UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Nuevos Protocolos y Esquemas de Seguridad para Redes Ad-hoc Móviles Inalámbricas

Autor:
D. Oscar Delgado Mohatar

Director/es:
Dr. D. José María Sierra Cámara
Dra. Dª Amparo Fúster Sabater

Departamento de Informática

Leganés, Noviembre 2010

TESIS DOCTORAL

Nuevos Protocolos y Esquemas de Seguridad para Redes Ad-hoc Móviles Inalámbricas

Autor: D. Oscar Delgado Mohatar

Director/es: Dr. D. José María Sierra Cámara
Dra. Dª Amparo Fúster Sabater

Firma del Tribunal Calificador:

Firma

Presidente:

Vocal:

Vocal:

Vocal:

Secretario:

Calificación:

Leganés, de de

Índice general

Índice de figuras

ÍNDICE DE CUADROS

CAPÍTULO I

INTRODUCCIÓN

I.1. EL NUEVO ESCENARIO DE JUEGO

Resulta innegable que, recién comenzado el siglo XXI, Internet ha modificado nuestro estilo de vida hasta niveles inconcebibles hace tan sólo unos años. Algunas voces han denominado ya el periodo histórico que vivimos como el *Siglo Internet* y otras se han atrevido a comparar la Red con la invención de la imprenta por Gutenberg en el siglo XV, argumentando que el impacto de ambos avances en la sociedad es similar.

Al margen de que estas afirmaciones se nos antojan algo exageradas (¿es realmente comparable a la auténtica revolución que supuso la generalización de la energía eléctrica en los hogares o el descubrimiento de la penicilina?), lo cierto es que la Web, el correo electrónico o la mensajería instantánea han revolucionado todos los aspectos de nuestras vidas, modificando hábitos y costumbres.

Estas herramientas se han convertido en instrumentos de trabajo imprescindibles en cualquier organización o empresa y cada vez resulta más habitual sorprendernos a nosotros mismos u oír de labios de los más jóvenes reflexiones como: "Pero, ¿cómo se hacía esto antes de que existiera Internet?".

I.1.1. La Internet de las cosas

Sin embargo, los cambios se suceden a una velocidad de vértigo y lo visto hasta ahora no es más que la punta del iceberg. La denominada *Internet de las cosas* está considerada por muchos como la próxima gran revolución en la actual Sociedad de la Información y en nuestro actual modo de vida [1].

En pocas palabras, podría decirse que la próxima generación de Internet consiste en *una red de todo tipo de objetos con capacidad de conexión entre sí, junto con una serie de servicios, típicamente Web, que interactúan con dichos objetos*. Entre algunas de las tecnologías que soportan este nuevo enfoque se encuentran RFID (identificación por radio frecuencia), sensores o los propios teléfonos móviles de última generación (*smartphones*).

Aunque algunos de estos escenarios puedan parecer innecesarios, o incluso frívolos, lo cierto es que este nuevo paradigma de comunicación puede proporcionar otros usos realmente beneficiosos.

Es el caso de las *redes de monitorización de pacientes en hospitales*, que harían innecesarios los incómodos cables y permitirían tener a un paciente permanentemente monitorizado, incluso cuando pasea por el pasillo.

Como estas redes no necesitan de una infraestructura previa de comunicaciones, pueden resultar muy útiles en casos de desastre natural o emergencias, donde es habitual que las redes tradicionales resulten dañadas.

Otros usos menos altruistas, pero probablemente mucho más rentables económicamente, incluyen una mejora sustancial en todo el proceso de manufactura de bienes, desde su

fabricación hasta su consumo, a través de la tecnología RFID. Un consumidor, por ejemplo, podría leer la etiqueta de la verdura que acaba de adquirir y conocer cuándo fue producida, empaquetada, cuánto ha viajado desde su lugar de origen, si la temperatura de almacenamiento ha sido óptima en todo momento o si contiene algún tipo de aditivo o conservante.

Las nuevas comunicaciones Todo esto ha sido y será posible gracias a la conjunción de muchos factores, entre los que destaca el avance imparable de la *miniaturización electrónica*, que permite crear dispositivos cada vez más capaces y, al mismo tiempo, reducir su tamaño.

Pero parece claro que el verdadero factor detonante ha sido la evolución de las *comunicaciones inalámbricas* que, aunque presentes y ubicuas desde hace décadas, sólo en los últimos años han alcanzado el grado de sofisticación necesario para hacer realidad los escenarios que hemos descrito.

I.2. LAS REDES INALÁMBRICAS Y LA TELEFONÍA MÓVIL

De los múltiples criterios utilizados para clasificar las redes de comunicaciones, entre los que se incluyen su escala, su método de conexión, la topología que forman o los protocolos que utilizan, en los últimos años ha cobrado especial importancia el *medio de transmisión*.

Cuando el cable tradicional se sustituye por transmisión a través del aire se habla de *redes inalámbricas*. En este caso la comunicación se lleva a cabo utilizando un medio no guiado, mediante ondas electromagnéticas, y haciendo uso de antenas.

No cabe duda de que la tecnología inalámbrica está ocupando rápidamente las preferencias de todo tipo de usuarios. La telefonía móvil está cada vez más cerca de convertirse en un sistema de comunicación personal universal en el mundo occidental y, desde hace unos años, todo tipo de ordenadores están librándose también de sus ataduras cableadas. Cada vez son más los hogares, cafés, pequeñas empresas, aeropuertos o grandes compañías en los que se dispone de redes inalámbricas de ordenadores.

Aunque las tecnologías que hacen posible las comunicaciones inalámbricas, como el láser, los rayos infrarrojos o las ondas de radio, existen desde hace muchas décadas, su implantación comercial y su aplicación a las comunicaciones no ha sido posible hasta fechas recientes. El primer servicio que se liberó del cable fue la telefonía y no lo hizo hasta los años 80 en EEUU y bastantes años después en España. Desde aquellos aparatos que pesaban varios kilos y costaban miles de dólares, la telefonía móvil no ha dejado de crecer espectacularmente hasta superar a la telefonía fija en cuanto al número de líneas.

En este sentido, dispositivos como el iPhone de Apple, un auténtico fenómeno de masas, están redefiniendo los límites de los teléfonos móviles tradicionales y evolucionan hacia dispositivos ubicuos de computación [2, 3]. Cualquiera de estos teléfonos de última generación podría jugar un papel predominante en la *Internet de las cosas*, el nuevo tipo de redes que se considera en esta tesis.

Inconvenientes de las redes inalámbricas tradicionales Las redes de comunicaciones inalámbricas tradicionales, como GSM o las más recientes 3G, utilizan un esquema típicamente centralizado [4] que no puede utilizarse en todas las situaciones. Algunos ejemplos significativos pueden ser escenarios de crisis, en los que son necesarias unas comunicaciones fiables, eficientes y dinámicas, o redes de vigilancia de lugares de difícil acceso.

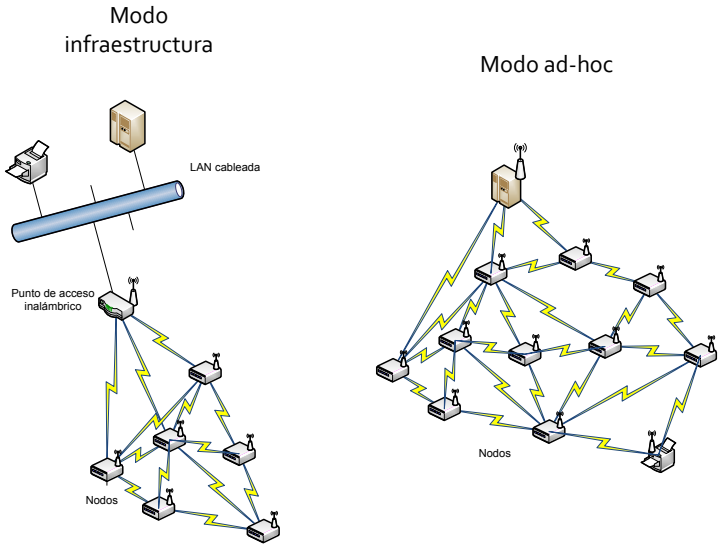


Figura I.1: Funcionamiento esquemático de los modos infraestructura y ad-hoc

Las soluciones aportadas para estos escenarios por otros protocolos, como el estándar 802.11 o el más reciente 802.16 (WiMax), no resultan del todo adecuadas. El primero de los protocolos, sin duda el más extendido en la actualidad, tiene dos modos básicos de funcionamiento, llamados *modo infraestructura* y *modo ad hoc*, que aparecen esquematizados en la Figura I.1 y analizados en [5, 6].

Ambos métodos tienen una serie de inconvenientes importantes en los escenarios planteados. El primero de ellos necesita, como su nombre indica, una infraestructura de comunicaciones *pre-existente*, que consiste habitualmente en un router. Este requerimiento lo hace inadecuado para, por ejemplo, situaciones de crisis donde es imposible prever dónde será necesario implantar la red de comunicaciones.

Respecto al segundo método, el llamado modo ad-hoc, adolece de serios inconvenientes:

- La comunicación es siempre **punto a punto**. Esto implica que el número de conexiones crece exponencialmente con el número de nodos de la red, lo que la convierte en inmanejable cuando éstos superan las pocas decenas.

- Aunque es cierto que no necesita una infraestructura pre-existente, la comunicación entre los nodos **no es multisalto**, lo que implica que no se aprovecha al máximo el carácter distribuido de la red. En este contexto, el término *multisalto* hace referencia a la capacidad de la red para que la comunicación entre dos nodos cualesquiera pueda utilizar los nodos intermedios a modo de "repetidores".

En la Figura I.2 se aprecia la diferencia entre el comportamiento de una red 802.11 en modo ad-hoc y el enrutamiento multisalto:

Como puede apreciarse, todos estos requisitos se traducen en que las redes tradicionales resulten inadecuadas para los nuevos escenarios planteados en la Sociedad de la Información. Por tanto, es necesario desarrollar nuevos esquemas de comunicaciones como las denominadas *redes MANets* (Mobile Ad-hoc Networks).

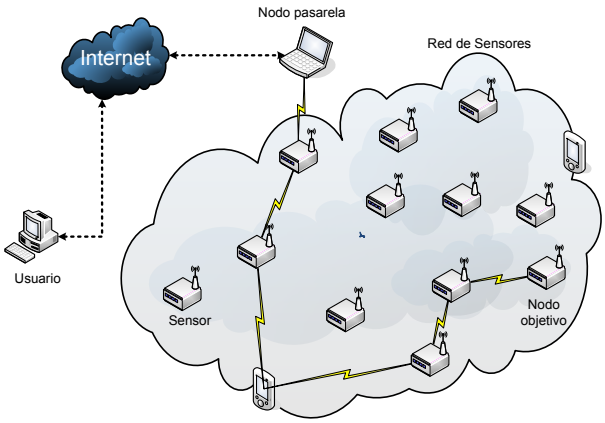


Figura I.2: Comportamiento multisalto

I.2.1. Redes móviles ad-hoc

Una red MANet no es más que un *conjunto de nodos con topología de red ad-hoc, es decir no necesita de una infraestructura previa de comunicaciones pero sí tiene capacidad de movimiento*. Como es previsible, esta definición tan amplia da cobijo a multitud de diversos enfoques, clasificados en base a factores tales como la capacidad de proceso de los nodos, de su tipo de alimentación, de su movilidad, del ancho de banda disponible o de su uso proyectado.

