde estudiar la energía de radiofrecuencia que emite el chip, desencapsular el hardware para exponer la configuración de puertas lógicas, obtener información desde memoria no volátil... A los ataques de este tipo se les suele denominar *side-channel attack*.

Para evitar este tipo de ataques, un TRNG debe evitar emitir energía de Radiofrecuencia (RF) detectable, y de cierta forma enmascarar su actividad. Su salida debería permanecer impasible ante cambios en tiempo, voltaje, calor, y otros fenómenos.

Idealmente, la entrada de entropía, o de aleatoriedad real de un TRNG debería ser una variable aleatoria uniformemente distribuida y sin sesgos. Sin embargo, las fuentes de que se extrae la entropía, como el ruido térmico que hay en los circuitos, pequeños defectos hardware de fabricación, y otras posibilidades, suelen tener algunos sesgos. Por ello, a veces se les aplica un proceso de condicionamiento de eliminación de sesgo, llamado *whitening*.

Capítulo 3

Test Estadísticos

Medir cuán aleatoria es una secuencia de números es un problema estadístico complejo. Para dar por bueno un generador de números aleatorios, se le aplican diferentes baterías de test estadísticos, los cuales tratan de buscar ciertas debilidades en las secuencias. Uno de los precursores más reseñables de estas baterías es el *Federal Information Processing Standards* (FIPS) 140-2 [Cad11].

Hoy en día, algunas de las baterías más conocidas son las siguientes:

- NIST SP 800-22 [BRS+10]
- ENT (de Entropía) [Wal08]
- Dieharder [Bro], llevada a cabo por Robert G. Brown. Es el sucesor de la batería Diehard, la cual deja obsoleta.

Como hemos comentado, medir la aleatoriedad es extremadamente complicado, por eso a la hora de evaluar secuencias estas baterías tratan de buscar ciertas vulnerabilidades específicas que permitan concluir que una secuencia no es aleatoria, y se dan por buenos aquellos generadores a los que no se les encuentre debilidades. La formalización matemática de estas pruebas son los test de hipótesis.

3.1. Test de hipótesis

Un test de hipótesis es una prueba estadística que pone a prueba una cierta **hipótesis nula**, H_0. En nuestro caso, la hipótesis es que la secuencia de números es aleatoria. Esto quiere decir que la cadena de ceros y unos proviene de n variables aleatorias independientes e idénticamente distribuidas, con una distribución de Bernoulli con parámetro $\theta = 0.5$ [BRS+10]. Es decir, que cada bit de la secuencia es equivalente a lanzar una moneda no sesgada al aire y tomar unos por caras y ceros por cruces, o viceversa. La hipótesis alternativa es el complementario de la hipótesis nula, H_a. En nuestro caso, significa que la secuencia no es aleatoria.

La forma de proceder es la siguiente:

- Tomar un estadístico (es, sencillamente, una función de la secuencia).
- Hallar la distribución probabilística que tiene esa función, bajo la hipótesis nula.
- Calcular la probabilidad de obtener el estadístico que se ha observado, *o uno más extremo*, en bajo la hipótesis nula. A esta probabilidad se le llama **p-valor**.
- Si el **p-valor** es menor que un número α, rechazar la hipótesis nula. Si no es así, no rechazarla. El valor de α lo elegimos nosotros y se llama *nivel de significancia*.

Para usos criptográficos, se suele utilizar α entre 0.001 y 0.01 [BRS+10]. Intuitivamente, esta es la proporción de veces que se permite fallar el test cuando la hipótesis es cierta. Este test puede resultar en dos tipos de fallos distintos. El primero es el recién mencionado, rechazar la hipótesis nula cuando esta es cierta, y se llama error de Tipo I. El otro es no rechazar la hipótesis nula cuando esta no es cierta, y se llama error de Tipo II. Véase la Tabla 3.1.

Tabla 3.1: Tipos de errores en test de hipótesis.

Situación real	**Resultado test**	
	No rechazar H_0	Rechazar H_0 (Aceptar H_a)
H_0 es cierta	No hay error	Error Tipo I
H_a es cierta	Error Tipo II	No hay error

3.2. NIST SP 800-22

La publicación especial 800-22 del NIST es una batería de test estadísticos introducida por primera vez en 2001, y es seguramente la prueba más importante para medir la aleatoriedad de PRNGs y TRNGs con usos criptográficos. Es ampliamente utilizada y muy estudiada.

En su versión actual [BRS+10], la batería consta de 15 pruebas a partir de test de hipótesis, más variantes de estas pruebas. Las pruebas se dividen en dos tipos:

- **Pruebas de dos colas**. En estos test, se toman unos estadísticos que, bajo hipótesis de aleatoriedad, aproximan una distribución normal estándar $\mathcal{N}(0,1)$ por el Teorema Central del Límite [CLT]. Por lo tanto, una vez tomada la muestra observada, se mide la probabilidad de los valores que tienen valor absoluto mayor que la muestra, sumando las dos colas, en el extremo positivo y negativo. Este es el p-valor que se compara con α. Se muestra en la Figura 3.1, de [ZML+16].
- **Pruebas de una cola**. En estos test, se divide la secuencia en N bloques. Se crean unas categorías en las que, dependiendo de cierta característica de los bloques, se hace una partición de los mismos. Tras hallar cuántos bloques se puede esperar tener en cada categoría bajo hipótesis de aleatoriedad, se procede a realizar una prueba de bondad de ajuste, con una distribución χ^2, que mide cuán plausible es obtener los resultados observados. Este es el caso de la Figura 3.2, también de [ZML+16].

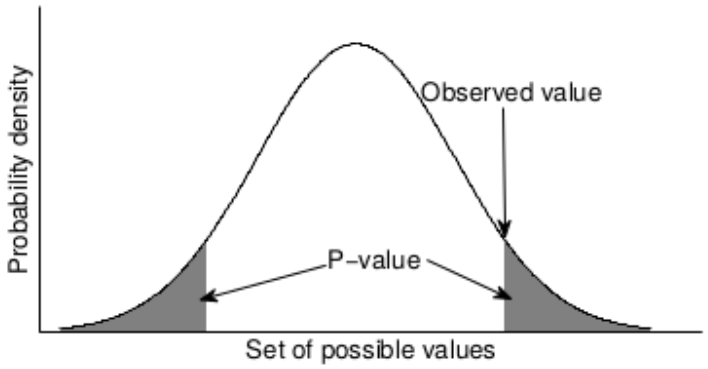


Figura 3.1: Test de dos colas.

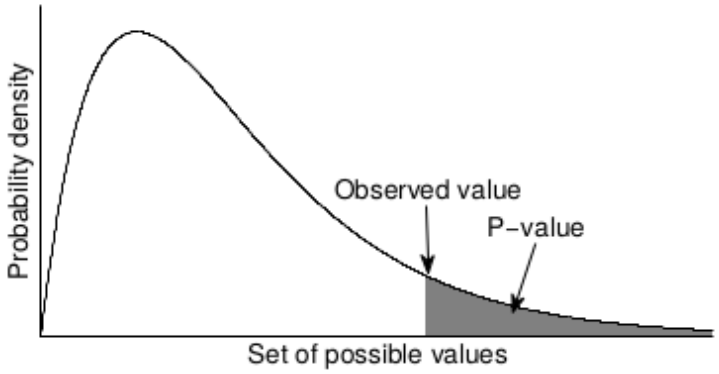


Figura 3.2: Test de una cola.

Procedemos a enumerar y explicar brevemente los test. Denotaremos la secuencia a evaluar por $\epsilon = \epsilon_0 ... \epsilon_{n-1}$, con $\epsilon_i \in \{0, 1\}\ \forall\, i$.

3.2.1. Test de frecuencia

Este test pone a prueba la proporción de ceros y unos en la secuencia. Primero hacemos la transformación $X_i = 2 \cdot \epsilon_i - 1 \in \{-1, 1\}$. Después, realizamos la suma $S_n = X_0 + ... + X_{n-1}$. Por el *Teorema Central del Límite* (TCL) [CLT], se tiene que

$$\lim_{n\to\infty} P\left(\frac{S_n}{\sqrt{n}} \leq z\right) = \Phi(z)$$

donde Φ es la función de distribución acumulada de la distribución normal estándar $\mathcal{N}(0, 1)$.

Por lo tanto

$$\lim_{n\to\infty} P\left(\frac{|S_n|}{\sqrt{n}} \leq z\right) = 2\Phi(z) - 1.$$

Entonces si nuestro valor observado es $z = \frac{|X_0+...+X_{n-1}|}{\sqrt{n}}$, obtenemos el p-valor $2\Phi(z) - 1$ de dos colas. Si este valor es menor que α, por defecto $\alpha = 0.01$, la secuencia falla el test. En caso contrario, la secuencia pasa el test. En adelante, el test se denominará *Frequency*.

3.2.2. Test de frecuencia por bloques

Parecido al anterior test, en este se trata de ver la frecuencia de forma local, dividiendo toda la secuencia en bloques, y realizando una prueba de bondad de ajuste.

La secuencia se divide en N bloques de M bits. Se denota por π_i: $\frac{1}{M}\sum_{j=0,...,M-1} \epsilon_{M\cdot i+j}$ la proporción de unos en el bloque $i \in \{0, ..., N-1\}$.

Dado que cada bloque tiene, bajo hipótesis de aleatoriedad, una proporción de unos de $1/2$, se puede obtener el siguiente estadístico:

$$X^2(obs) = 4M \sum_{i=0}^{N-1} \left[\pi_i - \frac{1}{2}\right]^2.$$

Bajo la hipótesis nula de aleatoriedad, el estadístico proviene de una distribución χ^2 de N grados de libertad, y se puede calcular la probabilidad $P(\chi^2_N \geq X^2(obs))$ para realizar una prueba de una cola. Si esa cola tiene valor menor que α, se rechaza la hipótesis de aleatoriedad para la secuencia. Si no, no se rechaza. Nos referiremos al test como *BlockFrequency*.

3.2.3. Test de rachas

Una racha es una subcadena de la secuencia compuesta únicamente por 0s o 1s. Se mide si la oscilación entre cadenas de ceros y unos es demasiado lenta o rápida. Como se conocen la media y la varianza del número de rachas esperado, se tiene

$$\lim_{n\to\infty} P\left(\frac{V_n - 2n\pi(1-\pi)}{2\sqrt{n}\pi(1-\pi)} \leq z\right) = \Phi(z)$$

donde V_n es el número de rachas en la secuencia, y π la proporción de unos en la secuencia.

Por lo tanto, podemos tomar el valor observado $z = \frac{|V_n - 2n\pi(1-\pi)|}{2\sqrt{n}\pi(1-\pi)}$, y se tiene el p-valor de dos colas $2\Phi(z) - 1$, procediendo como en el test de la Sección 3.2.1. También lo denotaremos *Runs*.

3.2.4. Test de racha de unos más larga

Para este test se divide la secuencia en N bloques de M bits. A continuación, se mide la racha de unos más larga que hay en cada bloque.

Se crean $K+1$ categorías. Para $i = 0, .., K-1$, ν_i es la cantidad de bloques cuya racha de unos más larga tiene longitud i. ν_K es la cantidad de bloques cuya racha más larga tiene longitud mayor o igual que K. Nótese que entonces, $\sum_{i=0}^{K} \nu_i = N$.

De esta forma, y como se conocen las probabilidades teóricas de que un bloque aleatorio pertenezca a las clases $\nu_0, ..., \nu_K$, denotadas $\pi_0, .., \pi_K$, se puede proceder por bondad de ajuste:

$$X^2 = \sum_{i=0}^{K} \frac{(\nu_i - N\pi_i)^2}{N\pi_i}.$$

El p-valor se obtiene de la distribución χ^2 de K grados de libertad, y se procede como en el test de la Sección 3.2.2. Nos dirigiremos a este test como *LongestRun*.

3.2.5. Test de rango de matriz binaria

Este test comprueba dependencias lineales entre subcadenas de la secuencia. Se construyen matrices a partir de ella, y se hallan sus rangos.

Se toman N matrices cuadradas $M \times M$, y se crean tres categorías: las de rango igual a M, las de rango igual a $M-1$, y las de rango menor que $M-1$. Como se conocen sus probabilidades analíticas, se procede con una prueba de una cola:

- F_m : de todas las $N = [n/(M \cdot M)]$ matrices, el número de las que tienen rango máximo m (con $[x]$ nos referimos a la parte entera de un número x, es decir, el mayor entero que es menor o igual que x).
- F_{m-1} : el número de matrices que tienen rango $m-1$.

$$X^2 = \frac{(F_m - 0,2888N)^2}{0,2888N} + \frac{(F_{m-1} - 0,5776N)^2}{0,5776N} + \frac{(N - F_m - F_{m-1} - 0,1336N)^2}{0,1336N}$$

También lo denotaremos *Rank*.

3.2.6. Test de transformada rápida de Fourier o espectral

Este test detecta componentes periódicos en la cadena de bits. Estos componentes son números complejos. Se mide cuántos de estos números son menores (en módulo) que un cierto umbral. Se aproxima la media y varianza de este valor, y se procede con un test de dos colas como en la Sección 3.2.1. Veremos que esta aproximación hace que este test haya dado problemas. En inglés nos referiremos a él sencillamente como *Fast Fourier Transform* (FFT).

3.2.7. Test de patrones sin solapamiento

Se prefija un patrón específico que se va a estudiar en la secuencia. A continuación, se divide la secuencia en N bloques, y se cuentan las apariciones del patrón en cada bloque. Al no solaparse, una vez se encuentra un patrón se procede la búsqueda desde el final de la coincidencia, no desde el bit siguiente al primer bit de la misma. Se tiene que para un patrón de longitud m la esperanza y varianza de las apariciones en un bloque son las siguientes.

$$\mu = \frac{n - m + 1}{2^m}$$

$$\sigma^2 = n\left[\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right]$$

Por consiguiente, se puede proceder con una prueba de bondad de ajuste de una cola, como ya hemos explicado en repetidas ocasiones.

$$X^2 = \sum_{j=0}^{N-1} \frac{(\text{Apariciones en el bloque j} - \mu)^2}{\sigma^2}.$$

En el código de [BRS+10] se repite esta prueba con muchos patrones diferentes. De esta manera, se crean 148 pruebas de este tipo, para 148 patrones. Veremos que no son independientes. Para sus 148 variantes, nos referiremos a ellos como $NonOverlappingTemplateK$, $K = 0, 1, ..., 147$

3.2.8. Test de patrones con solapamiento

Para este test, como en el test 3.2.7, se cuentan las apariciones de un patrón. En este caso, un patrón de m unos, con posible solapamiento. Es decir, si hubiera $m + 2$ unos seguidos se contarían tres apariciones.

La secuencia es dividida en N bloques, y se crean seis clases $\nu_0, ..., \nu_5$. Para $i = 0, ..., 4$, ν_i registra el número de bloques con exactamente i apariciones del patrón. ν_5 registra el número de bloques con 5 o más apariciones. En inglés nos referiremos a este test como *OverlappingTemplate*.

Las probabilidades de las clases π_i se aproximan por una fórmula [BRS+10]. Sin embargo, esta aproximación fue objeto de estudio [HK07], y tuvo que ser corregida en una revisión. Se acaba obteniendo el p-valor de una cola del estadístico de tipo χ^2:

$$X^2 = \sum_{i=0}^{5} \frac{(\nu_i - N\pi_i)^2}{N\pi_i}.$$

3.2.9. Test universal de Maurer

Este test tiene por objetivo detectar "cualquiera de los defectos estadísticos generales que se pueden modelar por una fuente ergódica de memoria finita". Según su autor, Ueli Maurer, este test integra la capacidad de muchos otros test.

Se divide la secuencia en Q bloques de inicialización, y en K bloques de test, de longitud L. Se recorre la secuencia analizando cada segmento de L bits, y hallando su aparición anterior. Se suma el logaritmo de la distancia entre ambos bloques. De esta suma se conoce la esperanza y la varianza bajo hipótesis de aleatoriedad, con lo cual se finaliza con un test de dos colas. También lo denotaremos *Universal*.

3.2.10. Test de complejidad lineal

El objetivo de este test es tratar de averiguar cuán largo (de cuántos bits) debería ser un registro de desplazamiento con retroalimentación, como hemos definido en el apartado 2.5.1, para poder generar la secuencia. Una secuencia que se pueda generar con un LFSR con pocos bits da prueba de baja aleatoriedad.

Al número de bits mínimo que basta para generar una secuencia con un LFSR se le denomina complejidad lineal. Por ejemplo, en la secuencia 1101011110001, se tiene que la complejidad lineal es 4, ya que a partir del 5º bit (inclusive) todos se pueden obtener sumando el primero y el segundo bit de los cuatro bits anteriores módulo 2.

Para el test, la secuencia se divide en N bloques, se mide la complejidad lineal de cada uno mediante el algoritmo de Berlekamp-Massey [GG70], y se dividen en categorías para realizar una prueba de χ^2 de una cola. Llamaremos a este test *LinearComplexity*.

3.2.11. Test serial

Este test se centra en la uniformidad de la distribución de patrones de una cierta longitud. En el caso de longitud uno, se reduciría al test de frecuencia.

Sea $i_0...i_{m-1}$ una secuencia de longitud m de las 2^m posibles que hay. Denotamos por $\nu_{i_0...i_{m-1}}$ el número de apariciones del patrón $i_0...i_{m-1}$ en la secuencia "circularizada", e.d. $\bar{\epsilon} = \epsilon_0...\epsilon_{n-1}\epsilon_0...\epsilon_{m-2}$. Podemos obtener algo parecido a un estadístico de chi cuadrado, para proceder con un test de una cola:

$$\Psi_m^2 = \frac{2^m}{n} \sum_{i_1,...,i_m} \left(\nu_{i_1,...,i_m} - \frac{n}{2^m}\right)^2 = \frac{2^m}{n} \sum_{i_1,...,i_m} \nu_{i_1...i_m}^2 - n$$

Según [BRS+10], Ψ_m^2 no sigue una distribución χ^2. Sin embargo, si llamamos $\nabla\Psi_m^2 = \Psi_m^2 - \Psi_{m-1}^2$, sí podemos usar una distribución χ^2. Se obtienen dos p-valores, uno con $\nabla\Psi_m^2$, y otro con $\nabla^2\Psi_m^2$. Nos referiremos a estos dos resultados como *Serial0* y *Serial1*.

3.2.12. Test de entropía aproximada

Este test lidia con el concepto de entropía de la Teoría de la Información. La entropía es, intuitivamente, la cantidad de información incomprimible que guarda una secuencia de bytes. Se mide la uniformidad de secuencias de longitudes m y $m+1$, y se ve cómo cambia la estimación de entropía entre las dos longitudes.

Primero se mide la frecuencia relativa de cada palabra de m bits. Esto es, dada una palabra $x = x_0...x_{m-1}$ (e.d. cualquier número entre 0 y $2^m - 1$), se toma $F_x = \frac{\text{Apariciones de } x \text{ en la secuencia}}{[n/m]}$ esto es, su número de apariciones partido por el número de palabras de m bits en toda la secuencia. Después, se suman los logaritmos de todas estas frecuencias y se dividen por el número total de palabras. Esto se denota por $\Phi^{(m)}$. Análogamente se halla $\Phi^{(m+1)}$, con el tamaño $m+1$.

El p-valor de dos colas es obtenido mediante una aproximación normal por el Teorema Central del Límite [CLT], aproximando $\Phi^{(m)} - \Phi^{(m+1)}$ mediante una distribución normal. A este test lo denominaremos *ApproximateEntropy*.

3.2.13. Test de sumas acumuladas

Este test estudia el máximo de las sumas parciales de la secuencia, interpretada como secuencia de 1 y -1. Primero se realiza la transformación $X_i = 2 \cdot \epsilon_i - 1$, después se estudian las sumas $S_k = \sum_{i=0}^{k-1} X_i$, y se toma el valor absoluto más grande de ellas para todo k.

Un p-valor bajo en este test indica que en un prefijo de la secuencia se acumulan demasiados ceros o demasiados unos. Por otra parte, se realiza la versión dual de esta parte, en la cual se consideran sumas acumuladas empezadas desde atrás $S'_k = \sum_{i=n-1}^{n-k} X_i$.

El p-valor se obtiene de una aproximación numérica, a partir del estadístico $\text{máx}_{1\le k\le n} |S_k|$, y se obtiene otro p-valor de la versión dual con los S'_k. Así, separaremos *CumulativeSums0* y *CumulativeSums1*.

3.2.14. Test de paseos aleatorios

Considerando las sumas acumuladas de los test 3.2.13 como un paseo aleatorio, se estudia la desviación de la distribución de cuántas veces "se visita" en las sumas parciales cada número $1, ..., z$, donde z es la misma que en los test 3.2.13. Se exige que haya un mínimo umbral de pares de puntos $l, l+k$ donde $S_l = 0 = S_{l+k}$, $S_{l+j} \neq 0 \; \forall \, 1 \leq j \leq k-1$. Si no hay suficientes ciclos así, no se realiza el test. Este hecho es importante porque ha sido ignorado en algunos estudios, contando p-valores para secuencias para las que no se realiza la prueba. Por cada valor $1, 2, ..., z$, se puede realizar una prueba de chi cuadrado. En el código por defecto de [BRS+10], se realizan 8 pruebas, dependiendo de un parámetro X, dando lugar a ocho test, denominados $RandomExcursionsK$, con $K = 0, ..., 7$.

3.2.15. Variante del test de paseos aleatorios

Es muy similar al test 3.2.14, solo que este se refiere a la secuencia completa, mientras que el anterior trabaja por bloques. Análogamente, exige la cantidad mínima de ciclos para poder realizar el test. Cuando este test y el test 3.2.14 no se pueden realizar por no haber ciclos suficientes, la salida es un p-valor 0, pero puede ser problemático utilizar esto como si fuera un p-valor auténtico. Este variante tiene 18 pruebas diferentes, para 18 valores de un parámetro X, nombrando las sucesivas pruebas $RandomExcursionsVariantK$, $K = 0, ..., 17$. Nos referiremos a los test con las abreviaturas especificadas en la Tabla 3.2.

Tabla 3.2: Abreviaturas de los test para ofrecer los datos en tablas.

Nombre completo en inglés	**Abreviatura**	**Explicación**
Frequency	Fr	Frecuencia
BlockFrequency	BlF	Frecuencia por bloques
CumulativeSums0	CuS0	Sumas acumuladas (hacia delante)
CumulativeSums1	CuS1	Sumas acumuladas (hacia atrás)
Runs	Ru	Rachas
LongestRun	LoR	Racha más larga de unos
Rank	Ra	Rango de matriz binaria
FFT	FFT	Espectral (transformada rápida de Fourier)
NonOverlappingTemplate0	NoOT0	Patrones sin solapamiento, patrón 000000001
OverlappingTemplate	OvT	Patrones con solapamiento, patrón 111111111
Universal	Un	Universal
ApproximateEntropy	ApE	Entropía aproximada
RandomExcursions0	RaE0	Paseos aleatorios, X=-4
RandomExcursionsVariant0	RaEV0	Variante de paseos aleatorios, X=-9
Serial0	Se0	Serial con $\nabla\psi_m^2$
Serial1	Se1	Serial con $\nabla^2\psi_m^2$
LinearComplexity	LiC	Complejidad lineal

Capítulo 4

Estado del Arte

La batería SP 800-22 ha sido objeto de muchas investigaciones. Es la batería de test más establecida y estandarizada, y fue usada para medir candidatos al algoritmo *Advanced Encryption Standard* (AES), ya que una propiedad deseada del cifrado era una salida aparentemente aleatoria. El escrutinio al que se ha sometido ha sido muy intenso, y han realizado pequeños cambios en el código a lo largo del tiempo.

En un inicio contaba con 16 test, uno más que ahora, ya que incluía el *Lempel Ziv Compression Test*. Este test fue eliminado por tener debilidades, como la encontrada en [DG06]. Según el texto:

> Este test solo se podía aplicar a datos de una longitud específica: 10^6 bits. Además, el test utilizaba datos empíricos generados por *Secure Hash Algorithm 1* (SHA-1) para estimar la esperanza (bajo hipótesis de aleatoriedad) de la complejidad de Lempel-Ziv de las secuencias de 10^6 bits. Aparentemente, [...] no era una buena estimación, ya que, por ejemplo, los primeros 10^6 bits de la expansión binaria del número e fallaban el test de aleatoriedad.

Antes, otros fallos en el test habían sido encontrados en [KUH04]. Asimismo, en este artículo se encontró que la varianza y el umbral usados en el test espectral o de Fourier eran incorrectos. Este test ha sido objeto de muchas críticas, y tras las correcciones de [KUH04], que seguían dependiendo de aproximaciones numéricas de la distribución teórica, recientemente en [OU17] se propuso una distribución teórica con la que mejorar los resultados.

Uno de los autores de [KUH04] y [OU17], Kim Hamano, también fue coautor de [HK07]. A raíz de este artículo y de [KUH04], el NIST decidió cambiar parámetros en los test de Fourier y en el de patrones solapados. Hoy día, en la página oficial del código `http://csrc.nist.gov/projects/random-bit-generation/documentation-and-software` se detallan los cambios realizados, y se recomienda que el test de Fourier se realice solo para secuencias de longitud 10^6, a pesar de que esto no case con los principios de escalabilidad.

4.1. Estudios de dependencias

En [BRS+10] se asegura que, para secuencias aleatorias, los test son independientes y están bien formulados. Esto tiene las siguientes implicaciones:

- Dados dos test T_i, T_j, $i \neq j$, se tiene $P(T_i|T_j) = P(T_i)$. Esto quiere decir que el resultado obtenido en un test para una secuencia no influencia en modo alguno el resultado de otros test a la misma.

- Si se aplican sobre datos aleatorios, los p-valores que se obtienen de los test están distribuidos uniformemente entre 0 y 1, luego para cada test T_i, $P(T_i) \sim \mathcal{U}(0,1)$. Además, estas distribuciones uniformes son independientes dos a dos.

Que los test de una batería sean **independientes** entre sí es **fundamental**. De no ser el caso, se está dando información redundante, perdiendo tiempo de cómputo, y sobre todo, sobrerrepresentando o menospreciando un único fenómeno en varios test. A pesar de que las aseveraciones de independencia de [BRS+10], varios estudios han hallado relaciones entre los test de la batería NIST SP 800-22.

Un estudio reseñable es [SUKD17]. En este artículo, se introduce un concepto novedoso e ingenioso: la cobertura de un conjunto de pruebas. Esto se define como la proporción de secuencias que fallan al menos una de las pruebas. Supongamos que tenemos k pruebas estadísticas, y cada una se pudiera fallar con probabilidad α. Entonces, en un escenario ideal, donde los test fueran independientes, tendríamos que la probabilidad de fallar alguna prueba es el complementario de pasar todas, con lo cual la cobertura teórica del conjunto vendría dada por

$$1 - (1 - \alpha)^k.$$

El artículo estudia un subconjunto de los test que se aplica a secuencias cortas, y anota lo que llama el "beneficio marginal" de los test: la diferencia de cobertura que hay entre no incluir un test, e incluirlo. No obstante, este "beneficio marginal" se calcula sobre un conjunto fijo de 9 test de la batería. Esto puede resultar insuficiente, y al ser los resultados relativos a ese conjunto de test, no se aclara cómo se pueden extrapolar resultados generales, y si los beneficios marginales cambiarían drásticamente de introducirse algunos de los otros test de la batería.

Para investigar las correlaciones dos a dos, se propone esta ingeniosa medida: dados dos test T_i, T_j, se define la correlación como

$$Corr(T_i, T_j) = \frac{|\{\text{Secuencias que fallan } T_i\} \cap \{\text{Secuencias que fallan } T_j\}|}{\text{nº de secuencias total}}$$

Si bien esta métrica de correlación resulta intuitiva e interesante, es difícil saber determinar qué valores pueden ser significativos en este test, y cómo afecta el número de secuencias y el tamaño de las mismas a los resultados. Es decir, no se sabe decir cómo de significativos son los resultados estadísticamente. Además, propone usar el test de frecuencia y el de sumas acumuladas para un subconjunto eficiente, y a la postre se demostraron altamente dependientes.