Con el fin de obtener un marco de trabajo más preciso, a continuación se presenta una breve clasificación o taxonomía en base a lo que constituye, a juicio de los autores, los dos factores más característicos de este tipo de redes: la *capacidad de cómputo* y la *movilidad de sus nodos*.

Una esquematización de dicha clasificación puede observarse en la Figura II.1.

En primer lugar aparecen las *redes MANet* que, además de dar nombre al concepto global de red móvil ad-hoc, constituyen uno de sus tipos. Son, quizás, las más extendidas y desarrolladas en la actualidad. Han sido clasificadas como poseedoras de una capacidad de computo media-alta y una movilidad media-baja. Tradicionalmente una MANet está formada por ordenadores portátiles o teléfonos móviles de última generación, cuyos usuarios se mueven a pie y, por tanto, tienen una movilidad media o baja.

A continuación y por orden de relevancia se encuentran las *redes de sensores*, clasificadas con una capacidad de proceso medio y movilidad baja. Estas redes están mayoritariamente compuestas por nodos, cuya CPU es un microcontrolador, que no se mueven o si lo hacen es muy lentamente.

Continuando el descenso en la escala de capacidad de proceso, aparece la *tecnología RFID*, que puede considerarse extrema en este sentido, pues la mayoría de sus dispositivos no disponen ni siquiera de alimentación energética propia. Se trata de elementos muy baratos, que se implementan habitualmente en forma de etiquetas adhesivas, y utilizados en un sinfín de aplicaciones como el seguimiento de productos en almacenes y centros comerciales.

Finalmente, en el extremo más alto de la movilidad, se encuentran las *redes VANet*, que podrían considerarse un subconjunto especial de redes ad hoc, implantadas en vehículos, ya sean coches o camiones, que circulan por carreteras convencionales.

Estos tipos de redes serán analizados con mayor detalle en la sección II.2.1 del

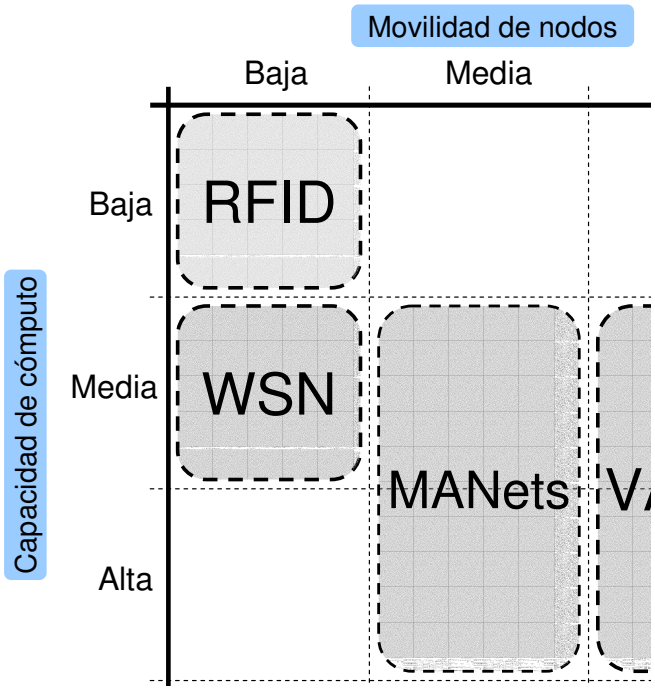


Figura I.3: Taxonomía de los diferentes tipos de redes ad hoc, clasificadas en base a la capacidad de proceso y movilidad de sus nodos

Capítulo II.

I.3. NECESIDAD DE LA TESIS

Como cabe esperar, la introducción de nuevos escenarios abre la puerta a nuevos requisitos y dificultades que deben ser resueltas. Una de las principales, desde luego, consiste en una necesidad imperiosa de esquemas de protección robustos, que proporcionen un adecuado nivel de seguridad. Es fácil imaginar lo que podría suceder si un atacante pudiese suplantar, por ejemplo, al dueño de una casa, y abrir la puerta del garaje a distancia, reprogramar la nevera para realizar compras en su nombre o subir y bajar persianas a voluntad.

Resulta obvio, por tanto, que la seguridad es uno de sus pilares básicos, sin el que resulta imposible el desarrollo de este tipo de redes. Sin embargo, *seguridad* es un concepto muy polifacético, más aún en redes tan complejas. En cualquier caso, los aspectos de *privacidad y autenticación* de la información deben formar parte indiscutiblemente de cualquier esquema que se proponga.

Por otro lado, teniendo en cuenta las restricciones ya comentadas de unas reducidas capacidades de proceso y energía de los nodos que forman parte de las redes ad hoc, el diseño de nuevos algoritmos para la protección de la información se convierte en una tarea desafiante.

Esta tesis surge, pues, de la necesidad clara de nuevos protocolos y mecanismos de seguridad para este tipo de redes. Un análisis de las propuestas existentes en la literatura especializada, que será expuesto con mayor detalle en la sección II.3 del Capítulo

II, muestra una serie de carencias significativas.

La primera de ellas es que un alto porcentaje de los autores trabajan con supuestos de partida que 'rompen' claramente las reglas del juego, puesto que no respetan alguna de las características definitorias de estas redes.

Por un lado, existen una serie de propuestas, como [7, 8], que introducen esquemas con necesidad de una infraestructura previa y centralizada de comunicaciones. Además, utilizan criptografía asimétrica, completamente fuera de las posibilidades reales de la mayoría de los dispositivos que conforman este tipo de redes.

Otras asumen la existencia de hardware anti-manipulación en los dispositivos [7, 9, 10]. Aunque es un campo activo de investigación y puede sin duda aceptarse en algunos escenarios, como la transmisión multimedia y la protección de contenidos digitales, resulta sin embargo inviable en la mayoría de las situaciones, debido al importantísimo incremento en coste que supone.

En resumen, el análisis anterior ha revelado la existencia de un hueco en la investigación realizada hasta el momento dentro del área que pretende cubrir esta tesis. Este espacio por desarrollar consiste en la inexistencia de buenas propuestas de protocolos de seguridad ligeros, que proporcionen tanto privacidad como autenticación, y que puedan ser ejecutados sobre dispositivos de gama baja; es decir dispositivos con una capacidad de cómputo limitada y que no dispongan de hardware anti-manipulación. De esta forma, pretendemos encontrar soluciones *realistas y económicas* que puedan ejecutarse con el hardware que se encuentra fácilmente en el mercado.

I.4. OBJETIVOS DE LA TESIS

Tomando como base las necesidades recién comentadas, se analizan a continuación los objetivos esenciales de este trabajo de tesis, detallando también cada uno de los hitos parciales que han permitido su consecución.

De forma muy general, en esta tesis se pretende plantear soluciones factibles y realistas a algunos de los nuevos retos de seguridad asociados a las redes ad hoc móviles. Algunos autores [11] consideran que son cuatro los grandes retos de seguridad por resolver antes que estas redes puedan generalizarse:

- La *autenticación* de dispositivos.
- El establecimiento seguro de *claves de sesión* entre dispositivos, para poder proporcionar privacidad a las comunicaciones.
- El *encaminamiento seguro* en redes multisalto.
- El *almacenamiento seguro* de claves criptográficas en este tipo de dispositivos.

A los anteriores requisitos, los autores añadirían la necesidad de una mayor investigación en el ámbito de la privacidad de las comunicaciones, a través de algoritmos de cifrado especialmente diseñados para ser ejecutados en dispositivos de baja capacidad computacional. Dado el enfoque eminentemente criptográfico de esta tesis, el esfuerzo principal se ha centrado en los mecanismos de cifrado y autenticación que, por otra parte, constituyen los pilares básicos de toda solución de seguridad.

Por estas razones, y a grandes rasgos, los objetivos principales que se pretenden alcanzar en este trabajo de tesis son:

- Diseño de *nuevos mecanismos y protocolos de seguridad para una arquitectura de comunicaciones ad hoc*, que pueden subdividirse en dos grandes grupos:

Confidencialidad: Análisis y desarrollo de un cifrador en flujo adecuado para este tipo de redes.

Autenticación: Desarrollo de un protocolo de autenticación ligero, energéticamente eficiente y capaz de ejecutarse sobre dispositivos de baja capacidad de cómputo.

- Análisis del nivel de seguridad de las soluciones diseñadas e implementación de las mismas.
- Aplicación de los protocolos en un entorno real, a través de un caso de estudio concreto consistente en la transmisión segura de contenidos multimedia (audio y vídeo) en redes ad hoc.

A continuación se describen estos objetivos con mayor detalle, comenzando con los relacionados con la protección de la privacidad de las comunicaciones.

Objetivos de confidencialidad El cifrador en flujo que se presenta está basado en el uso de los denominados *registros de desplazamiento con realimentación lineal (LFSRs)*. Tradicionalmente estas estructuras se definen sobre el cuerpo algebraico binario $GF(2)$, que proporciona un bit de salida en cada pulso de reloj. De hecho, todo el estudio matemático y las tablas habitualmente utilizadas para el diseño de LFSRs se encuentran definidas para este cuerpo [12].

Como veremos en este trabajo de tesis, esta aproximación puede ser adecuada en implementaciones hardware pero nunca en implementaciones software. Dado que el tamaño de los registros de los ordenadores actuales es de, como mínimo, 8 bits en las máquinas menos potentes y de hasta 64 bits en las de mayor potencia, realizar implementaciones orientadas al bit es claramente ineficiente.

El objetivo es, por tanto, utilizar cuerpos finitos más apropiados para la arquitectura de los actuales microprocesadores. En este sentido, las elecciones naturales son los cuerpos de Galois extendidos, con 2^n elementos, donde n está relacionado con el tamaño de los registros del propio microprocesador, que normalmente serán bytes o palabras de 16 ó 32 bits.

En este sentido, el conjunto de hitos parciales planteados son los siguientes:

- Análisis de la estructura de registros de desplazamiento definidos sobre cuerpos algebraicos compuestos, en contraposición a los tradicionales $GF(2)$, y obtención de una metodología para su construcción.
- Estudio del rendimiento obtenido por estos registros, que denominaremos *LFSRs extendidos*, en comparación con los tradicionales. Se muestran los beneficios de este enfoque, proporcionando datos empíricos que avalan los resultados, y que demuestran que éste método resulta mucho más eficiente que los *LFSRs* tradicionales.
- Asimismo, se realiza un análisis exhaustivo de los diferentes métodos para la implementación software de *LFSRs*, comparándolos y proporcionando un veredicto final en base a los datos empíricos obtenidos.

Objetivos de autenticación El otro gran pilar, imprescindible en toda solución de seguridad, consiste en el proceso de *autenticación*.

En este trabajo de tesis se tratará, por tanto, de encontrar un esquema de autenticación que tenga en cuenta las grandes restricciones en este tipo de escenarios, anteriormente mencionadas, y que cuente además con las siguientes características:

- *Resistencia contra captura de nodos*. Como escenario de trabajo se asumirá que el adversario puede atacar físicamente un nodo de manera sencilla y extraer de él toda la información secreta almacenada en su memoria. Una métrica habitual para medir esta resistencia es calcular la fracción de todas las comunicaciones de la red que se ven comprometidas por la captura de x nodos [13].
- *Resistencia contra replicación de nodos*. Mide la capacidad del adversario para insertar nodos hostiles en la red, utilizando la información secreta obtenido tras el compromiso de alguno(s) de ellos.
- *Escalabilidad*. El mecanismo propuesto debe escalar adecuadamente, es decir, debe tener la capacidad de adaptarse a tamaños de redes cada vez mayores, sin que la solución deje de funcionar o vea rebajados los niveles de seguridad que proporciona.