4.1.1. Correlación

Un indicador de posibles dependencias lineales es la correlación de Pearson [Kre11]. Dados los p-valores de dos test de la batería, como $x = (x_0, ..., x_{L-1})$, $y = (y_0, ..., y_{L-1})$ para L secuencias, se tiene que el coeficiente de correlación de Pearson muestral viene dado por:

$$r = \frac{\sum_{k=0}^{L-1}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=0}^{L-1}(x_k - \bar{x})^2}\sqrt{\sum_{k=0}^{L-1}(y_k - \bar{y})^2}}$$

donde $\bar{x} = \frac{1}{L}\sum_{i=0}^{L-1} x_i$ e $\bar{y} = \frac{1}{L}\sum_{i=0}^{L-1} y_i$ son las medias muestrales de x e y.

Este coeficiente mide cómo dependen linearmente x e y uno del otro. Si x e y tienen una relación lineal perfecta positiva, entonces $r = 1$. Si tienen una relación lineal perfecta negativa, $r = -1$. Si, en cambio, no hay ningún tipo de linealidad entre ambas, se tiene $r = 0$.

El coeficiente muestral aproxima el coeficiente de correlación de Pearson poblacional de dos variables aleatorias, dado por

$$\rho_{X,Y} = \frac{Cov(X, Y)}{\sigma_X \sigma_Y}$$

donde $Cov(X, Y)$ denota la covarianza entre X e Y, y σ_X y σ_Y sus desviaciones estándares. Si dos variables aleatorias X e Y son independientes, su coeficiente de correlación poblacional $\rho_{X,Y}$ es cero, pues tendrían covarianza nula. Por lo tanto, un coeficiente de correlación muestral alto nos indicará con un alto grado de probabilidad que dos variables no son independientes.

En [DSU+17] se realizan pruebas de correlación, tanto para secuencias cortas, de 1024 bits, como para secuencias largas, de 2^{20} bits. Se concluye que el test de Frecuencia, el test de Sumas Acumuladas hacia adelante *CumulativeSums0*, y el de Sumas Acumuladas hacia atrás *CumulativeSums1* están altamente correlacionados dos a dos. Asimismo, encuentra que los test *Serial0*, *Serial1* y *ApproximateEntropy* están correlacionados, a pesar de que el coeficiente de correlación entre *Serial1* y *ApproximateEntropy* sea 0.001, lo cual no parece muy razonable. Si bien *Serial0* está altamente correlacionado con los otros dos, no se aplica transitividad. Además, se propone estudiar cómo afectan ciertas transformaciones sobre la secuencia a los resultados del test, asociando test que son igualmente sensibles a las transformaciones.

Sin embargo, no se relacionan los resultados obtenidos en las secuencias cortas con los de las secuencias largas, y, parecido a [SUKD17], no se aclara qué se considera como una correlación alta, y en qué grado. Los resultados tampoco se contrastan con otra fuente de datos. Por tanto, dan una imagen un tanto estática y puntual de los resultados.

Un estudio parecido es realizado por parte de [BS19], donde se someten secuencias de 1.000.000, 2.000.000, 5.000.000 y 6.000.000 de bits a los test, y se estudian sus correlaciones de Pearson. Análogamente a [DSU+17], se encuentra correlación entre *Frequency*, *CumulativeSums0* y *CumulativeSums1*. Asimismo, encuentra que *Serial0* y *Serial1* están altamente correlacionados (aunque no con *ApproximateEntropy*). Para las primeras dos longitudes, también muestra correlaciones reseñables para *RandomExcursions* y *RandomExcursionsVariant*. La nueva contribución de este estudio es que analiza varias longitudes de secuencia. Las dependencias que encuentra están incluidas mayoritariamente

en [DSU+17]. Sin embargo, el umbral que ponen para considerar una correlación importante, que sitúan en 0.5, es completamente arbitrario. Además, sus mediciones de tiempo apuntan a que todos los test tienen un coste de tiempo casi idéntico, lo cual carece de sentido al ser test con una serie de complejidades muy distintas. También parece que, a juzgar por sus datos, han contado datos que en *RandomExcursions0* y *RandomExcursionsVariant0* daban salida cero por no haber suficientes ciclos (ver 3.2.14), como si fueran p-valores 0.

Una de las limitaciones de todos estos estudios consiste en que no cuantifican la probabilidad de obtener estas correlaciones altas, y como tal no permiten hacer una inferencia en términos numéricos, o decir con qué nivel de significancia se puede rechazar que los pares de test sean independientes. Si bien pueden señalar qué pares de test tienen correlaciones más grandes respecto a los demás, y son correlaciones que parecen excesivamente altas en términos absolutos, el umbral designado para discriminar entre significativo y no significativo es completamente arbitrario. Tampoco utilizan variedad de fuentes de datos.

Por consiguiente, vimos la oportunidad de realizar un estudio de correlación con un método de proceder estrictamente estadístico, y dar probabilidades a cuán dependientes son los pares de pruebas.

4.1.2. Dependencias mediante p-valores de p-valores

Este otro paradigma para detectar la falta de independencia se basa en la premisa anteriormente citada: si los p-valores de los test están uniformemente distribuidos, y estas distribuciones son independientes, entonces es posible para un par de test hallar la distribución de la resta de los p-valores de uno y otro. Esta propuesta fue presentada por [FCG14], y la replicaremos más adelante en la Sección 5.7. La idea es sencilla como ingeniosa. Dado que dos test independientes aplicados a un conjunto de secuencias producen, en teoría, dos distribuciones uniformes $\mathcal{U}(0,1)$ independientes, podemos teorizar qué distribución tiene la resta de los p-valores de ambos test. Así, una vez hallada la distribución teórica de esas restas, se puede ver si las restas observadas parecen provenir de esa distribución o no. Para determinar esto último, se realiza un test de bondad de ajuste, del que se pueden obtener p-valores, y por tanto hacer un tratamiento estadístico de las dependencias. Este enfoque es innovador, y aporta un rigor estadístico sólido. De varios generadores que utilizan para la prueba, solo muestran datos del generador *Blum Blum Shub* (BBS).

Si bien este estudio es, tal vez, el más robusto en términos estadísticos, solo ofrece datos de una determinada longitud y para un único generador. Los resultados que arroja están por tanto bien fundamentados, pero no es difícil enmarcarlos en una imagen global de cómo evolucionan las dependencias independientemente del generador utilizado, a medida que cambia la longitud.

Capítulo 5

Estudio Sobre las Dependencias y Eficiencia de la Batería NIST SP 800-22

5.1. Descripción del experimento

Primero generamos una gran cantidad de datos, 5.000 archivos de 12 MB (aunque no se llegan a usar todos), a través del fichero especial de *UNIX /dev/urandom*, tras haber obtenenido una cierta entropía. Este fichero especial por caracteres provee una interfaz con el generador de números aleatorios del kernel, proveyendo números a través de un PRNG cuya semilla proviene del *pool* de entropía de nuestro dispositivo [man]. Un pequeño código en C basta para conseguir estos ficheros de forma rápida.

Después, procesamos secuencias de longitudes 2^{20} bits, 2^{21} bits, 2^{22} bits, 2^{23} bits y 2^{24} bits, ya que es importante ver si las relaciones encontradas se mantienen a medida que se escala la longitud de la secuencia, y tal vez detectar alguna relación asintótica. Para la primera longitud, se toman 5.000 secuencias, para las demás, por el alto coste computacional de la batería, 500. A partir de todos estos p-valores, realizamos estudios de dependencia. Después, para corroborar resultados, y ver que no están afectados por particularidades de este generador, tomamos también 500 muestras de un TRNG, llamado *ChaosKey*, para cada una de las cinco longitudes.

Para realizar estas pruebas, se han generado scripts en Python que lidian con descriptores de ficheros y procesos, con el fin de automatizar la generación de datos, la aplicación de la batería a los mismos, la recogida de resultados, su procesamiento y, finalmente, su estudio estadístico. Para esto último, utilizamos las librerías SciPy, NumPy, Pandas, y también el lenguaje R. Los scripts, así como todas las tablas con todos los datos, que por espacio hemos omitido aquí, están en el espacio habilitado para código fuente del trabajo.

5.2. Resultados de aleatoriedad

Para cada test de la batería, dado un conjunto de secuencias, se aprueba el test de aleatoriedad si se da simultáneamente que:

- El porcentaje de secuencias que obtienen un p-valor mayor o igual que $\alpha = 0.01$ es de al menos
$$1 - \alpha - 3 \cdot \sqrt{\frac{\alpha \cdot (1-\alpha)}{n}}. \tag{5.1}$$
- La prueba de uniformidad de χ^2 de los p-valores tiene p-valor mayor o igual que $\alpha^2 = 0.0001$. Es decir, llamamos A_i al número de secuencias que caen en el intervalo I_i, siendo estos intervalos $I_i = [\frac{i-1}{10}, \frac{i}{10})$ para $i = 1, ..., 9$, $I_{10} = [0.9, 1]$. Se toma el estadístico:
$$X^2 = \sum_{i=1}^{10} \frac{(A_i - n/10)^2}{n/10}.$$
Si la probabilidad $P(\chi^2_9 \geq X^2)$ fuera menor que 0.0001, eso quiere decir que algún intervalo tiene demasiados o demasiados pocos p-valores, y estos no son uniformes.

Nótese que de los 15 test, los siguientes tienen variantes: *CumulativeSums* (2), *NonOverlappingTemplate* (148), *RandomExcursions* (8), *RandomExcursionsVariant* (18), *Serial* (2). Por lo tanto hay en total $15 + 1 + 147 + 7 + 17 + 1 = 188$. Mostramos cuántos test han pasado cada conjunto de secuencias en la Tabla 5.1.

Tabla 5.1: Test pasados satisfactoriamente por cada conjunto de secuencias, del total de 188.

Longitud	*/dev/urandom*	*ChaosKey*
2^{20}	186	187
2^{21}	186	187
2^{22}	186	185
2^{23}	185	187
2^{24}	187	186

Algo destacable es que, a pesar de aprobar tantísimos test, las secuencias de **ambos** generadores tienen un mal comportamiento con *OverlappingTemplate*. Hablaremos de ello más adelante.

5.3. Correlación

Mostramos los resultados más importantes en las Tablas 5.2, 5.3, 5.4, 5.5 y 5.6. Mostramos todas las pruebas que tuvieron coeficiente mayor que 0.1 con alguna otra. Las tablas completas, con todos los pares de test, están disponibles junto al código en el espacio habilitado para el código fuente del trabajo.

Tabla 5.2: Correlaciones relevantes para longitud 2^{20}.

	Fr	**CuS0**	**CuS1**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**
Fr	1.000	0.773	0.772	-0.001	0.014	-0.026	0.008	0.010
CuS0	0.773	1.000	0.725	-0.003	0.000	-0.016	0.016	0.013
CuS1	0.772	0.725	1.000	-0.015	0.001	-0.012	0.023	0.022
ApE	-0.001	-0.003	-0.015	1.000	0.017	0.032	0.173	-0.012
RaE0	0.014	0.000	0.001	0.017	1.000	0.112	0.017	0.010
RaEV0	-0.026	-0.016	-0.012	0.032	0.112	1.000	0.036	0.045
Se0	0.008	0.016	0.023	0.173	0.017	0.036	1.000	0.686
Se1	0.010	0.013	0.022	-0.012	0.010	0.045	0.686	1.000

Tabla 5.3: Correlaciones relevantes para longitud 2^{21}.

	Fr	**CuS0**	**CuS1**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**	**LiC**
Fr	1.000	0.789	0.796	-0.026	0.017	-0.049	0.019	0.027	0.035
CuS0	0.789	1.000	0.729	-0.027	-0.017	-0.057	-0.008	0.010	0.037
CuS1	0.796	0.729	1.000	-0.074	-0.002	-0.028	0.059	0.054	0.049
ApE	-0.026	-0.027	-0.074	1.000	-0.023	0.069	0.119	0.012	0.050
RaE0	0.017	-0.017	-0.002	-0.023	1.000	0.104	0.009	0.005	0.000
RaEV0	-0.049	-0.057	-0.028	0.069	0.104	1.000	0.045	0.038	-0.047
Se0	0.019	-0.008	0.059	0.119	0.009	0.045	1.000	0.736	0.030
Se1	0.027	0.010	0.054	0.012	0.005	0.038	0.736	1.000	0.115
LiC	0.035	0.037	0.049	0.050	0.000	-0.047	0.030	0.115	1.000

Tabla 5.4: Correlaciones relevantes para longitud 2^{22}.

	Fr	**CuS0**	**CuS1**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**
Fr	1.000	0.785	0.772	-0.003	0.034	-0.034	0.029	-0.033
CuS0	0.785	1.000	0.718	0.000	0.040	-0.012	0.011	-0.025
CuS1	0.772	0.718	1.000	-0.039	0.014	-0.037	0.054	0.021
ApE	-0.003	0.000	-0.039	1.000	-0.042	0.008	0.140	0.023
RaE0	0.034	0.040	0.014	-0.042	1.000	0.135	0.067	0.060
RaEV0	-0.034	-0.012	-0.037	0.008	0.135	1.000	0.000	0.051
Se0	0.029	0.011	0.054	0.140	0.067	0.000	1.000	0.720
Se1	-0.033	-0.025	0.021	0.023	0.060	0.051	0.720	1.000

Tabla 5.5: Correlaciones relevantes para longitud 2^{23}.

	Fr	**CuS0**	**CuS1**	**LoR**	**Ra**	**ApE**	**Se0**	**Se1**
Fr	1.000	0.739	0.770	-0.052	-0.059	-0.018	0.021	0.016
CuS0	0.739	1.000	0.716	-0.049	-0.039	-0.007	0.038	0.004
CuS1	0.770	0.716	1.000	-0.036	-0.056	-0.001	0.029	-0.009
LoR	-0.052	-0.049	-0.036	1.000	0.105	0.050	0.051	0.064
Ra	-0.059	-0.039	-0.056	0.105	1.000	0.032	0.056	0.026
ApE	-0.018	-0.007	-0.001	0.050	0.032	1.000	0.155	-0.001
Se0	0.021	0.038	0.029	0.051	0.056	0.155	1.000	0.648
Se1	0.016	0.004	-0.009	0.064	0.026	-0.001	0.648	1.000

Tabla 5.6: Correlaciones relevantes para longitud 2^{24}.