I.5. ORGANIZACIÓN DE LA MEMORIA

La presente memoria se organiza de la siguiente forma. Tras el presente capítulo de introducción, el Capítulo II presenta los detalles previos e imprescindibles que deben conocerse sobre este tipo de redes, junto con un análisis del actual estado del arte en el área.

Posteriormente el Capítulo III introduce algunos conceptos teóricos sobre los registros de desplazamiento para, a continuación, presentar los denominados *registros extendidos* y analizar sus características más importantes. A continuación, se compararán diferentes métodos de multiplicación en cuerpos algebraicos extendidos, que resultan esenciales para el funcionamiento de estos registros. Finalmente, se presentarán y analizarán una serie de técnicas, cuyo fin es mejorar, en la medida de lo posible, el rendimiento de las implementaciones de estos registros en software.

A continuación, el Capítulo IV presenta el diseño de un nuevo cifrador, denominado *LFSRe*, que utiliza como componente principal los registros descritos con anterioridad. Se presentan, también, diversos aspectos relacionados con su funcionamiento e implementación, para acabar analizando su rendimiento en comparación con otros cifradores de vanguardia.

El Capítulo V da respuesta a otro de los grandes objetivos de esta tesis, asegurar la autenticidad de las comunicaciones, aportando un nuevo protocolo de autenticación específicamente diseñado para redes de sensores.

Por otro lado, el Capítulo VI describe las implementaciones de las soluciones anteriores en diversos entornos reales, con el fin de demostrar su viabilidad y obtener datos fiables y más precisos sobre su rendimiento y otros aspectos, como su consumo energético.

Finalmente, las conclusiones presentes en el Capítulo VII y la Bibliografía completan esta tesis.

CAPÍTULO II

ANTECEDENTES Y ESTADO DE LA CUESTIÓN

Analizando los grandes objetivos planteados en el capítulo anterior, resulta evidente que la presente memoria es en gran medida *multidisciplinar*, pues para su realización ha necesitado profundizar en varios ámbitos bien diferenciados. El primero de ellos, relacionado con el objetivo de proporcionar *privacidad*, incluye aspectos matemáticos, como los cuerpos de Galois extendidos y su álgebra asociada, necesarios para definir de forma rigurosa el nuevo cifrador en flujo presentado. Todos estos fundamentos matemáticos serán presentados en la sección II.1.

Por otro lado, las soluciones de seguridad aportadas, tanto las relacionadas con la privacidad como con la autenticidad de las comunicaciones, no pueden desligarse de las redes para las que han sido diseñadas. Se hace imprescindible, por tanto, aportar una visión global de las mismas, que resuma sus características más importantes y haga especial hincapié en aquéllas que afectan a las soluciones anteriores. Estos aspectos mayoritariamente tecnológicos se analizarán en la sección II.2.

Por último, esta primera parte de antecedentes dará paso en el punto II.3 al estado de la cuestión sobre el que se cimienta esta memoria, completando así el presente capítulo.

II.1. FUNDAMENTOS DE LOS CUERPOS DE GALOIS EXTENDIDOS

Esta sección describirá brevemente un conjunto de definiciones y propiedades básicas de los denominados *cuerpos algebraicos finitos*, que resultan imprescindibles para la posterior exposición del trabajo realizado. Aunque la mayoría de los resultados y teoremas se presentan sin demostración, por no resultar imprescindibles para la comprensión de las aportaciones realizadas, éstas pueden encontrarse junto con una información más detallada y ampliada en cualquiera de las referencias citadas a continuación.

A pesar de la tremenda importancia del área, resulta llamativo el pequeño número de publicaciones dedicadas por completo al estudio de los cuerpos finitos. Durante mucho tiempo, de hecho, ha sido más fácil encontrar esta información en libros orientados a codificación algebraica, como el clásico de Peterson *"Error-correcting codes"* [1]. Afortunadamente esta situación se ha corregido, pudiendo citar a Lidl y Niederreiter's [2] y a McEliece [3] como referencias imprescindibles en este campo.

II.1.1. Grupos, cuerpos y sus propiedades

II.1 Definición. (Grupo): Un *grupo* es un conjunto de elementos G sobre el que se define una *operación binaria* $*$ que cumple las siguientes propiedades:

- *Propiedad asociativa*: Para todo $a, b, c \in G$, $a * (b * c) = (a * b) * c$.
- *Elemento neutro*: Existe un único $e \in G$ tal que para todo $a \in G$ se cumple que $e * a = a * e = a$.

- *Elemento inverso*: Para todo $a \in G$ existe un único $a' \in G$ tal que $a * a' = a' * a = e$.

II.2 Definición. Si las propiedades 1), 2) y 3) anteriores se cumplen, $(G, *)$ se consiera un grupo. Además, si $(a * b) = (a \cdot b)$, se dice que G es un *grupo multiplicativo*. Por otro lado, si $(a * b) = a + b$, entonces G forma un *grupo aditivo*. Si, además, se cumple que para todo $a, b \in G,\ a * b = b * a$, entonces se dice que G es un grupo *abeliano o conmutativo*.

II.1 Ejemplo. *El conjunto de los números enteros $\mathbb{Z}$ y la operación suma definen un grupo abeliano. El elemento neutro es el 0, mientras que el inverso de un elemento $a \in \mathbb{Z}$ es $-a$.*

II.2 Ejemplo. *Sin embargo, el mismo conjunto de los números enteros $\mathbb{Z}$ bajo la operación multiplicación no define un grupo abeliano, ya que ningún elemento excepto el par $\{+1, -1\}$ tiene inverso.*

II.3 Definición. (Orden de un grupo): Un grupo G puede ser finito o infinito. Si el grupo es finito se define su *orden*, $ord(G)$ ó $|G|$, como el número de elementos contenidos en el conjunto que lo define.

II.3 Ejemplo. *El orden del grupo abeliano aditivo $\mathbb{Z}_n$ es n. Por otro lado, el orden del grupo multiplicativo $\mathbb{Z}_n^*$ es $\phi(n)$, siendo ϕ la función de Euler.*

II.4 Definición. (Subgrupo): Sea G un grupo abeliano y H un subconjunto no vacío de G tal que:

- $a * b \in H$ para todo $a, b \in H$ y
- $a' \in H$ para todo $a \in H$

Entonces se dice que *H es un subgrupo de G.*

II.4 Ejemplo. *Si $n \in \mathbb{Z}$, entonces $H = n\mathbb{Z}$ es un subgrupo de $\mathbb{Z}$.*

II.1 Proposición. *Sea G un grupo y $A \subseteq G$. Se tiene lo siguiente:*

- *$\langle A \rangle$ es un subgrupo de G*
- *$\langle A \rangle$ es el menor subgrupo que contiene a A. Es decir, si existe un subgrupo H que contiene a A, entonces también contiene a $\langle A \rangle$.*

II.5 Definición. (Subgrupo generador): Al subgrupo $\langle A \rangle$ se le denomina *subgrupo generado* por A. Se dice que A es un sistema de generadores de $\langle A \rangle$. Si $G = \langle A \rangle$, entonces A es un sistema de generadores de G.

II.6 Definición. (Subgrupo cíclico): Sea G un grupo, $a \in G$ y $A = \{a\}$. Entonces

$$\langle \{a\} \rangle = \{a^n : n \in \mathbb{Z}\}$$

y se le denomina *subgrupo cíclico* de G generado por a. En otras palabras, un grupo cíclico es un grupo que puede ser generado por un solo elemento o, lo que es lo mismo, existe un elemento a del grupo G, denominado *generador de G*, tal que todo elemento de G puede ser expresado como una potencia de a.

II.7 Definición. (Teorema de Lagrange): El orden de un subgrupo H de G divide al orden de G.

II.*1 Observación.* Es importante destacar que el enunciado recíproco del teorema de Lagrange es falso, pues existen grupos de orden m que no cuentan con subgrupos de orden n a pesar de que $n \mid m$. Sin embargo, en el caso de grupos finitos abelianos, el recíproco sí se cumple.

II.8 Definición. (Cuerpo): Un *cuerpo* es una terna $< F, \cdot, + >$ donde $< F, \cdot, + >$ es un grupo abeliano aditivo con elemento neutro 0, $< F\backslash\{0\}, \cdot >$ es un grupo multiplicativo con elemento identidad 1 y $< F, \cdot, + >$ verifica la propiedad distributiva.

En otras palabras, dicho grupo forma un cuerpo si todo elemento distinto de 0 tiene un inverso multiplicativo. Es fácil ver que el inverso multiplicativo de a es único, por lo que podemos denotarlo con un símbolo especial a^{-1} . Es decir, si $a \neq 0$:

$$aa^{-1} = a^{-1}a = 1$$

.

II.5 Ejemplo. *Los ejemplos clásicos son los cuerpos de números* $\mathbb{Q}$, $\mathbb{R}$ *y* $\mathbb{C}$.

II.9 Definición. (Característica de un cuerpo): Se dice que la *característica* de un cuerpo F es el menor entero positivo p tal que para todo $x \in F$:

$$px = \underbrace{x + x + \ldots + x}_{p \text{ veces}} = 0$$

Un cuerpo $< F, \cdot, + >$ suele denotarse F_{p^d}, donde p es la característica del cuerpo y p^d es el número de elementos del mismo. Si tal p no existe, entonces la característica de F es 0.

II.6 Ejemplo. *Algunos ejemplos:*

1. $\mathbb{Q}$, $\mathbb{R}$ *y* $\mathbb{C}$ *tienen característica 0.*
2. $\mathbb{Z}_p$*, con p primo, tiene característica p.*

Considérese el caso en que $F_p = Z_p = \{0, 1, \ldots, p-1\}$. Puede demostrarse que F_p es un cuerpo si y sólo si p es primo. Considérese ahora un polinomio sobre el cuerpo F_p de grado d de la forma:

$$f(x) = \sum_{i=0}^{d} b_i \cdot x^i, \quad b_i \in F_p, \; b_d \neq 0.$$

II.2 Proposición. *La característica de un cuerpo es siempre un número primo.*

Demostración. Si $p = r \cdot s$ entonces $p \cdot a = r \cdot s \cdot a = r \cdot (s \cdot a)$. Sin embargo, $s \cdot a = b \neq 0$ si $a \neq 0$ y $r \cdot b \neq 0$, ya que r y s son menores que p, lo que llevaría a $p \cdot a \neq 0$, contradiciendo la definición de característica. □

II.1 Teorema. *(Teorema de la descomposición): Todo grupo abeliano con un número finito de generadores viene dado por el producto de los subgrupos cíclicos $G_1, G_2, \ldots, G_n$, donde el orden de G_i divide al orden de G_{i+1} para $i = 1, 2, \ldots, n-1$ y n es el número de elementos del sistema generador mínimo de G.*

II.1 Corolario. *Los elementos no nulos de un cuerpo finito constituyen un grupo cíclico.*

II.2 *Observación.* Es necesario distinguir entre las notaciones F_{16}, F_2^4 y F_{2^4}. La primera, F_{16}, hace referencia al conjunto de enteros comprendido entre el 0 y el 15. F_2^4 es un conjunto de vectores binarios de dimensión 4, como por ejemplo $\{(0,1,0,1),(0,1,1,1)\}$ y, finalmente, F_2^4 es un cuerpo finito, cuyo polinomio generador es de grado 4 y su cuerpo base F_2.

II.7 Ejemplo. *(Cuerpos finitos y su representación): Sea el polinomio generador $f(x) = x^4 + 2x^3 + 2x^2 + x + 1$ sobre F_3, que también es irreducible. Por tanto, cualquier elemento $\alpha(x)$de F_{3^4}es de la forma:*

$$\alpha(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0,\ a_0, a_1, a_2, a_3 \in F_3 = \{0, 1, 2\}$$

El número de elementos (polinomios) del cuerpo es $3^4 = 81$.

II.10 Definición. (Función de Euler): Dado $n \in \mathbb{Z}^+$, se llama indicador o *función de Euler* de n, $\phi(n)$, a la aplicación:

$$\begin{gathered}\mathbb{Z}^+ \xrightarrow{\phi} \mathbb{Z}^+ \\ n \mapsto \phi(n) = |\{d \in Z^+ : mcd(d, n) = 1, 1 \leq d < n\}|\end{gathered}$$

En otras palabras, la función $\phi(n)$ se define como el número de enteros positivos menores o iguales que n y coprimos con él.

II.8 Ejemplo. *Así, $\phi(9)$=6, ya que los seis números 1, 2, 4, 5, 7 y 8 son primos relativos con 9; esto es, no comparten divisores comunes.*

II.3 Proposición. *Función de Euler*

1. La función de Euler es una aplicación multiplicativa, es decir, si $mcd(m, n) = 1$, entonces se cumple que $\phi(nm) = \phi(n)\phi(m)$.
2. Si p es primo:
$$\phi(p) = p - 1,$$
$$\phi(p^k) = (p-1)p^{k-1} = p^k(1 - \tfrac{1}{p}).$$
En particular, en nuestro caso, $p = 2$, por lo que:
$$\phi(2^k) = 2^{k-1}. \tag{II.1}$$
3. Si $n = p_1^{k_1}p_2^{k_2}\ldots p_m^{k_m}$ es la descomposición en factores primos de n, se tiene que:
$$\phi(n) = \prod_{i=1}^{m} p_i^{k_i}(1 - \frac{1}{p_i}) = n\prod_{p|n}^{m}(1 - \frac{1}{p}).$$

4. Para cualquier $n \in \mathbb{Z}^+$ se cumple:

$$\sum_{d|n} \phi(d) = n.$$

II.3 *Observación.* Sabiendo que el orden de un elemento primitivo sobre $\mathbb{F}_q$, de característica p, es $p^m - 1$ y que el número de raíces conjugadas de un polinomio primitivo definido sobre dicho cuerpo es m, entonces el número de polinomios primitivos vendrá dado por la expresión $\frac{\phi(p^m-1)}{m}$.

II.1.2. Cuerpos de Galois

Los cuerpos de Galois, llamados así en honor a Évariste Galois que caracterizó este tipo de cuerpos en uno de los pocos artículos que publicó a lo largo de su vida, serán una pieza central en el desarollo de este capítulo, por lo que serán descritos a continuación con cierto detalle.

II.11 Definición. (Cuerpo de Galois): Un *cuerpo finito o de Galois* de orden q, denotado *GF(q)*, es un cuerpo de orden finito q; es decir, con un número finito de elementos.

Un cuerpo de Galois cumple las siguientes propiedades:

- El orden, o número de elementos, de un cuerpo finito es de la forma p^n, donde p es un número primo llamado *característica* del cuerpo y n es un entero positivo, denominado *grado*. Un cuerpo de Galois se denota, entonces, $GF(p^n)$, o $\mathbb{F}_{p^n}$.
- Para cada primo p y entero positivo n, exite un cuerpo finito de p^n elementos.
- Dos campos cualesquiera con el mismo número de elementos son *isomorfos*. Esto es, si se realiza una correspondencia adecuada entre la representación de los elementos en cada cuerpo, entonces las tablas que rigen las operaciones de suma y multiplicación resultan idénticas.
- Los elementos $0, 1, 2, \ldots, p-1$ de un cuerpo $GF(p^n)$ forman un subcuerpo (o una extensión) que es isomorfo a los enteros módulo p, denominado *subcuerpo primo o extensión de* $GF(p)$. Nótese que $a = b$ en $GF(p)$ es equivalente a $a \equiv b (\text{mod p})$. Sin embargo, $2 \times 2 \equiv 0 (\text{mod } 4)$ en el anillo de residuos módulo 4, por lo que 2 no tiene recíproco y, por tanto, el anillo de residuos módulo 4 es diferente al cuerpo finito de 4 elementos. Por esta razón, para evitar ambigüedades, los cuerpos finitos se denotan como $GF(p^n)$, en lugar de $GF(k)$, aunque $k = p^n$.

Los cuerpos finitos son muy importantes en diferentes áreas de las matemáticas, como la teoría de números o la geometría algebraica, y tienen un sinfín de aplicaciones prácticas que incluyen los códigos correctores de errores y, por supuesto, la criptografía en vertientes tan variadas como los LFSRs o el algoritmo de cifrado Rijndael [4].

II.9 Ejemplo. *El cuerpo finito GF(2) contiene únicamente los elementos 0 y 1, y satisface las siguientes tablas de suma y multiplicación.*

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

Potencia	Polinomio	Vector	Regular
0	0	(000)	0
x^0	1	(001)	1
x^1	x	(010)	2
x^2	x^2	(100)	4
x^3	$x+1$	(011)	3
x^4	x^2+x	(110)	6
x^5	x^2+x+1	(111)	7
x^6	x^2+1	(101)	5

Cuadro II.1: Representación de los elementos de $GF(2^3)$ en diferentes formas.

II.12 Definición. (Elemento primitivo): Un elemento de orden $q-1$ en $GF(q)$ se denomina *elemento primitivo* de $GF(q)$. Cada cuerpo $GF(q)$ contiene, al menos, un elemento primitivo. Por otro lado, cualquier elmenento $\alpha \in GF(q)$ distinto de cero puede ser representado como $(q-1)$ potencias consecutivas de un elemento primitivo.

Representación de elementos en cuerpos de Galois

Cuando $n > 1$, el cuerpo $GF(p^n)$ puede representarse como el cuerpo de clases de equivalencia de los polinomios cuyos coeficientes pertenecen a $GF(p)$. Por tanto, cualquier polinomio irreducible de grado n representa el mismo cuerpo finito, salvo isomorfismos.

II.10 Ejemplo. *Por ejemplo, para $GF(2^3)$, podemos utilizar cualquiera de los dos polinomios irreducibles x^3+x^2+1 y x^3+x+1 como módulo. Si usamos el último de ellos, los elementos de $GF(2^3)$ pueden representarse como polinomios de grado menor que 3. De esta forma, quedaría:*
$x^3 = -x-1 = x+1$
$x^4 = x(x^3) = x(x+1) = x^2+x$
$x^5 = x(x^2+x) = x^3+x^2 = x^2-x-1 = x^2+x+1$
$x^6 = x(x^2+x+1) = x^3+x^2+x = x^2-1 = x^2+1$
$x^7 = x(x^2+1) = x^3+x = -1 = 1 = x^0$

Existen, sin embargo, otras formas, todas equivalentes, de representar los elementos de un cuerpo finito. En el Cuadro II.1 pueden encontrarse estas formas resumidas. De izquierda a derecha, las columnas muestran las representaciones que utilizan las diferentes potencias del elemento generador, la representación polinomial, la representación en forma de n-tuplas binarias, que codifican los coeficientes de la representación polinomial y, por último, la equivalencia decimal de la representación binaria.

El conjunto de polinomios de la segunda columna es cerrado respecto a las operaciones de suma y multiplicación módulo x^3+x+1 y forma, por tanto, un cuerpo finito. Este cuerpo particular se denomina cuerpo extensión de grado 3 de $GF(2)$, denotado $GF(2^3)$, donde el cuerpo $GF(2)$ se llama *cuerpo base*.

Este concepto resulta esencial para este trabajo, por lo que será ampliado en la siguiente sección.

II.1.3. Extensión de cuerpos algebraicos

En Álgebra, las extensiones de cuerpo constituyen uno de los problemas fundamentales de la Teoría de Cuerpos. Informalmente, un cuerpo es un conjunto en el que están

definidas las operaciones de suma y producto y éstas funcionan de forma *adecuada*.

Cuando se construye una extensión de un cuerpo, se busca un conjunto más grande en el que estas operaciones sigan siendo correctas y, además, se puedan resolver ecuaciones polinómicas.

II.13 Definición. (Extensión de cuerpos) Un cuerpo L es *una extensión* de otro cuerpo K, denotado por $L \leq K$, si K es un subcuerpo de L.

Formalmente, un cuerpo L es una extensión de K si K es un subcuerpo de L, es decir si $< L, +, \cdot >$ es un cuerpo y $< K, +, \cdot >$ es un cuerpo con la restricción sobre K de las operaciones + y $\cdot$ en L. Si L es extensión sobre K se denota $L : K$ ó L/K.

II.11 Ejemplo. *El conjunto de los números complejos, $\mathbb{C}$, forma un cuerpo extensión de los números reales $\mathbb{R}$ y éstos, a su vez, constituyen una extensión de los números racionales $\mathbb{Q}$.*

II.2 Teorema. *(Kronecker): Sea L un cuerpo y sea $f(x)$ un polinomio no constante en $K[x]$. Entonces existe una extensión L de K y algún $\alpha \in K$ tal que $f(\alpha) = 0$.*

II.14 Definición. (Elementos algebraicos) Un elemento α de una extensión de cuerpo L de un cuerpo K es *algebraico sobre K* si $f(x) = 0$ para algún $f(x) \in K[x]$ distinto de cero. Si α no es algebraico sobre K, entonces α es *trascendente sobre K*.

II.12 Ejemplo. *Como se ha visto en el ejemplo anterior, $\mathbb{C}$ es una extensión de $\mathbb{Q}$. Como $\sqrt{2}$ es un cero de $x^2 - 2$, $\sqrt{2}$ es también un elemento algebraico sobre $\mathbb{Q}$. Además, i es un elemento algebraico sobre $\mathbb{Q}$, pues es un cero de $x^2 + 1$.*

II.1.4. Polinomios irreducibles sobre $\mathbb{F}$

Considérese la extensión de $\mathbb{R}$ sobre $\mathbb{Q}$. Sabemos que $\sqrt{2}$ es algebraico sobre $\mathbb{Q}$ y es un cero de $x^2 - 2$. Es fácil observar que $\sqrt{2}$ es también un cero de $x^3 - 2x$ y de $x^4 - 3x^2 + 2 = (x^2 - 2)(x^2 - 1)$. Como puede verse, estos polinomios son todos múltiplos de $x^2 - 2$. El siguiente teorema demuestra que esto no es casual, sino una situación general. Este teorema desempeña un papel fundamental en nuestro desarrollo posterior.

II.3 Teorema. *Sea L una extensión de un cuerpo K y sea $\alpha \in L$ donde α es algebraico sobre K. Entonces existe algún polinomio irreducible $p(x) \in K[x]$ tal que $p(\alpha) = 0$. Este polinomio irreducible $p(x)$ está determinado de manera única salvo un factor constante en K y es un polinomio de grado mínimo ≥ 1 en $K[x]$ que tiene α como un cero.*

Multiplicando con una constante adecuada en K, podemos suponer que el coeficiente de la potencia mayor de x que aparece en $p(x)$ del teorema anterior es 1. Dicho polinomio se denomina *polinomio mónico*.