	Fr	**CuS0**	**CuS1**	**Ru**	**Ra**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**	**LiC**
Fr	1.000	0.793	0.799	0.065	-0.023	0.009	-0.043	0.019	-0.036	-0.043	-0.107
CuS0	0.793	1.000	0.774	0.028	-0.063	-0.001	-0.002	0.006	-0.044	-0.042	-0.094
CuS1	0.799	0.774	1.000	0.016	-0.065	-0.029	-0.030	0.013	-0.081	-0.067	-0.136
Ru	0.065	0.028	0.016	1.000	0.018	-0.007	0.000	0.108	-0.023	-0.011	-0.061
Ra	-0.023	-0.063	-0.065	0.018	1.000	0.049	0.003	-0.025	0.021	0.017	0.122
ApE	0.009	-0.001	-0.029	-0.007	0.049	1.000	0.050	-0.015	0.113	-0.027	0.005
RaE0	-0.043	-0.002	-0.030	0.000	0.003	0.050	1.000	0.093	0.104	0.133	0.119
RaEV0	0.019	0.006	0.013	0.108	-0.025	-0.015	0.093	1.000	0.053	-0.006	0.016
Se0	-0.036	-0.044	-0.081	-0.023	0.021	0.113	0.104	0.053	1.000	0.639	-0.032
Se1	-0.043	-0.042	-0.067	-0.011	0.017	-0.027	0.133	-0.006	0.639	1.000	-0.054
LiC	-0.107	-0.094	-0.136	-0.061	0.122	0.005	0.119	0.016	-0.032	-0.054	1.000

Notamos algunas cosas. Primero, que los resultados son bastante similares a los obtenidos por [DSU+17] y [BS19]. No obstante, hay una salvedad: En *RandomExcursions* y *RandomExcursionsVariant*, nuestras tablas muestran correlaciones mucho menores. Esto tiene una explicación muy sencilla. Hemos decidido contar solo los datos de las secuencias en las cuales se daban las condiciones para realizar ambos test, ya que en caso contrario se devuelve un 0. Esto hace que haya muchas apariciones de 0 conjuntas en los datos de ambos test, y parece que ha sesgado la investigación de la correlación en [DSU+17] y [BS19]. Si no descartamos esos ceros, hallamos correlaciones parecidas a las de estos estudios (ver Tabla 5.7).

Tabla 5.7: Correlaciones entre *RandomExcursions0* y *RandomExcursionsVariant0* si no se filtran ceros.

Longitud	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Correlación	0.59	0.50	0.48	0.48	0.32

Por lo tanto, es de esperar que los altos datos de los estudios se deban al haber utilizado los ceros de las secuencias inválidas como si fueran p-valores válidos. Sin embargo, nótese que, contando los falsos ceros como p-valores, la correlación entre ambos test va tendiendo a menos según se incrementa la longitud de secuencia. Esto puede deberse a que, con mayores longitudes, se cumple más fácilmente la precondición necesaria de que haya ciclos suficientes.

A simple vista parece que los altos datos de correlación hacen bastante improbable que ciertos pares de test sean independientes. Nos ocuparemos en trabajar en la cuantificación de esta probabilidad.

5.4. Uniformidad de los p-valores

El otro aserto que se realiza en [BRS+10] que hemos nombrado al inicio del apartado 4.1 se refiere a la uniformidad de los p-valores. Por lo tanto, si los test están bien formulados y son independientes, tal como se asegura en la publicación, los p-valores obtenidos en los test deben responder a distribuciones uniformes $\mathcal{U}(0,1)$ independientes dos a dos.

Para ponerlo a prueba, utilizamos el test no paramétrico de Kolmogorov-Smirnov [Wik]. Este test compara una serie de muestras con una distribución de probabilidad. Nuestras muestras consisten en los p-valores obtenidos por las secuencias en un cierto test, y la distribución de referencia es la distribución uniforme $\mathcal{U}(0,1)$.

Se toma la función de distribución empírica. Dadas n observaciones independientes e idénticamente distribuidas, y ordenadas crecientemente, llamadas X_i, la función de distribución es:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{[-\infty,x]}(X_i) \qquad x \in \mathbb{R} \tag{5.2}$$

siendo $I_{[-\infty,x]}(X_i)$ la función indicatriz, es decir, 1 si $X_i \leq x$ y 0 en otro caso.

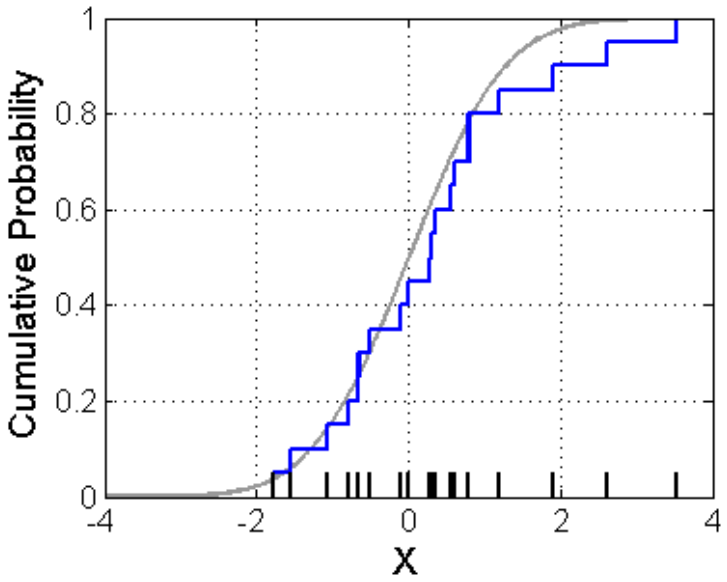


Figura 5.1: Función de distribución empírica (escalonada) vs. teórica (continua).

El estadístico que se toma es el supremo para todo x de $D_n = |F_n(x) - F(x)|$, siendo F la función de distribución teórica. Es decir, mide la mayor separación entre la distribución teórica y la observada. De este supremo se obtiene un p-valor, pues la distribución del estadístico D_n es conocida.

Aplicamos el test a los p-valores para las distintas longitudes, para ver si no se cumple la uniformidad en algún test. Los resultados se recogen en la Tabla 5.8.

Tabla 5.8: Estadístico y p-valor para el test Kolmogorov-Smirnov.

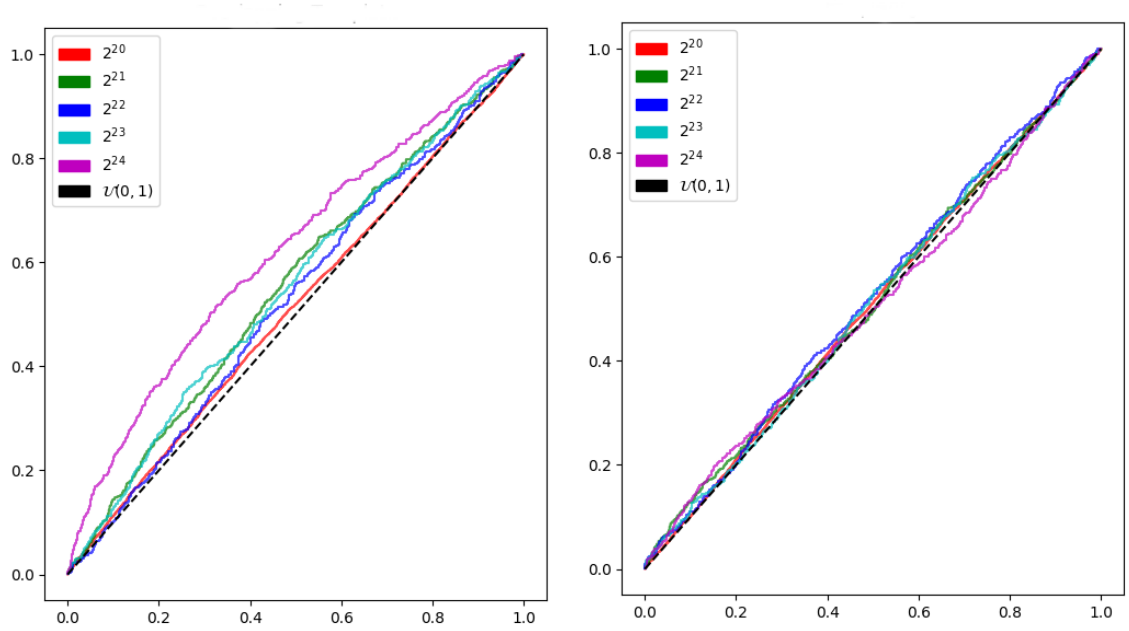
	2^{20}		2^{21}		2^{22}		2^{23}		2^{24}	
	KS	**pvalue**	**KS**	**pvalue**	**KS**	**pvalue**	**KS**	**pvalue**	**KS**	**pvalue**
Fr	0.0180	0.0789	0.0362	0.4742	0.0392	0.4124	0.0339	0.6094	0.0439	0.2834
BlF	0.0088	0.8381	0.0430	0.2633	0.0436	0.2839	0.0301	0.7483	0.0309	0.7277
CuS0	0.0199	0.0385	0.0406	0.3284	0.0368	0.4974	0.0219	0.9683	0.0364	0.5196
CuS1	0.0203	0.0327	0.0315	0.6556	0.0426	0.3096	0.0321	0.6740	0.0359	0.5388
Ru	0.0186	0.0622	0.0397	0.3547	0.0288	0.7951	0.0217	0.9713	0.0438	0.2848
LoR	0.0088	0.8312	0.0422	0.2841	0.0303	0.7417	0.0467	0.2142	0.0583	0.0644
Ra	0.0117	0.4960	0.0396	0.3585	0.0208	0.9807	0.0385	0.4335	0.0315	0.7042
FFT	0.0262	0.0021	0.0333	0.5898	0.0414	0.3425	0.0291	0.7864	0.0257	0.8971
NoOT0	0.0117	0.4995	0.0298	0.7223	0.0419	0.3278	0.0283	0.8131	0.0276	0.8414
OvT	0.0269	0.0015	0.0986	0.0001	0.0638	**0.0309**	0.0939	0.0002	0.1895	0.0000
Un	0.0197	0.0417	0.0255	0.8730	0.0471	0.2045	0.0348	0.5752	0.0451	0.2545
ApE	0.0122	0.4420	0.0408	0.3228	0.0607	0.0462	0.0521	0.1238	0.0450	0.2560
RaE0	0.0236	0.0629	0.0229	0.9875	0.0809	0.0116	0.1016	0.0003	0.0337	0.6921
RaEV0	0.0099	0.9236	0.0488	0.3056	0.0593	0.1248	0.0463	0.3126	0.0315	0.7675
Se0	0.0098	0.7172	0.0288	0.7623	0.0524	0.1199	0.0273	0.8442	0.0463	0.2274
Se1	0.0137	0.3006	0.0378	0.4177	0.0362	0.5177	0.0364	0.5109	0.0421	0.3299
LiC	0.0091	0.8005	0.0259	0.8629	0.0475	0.1967	0.0259	0.8867	0.0467	0.2182

OvT para el generador *ChaosKey*:

	2^{20}		2^{21}		2^{22}		2^{23}		2^{24}	
	KS	**pvalue**	**KS**	**pvalue**	**KS**	**pvalue**	**KS**	**pvalue**	**KS**	**pvalue**
OvT	0.0364	0.5180	0.0823	0.0021	0.0601	0.0517	0.1002	0.0001	0.1683	0.0000

En la Tabla 5.8 observamos que en general los p-valores obtenidos en los test están distribuidos uniformemente, ya que la probabilidad de que la probabilidad de obtener sus estadísticos de Kolmogorov-Smirnov o uno más grande no es menor que $\alpha = 0.01$. No obstante, son acentuadamente bajos en *OverlappingTemplate*, el test de patrones con solape, donde vemos que para las distintas longitudes de secuencia, de 4 p-valores, tres son menores que 0.01, mientras que el otro es algo mayor pero menor que 0.05. Si estos resultados se presentaran por sí solos, se daría por hecho que */dev/urandom* tiene una vulnerabilidad que sale a relucir cuando se buscan patrones solapados de unos en sus secuencias. No obstante, viendo que la tónica se repite para las secuencias de *ChaosKey*, y teniendo en cuenta que el test ha tenido errores con anterioridad, parece probable que el test *OverlappingTemplate* esté siendo realmente demasiado estricto.

Para ver gráficamente cómo distan los p-valores de *OverlappingTemplate* de estar distribuidos uniformemente, trazamos la función de distribución empírica de este test frente al test de frecuencia *Frequency*. Los resultados se muestran en la Figura 5.2. Tal como se aprecia, para el primer test la distribución dista mucho de la teórica, mientras que el segundo tiene un ajuste más preciso.



(a) *Overlapping Template* (b) *Frecuency*

Figura 5.2: Funciones empíricas de distribución acumulada

5.5. Estudio de la probabilidad de las correlaciones

Tal como se explicó en la Sección 4.1.1, si dos variables aleatorias son independientes, el coeficiente de correlación poblacional ρ es cero. Resulta de interés, entonces, medir cuán probable es que el coeficiente muestral r (el observado) aporte números tan altos como los vistos en las Tablas 5.2, 5.3, 5.4, 5.5 y 5.6.

En aquellas tablas, hemos decidido mostrar los test que tuvieran correlaciones mayores que 0.1 con algunos otros test. Pero, ¿en qué se basa esta decisión? ¿Es lo mismo un 0.1 de correlación para nuestro conjunto de 5.000 secuencias que para los de 500? Hay cierta componente de arbitrariedad ahí en lo que se considere correlación "suficientemente grande". Por ello, para cuantificar esta probabilidad, echamos mano nuevamente de los test de hipótesis. Esto nos aportará un enfoque nuevo frente a [BS19] y [DSU+17].

Es decir, nos proponemos obtener p-valores a partir de la correlación de los p-valores. Esto se puede realizar mediante una prueba T de Student. El objetivo es el siguiente: dado el coeficiente de correlación muestral $r_{X,Y}$, obtener la probabilidad de que el $r_{X,Y}$ sea tan distinto de cero, bajo la premisa de aleatoriedad, en la cual $\rho_{X,Y} = 0$ por ser los test X e Y independientes.

Podemos obtener el siguiente t-estadístico [Kre11]:

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \tag{5.3}$$

donde r es el coeficiente de correlación, y n el número de p-valores para cada test. Si los datos que estamos comparando siguen una distribución normal de dos variables, el t-estadístico seguirá una distribución T de Student de $df = n - 2$ grados de libertad. Al conocer esta distribución, la probabilidad de tener una correlación tan diferente de cero como la r observada viene dada por

$$P(|T| \geq |t|) \tag{5.4}$$

donde T denota la distribución T de Student de df grados de libertad. De esta forma, se obtiene un p-valor de dos colas, como el de la Figura 3.1, que da una idea de cuán significativamente distinto de cero es un coeficiente r observado.