II.15 Definición. Sea L una extensión de un cuerpo K y sea $\alpha \in L$ algebraico sobre K. El único polinomio mónico del teorema anterior es el polinomio irreducible de α sobre K y se denotará por $irr(\alpha, F)$. El grado de $irr(\alpha, F)$ es el grado de α sobre K y se denota por $grad(\alpha, F)$.

II.1.5. Generadores

El concepto de *generador*, y sus propiedades asociadas, representan otro de los pilares básicos sobre los que se apoya el desarrollo de la propuesta realizada en esta tesis. Los generadores son imprescindibles para la construcción de polinomios primitivos en $GF(2^n)^m$ y, éstos, a su vez, necesarios para el desarrollo de un LFSR definido sobre este cuerpo extendido.

Por estas razones se describirán a continuación sus características más importantes y dos métodos diferentes para su obtención.

II.1.5.1. Definición de generador

Tal y como se apuntó en la definición II.6 (pág. 22), el grupo multiplicativo de cualquier cuerpo finito es cíclico, por lo que debe tener un *elemento generador*. Si el elemento $\alpha = x$ genera el conjunto multiplicativo del cuerpo, se dice entonces que el polinomio generador del cuerpo es *primitivo*.

II.13 Ejemplo. *Considérese Z_{12} con generador $\alpha = 1$. Como el máximo común divisor de 3 y 12 es 3, 3=3·1 genera un subgrupo de $12/3 = 4$ elementos:*

$$< 3 >= \{0, 3, 6, 9\}$$

Por otro lado, como el mcd de 8 y 12 es 4, 8 genera un subgrupo de $12/4 = 3$ elementos:

$$< 8 >= \{0, 4, 8\}$$

Finalmente, puesto que el mcd de 12 y 5 es 1, 5 genera un subgrupo de $12/1 = 12$ elementos. Esto es, 5 es un generador de todo el grupo Z_{12}.

II.1.5.2. Obtención de generadores

Calcular el número de generadores de un cuerpo resulta muy sencillo. Según se ha visto en la definición II.10 (pág. 24), el número de generadores de, por ejemplo, el cuerpo de Galois $GF(2^8)$ es $\phi(2^8) = 2^{8-1} = 128$, donde ϕ es la función de Euler. De la misma forma, para $GF(2^{16})$ el número de generadores asciende a 32,768 y a 2,147,483,648 para $GF(2^{32})$.

Sin embargo, obtener el conjunto de *todos los elementos generadores* de un cuerpo dado ya no resulta tan sencillo, pues desgraciadamente no existe un algoritmo simple y efectivo para encontrar todos los elementos generadores de un cuerpo finito [5, 6].

Los métodos existentes son muy ineficientes, y están basados en aplicar directamente la definición de generador. Así, se escoge un elemento al azar del cuerpo correspondiente y se comprueba si cumple los criterios para ser considerado un elemento generador. De ser así, se añade el elemento al conjunto de generadores y se continúa el procedimiento. Se trata, por tanto, de un método probabilístico muy ineficiente.

A continuación se estudia este enfoque, que hemos denominado método básico. En la sección II.1.5.2.2 se presenta una optimización clara, basada en el uso de ciertas propiedades inherentes a los cuerpos finitos. Finalmente, en II.1.5.2.3 se comparan ambos métodos, sus respectivos rendimientos, y se presentan algunas conclusiones.

II.1.5.2.1. Método básico

La primera aproximación consiste, básicamente, en utilizar la propia definición de generador. Sea F_q un cuerpo finito de orden q. Se elige entonces un elemento $\alpha \in F_q$ de forma aleatoria o sistemática y se comprueba si $\alpha^i = \beta \in F_q,\ i =$

Ciclo	Número de elementos
G_1	85 (255/85=3)
G_2	51 (255/51=5)
G_3	15 (255/15=17)

Cuadro II.2: Subgrupos cíclicos de $GF(2^8)$

$1, \ldots, q-1$. O lo que es lo mismo, se comprueba si elevando un elemento candidado a generador α a cada posible exponente, se obtienen (generan) todos los elementos del cuerpo.

El pseudo-código correspondiente al algoritmo sería el mostrado en el Listado 1.

Algoritmo 1 Obtención de generadores

```
1  while (queden elementos en el cuerpo){
2  Se elige un elemento α del cuerpo al azar o de forma sistemática.
3  for i=1 to q{
4    if (α^i ∉ F_q) then
5  break;
6  }
7  print("α es un elemento generador");
8 }
```

Claramente este enfoque es el más ineficiente, pues necesitaría en el peor de los casos $q-1$ exponenciaciones modulares que resultarían computacionalmente muy costosas. En término medio serían necesarias $(q-1)/2$ operaciones de exponenciación y, por tanto, la complejidad del algoritmo sería exponencial, $O(n^2)$, siendo $n \approx q$.

Es posible, sin embargo, mejorar el algoritmo en gran medida haciendo uso del siguiente resultado, que puede derivarse del Teorema de Lagrange (véase II.7).

II.4 Teorema. *Sea G un grupo finito cíclico de orden n. Entonces para cada $d \mid n$ con $d \geq 1$ existe un único subgrupo H de orden d.*

Teniendo en cuenta este hecho, sabemos que el cuerpo F estará compuesto por una serie de subgrupos cíclicos $G_1, G_2, \ldots, G_n$, cuyos órdenes dividen al de F. Por ejemplo, para $GF(2^8)$, los subgrupos cíclicos de generadores son los mostrados en el Cuadro II.2.

A la vista de estos resultados, la optimización del algoritmo es clara: no es necesario comprobar las $q-1$ exponenciaciones, 254 en $GF(2^8)$, sino únicamente 86, puesto que si α ha generado correctamente 86 elementos, entonces obligatoriamente debe generar el resto de los mismos.

En el Cuadro II.3 se muestran, a modo de ejemplo, algunos de los 128 generadores del cuerpo $GF(2^8)$ encontrados con el método anterior.

II.1.5.2.2. Método de los factores primos A continuación se presenta una contribución de este trabajo de tesis, que consiste en un nuevo test para la verificación de que un elemento es o no generador, mucho más eficiente que el procedimiento tradicional. Seguidamente se presentan algunas definiciones y conceptos necesarios para este desarrollo.

II.16 Definición. Sea a un elemento de $GF(2^n)$. El menor número entero positivo s tal que $a^s = 1$ se denomina *orden* de a, denotado por $ord(a)$. Los elementos de orden máximo, $s = n-1$, se denominan *elementos primitivos*.

Polinomio	Representación decimal
x^4	16
$x^6 + x^2$	68
$x^6 + x^5 + x^2$	100
x^7	128
$x^7 + x^5 + x^4$	176
$x^7 + x^6$	192
$x^7 + x^6 + x^4$	208
$x^7 + x^6 + x^5 + x^4 + x^3$	248

Cuadro II.3: 8 de los 128 generadores de $GF(2^8)$

II.4 Proposición. *Si a es de orden finito entonces se tienen las siguientes propiedades:*

- *Existe un* $n > 0$ *tal que* $a^n = 1$.
- *Sea* $ord(a) = m$. *Entonces los distintos* a^i, $0 \leq i < m$ *son distintos entre sí y* $< a >= \{a^0, a^1, \ldots, a^{m-1}\}$.
- *Si* $ord(a) = n$ *y* $a^m = 1$, *entonces* $n|m$.

De la proposición anterior se desprende que: si a es un generador, entonces debe tener orden máximo y, por tanto, no se ha de verificar que $a^i = 1$ para $i < n$. Si esto fuera cierto, entonces a no sería un generador. Pero según la propiedad anterior, si $a^m = 1$, entonces $n|m$. Es decir $a^n = 1$ ó $a^m = 1$ para alguno de los divisores máximos de n. Por ejemplo, para $GF(2^8)$ estos divisores pueden obtenerse fácilmente de los factores primos de n, $d = \{255/3 = 85, 255/5 = 51, 255/17 = 15\}$.

Con estos conceptos, ya se puede enunciar la optimización del algoritmo: no resulta necesario realizar las $2^n - 1$ exponenciaciones del primer método, sino únicamente comprobar si $a^{(2^n-1)/l} \neq 1$ para todos los factores primos $l \mid (2^n - 1)$.

El algoritmo final para la obtención de generadores, con la optimización presentada, quedaría finalmente de la forma mostrada en el Listado 2.

Algoritmo 2 Obtención de generadores

```
1  Obtener la factorización del orden del cuerpo en cuestión.
2
3  while (queden elementos en el cuerpo){
4    Se elige un elemento a del cuerpo al azar o de forma sistemática.
5    foreach (factor primo l del orden del cuerpo){
6      calcular d = (2^n - 1)/l
7      if (a^d = 1) then
8    break;
9    }
10   print("a es un elemento generador");
11 }
```

Por ejemplo, para el cuerpo $GF(2^{32})$, que supone el caso más desfavorable, el número de exponenciaciones necesarias se reduce en el peor de los casos a 5. En el Cuadro II.4 puede encontrarse una comparativa entre este método y el presentado en la sección anterior. Puede encontrarse también la descomposición en factores primos del orden

	Factor de mejora
Método básico	28.66
Método de los factores primos	46.76

Cuadro II.4: Comparación de los dos métodos de obtención de generadores

de los cuerpos $GF(2^8)$, $GF(2^{16})$ y $GF(2^{32})$ y, por tanto, el número de exponenciaciones necesarias con este método.

Nótese que *en el peor de los casos* serían necesarias tan solo 5 exponenciaciones, pues se trata de un método probabilístico. La probabilidad de que un elemento $a \in GF(2^n)$ elegido al azar sea un generador resulta:

$$P(a = generador) = \frac{\text{Número de generadores}}{\text{Número de elementos del cuerpo}} = \frac{2^{k-1}}{2^k} = 0.5$$

II.1.5.2.3. Conclusiones La mejora del nuevo método respecto al procedimiento tradicional es realmente sustancial, pues la complejidad algorítmica del mismo se reduce un orden de magnitud, pasando de una complejidad exponencial $O(n^2)$ a una lineal $O(n)$, como puede observarse en el Cuadro II.4. De esta forma, el método 1 tarda 26.5 minutos[1] en obtener los 128 generadores de $GF(2^8)$. El mismo proceso tarda sólo 34 segundos utilizando el segundo método propuesto, lo que supone un factor de mejora de 46.76.

II.1.6. El cuerpo $GF(2^n)^m$

En esta sección se describirán algunas de las propiedades de los polinomios definidos sobre cuerpos finitos. Se revisará el concepto básico de *extensión de cuerpos*, presentado en II.1.3 y un tipo especial de estas extensiones, que forman los *cuerpos compuestos*.

II.1.6.1. Cuerpos compuestos

II.17 Definición. Sea $GF(2^k)$ la *extensión binaria* del cuerpo definido a su vez sobre el cuerpo primo $GF(2)$. Si la extensión no se define directamente sobre el cuerpo binario sino sobre una de sus extensiones, entonces se obtiene un *cuerpo compuesto*.