La comprobación de que los datos estudiados provienen de una normal bivariante se suele omitir, especialmente cuando el número de muestras es grande. No obstante, la omisión de este requisito puede tener resultados adversos [Kow72]. Por ello, primero hallamos p-valores con la práctica habitual de omitir la normalidad, y después hicimos una comprobación mediante una transformación, tras convertir los datos uniformes en muestras de una distribución normal estándar $\mathcal{N}(0, 1)$.

Esto es, tal como se menciona en [Kow72], y teniendo en cuenta que los p-valores de dos test X, Y cumplen, para secuencias aleatorias, y según los asertos de [BRS+10], $X, Y \sim \mathcal{U}(0, 1)$ independientes e idénticamente distribuidas (i.i.d.), entonces $\Phi^{-1}(X)$ y $\Phi^{-1}(Y)$ están distribuidas normalmente con una $\mathcal{N}(0, 1)$, donde Φ^{-1} denota la función inversa de la función de distribución normal estándar. Si X e Y fueran independientes, también deben serlo $\Phi^{-1}(X)$ y $\Phi^{-1}(Y)$.

A pesar de que no se conozca una forma cerrada (fórmula analítica explícita) para Φ^{-1}, existen diversas aproximaciones numéricas para la misma. Por lo tanto, si tomamos los p-valores uniformemente repartidos (tal como se ha observado en la Tabla 5.8), a través de la aproximación numérica de Φ^{-1} tenemos una muestra de una normal estándar, sobre cuyas correlaciones ya podemos aplicar el test T de Student teniendo en cuenta las advertencias de [Kow72].

Nótese que como la independencia de X, Y implica la independencia de $\Phi^{-1}(X), \Phi{-}1(Y)$, por contrarrecíproco, si conseguimos encontrar dependencias en $\Phi^{-1}(X), \Phi^{-1}(Y)$, estaremos demostrando dependencias entre X e Y.

5.5.1. P-valores pre normalización

En las Tablas 5.9, 5.10, 5.11, 5.12 y 5.13 mostramos los p-valores obtenidos tras aplicar las ecuaciones 5.3 y 5.4.

Tabla 5.9: P-valores del test T de Student pre normalización para longitud 2^{20}.

	Fr	**CuS0**	**CuS1**	**Ru**	**LoR**	**NoOT0**	**OvT**	**Un**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**	**LiC**
Fr	0.000	0.000	0.000	0.430	0.228	0.000	0.196	0.225	0.916	0.448	0.155	0.579	0.494	0.727
CuS0	0.000	0.000	0.000	0.209	0.084	0.000	0.204	0.169	0.835	0.987	0.367	0.243	0.373	0.480
CuS1	0.000	0.000	0.000	0.243	0.306	0.000	0.168	0.159	0.276	0.948	0.521	0.108	0.121	0.589
Ru	0.430	0.209	0.243	0.000	0.364	0.441	0.657	0.719	0.009	0.649	0.156	0.008	0.973	0.533
LoR	0.228	0.084	0.306	0.364	0.000	0.782	0.000	0.858	0.296	0.686	0.056	0.169	0.480	0.504
NoOT0	0.000	0.000	0.000	0.441	0.782	0.000	0.772	0.044	0.670	0.355	0.323	0.542	0.501	0.087
OvT	0.196	0.204	0.168	0.657	0.000	0.772	0.000	0.653	0.650	0.719	0.785	0.690	0.106	0.644
Un	0.225	0.169	0.159	0.719	0.858	0.044	0.653	0.000	0.967	0.925	0.512	0.438	0.511	0.010
ApE	0.916	0.835	0.276	0.009	0.296	0.670	0.650	0.967	0.000	0.350	0.078	0.000	0.382	0.933
RaE0	0.448	0.987	0.948	0.649	0.686	0.355	0.719	0.925	0.350	0.000	0.000	0.350	0.566	0.986
RaEV0	0.155	0.367	0.521	0.156	0.056	0.323	0.785	0.512	0.078	0.000	0.000	0.043	0.013	0.709
Se0	0.579	0.243	0.108	0.008	0.169	0.542	0.690	0.438	0.000	0.350	0.043	0.000	0.000	0.204
Se1	0.494	0.373	0.121	0.973	0.480	0.501	0.106	0.511	0.382	0.566	0.013	0.000	0.000	0.715
LiC	0.727	0.480	0.589	0.533	0.504	0.087	0.644	0.010	0.933	0.986	0.709	0.204	0.715	0.000

Tabla 5.10: P-valores del test T de Student pre normalización para longitud 2^{21}.

	Fr	**CuS0**	**CuS1**	**ApE**	**Se0**	**Se1**	**LiC**
Fr	0.000	0.000	0.000	0.547	0.654	0.525	0.414
CuS0	0.000	0.000	0.000	0.534	0.858	0.814	0.396
CuS1	0.000	0.000	0.000	0.087	0.169	0.208	0.258
ApE	0.547	0.534	0.087	0.000	0.006	0.778	0.243
Se0	0.654	0.858	0.169	0.006	0.000	0.000	0.482
Se1	0.525	0.814	0.208	0.778	0.000	0.000	0.008
LiC	0.414	0.396	0.258	0.243	0.482	0.008	0.000

Tabla 5.11: P-valores del test T de Student pre normalización para longitud 2^{22}.

	Fr	**CuS0**	**CuS1**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**
Fr	0.000	0.000	0.000	0.938	0.503	0.506	0.520	0.452
CuS0	0.000	0.000	0.000	0.996	0.435	0.810	0.802	0.578
CuS1	0.000	0.000	0.000	0.381	0.787	0.470	0.228	0.645
ApE	0.938	0.996	0.381	0.000	0.409	0.877	0.002	0.605
RaE0	0.503	0.435	0.787	0.409	0.000	0.008	0.189	0.234
RaEV0	0.506	0.810	0.470	0.877	0.008	0.000	0.995	0.320
Se0	0.520	0.802	0.228	0.002	0.189	0.995	0.000	0.000
Se1	0.452	0.578	0.645	0.605	0.234	0.320	0.000	0.000

Tabla 5.12: P-valores del test T de Student pre normalización para longitud 2^{23}.

	Fr	**CuS0**	**CuS1**	**ApE**	**Se0**	**Se1**
Fr	0.000	0.000	0.000	0.680	0.633	0.722
CuS0	0.000	0.000	0.000	0.875	0.397	0.925
CuS1	0.000	0.000	0.000	0.990	0.508	0.833
ApE	0.680	0.875	0.990	0.000	0.000	0.975
Se0	0.633	0.397	0.508	0.000	0.000	0.000
Se1	0.722	0.925	0.833	0.975	0.000	0.000

Tabla 5.13: P-valores del test T de Student pre normalización para longitud 2^{24}.

	Fr	**CuS0**	**CuS1**	**Ru**	**LoR**	**Ra**	**RaE0**	**Se0**	**Se1**	**LiC**
Fr	0.000	0.000	0.000	0.145	0.966	0.612	0.360	0.426	0.338	0.016
CuS0	0.000	0.000	0.000	0.531	0.762	0.157	0.972	0.326	0.352	0.037
CuS1	0.000	0.000	0.000	0.714	0.741	0.144	0.533	0.072	0.136	0.002
Ru	0.145	0.531	0.714	0.000	0.005	0.690	0.995	0.604	0.811	0.173
LoR	0.966	0.762	0.741	0.005	0.000	0.569	0.527	0.291	0.224	0.077
Ra	0.612	0.157	0.144	0.690	0.569	0.000	0.954	0.632	0.699	0.006
RaE0	0.360	0.972	0.533	0.995	0.527	0.954	0.000	0.028	0.005	0.012
Se0	0.426	0.326	0.072	0.604	0.291	0.632	0.028	0.000	0.000	0.470
Se1	0.338	0.352	0.136	0.811	0.224	0.699	0.005	0.000	0.000	0.231
LiC	0.016	0.037	0.002	0.173	0.077	0.006	0.012	0.470	0.231	0.000

5.5.2. P-valores tras normalización

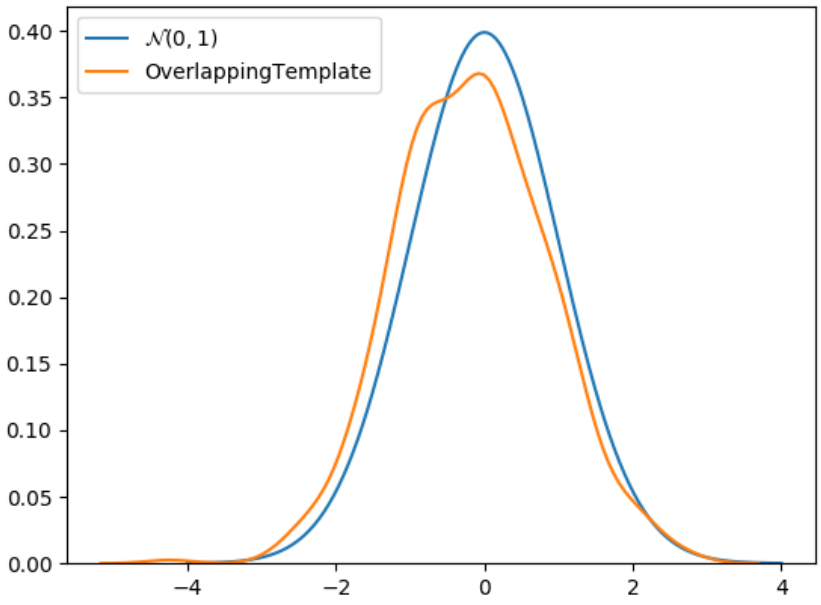
Mostramos un ejemplo de los resultados de las transformaciones en la Figura 5.3. En ella, mostramos los resultados por una estimación de kernel de las muestras obtenidas, y los comparamos con la distribución normal estándar. Como vemos gráficamente, se reflejan los resultados obtenidos en la Tabla 5.8, donde se ponía en manifiesto la poca uniformidad de los p-valores del test *OverlappingTemplate*, o patrones con solape, frente a los datos del test de frecuencia, mejor repartidos.

Ahora, si se mantiene cierto el aserto de que los p-valores de los test provienen de una distribución uniforme $\mathcal{U}(0,1)$, se cumple la hipótesis de que los datos transformados provienen de una distribución normal estándar $\mathcal{N}(0,1)$, al menos para aquellos distintos del *OverlappingTemplate*. Así, podemos reevaluar las correlaciones, y realizarles un test T de Student, para medir cuán probable es obtener las correlaciones observadas.

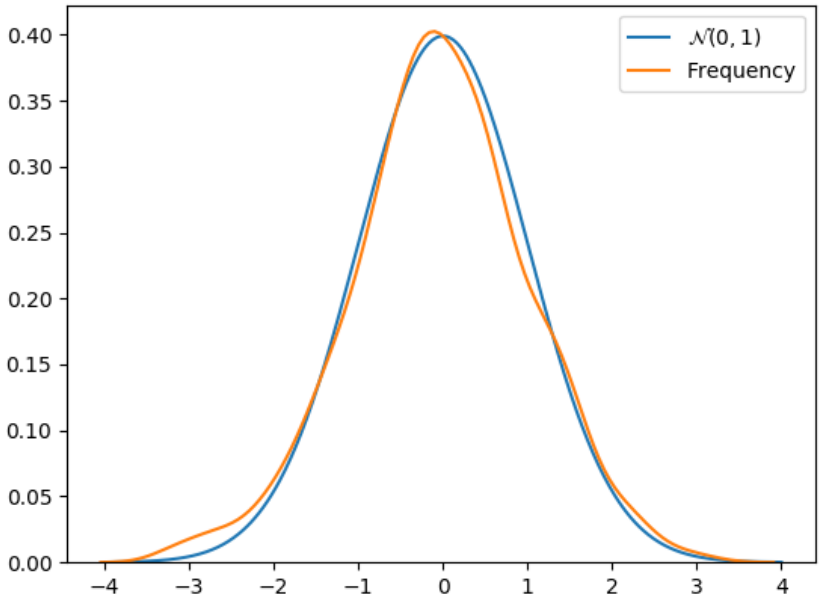
Mostramos los test con alguna entrada con p-valores menores a $\alpha = 0.01$.

Por los resultados obtenidos en las Tablas 5.14, 5.15, 5.16, 5.17 y 5.18, podemos decir con nivel de confianza del 0.99 que *Frequency*, *CumulativeSums0* y *CumulativeSums1* no son independientes. Tampoco lo son *Serial0* y *Serial1*, y tampoco lo son *Serial0* y *ApproximateEntropy*. Estos resultados se mantienen a medida que se alarga la secuencia por lo que son consistentes. Para estudiar las demás entradas con valores muy bajos, habría que ver por qué en esas longitudes precisas son así.

Es reseñable de estas tablas que, al compararlas respectivamente con las Tablas 5.9, 5.10, 5.11, 5.12 y 5.13, se destacan en los respectivos pares de tablas los mismos pares de test críticos. Por lo tanto, si bien hemos decidido ser precavidos para la corrección teórica, los datos pre normalizados también valen.



(a) Test de patrones con solape



(b) Test de frecuencia

Figura 5.3: Resultados de transformar los p-valores frente a la distribución normal estándar $\mathcal{N}(0,1)$ para longitud 2^{23}.

Tabla 5.14: P-valores del test T de Student tras normalización para longitud 2^{20}.

	Fr	**CuS0**	**CuS1**	**Ru**	**LoR**	**NoOT0**	**OvT**	**Un**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**	**LiC**
Fr	0.000	0.000	0.000	0.319	0.242	0.000	0.116	0.444	0.662	0.286	0.318	0.395	0.394	0.714
CuS0	0.000	0.000	0.000	0.119	0.054	0.000	0.199	0.344	0.862	0.947	0.385	0.153	0.231	0.308
CuS1	0.000	0.000	0.000	0.121	0.206	0.000	0.157	0.181	0.759	0.781	0.468	0.076	0.102	0.710
Ru	0.319	0.119	0.121	0.000	0.607	0.283	0.393	0.369	0.006	0.665	0.346	0.004	0.538	0.267
LoR	0.242	0.054	0.206	0.607	0.000	0.963	0.000	0.848	0.302	0.698	0.163	0.163	0.414	0.704
NoOT0	0.000	0.000	0.000	0.283	0.963	0.000	0.845	0.061	0.933	0.602	0.244	0.519	0.794	0.140
OvT	0.116	0.199	0.157	0.393	0.000	0.845	0.000	0.968	0.751	0.603	0.860	0.639	0.221	0.710
Un	0.444	0.344	0.181	0.369	0.848	0.061	0.968	0.000	0.843	0.738	0.659	0.270	0.328	0.006
ApE	0.662	0.862	0.759	0.006	0.302	0.933	0.751	0.843	0.000	0.546	0.041	0.000	0.353	0.785
RaE0	0.286	0.947	0.781	0.665	0.698	0.602	0.603	0.738	0.546	0.000	0.000	0.417	0.910	0.801
RaEV0	0.318	0.385	0.468	0.346	0.163	0.244	0.860	0.659	0.041	0.000	0.000	0.018	0.007	0.994
Se0	0.395	0.153	0.076	0.004	0.163	0.519	0.639	0.270	0.000	0.417	0.018	0.000	0.000	0.398
Se1	0.394	0.231	0.102	0.538	0.414	0.794	0.221	0.328	0.353	0.910	0.007	0.000	0.000	0.936
LiC	0.714	0.308	0.710	0.267	0.704	0.140	0.710	0.006	0.785	0.801	0.994	0.398	0.936	0.000

Tabla 5.15: P-valores del test T de Student tras normalización para longitud 2^{21}.

	Fr	**CuS0**	**CuS1**	**ApE**	**Se0**	**Se1**	**Lic**
Fr	0.000	0.000	0.000	0.614	0.531	0.352	0.368
CuS0	0.000	0.000	0.000	0.374	0.779	0.619	0.420
CuS1	0.000	0.000	0.000	0.094	0.155	0.113	0.318
ApE	0.614	0.374	0.094	0.000	0.005	0.891	0.394
Se0	0.531	0.779	0.155	0.005	0.000	0.000	0.413
Se1	0.352	0.619	0.113	0.891	0.000	0.000	0.008
LiC	0.368	0.420	0.318	0.394	0.413	0.008	0.000

Tabla 5.16: P-valores del test T de Student tras normalización para longitud 2^{22}.

	Fr	**CuS0**	**CuS1**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**
Fr	0.000	0.000	0.000	0.838	0.469	0.367	0.497	0.692
CuS0	0.000	0.000	0.000	0.793	0.344	0.758	0.539	0.866
CuS1	0.000	0.000	0.000	0.556	0.672	0.380	0.154	0.359
ApE	0.838	0.793	0.556	0.000	0.333	0.926	0.005	0.780
RaE0	0.469	0.344	0.672	0.333	0.000	0.001	0.359	0.351
RaEV0	0.367	0.758	0.380	0.926	0.001	0.000	0.887	0.422
Se0	0.497	0.539	0.154	0.005	0.359	0.887	0.000	0.000
Se1	0.692	0.866	0.359	0.780	0.351	0.422	0.000	0.000

Tabla 5.17: P-valores del test T de Student tras normalización para longitud 2^{23}.

	Fr	**CuS0**	**CuS1**	**ApE**	**Se0**	**Se1**
Fr	0.000	0.000	0.000	0.684	0.761	0.442
CuS0	0.000	0.000	0.000	0.646	0.589	0.678
CuS1	0.000	0.000	0.000	0.961	0.634	0.753
ApE	0.684	0.646	0.961	0.000	0.000	0.720
Se0	0.761	0.589	0.634	0.000	0.000	0.000
Se1	0.442	0.678	0.753	0.720	0.000	0.000

Tabla 5.18: P-valores del test T de Student tras normalización para longitud 2^{24}.

	Fr	**CuS0**	**CuS1**	**Ru**	**LoR**	**Ra**	**ApE**	**RaE0**	**RaEV0**	**Se0**	**Se1**	**LiC**
Fr	0.000	0.000	0.000	0.116	0.939	0.961	0.621	0.453	0.823	0.642	0.207	0.011
CuS0	0.000	0.000	0.000	0.472	0.655	0.296	0.952	0.927	0.912	0.460	0.306	0.030
CuS1	0.000	0.000	0.000	0.696	0.511	0.229	0.642	0.611	0.834	0.141	0.131	0.004
Ru	0.116	0.472	0.696	0.000	0.005	0.740	0.762	0.667	0.005	0.510	0.736	0.099
LoR	0.939	0.655	0.511	0.005	0.000	0.789	0.209	0.667	0.679	0.381	0.439	0.032
Ra	0.961	0.296	0.229	0.740	0.789	0.000	0.150	0.665	0.557	0.283	0.401	0.008
ApE	0.621	0.952	0.642	0.762	0.209	0.150	0.000	0.479	0.878	0.002	0.945	0.621
RaE0	0.453	0.927	0.611	0.667	0.667	0.665	0.479	0.000	0.044	0.021	0.002	0.016
RaEV0	0.823	0.912	0.834	0.005	0.679	0.557	0.878	0.044	0.000	0.138	0.942	0.933
Se0	0.642	0.460	0.141	0.510	0.381	0.283	0.002	0.021	0.138	0.000	0.000	0.809
Se1	0.207	0.306	0.131	0.736	0.439	0.401	0.945	0.002	0.942	0.000	0.000	0.404
LiC	0.011	0.030	0.004	0.099	0.032	0.008	0.621	0.016	0.933	0.809	0.404	0.000

Además, cabe destacar que tanto la Tabla 5.9 como 5.13 muestran muchos pares correlacionados. Recordamos que para la longitud 2^{20} se han utilizado 5.000 secuencias, mientras que para las demás se han usado 500. Por tanto, es posible que algunas relaciones sean más aparentes cuando tanto la longitud de la secuencia como el número de secuencia tienden hacia números grandes.

5.6. Elección de patrones para *NonOverlappingTemplate*

El código oficial de NIST SP 800-22 utiliza 148 patrones diferentes para esta prueba, resultando en 148 test diferentes. Sin embargo, algunos de estos pueden ofrecer información redundante. Como hemos visto en la Tabla 5.9, la prueba de patrones sin solapar con el patrón 000000001 tiene dependencias con la prueba de sumas acumuladas y de frecuencia, por ejemplo.

Al haber 148 patrones, hemos obtenido los p-valores de los $\frac{148 \cdot 147}{2}$ pares de test. Esta vez lo hemos hecho sin normalizar, tras ver en el apartado 5.5.2 que la normalización no altera las relaciones críticas. Al haber en total 10878 pares de test, esperaríamos observar $10878 \cdot 0.01 = 108.78$ fallos. Sin embargo, el valor observado es 388. Es más, si los test realmente fueran independientes, y tuviéramos probabilidad 0.01 de fallar cada par, se tendría una distribución Binomial $B(10878, 0.01)$, donde se entendiera fallar el test como un "éxito", y contabilizáramos los éxitos de 10878 intentos. Pues bien, se tiene que la probabilidad de obtener tantos "éxitos" si este fuera el caso es ínfima: $P(B(10878, 0.01) \geq 388) < 10^{-6}$.

Esto quiere decir que se deberían elegir los patrones con más cuidado, especialmente teniendo en cuenta que la gran cantidad de los patrones probados resulta en una demora sustancial, tal como se observa en la Figura 5.4.

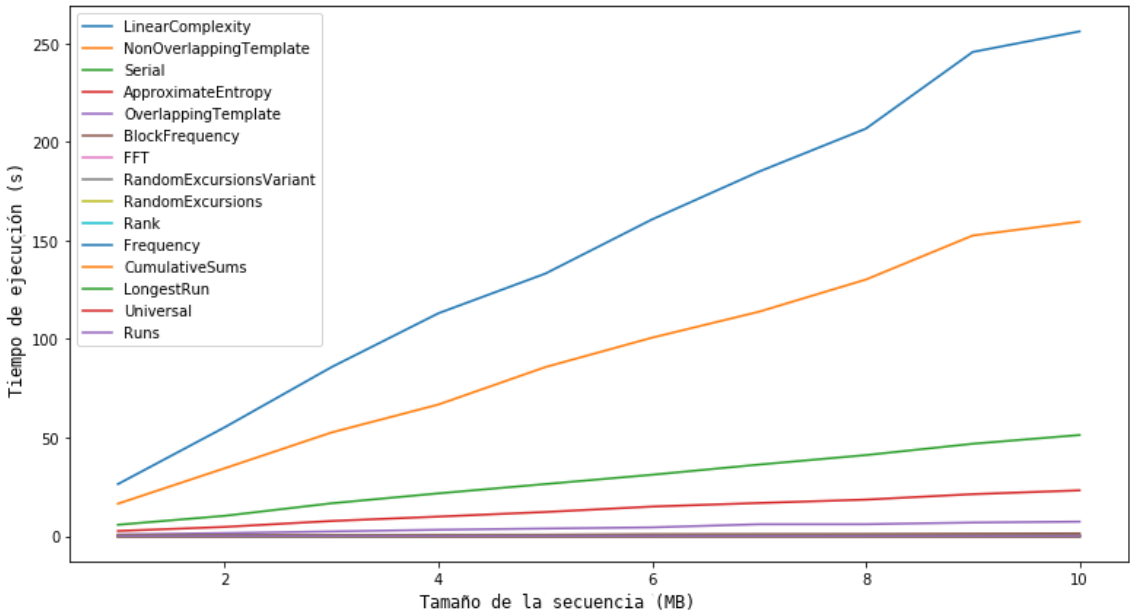


Figura 5.4: Tiempo de ejecución de cada test en segundos (incluyendo variantes) vs. tamaño en MB de la secuencia. Leyenda ordenada por tiempo de ejecución (de arriba abajo: de mayor tiempo a menor tiempo).

5.7. Estudio alternativo de independencia

En la Sección 5.5 hemos presentado los test que, a todas luces, no son independientes en la batería. También hemos visto algunas otras anomalías que podrían indicar dependencias en otros pares de test. Adoptamos otra perspectiva, propuesta en [FCG14], para ver si estos resultados son consistentes.

La idea de este artículo es bastante sencilla. Dado un par de test, suponiendo que sus p-valores son dos distribuciones uniformes $\mathcal{U}(0,1)$ e independientes, podemos hallar analíticamente la distribución de la resta de los p-valores de un test y el otro.

Si $X, Y \sim \mathcal{U}(0,1)$, entonces, la función de densidad de la resta $X - Y$ se puede calcular fácilmente. Sea $Z = X - Y$. Conocemos las funciones de densidad de X y $-Y$ por ser distribuciones uniformes, y por lo tanto solo queda tomar la función de densidad de su suma. Usamos la función indicatriz I al igual que en la Ecuación 5.2.

$$f_X(x) = I_{[0,1]}(x) \tag{5.5}$$

$$f_{-Y}(y) = I_{[-1,0]}(y) \tag{5.6}$$

$$f(z) = \int_0^1 f_X(x) \cdot f_{-Y}(z - x)dx \tag{5.7}$$

$$= \begin{cases} z + 1 & -1 \leq z \leq 0 \\ -z + 1 & 0 < z \leq 1 \\ 0 & e.o.c. \end{cases} \tag{5.8}$$

Teniendo esta distribución, podemos dividir el intervalo $[-1, 1]$ en n_i intervalos

$$I_0 = \left[-1, -1 + \frac{1}{n_i}\right), I_1 = \left[-1 + \frac{1}{n_i}, -1 + \frac{2}{n_i}\right), \ldots$$
$$I_{n_{i-1}} = \left[-1 + \frac{n_i - 1}{n_i}, 1\right]$$

De esta forma, podemos calcular las probabilidades de que la resta $X - Y$ pueda caer en cada uno de los intervalos. Así, se toman los S p-valores obtenidos en un **TestA** y un **TestB**, los cuales denotamos, respectivamente $(pva_0, ..., pva_{S-1})$ y $(pvb_0, ..., pvb_{S-1})$. Después se toman sus restas $(pva_0 - pvb_0, ..., pva_{S-1} - pvb_{S-1})$. Entonces se cuenta cuántas restas quedan en cada intervalo I_i. Esta cuenta la denotamos por S_i, y se compara con el número teórico de apariciones en ese intervalo, $S \cdot P_i$, donde $P_i = \int_{a_i}^{b_i} f(z)dz$, siendo a_i y b_i los extremos del intervalo I_i, y la f definida en la Ecuación 5.7.

A partir de estos datos podemos realizar una prueba de chi cuadrado, de la cual hacemos un test de una cola como se mostraba en la Figura 3.2:

$$X^2 = \sum_{i=0}^{n_i} \frac{(S_i - S \cdot P_i)^2}{S \cdot P_i} \tag{5.9}$$

Si $P(\chi^2_{S-1} \geq X^2) < \alpha = 0.01$, entonces estas S secuencias fallan el test. En caso contrario, lo aprueban. Repitiendo este test para C conjuntos de S secuencias, la expresión 5.1 nos da la proporción mínima de conjuntos que deberían aprobar el test.

Replicamos el experimento con la longitud 2^{20}, ya que es el que más secuencias tiene, 5.000, de las cuales tomamos 30 conjuntos de 166 secuencias cada uno. En [FCG14] se usan 30.000 secuencias. Mostramos los resultados en los que ha habido algún suspenso en la Tabla 5.19, pues por la expresión 5.1 deben aprobarlo al menos 28 test.

Tabla 5.19: El número de conjuntos, del total de 30, que pasan la prueba para cada par de test.

	Fr	**CuS0**	**CuS1**	**ApE**	**Se0**	**Se1**
Fr	0	0	0	30	30	29
CuS0	0	0	0	30	30	29
CuS1	0	0	0	30	29	30
ApE	30	30	30	0	26	28
Se0	30	30	29	26	0	0
Se1	29	29	30	28	0	0

Para un caso particular, vemos gráficamente en la Figura 5.5 el reparto de las restas en cada conjunto. Cada conjunto corresponde a una línea discontinua, y la línea continua con cuadrados muestra cómo deberían estar repartidas las restas en cada conjunto idealmente. Para el test *Frequency* y *CumulativeSums0*, tenemos que sus diferencias se concentran excesivamente en los intervalos centrales, o lo que es lo mismo, las restas de los p-valores pva_i y pvb_i se aglutinan en demasía alrededor del cero, de una forma que no cabría esperar si realmente fueran independientes.