Formalmente, un cuerpo se dice compuesto si el par:

$$[\{Q(y) = y^n + \sum_{i=0}^{n-1} q_i y^i\}, \{P(x) = x^m + \sum_{i=0}^{m-1} p_i x^i\}]$$

verifica que:

- $GF(2^n)$ se construye a partir de $GF(2)$ utilizando $Q(y)$
- $GF(2^n)^m$ se construye a partir de $GF(2^n)$ utilizando $P(x)$.

El cuerpo compuesto resultante se denota por $GF(2^n)^m$ donde $GF(2^n)$ se conoce como el *cuerpo base* sobre el que se define el cuerpo compuesto.

Un cuerpo compuesto $GF(2^n)^m$ es isomorfo al cuerpo $GF(2^k)$, con $k = nm$ [7]. Esto significa que, por ejemplo, puede trabajarse indistintamente en $GF(2^4)^4$ o $GF(2^{16})$. Sin

[1] El hardware utilizado es un PC de sobremesa estándar, de características medias. Los detalles de su especificación pueden encontrarse en el Cuadro III.5, en la página 64.

embargo, a pesar de que los dos cuerpos de orden 2^{nm} son isomorfos, la complejidad algorítmica de sus operaciones básicas, como la suma y la multiplicación, son claramente diferentes y depende principalmente de la elección de n y m y de los polinomios $Q(y)$ y $P(x)$.

Por otra parte, una extensión $GF(2^n)$ del cuerpo $GF(2)$ puede interpretarse como un espacio vectorial n-dimensional sobre $GF(2)$. Por tanto, cada elemento de $GF(2^n)$ puede representarse como una combinación lineal de elementos del cuerpo base $GF(2)$.

II.18 Definición. El conjunto $\{1, \alpha, \alpha^2, ..., \alpha^{m-1}\}$, donde α es una raíz del polinomio primitivo $P(x)$ de grado m, se denomina *base canónica* o estándar.

De forma similar, asumiendo que ω es una raíz de $Q(y)$, α una de $P(x)$ y que ambos polinomios son primitivos, los elementos del cuerpo base $GF(2^n)$ pueden representarse como $\{1, \omega, \omega^2, \ldots, \omega^{2^n-1}\}$. A su vez, los elementos del cuerpo compuesto $GF(2^n)^m$ pueden representarse como $\{1, \alpha, \alpha^2, \ldots, \alpha^{2^{nm}-1}\}$.

De esta forma, un elemento $A \in GF(2^n)^m$ puede escribirse como:

$$A = \sum_{i=0}^{m-1} a_i \alpha^i$$

donde $a_0, a_1, \ldots, a_{m-1} \in GF(2^n)$. En el cuadro II.14 (página 33) puede encontrarse un ejemplo. Como los coeficientes en la representación del cuerpo compuesto no pertenecen ya al cuerpo primo base $GF(2)$, se hace necesario una nueva forma de realizar cálculos en $GF(2^n)$, que se analizará en detalle en la sección III.2.2.

II.1.6.2. Polinomios primitivos

Se define un *polinomio sobre* $GF(q)$ como un polinomio $A(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_0$, cuyos coeficientes a_i son elementos del cuerpo $GF(q)$. A su vez, un polinomio es *mónico* si su coeficiente a_m es 1.

II.19 Definición. (Polinomio irreducible) Un polinomio $A(x)$ es *irreducible* sobre $GF(q)$ si únicamente es divisible por c ó por $cA(x)$, con $c \in GF(q)$. En adelante, se denotará "a divide a b" como $a \mid b$, donde a y b pueden ser tanto números como polinomios.

II.20 Definición. (Orden de un polinomio) Sea $P(x)$ un polinomio de grado m sobre $GF(q)$ con $P(0) \neq 0$. El *orden* de $P(x)$ se define como el menor entero s para el que $P(x) \mid x^s - 1$.

II.5 Teorema. *El orden s de todo polinomio irreducible de grado m sobre $GF(q)$ cumple la condición: $s \mid (q^m - 1)$.*

Una consecuencia de este teorema es que el orden máximo de un polinomio irreducible es $s = (q^m - 1)$.

II.21 Definición. (Polinomio primitivo) Se define como *polinomio primitivo* a un polinomio mónico de grado m con orden máximo $s = (q^m - 1)$.

En cuerpos finitos, un *polinomio primitivo sobre* $GF(2^n)$ es el polinomio mínimo de algún elemento primitivo de $GF(2^n)$. Un elemento x se dice primitivo si x es un generador del cuerpo $GF(2^n)$.

Esto significa que cada elmento distinto de cero del cuerpo puede ser expresado como el elemento primitivo elevado a alguna potencia entera. Como consecuencia, un polinomio primitivo tiene grado n y es irreducible.

II.14 Ejemplo. *En este ejemplo se analizará cómo se define el cuerpo compuesto $GF(2^3)^2$). Sea $Q(y) = y^3 + y^2 + 1$ un polinomio primitivo en $GF(2)$, que se utilizará para formar el cuerpo base $GF(2^3)$. Sea a una raíz del polinomio primitivo. Se cumple por tanto que:*

$$Q(a) = a^3 + a^2 + 1 = 0$$

lo que, a su vez, implica:

$$a^3 = a^2 + 1$$

ya que la característica del cuerpo es 2; es decir, para cualquier $a \in GF(2^3)$ se cumple que $a + a = 2a = 0$. Llegados a este punto, podemos ya obtener los elementos del cuerpo utilizando la base $\{\omega^2, \omega, 1\}$:

$$\begin{aligned}
\omega^0 &\mapsto 1\\
\omega^1 &\mapsto \omega\\
\omega^2 &\mapsto \omega^2\\
\omega^3 &\mapsto \omega^2 + 1\\
\omega^4 = \omega^3\omega = (\omega^2 + 1)\omega = \omega^3 + \omega &\mapsto \omega^2 + \omega + 1\\
\omega^5 = \omega^3\omega^2 = (\omega^2 + 1)\omega^2 = \omega^4 + \omega^2 = \cancel{\omega^2} + \omega + 1 + \cancel{\omega^2} &\mapsto \omega + 1\\
\omega^6 = \omega^3\omega^3 = (\omega^2 + 1)(\omega^2 + 1) = \omega^4 + \omega^2 + \omega^2 + 1 = \omega^2 + \omega + \cancel{1} + \cancel{\omega^2} + \cancel{\omega^2} + \cancel{1} &\mapsto \omega^2 + \omega
\end{aligned}$$

Es fácil comprobar que, en efecto, $\omega^7 = 1$, ya que $\omega^7 = \omega^6\omega = (\omega^2+\omega)\omega = \omega^3+\omega^2 = \cancel{\omega^2}+1+\cancel{\omega^2} = 1$. Si los elementos se representan como $\sum_i a_i\alpha^i$, para $a_i \in \{0, 1\}$ es posible asignar un código binario a cada uno de ellos, lo que daría lugar a:

$$\begin{aligned}
\omega^0 &\mapsto 1\\
\omega^1 &\mapsto \omega\\
\omega^2 &\mapsto \omega^2\\
\omega^3 &\mapsto \omega^2 + 1\\
\omega^4 = \omega^3\omega = (\omega^2 + 1)\omega = \omega^3 + \omega &\mapsto \omega^2 + \omega + 1\\
\omega^5 = \omega^3\omega^2 = (\omega^2 + 1)\omega^2 = \omega^4 + \omega^2 = \cancel{\omega^2} + \omega + 1 + \cancel{\omega^2} &\mapsto \omega + 1\\
\omega^6 = \omega^3\omega^3 = (\omega^2 + 1)(\omega^2 + 1) = \omega^4 + \omega^2 + \omega^2 + 1 = \omega^2 + \omega + \cancel{1} + \cancel{\omega^2} + \cancel{\omega^2} + \cancel{1} &\mapsto \omega^2 + \omega
\end{aligned}$$

II.1.6.3. Búsqueda de polinomios primitivos en $GF(2^n)^m$

La técnica que se presenta a continuación para encontrar polinomios primitivos sobre $GF(2^n)^m$ se debe al trabajo de tesis de Christof Paar [8]. A su vez, su trabajo se apoya en el algoritmo original de Alanen y Knuth [9] y, particularmente, en dos resultados concretos ligeramente modificados para encajar mejor en el tipo de cuerpos finitos que se consideran en este trabajo.

Estos resultados son los siguientes:

II.6 Teorema. *(Lema 1 en [9]): Si un polinomio mónico de grado m definido sobre $GF(2^n)$ tiene orden $s = 2^{nm} - 1$, entonces es primitivo, y no es necesario comprobar si es irreducible.*

II.7 Teorema. *(Teorema en [9]): El coeficente del término independiente p_0 de un polinomio primitivo de grado m sobre $GF(2^n)$ es una raíz primitiva, es decir, un elemento de orden $2^n - 1$, en el cuerpo base $GF(2^n)$.*

Utilizando estos resultados, el algoritmo para determinar si un polinomio $P(x)$ sobre $GF(2^n)^m$ es primitivo comprueba, básicamente, el orden del polinomio x módulo $P(x)$.

Es importante destacar que se trata de un método para comprobar si un polinomio es o no primitivo, pero nunca para la búsqueda exhaustiva de todos los polinomios primitivos de un cuerpo dado. Por tanto, el algoritmo recibe como entrada un polinomio $P(x)$, y produce una salida binaria, verdadera o falsa, indicando si $P(x)$ es o no primitivo. Si tal término existiera, podría decirse que se trata de un test de "primitividad".

Definición del algoritmo El algoritmo de comprobación es esencialmente un algoritmo sencillo. Sea $k = 2^{nm} - 1$. Si $x^k \equiv 1 \bmod P(x)$, entonces $P(x)$ *podría* ser primitivo. Su orden debe dividir a k, y para que $P(x)$ sea primitivo, no debe existir un $k' < k$ tal que $x^{k'} = 1 \bmod P(x)$. Sólo es necesario comprobar los divisores máximos de k, es decir, cada uno de los factores primos p de k, $d = k/p$, porque cualquier otro orden dividiría a uno de estos factores si no fuese primitivo.

La forma de comprobar el resto de $x^k \bmod P(x)$ es tomar el polinomio x y elevarlo a la k-ésima potencia utilizando el método tradicional de Knuth [10], reduciéndolo módulo $P(x)$ en cada paso. De esta forma sólo es necesaria la representación binaria de k, ya que realmente no se realizan cálculos con su valor decimal.

Finalmente, es posible eliminar ciertos polinomios antes de iniciar el proceso del algoritmo. Como se ha comentado, el término independiente debe ser distinto de cero y un generador del cuerpo base $GF(2^n)$. Si esta comprobación es negativa, $P(x)$ no puede ser primitivo.

A continuación se muestra el algoritmo resumido:

Algoritmo 3 Algoritmo de búsqueda de polinomios primitivos en un cuerpo compuesto

1. Calcular 2^{nm-1} y sus r màximos divisores d_i, $i = 1, 2, ..., r$.
2. Comprobar si $x = 1$ es una raíz o si p_0 no es un generador en el cuerpo base $GF(2^n)$. Si es así, $P(x)$ NO es primitivo.
3. Comprobar si $x^{2^{nm-1}} \equiv 1 \bmod P(x)$. Si no lo es, $P(x)$ NO es primitivo. En caso contrario, continuar al paso 4.
4. Comprobar si $x^{d_i} \equiv 1 \bmod P(x)$, $i = 1, 2, ..., r$. Si esta condición es cierta para algún i, entonces el orden de $P(x)$ en menor de 2^{nm-1} y, por tanto, no es primitivo. Por el contrario, si ningún i satisface la condición, el polinomio $P(x)$ es finalmente primitivo.

II.2. ANTECEDENTES TECNOLÓGICOS

Una vez presentados los antecedentes de corte esencialmente teórico, relacionados en su mayoría con el cifrador en flujo que se presentará en el Capítulo III, se describen a continuación otros aspectos tecnológicos de esta tesis doctoral.

La sección comenzará con una clasificación de los tipos de redes ad-hoc existentes en base a los dos criterios más diferenciadores, como son la *movilidad* y *capacidad de proceso* de los nodos que las conforman. Esta clasificación permitirá ubicar correctamente las distintas soluciones de seguridad que se propondrán en los capítulos posteriores.

Finalmente, se realizará un breve análisis de los protocolos de enrutamiento más utilizados en este tipo de redes, dada su importancia tanto a nivel de funcionamiento como a nivel de la seguridad que finalmente proporcionan.

II.2.1. **Tipos de redes ad-hoc**

Dada la disparidad y variedad de las características definitorias de las redes ad-hoc, no resulta sencillo proporcionar una única y concisa definición que englobe todas sus posibles variantes. Por tanto, es necesario hacer depender esta definición de diversos conceptos, como la capacidad de proceso de los nodos, de su tipo de alimentación energética, de su movilidad, del ancho de bando disponible para comunicaciones o de su uso proyectado.

Con el fin de avanzar en esta clasificación, la Figura II.1 muestra una esquematización de los grandes tipo de redes ad-hoc. Atendiendo al criterio de capacidad computacional de sus nodos, en esta taxonomía encontramos en primer lugar las denomiandas redes MANet, poseedoras de una capacidad de cómputo media-alta y una movilidad media. Los nodos de una típica red MANet estan formados por ordenadores portátiles o teléfonos móviles de última generación. Este tipo de redes será analizado con más detalle en la próxima sección.

Bajando en la escala conformada por este criterio de clasificación encontraríamos las redes VANet (sección II.2.1.4), las redes de sensores (sección II.2.1.2) y, finalmente, la tecnología RFID (sección II.2.1.3).

II.2.1.1. Redes MANets

En pocas palabras, y de una forma muy global, una *red MANet* es *una red autónoma, inalámbrica, formada por nodos con cierta capacidad de movimiento* [11]. Por supuesto, puede incluir nodos cableados y estáticos (que no son más que un caso particular de los primeros).

Principales características La principal característica de una MANet, como analizaremos con mayor profundidad a continuación, es su gran *flexibilidad*, que permite establecer de forma espontánea una red en cualquier localización, sin necesidad de contar con infraestructura de red pre-existente (como routers o una instalación cableada). Por tanto, en determinados escenarios como una situación de crisis, una red MANet puede suponer la única opción viable de comunicaciones, debido a que:

- El lugar no cuenta con infraestructuras de comunicaciones o éstas han sido seriamente dañadas.
- Es imposible prever todas las necesidades de comunicación y, por tanto, es imprescindible la flexibilidad.

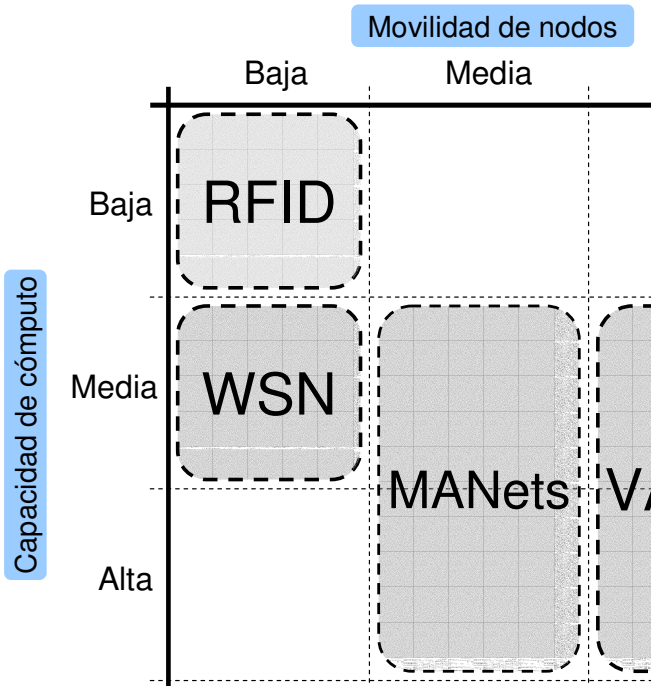


Figura II.1: Taxonomía de los diferentes tipos de redes ad hoc, clasificadas en base a la capacidad de proceso y movilidad de sus nodos

Como característica adicional, y a diferencia de las redes ad-hoc tradicionales, las redes MANet presentan lo que se denomina *comportamiento multisalto*, definido de forma implícita e inserapable del diseño de la red. El aspecto multisalto, en contraposición al tradicional enrutamiento punto a punto, otorga a la red la capacidad de utilizar a modo de repetidores los nodos intermedios entre origen y destino. Esto confiere a este tipo de redes una enorme robustez ante, por ejemplo, el fallo de uno o varios de sus nodos [12].

Como hemos analizado en el capítulo de introducción, el creciente interés en las redes MANet se sustenta en la diversidad y utilidad de las aplicaciones a las que este tipo de redes se prestan. Otros de los múltiples ejemplos posibles los constituyen los sistemas de difusión de anuncios y/o alertas en zonas usualmente frecuentadas por multitud de usuarios (como los centros comerciales, las estaciones de trenes o los aeropuertos), o las aplicaciones de intercambio de información entre varios usuarios [13].

En principio, cualquier dispositivo con cierta capacidad de cómputo y de comunicaciones puede formar parte de una MANet. Por ejemplo, una PDA, un ordenador portátil o un simple teléfono móvil. Esta cualidad confiere a este tipo de redes un abanico de posibilidades enorme, por lo que no resulta descabellado decir que la tecnología de las redes MANet puede considerarse una de las tecnologías con más futuro en los próximos años. Los usos imaginables son prácticamente infinitos, en todos los ámbitos de la tecnología, desde el personal y del hogar hasta el industrial.

Pero, como no podía ser de otra manera, no todo en las redes MANet son ventajas y características favorables. El diseño de este tipo de redes presenta una alta dificultad debido a su carácter espontáneo y dinámico. Máxime si tenemos en cuenta que

éstas se autogestionan automáticamente sin infraestructura previa y que, además, los dispositivos que la forman se mueven libremente y pueden aparecer o desaparecer en cualquier momento en la red [14]. La gestión de este tipo de redes suele basarse en la utilización de algoritmos de difusión de mensajes, que tratan de mantener la topología de la red en cada momento.

El problema se ve agravado debido a que los recursos energéticos de los dispositivos que forman este tipo de redes es limitado, puesto que suelen funcionar alimentados por baterías. Por esta razón, el consumo de este tipo de algoritmos de difusión debe ser minimizado. Un buen algoritmo debe maximizar el número de dispositivos alcanzados minimizando el uso de la red y el tiempo de procesamiento.

A modo de resumen, podríamos citar los siguientes aspectos como las características más importantes de las redes MANet:

- Permiten movilidad en sus nodos.
- Su operación es distribuida por naturaleza.
- Cada nodo puede operar como fuente y sumidero de información simultáneamente.
- No necesitan de infraestructura de comunicaciones previa.
- No disponen de nodos dedicados para las funciones básicas de la red como su administración, o el envío y enrutamiento de paquetes. Dichas funciones son llevadas a cabo por los nodos disponibles.

II.2.1.2. Redes de sensores

Las redes de sensores, *wireless sensor networks* o WSN en inglés, constituyen otro de los tipos particulares de red ad hoc, si bien comparten todas los aspectos característicos, como la total ausencia de infraestructura previa o la falta de un control central [15]. Sin embargo las diferencias son significativas en cuanto a los recursos disponibles por parte de los nodos, como la escasa capacidad de cómputo o ancho de banda, junto con importantes restricciones energéticas.

Estas redes de sensores pueden estar formadas por, potencialmente, cientos o miles de pequeños nodos sensores, comunicándose entre sí a través de canales de radio de baja potencia.

Composición de los nodos Por otro lado, un sensor suele estar compuesto por un microcontrolador, memoria, algún tipo de sensor ambiental, como sensores de temperatura, humedad o presencia, y un transmisor de radio. La potencia de este transmisor es la que define, en gran medida, el rango de distancias de comunicación que pueden alcanzar los nodos de la red. En el Cuadro II.5 pueden encontrarse las potencias habituales de los dispositivos típicos utilizados en este tipo de redes.

Las potenciales aplicaciones de las WSN son, como las de todos los tipos de redes ad-hoc móviles, prácticamente ilimitadas. Las más comunes incluyen escenarios de guerra o de conflicto, con el fin de monitorizar un área enemiga o amiga, y la monitorización de edificios o de entornos industriales. Por supuesto, la seguridad es crucial en estos escenarios, dada la importancia de la información que fluye por ellos.

Sin embargo, dados sus estrictos requisitos, proporcionar mecanismos de seguridad efectivos para redes WSN es una tarea especialmente compleja y delicada. En general, en este campo de investigación se considera que los retos globales a asumir son los siguientes:

- Naturaleza inalámbrica de la comunicación

Dispositivo	Fuente de energía	Potencia
Ordenador de sobremesa	Red eléctrica	150W-500W
Ordenador portátil	Baterías de alta capacidad	10W-120W
PDAs, teléfonos móviles	Baterías	100mW-10W
Redes de sensores	Baterías de baja capacidad	1mW-100mW
RFID	Campo electromagnético del lector	1μW-500μW

Cuadro II.5: Comparación de las fuentes de energía y potencia de operación de algunos dispositivos de redes ad hoc

- Limitación de recursos en los nodos
- Redes WSN de área extensa y alta ocupación de nodos
- Falta de una infraestructura fija
- Topología desconocida durante la implamantación de la red
- Alto riesgo de ataques físicos a los nodos

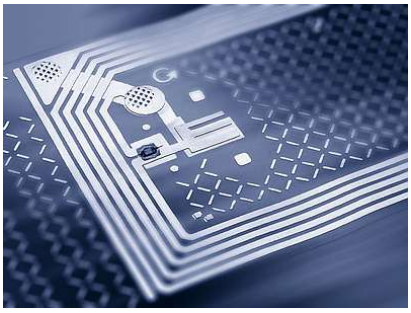
II.2.1.3. RFID

Continuando hacia abajo en la escala de capacidad de proceso, encontramos la tecnología *RFID* (**R**adio **F**requency **ID**entification). La mayoría de los nodos de una red de este tipo, denominados *etiquetas (tags) RFID*, son dispositivos de pequeño tamaño, en ocasiones incluso en forma de una simple pegatina adhesiva, que pueden ser adheridas o incorporadas a un producto, animal o persona. De hecho, cualquier visitante de un supermercado está familiarizado con esta tecnología, que ha sido utilizada durante años para evitar el hurto de productos.