Interpretando los resultados de la Tabla 5.19, observamos tres cosas:

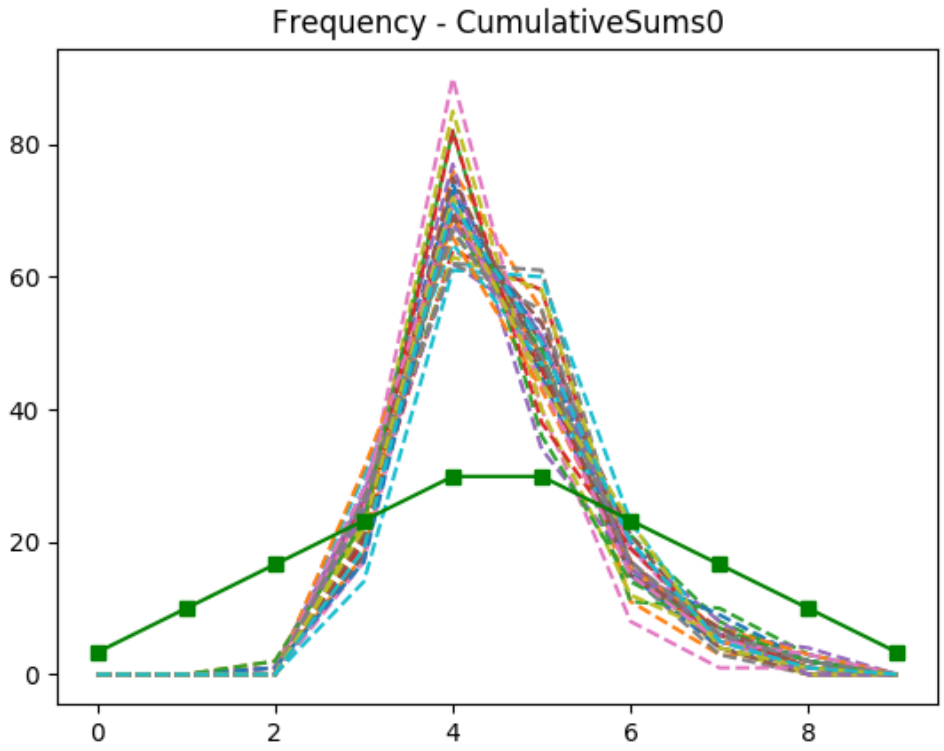


Figura 5.5: Reparto de los conjuntos (líneas discontinuas) vs. reparto teórico (línea verde).

- Los resultados de la Sección 5.5 son consistentes. *Frequency*, *CumulativeSums0* y *CumulativeSums1* son dependientes, como lo son *Serial0* y *Serial1*, y *Serial0* y *ApproximateEntropy*.
- Los resultados que hemos obtenido son menos pesimistas que en [FCG14]. Esto puede deberse a que han utilizado más datos de prueba, o a la naturaleza de sus datos, los cuales están obtenidos con el PRNG *Blum Blum Shub*. Unos datos que no tuvieran buenas propiedades de aleatoriedad no podrían utilizarse, ya que la premisa de que los p-valores estén repartidos uniforme e independientemente se tiene con datos aleatorios. Por otra parte, tal como hemos visto en la Tabla 5.9, el tener muchos datos nos ha servido para encontrar más relaciones, y es de esperar que por esto [FCG14] encontrara otras relaciones.

5.8. Interpretación y explicación analítica de los resultados

Ofrecemos la explicación de por qué hemos encontrado ciertas dependencias, y de qué manera proceder si se busca una elección óptima de los test, con el fin de decidir cuáles descartar en caso de dependencias.

- *Frequency* con *CumulativeSums0* y *CumulativeSums1*. Las pruebas de sumas acumuladas estudian el máximo valor absoluto de sumas parciales de la secuencia, interpretada como secuencia de 1s y -1s (3.2.13). Por consiguiente, las sumas acumuladas $S_n = \sum_{i=0}^{n-1} X_i = S'_n = \sum_{i=n-1}^{0} X_i$ estudian la suma de todos los bits de la secuencia, tal como lo hace la prueba de frecuencia. De esta forma, si esta suma tiene un valor absoluto demasiado alejado de cero, la secuencia fallará los tres test. Es decir, la debilidad que detecta el test de frecuencia también es detectada por las pruebas de sumas acumuladas. Por consiguiente, para una elección óptima de test, sería aconsejable decantarse por una sola prueba de las sumas acumuladas, y no usar la otra ni la de frecuencia.

- *CumulativeSums0* y *CumulativeSums1*. A pesar de que el primero está más centrado en las sumas acumuladas en los prefijos de la secuencia, y el segundo en los sufijos (3.2.13), salvo que las grandes diferencias estén localizadas muy en los extremos, sus resultados tienen una alta dependencia entre ellos.Esto hace que se pueda descartar uno de ellos. Cuál descartar es bastante indiferente, por sencillez podríamos descartar el de las sumas acumuladas hacia atrás.

- *ApproximateEntropy* y *Serial0*. Ambos test estudian la uniformidad de las palabras de un número fijo de bits, y toman la diferencia de cómo varía esta uniformidad al pasar de m a $m-1$ bits. Entonces, intuitivamente, si las 2^m y 2^{m-1} palabras de m y $m-1$ bits respectivamente están repartidas uniformemente en la secuencia, se obtendrá un buen resultado en ambos test. Si sin embargo, este no es el caso, se tenderá a resultados desfavorables. Por lo tanto, es comprensible esta dependencia. De elegir uno de los test, tal vez fuera más apropiado elegir el de entropía aproximada, conociendo el punto siguiente.

- *Serial0* y *Serial1*. Mientras que el primero estudia la uniformidad de patrones con longitudes m y $m-1$, el segundo también introduce los de $m-2$. Por consiguiente, al ser *Serial0* también dependiente con *ApproximateEntropy*, tal vez sea más adecuado decantarse por la segunda variante, *Serial1*.

- *RandomExcursions* y *RandomExcursionsVariant*. Aunque tengan dependencias para varias de las longitudes que hemos probado, no es evidente cuál puede ser más conveniente.

- *NonOverlappingTemplate*. Hemos visto que tienen bastantes dependencias entre ellos, y que también puede tener dependencias con *Frequency*, *CumulativeSums0* y *CumulativeSums1* al usar el patrón 000000001. Esto puede resultar intuitivo, ya que demasiadas o demasiadas pocas apariciones de esa subsecuencia pueden estar relacionadas con una prevalencia o déficit acentuado de ceros en la secuencia.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

Motivados primero por el estudio de de la criptografía para proteger comunicaciones 5G que permitan el uso extendido de la tecnología IoT, hemos avanzado al estudio estadístico de uno de los componentes criptográficos más importantes: los generadores de números aleatorios. Hemos estudiado la batería de test de aleatoriedad más utilizada en el mundo de la criptografía, para analizar qué información nos dan sus pruebas a la hora de detectar vulnerabilidades en la secuencia. Cualquier debilidad que tuviera un generador se podría explotar en su contra para abaratar ataques de fuerza bruta, o poder poner en entredicho la seguridad del sistema mediante criptanálisis.

Para realizar el experimento, hemos generado cinco mil ficheros de 12 MB del PRNG */dev/urandom*, y los hemos tratado sucesivamente como: 5.000 secuencias de 2^{20} bits, 500 de 2^{21}, 500 de 2^{22}, 500 de 2^{23} y 500 de 2^{24}. Con el TRNG *ChaosKey*, hemos hecho lo propio con 500 secuencias de 2^{20}, 2^{21}, 2^{22}, 2^{23} y 2^{24} bits, para asegurar que los resultados no fueran particulares del generador utilizado.

Hemos encontrado que, sin ningún lugar a dudas, los siguientes pares de test son dependientes:

- Frecuencia y sumas acumuladas hacia delante, *Frequency* - *CumulativeSums0*.
- Frecuencia y sumas acumuladas hacia atrás, *Frequency* - *CumulativeSums1*.
- Sumas acumuladas hacia delante y atrás, *CumulativeSums0* - *CumulativeSums1*.
- Entropía aproximada y primer test serial, *ApproximateEntropy* - *Serial0*.
- Las dos variantes del test serial, *Serial0* - *Serial1*.
- Paseos aleatorios y su variante, *RandomExcursions0* - *RandomExcursionsVariant0*.

Además, se ha encontrado que realizar el test de patrones sin solapar con el patrón 000000001 no es independiente de los test de frecuencia y de sumas acumuladas. Por lo tanto, de elegir un único patrón, sería mejor decantarse por uno diferente que este, que figura como primero en la configuración por defecto de la batería.

Por otra parte, se ha encontrado que el test de patrones con solape tal vez sea demasiado exigente tras sus ajustes realizados en [KUH04].

En lo referente a los test de patrones sin solape, se ha encontrado que, si usamos los patrones por defecto del código de NIST, se tiene un cierto nivel de redundancia, resultando en una demora inncesaria de tiempo de cómputo.

Hemos encontrado una gran cantidad de pares de test aleatorios que son dependientes con un nivel de confianza del 99 %. Si bien algunos de ellos no eran consistentes a través de las longitudes o para las dos fuentes de datos, puede haber dependencias que se manifiesten más explícitamente en longitudes específicas.

La novedad de nuestra contribución es que hemos tratado la correlación desde un punto de vista de inferencia estadística, pudiendo ofrecer p-valores que nos dieran información real sobre estas correlaciones, y sabiendo cuán significativos eran nuestros resultados, teniendo en cuenta los tamaños de las muestras. Además, hemos estudiando en profundidad la uniformidad de los p-valores, y hemos aplicado transformaciones a normalidad para atenernos a un marco de trabajo estricto en cuanto al uso de la estadística. Por otro lado, hemos relacionado las dependencias observadas con las definiciones de los test, ofreciendo una explicación lógica tras ellas.

Para ponerle el punto final, hemos razonado que, si queremos utilizar un subconjunto de los test que sean independientes, y así evitar redundancias mientras mejoramos el tiempo de cómputo, podemos decantarnos por descartar *Frequency*, *CumulativeSums1*, *Serial0*, e indiferentemente uno de *RandomExcursions0* o *RandomExcursionsVariant0*.

6.2. Trabajo futuro

Al ser el estudio de la aleatoriedad una tarea relativamente enrevesada, hay aún mucho que explorar. Al no haber ninguna prueba explícita que demuestre la independencia de los test de la batería SP 800-22, todavía es posible que se hallen muchas más dependencias. Debe tenerse en mente que investigaciones como la nuestra o [DSU+17], [FCG14], [BS19] y [SUKD17] pueden encontrar dependencias entre test, pero que estos estudios no puedan hallar que un par sea dependiente no quiere decir que sea independiente. Es decir, con estos test somos capaces de, en algunos casos, rechazar con cierto nivel de confianza la hipótesis de que dos test sean independientes. No ser capaces de rechazar la hipótesis no basta para concluir que la hipótesis es cierta. Por lo tanto, quedan diversas vías de avance en la materia, como por ejemplo:

1. Realizar más pruebas con los pares concretos cuyas correlaciones han obtenido p-valores bajos. Al localizarse la prueba en pocos pares, y pasar de procesar 188 pruebas a procesar muchas menos, se abre la posibilidad de probar una gran cantidad de datos, y secuencias de diversas longitudes, para explotar al máximo las posibles dependencias. Un caso especialmente interesante es el de los patrones sin solape, y ver qué patrón podemos elegir que sea independiente con todos los demás test.

2. Observar el comportamiento con secuencias aún más largas. Si bien hemos probado las secuencias más largas que existen en la literatura, aún queda margen para estudiar posibles comportamientos asintóticos, y tratar de analizar la supuesta escalabilidad de la batería.

3. Tratar de establecer relaciones de dependencia más sutiles que la correlación de Pearson. Nosotros hicimos algunos intentos insatisfactorios, con modelos de regresiones, análisis por clusters... Las dependencias no son sencillas de encontrar, y sobre todo cuesta explicar los resultados en un test en función de los obtenidos en otro. No obstante, puede haber otras propuestas interesantes como la de [FCG14] que permitan establecer dependencias que no se basen en la linealidad.

Finalmente, señalar que el presente Trabajo Fin de Grado ha dado lugar al siguiente artículo científico:

Bittor Alaña Olivares, Luis Javier García Villalba, Elena Salomé Almaraz Luengo, Darren Hurley Smith, Julio César Hernández Castro: "Further Analysis of the Statistical Independence of the NIST SP 800-22 Randomness Tests", actualmente en proceso de envío a la revista *IEEE Transactions on Information Forensics and Security*.

Capítulo 7

Introduction

7.1. Motivation

Ever since telematic communications exist, a myriad of possible threats rise: interception, tampering, or even sabotage of messages. We know that, even during the dusk of the Republic of Rome, Julius Caesar's generals would use a primitive cipher to mask their communications, in order to not reveal secrets should the messages be intercepted.

The field of mathematics which works on the confidentiality, integrity and authenticity of messages is cryptography. Through a series of algorithms, the contents of messages are hidden during the way between the sender and the receiver, such that the communication is kept secret.

As communication technologies advance, and the world approaches a scenario where all the devices we use every day are interconnected, there arises the need of protecting communications by the means of very limited hardware resources, as these devices will deal with very sensitive information. One of the most basic mechanisms to achieve this end, and which is pivotal for many other applications in telecommunications, is random number generation.

The so-called Internet of Things is an emerging field with massive potential. Billions of small devices, with low resources available in terms of hardware and power, will be connected and sharing information, in order to help us improve our day-to-day life from our homes to 4.0 Industry [ASHAM17]. The rapid advances in 5G networks are expected to be an important enabler of this technology, and hence it is paramount to protect the communications of these systems, especially considering they will cope with very sensitive information: people's health status, company secrets, the state of things in a house... One of the lightest ways of offering a cipher for these communications is the generation of random numbers.

Although it might seem trivial to generate numbers that appear to be random in a computer or embedded system, this is hardly the case. For their very nature, computers are deterministic instruments that follow algorithms. This means any procedure by which someone might generate numbers in a computer can lead to easily predictable outcomes.

Nonetheless, this unpredictability is a two-way road. On the one hand, for a sequence to be easily predictable is a problem if we want to use it as a secret key. On the other hand, if one could, starting from an initial secret, reproduce the same sequence in two separate

places, that is actually a desirable property. This would allow two agents who shared the initial secret to reproduce the same seemingly random sequence in two completely separate places, and they would then be able to use those random numbers that only they would know, in order to have ciphered communications.

Obtaining random and pseudo-random –this is how we refer to numbers which seem random but are deterministically obtained from an initial secret– numbers is a challenging task, though. A fundamental part of it is measuring if the output of a generator really behaves in a random fashion. In statistical terms, a metric is sought that will assess how solid these sequences are, so that they cannot be predicted without knowing the initial secret, so that any number is equally likely to show up in the sequence, and many other properties.

This is the aim of randomness test suites. They decide whether a given sequence behaves randomly or not. They use functions that we call statistics, which are functions that take a sequence of numbers as input, and output a certain value. Once these statistics are computed, and since one can theoretically prove what statistics one should expect to find in random sequences, the probability that these sequences do indeed come from a random source is computed.

Therefore, every time one of these randomness tests is performed, with a specific statistic, the statistic measures one specific property. This can be the amount of zeros and ones that the binary sequence has, the rhythm with which ones and zeros oscillate, the mean of the sequence regarded as a string of bytes treated as unsigned integers... Hence, if each test assesses one particular weakness, the next phenomenon occurs: on the one hand, passing all tests means no weakness was found in the sequence (which does not necessarily mean it is perfectly random), and on the other hand, it is fundamental that the weaknesses the tests seek do not overlap.

Tests must be independent. If a four test suite was used, out of which three tests were highly interdependent, and they essentially measured the same weakness, passing all three tests would give a fake feeling of safety, as only one kind of weakness would really be avoided. Reciprocally, failing the three would seem catastrophic, while actually only pointing at one deficiency, essentially.

That is why we set out to analyse one of the most widely used randomness test suites, NIST's SP 800-22. After thoroughly studying the state of the art, we carried out an experimental investigation, working with a vast amount of data, and analysing the from a statistical inference viewpoint, in order to achieve statistically significant results of the information that the tests really yield.

7.2. Context

This Final Year dissertation was done within the Group of Analysis, Security and Systems (Group GASS, https://gass.ucm.es/, Group 910623 of the UCM catalogue of recognised groups), and as part of the THEIA research project (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices), reference no. FEI-EU-19-04.

7.3. Object of the Investigation

NIST's SP 800-22 is the most well-established randomness test suite in the field of cryptography, and has been thoroughly scrutinised. As such, it is likely the biggest reference when assessing randomness, and it is vital that the information it provides is significant. This means their tests should be independent, and [BRS+10] states that this is the case.

Nonetheless, there has been an array of investigations which have questioned the independence of the tests, and showing varying results. We have made an effort to get a deep understanding of the state of the art, be able to replicate previous experiments, and then advance in the experimentation and interpretation of results.

From a statistical inference viewpoint, we have set the aim of quantifying with which degrees of confidence we can say a pair of tests is independent or not, we have offered a sound reasoning of why the observed dependencies take place, and we have suggested which decisions to take on the issue, and why. On top of that, we have looked further into the variants of the tests, with a focus on the computational cost that they incur on.

7.4. Workplan

The development of this work has been carried out in three phases:

1. **Research** in the beginning, we started by studying several implementations of PRNGs, especially for RFID systems. Afterwards, and having seen which metrics were used to assess these generators, we focused on studying randomness tests, by professor Julio Hernández-Castro's suggestion. We mainly set our attention on NIST SP 800-22, as it is the most widely used suite in cryptography, due to how demanding it is, and the backing it has from a highly reputed institution as NIST.

2. **Development**: we then went on to generate data for the experiment. This process was first carried out in the School of Mathematics. However, due to the confinement, the process had to be switched to our personal computers, which meant an important delay. Scripts were coded in C and Python in order to generate data, run the tests treating them as different length sequences, and to store these data in tables for their posterior statistical analysis. Two sources of randomness were tried, to be able to compare results.

3. **Statistical analysis and interpretation**: during the last months, the interpretation of the data observed was carried out, assessing the dependencies obtained, and the information redundancy that was found.

7.5. Struture of the Work

The work has 7 chapters from the introduction on, and is structured in the following fashion:

Chapter 2 sets the conceptual frame that the work is set on. That is, once the motivation and aims of the work are set, we set out to detail some of the most

basic concepts of cryptography and IoT. We introduce some of the physical aspects that IoT demands, some solutions that are devised for the light encryption problem, and basic notions on TRNGs and PRNGs, with one of the most relevant examples in telecommunications: LFSRs.

Chapter 3 explains how randomness is assessed in statistical terms, and in particular how NIST SP 800-22 does it. We introduce all the tests, with an informal explanation of their aim – to transmit what weaknesses they are designed to seek– and a mathematical formulation of the process of obtaining statistics and probabilities out of them.

Chapter 4 explains the state of the art, guiding the reader through previous investigations, and drawing conclusions from them. Moreover, the evolution of the suite over time since its inception in 2001 is explained. From the results reaped in other studies, and the areas that were left unexplored, we see where our investigation could make new contributions.

Chapter 5 is focused on our study. From the experimental part to the statistical analysis, we explain which data we generated and study, and then focus on the results, and make an interpretation of their meaning. We explain how we processed the data, why we transformed them, and linked our results with other results in the literature.

Chapter 6 shows the main conclusions and take away points of our work, as well as possible future ways of investigation to further study the topic.

Chapters 7 and 8 are English translations of the Introduction and Conclusions.

Capítulo 8

Conclusions and Future Work

8.1. Conclusions

Firstly motivated by the study of the use of cryptography to protect 5G communications that would enable IoT technologies, we have further advanced to the statistical study of one of the most important cryptography components: random number generators. We have studied the most widely used randomness test suite in the field of cryptography, to analyse what information their tests yield when detecting vulnerabilities in a sequence. Any weakness that a generator had could potentially be used to ease brute force attacks, or to compromise the security of the system through cryptanalysis.

To carry out the experiment, we generated five thousand files of 12 MB from the */dev/urandom* PRNG, and have subsequently treated them as: 5000 sequences of length 2^{20} bits and 500 sequences for respective lengths 2^{20} bits, 2^{21} bits, 2^{22} bits, 2^{23} bits and 2^{24} bits. We also used data from the TRNG *ChaosKey*, with 500 sequences for each of the former lengths. Therefore, we can say our results are not specific for the used generator.

We have found the following pairs of tests are undoubtedly dependent:

- Frequency and Cumulative Sums forwards.
- Frequency and Cumulative Sums backwards.
- Cumulative Sums forwards and backwards.
- Approximate Entropy and first Serial test.
- Both variants of the Serial test.
- Random Excursions and Random Excursions Variant.

Moreover, it was found that the Non Overlapping Template test with pattern 000000001 was not independent from the frequency and cumulative sums tests. Therefore, if one pattern were to be used, it would be preferable to choose a different one than this, which is the first default pattern in the suite.

Other than that, we found that the Overlapping Template test might be too strict, even after the corrections performed at [HK07].

On the Non Overlapping Template test variants, we found that using the default patterns by NIST results in redundancy, which in turn needlessly expands computation time.

We have found a big amount of pairs of randomness tests which are dependent with a degree of confidence of 99 %. Even if some of them did not hold for different lengths or different data sources, they might be worth looking into, in their specific lengths.

The novelty of our contribution is that we treated correlation from a statistical inference viewpoint, being able to offer p-values which give us real information on these correlations, and being able to tell how significant these results were, factoring in the sample sizes as well. Moreover, we have linked the dependencies observed with the test definitions, offering a logical explanation.

To top it off, we have reasoned that, if we want to use a subset of tests that will be independent, and avoid redundancies while improving computing costs, we can discard the Frequency test, the backwards Cumulative Sums test, the first variant of the Serial test, and either of the Random Excursions test or the Random Excursions Variant test.

8.2. Future Work

Since the study of randomness is a rather complicated issue, there is yet a lot to explore. Since there is no proof that the tests in the SP 800-22 suite are independent, many more dependencies might be found. We must bear in mind that investigations like our own, [DSU+17], [FCG14], [BS19] and [SUKD17] are able to find many dependencies between tests, but their failure in finding that a given pair is dependent does not make it independent. That is, these test are able to, in some cases, reject the hypothesis that two tests are independent. However, failing to reject that hypothesis does not make the hypothesis true. Therefore, there are many ways forward on the subject, as the following:

- Carrying out tests on the particular pairs for which we obtained low p-values. As these tests would be more localised, and the tests that would have to be processed far less, bigger amounts of data could be used, with sequences of many lengths, to exploit the possible dependencies thoroughly. An especially interesting case is the Non Overlapping Template test, so as to decide an optimal set of patterns.
- To observe the behaviour of even longer sequences. We have worked with the longest sequences in the literature, but there is still margin to study possible asymptotic behaviours, and analyse the scalability of the suite.
- Try and establish subtler dependency relations than the Pearson correlation. We have tried made some fruitless attempts, with regression models, cluster analysis... Dependencies are hard to find, and mostly it is hard to explain the results of a test as variables of results in other tests. Nonetheless, some innovative suggestions as the one in [FCG14] might be made, to establish dependencies that have nothing to do with linearity.

Finally, this dissertation has led to the following research paper: Bittor Alaña Olivares, Luis Javier García Villalba, Elena Salomé Almaraz Luengo, Darren Hurley Smith, Julio César Hernández Castro: "Further Analysis of the Statistical Independence of the NIST SP

800-22 Randomness Tests", currently in the process of sending to the *IEEE Transactions on Information Forensics and Security* journal.

Bibliografía

[AJ] David A. Jones. True Random Number Generators for a More Secure IoT. https://www.techdesignforums.com/practice/technique/true-random-number-generators-for-more-secure-systems/.

[ANSS15] Babacar A. Ndaw, Djiby Sow, and Mamadou Sanghare. Construction of maximum period linear feedback shift registers (LFSR) (Primitive Polynomials and Linear Recurring Relations). *British Journal of Mathematics & Computer Science*, 2015.

[ASHAM17] Godfrey Akpakwu, Bruno Silva, Gerhard Hancke, and Adnan Abu-Mahfouz. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access*, 5:3619 – 3647, 12 2017.

[Bro] Robert G. Brown. Dieharder: A random number test suite. http://webhome.phy.duke.edu/~rgb/General/dieharder.php.

[BRS+10] Lawrence E. Bassham, Andrew L. Rukhin, Juan Soto, James R. Nechvatal, Miles E. Smid, Elaine B. Barker, Stefan D. Leigh, Mark Levenson, Mark Vangel, David L. Banks, Nathanael Alan Heckert, James F. Dray, and San Vo. SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, USA, 2010.

[BS19] Paul Burciu and Emil Simion. A Systematic Approach of NIST Statistical Tests Dependencies. pages 1–6, 01 2019.

[Cad11] Tom Caddy. *FIPS 140-2*, pages 468–471. Springer US, Boston, MA, 2011.

[CLT] Central Limit Theorem. https://en.wikipedia.org/wiki/Central_limit_theorem.

[DG06] Ali Doğanaksoy and Faruk Göloğlu. On Lempel-Ziv Complexity of Sequences. In *Sequences and Their Applications – SETA 2006*, pages 180–189. Springer Berlin Heidelberg, 2006.

[DSU+17] Ali Doğanaksoy, Fatih Sulak, Muhiddin Uğuz, Okan Şeker, and Ziya Akcengiz. Mutual Correlation of NIST Statistical Randomness Tests and Comparison of their Sensitivities on Transformed Sequences. *Turkish Journal of Electrical Engineering & Computer Sciences*, 25:655–665, 2017.

[FCG14] Limin Fan, Hua Chen, and Si Gao. A General Method to Evaluate the Correlation of Randomness Tests. 8267:52–62, 01 2014.

[GG70] S. Gashkov and Igor Gashkov. The berlekamp-massey algorithm. a sight from theory of pade approximants and orthogonal polynomials. pages 561–564, 01 1970.

[Haa98] Dr. Mads Haahr. Random.org. www.random.org, 1998.

[HK07] Kenji Hamano and Toshinobu Kaneko. Correction of Overlapping Template Matching Test Included in NIST Randomness Test Suite. *IEICE Transactions*, 90-A:1788–1792, 09 2007.

[Kow72] Charles J. Kowalski. On the Effects of Non-Normality on the Distribution of the Sample Product-Moment Correlation Coefficient. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 21(1):1–12, 1972.

[Kre11] David Kremelberg. *Practical Statistics: a Quick and Easy Guide to IBM SPSS Statistics, STATA, and Other Statistical Software*. SAGE Publications, Los Angeles, 2011.

[KUH04] Song-Ju Kim, Ken Umeno, and Akio Hasegawa. Corrections of the NIST Statistical Test Suite for Randomness. 103, 02 2004.

[man] urandom(4): Linux man page. https://linux.die.net/man/4/urandom.

[OU17] Hiroki Okada and Ken Umeno. Randomness Evaluation with the Discrete Fourier Transform Test Based on Exact Analysis of the Reference Distribution. 2017.

[SKS+20] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim. Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios. *IEEE Access*, 8:23022–23040, 2020.

[SUKD17] Fatih Sulak, Muhiddin Uğuz, Onur Koçak, and Ali Doğanaksoy. On the independence of statistical randomness tests included in the NIST test suite. *Turkish Journal of Electrical Engineering & Computer Sciences*, 25:3673–3683, 01 2017.

[TBK+18] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A McKay, Mary L Baish, and Mike Boyle. Recommendation for the entropy sources used for random bit generation. Technical report, January 2018.

[Wal08] John Walker. ENT: A pseudorandom number sequence test program. https://www.fourmilab.ch/random/, 2008.

[Was] Rick Wash. Lecture Notes on Stream Ciphers and RC4. http://www.rickwash.com/papers/stream.pdf.

[Wik] Wikipedia. Kolmogorov-Smirnov test. https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test.

[ZML+16] Shuangyi Zhu, Yuan Ma, Jingqiang Lin, Jia Zhuang, and Jiwu Jing. More Powerful and Reliable Second-Level Statistical Randomness Tests for NIST SP 800-22. In *Advances in Cryptology – ASIACRYPT 2016*, pages 307–329. Springer Berlin Heidelberg, 2016.