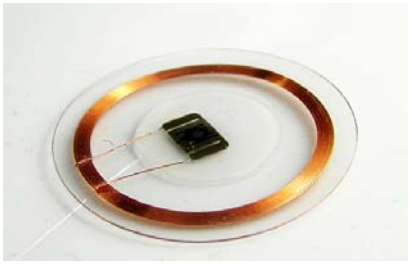
Una etiqueta RFID es un dispositivo muy sencillo consistente en un pequeño circuito integrado que contiene un número identificador, único para dicha etiqueta (a diferencia, por ejemplo, de los códigos de barras tradicionales, que identifican grupos de productos). Para completar el sistema, es necesario un *dispositivo lector*, que consulta las etiquetas situadas en sus proximidades y lee sus identificadores. Éstos se consultan posteriormente en una base de datos que contiene información adicional sobre los artículos asociados a las etiquetas.

Tipos de etiquetas RFID A grandes rasgos, existen dos grandes tipos de etiquetas RFID, que se clasifican en función de su capacidad energética y, en consecuencia, computacional:

- *Pasivas*: no disponen de alimentación propia, sino que obtienen la energía necesaria para su funcionamiento del campo electromagnético generado por el dispositivo lector. Obviamente, esto limita la capacidad de cómputo y de comunicación de dichas etiquetas. En la práctica, su misión va poco más allá de responder con un número de identificación cuando son alimentadas externamente.
- *Activas*: disponen de alimentación interna, aunque limitada. Disponen, por tanto, de cierta capacidad de proceso que les permite, incluso, implementar protocolos de autenticación tipo reto-respuesta y criptografía básica [16, 17]. En la Figura II.2 pueden encontrarse ejemplos de ambos tipos de etiquetas.



(a) Detalle del minúsculo microprocesador de la etiqueta pasiva, utilizadas habitualmente para evitar el hurto en centros comerciales



(b) Etiqueta activa, de mayor tamaño y capacidad computacional

Figura II.2: Ejemplos de etiquetas RFID pasivas y activas

Frecuencias de operación y usos Otro criterio habitual para la clasificación de los sistemas RFID consiste en el rango de frecuencias de operación que éstos utilizan. Hasta el momento, se han aceptado cuatro tipos de sistemas:

- De *baja frecuencia*: entre 125 ó 134,2 Kilohercios.
- De *alta frecuencia*: 13,56 Megahercios.
- UHF o de *frecuencia ultraelevada*: 868 a 956 Megahercios.
- *Microondas*: 2,45 Gigahercios.

Los sistemas más extendidos son, sin lugar a dudas, los dos primeros, pues los sistemas UHF, por ejemplo, no han sido regulados de forma global y común en todo el mundo, de forma que existen ciertas restricciones en su uso.

Estas frecuencias de operación determinan también en gran medida el coste, el alcance y las aplicaciones posibles. En general, los sistemas que emplean frecuencias bajas tienen asimismo costes bajos, pero también un rango de distancia de uso más limitado. Por estas razones, las etiquetas de baja frecuencia se utilizan comúnmente para la identificación de animales, seguimiento de stocks en almacenes o como llave de automóviles con sistema antirrobo.

Por otro lado, los que emplean frecuencias más altas proporcionan distancias y velocidades de lectura mayores. De esta forma permiten usos muy diversos, como el control de libros en bibliotecas, seguimiento de palés, control de acceso en edificios, seguimiento de equipaje en aerolíneas, identificación de artículos de ropa o seguimiento de pacientes en centros hospitalarios. En este sentido, un uso muy extendido de este tipo de etiquetas es la identificación de acreditaciones, substituyendo a las anteriores tarjetas de banda magnética. Sólo es necesario acercar estas insignias a un lector para autenticar al portador.

Por último, las etiquetas RFID de microondas se utilizan, por ejemplo, en el control de acceso en vehículos de gama alta.

Seguridad vs privacidad Debido a que cada etiqueta RFID está asociada de forma unívoca a un artículo concreto, surgen inmediatamente cuestiones relacionadas con la privacidad de su uso. Por ejemplo, es el caso de las prendas textiles que la persona lleva consigo y que permite desde ese momento su seguimiento.

Técnica	Descripción
Códigos 'de la muerte'	Estos códigos específicos desactivan de forma permanente la etiqueta y, por tanto, evitan su posterior lectura y seguimiento. Esta opción está especialmente indicada en aquellos artículos que requieren de una monitorización durante su proceso de fabricación y transporte, pero que no se desea que ésta siga siendo posible una vez vendidos.
Jaula de Faraday	Introducir el producto que contiene la etiqueta RFID en el interior de una jaula de Faraday[2] lo que hace imposible su lectura. Tiene la ventaja de que el usuario puede decidir cuándo la etiqueta puede ser leida o no (como en el caso de un pasaporte). Por el contrario, sufre el inconveniente de que, dependiendo del tamaño y objeto del producto, puede resultar incómodo o, incluso, imposible.
Bloqueador	El 'Bloquedador de etiquetas' [19] consiste en un dispositivo que provoca interferencias de forma artificial y deliberada, y que hacen creer a los lectores no autorizados que un gran número de etiquetas RFID están presentes. Así, el lector no autorizado no puede distinguir la etiqueta legítima de las falsas.

Cuadro II.6: Comparación de los diferentes tipos de métodos físicos para preservar la privacidad de los sistemas RFID

Para tratar de solucionar estos problemas, pueden utilizarse dos enfoques básicos. El primero consiste en utilizar medidas físicas, con el fin de desactivar o destruir la etiqueta, e impedir así su funcionamiento y los problemas de privacidad que ello acarrea. En el Cuadro II.6 puede encontrarse un resumen de los métodos propuestos hasta la fecha.

El segundo enfoque propone el uso de diversas técnicas criptográficas con el fin de conseguir que sólo las partes autorizadas puedan acceder al identificador real de la etiqueta. Un buen resumen de estas técnicas puede encontrarse en [18].

Debido a las importantes diferencias en cuanto a la capacidad computacional de cada tipo de etiqueta RFID, resulta necesario definir diversos mecanismos para proporcionar por un lado **seguridad** (es decir, la capacidad del sistema RFID para mantener la información transmitida entre etiqueta y lector secreta) y, por otro, **privacidad** (la capacidad del sistema para mantener oculto el *significado* de la información entre etiqueta y lector) que se adapten a estas características.

El futuro del RFID Las previsiones de crecimiento del mercado RFID no dejan de aumentar año tras año. De los 1970 millones de etiquetas vendidas en 2008 y de los 5250 millones de dólares de negocio en 2008, se pasó a 2350 y 5560 millones de etiquetas y negocio, respectivamente, en 2009 [20]. Sin embargo, la tendencia parece haberse acelerado en los últimos años y se espera que el mercado crezca aún a mayor ritmo en los próximos años, a pesar de la crisis ecónomica.

La tecnología RFID se convertirá en una parte importante de la futura y, ya presente en ciertos aspectos, *Internet de las cosas* y será, sin duda, cada vez más ubicua hasta estar presente en cada aspecto de la vida. Esta es una afirmación literal, pues la implantación de pequeños dispositivos RFID bajo la piel se está convirtiendo en una práctica relati-

(a) Proceso de inserción de la etiqueta RFID bajo la piel, para lo que se realiza una pequeña incisión con anestesia local

(b) Resultado final: las manos se han convertido en una tarjeta de crédito que nunca se olvida

Figura II.3: La tecnología RFID cuenta ya con usos insospechados inicialmente

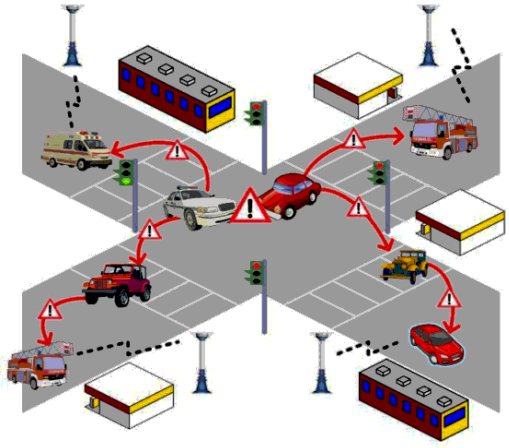


Figura II.4: Escenario de una red VANet típica

vamente habitual, sobre todo en EEUU, para la identificación de personas en sistemas de control de acceso o como medio de pago. En la Figura II.3 puede observarse cómo se realiza esta operación.

II.2.1.4. VANets

Las redes VANet (Vehicular Ad-Hoc Network) conforman otro de los grandes tipos de redes ad hoc móviles, con la particularidad de que los nodos son vehículos o el propio equipamiento de tráfico, como por ejemplo señales verticales (véase la Figura II.4).

El objetivo principal de estos sistemas es proporcionar un mejor conocimiento de las condiciones de la carretera a los conductores, con el fin de reducir el número de accidentes y hacer que la conducción sea más cómoda y fluida. Asímismo, estas redes permitirán el acceso a contenidos multimedia e Internet, como la compartición de archivos entre diferentes vehículos.

En lo relativo a la seguridad, al ser una extensión de las redes ad hoc móviles tradicionales, las VANets implican los mismos desafíos, aunque si cabe aún más complica-

dos. El principal inconveniente radica en la velocidad a la que se mueven los vehículos, que afecta a los protocolos de comunicación y las soluciones de seguridad viables que pueden aportarse.

A pesar del enorme potencial de desarrollo de las VANets, es necesario resolver una serie de problemas importantes, antes de que su despliegue sea efectivo:

- Definición de aplicaciones comerciales y de seguridad.
- Definición de los modelos de tráfico y movilidad que utilizarán los investigadores e ingenieros [21, 22].
- Definición de los mecanismos de seguridad y privacidad [23].
- Cuestiones de escalabilidad y disponibilidad
- Protocolos de encaminamiento a todos los niveles: físico, enlace y de red.

II.2.2. Protocolos de encaminamiento

Como se ha mencionado con anterioridad, una de las características esenciales de una red MANet es su comportamiento multisalto. Este comportamiento emerge a través del uso de nuevos protocolos de enrutamiento específicamente diseñados para este propósito.

El organismo internacional IETF ha definido varios de ellos, pero únicamente dos han sido ampliamente aceptados. Estos protocolos están basados en dos enfoques distintos:

- *Protocolos reactivos*: Estos protocolos establecen las rutas necesarias bajo demanda. Cuando un nodo quiere comunicarse con otro, con el que no existe una ruta establecida, se activa el protocolo de establecimiento para generar una nueva. El ejemplo más conocido de este tipo de protocolos es el denominado AODV [24].
- *Protocolos proactivos*: Por el contrario, estos protocolos utilizan el enfoque inverso; mantienen las rutas activas y "limpias" (sin bucles) constantemente, tras un paso inicial de establecimiento. El ejemplo más conocido es el OLSR [25].

Veamos ambos protocolos con más detalle a continuación.

II.2.2.1. AODV

El protocolo AODV (*Ad-Hoc On-Demand Distance Vector*), fue definido en 2003 por la IETF y se describe con detalle en [26]. La filosofía de AODV, como la de todos los protocolos reactivos, es que la información sobre la topología de la red sólo se envía cuando es necesario, es decir bajo demanda.

Cuando un nodo quiere enviar tráfico a otro con el que no existe ruta establecida, el primero genera un mensaje de petición de ruta (Route request, RREQ), que es enviado posteriormente de forma masiva a los nodos vecinos, como puede observarse en la Figura II.5.

Se considera que un ruta ha sido establecida cuando el mensaje RREQ ha llegado al destino o a un nodo intermedio con una ruta válida hasta el destino. Mientras la ruta exista entre ambos extremos, el protocolo AODV no lleva a cabo acciones de mantenimiento.

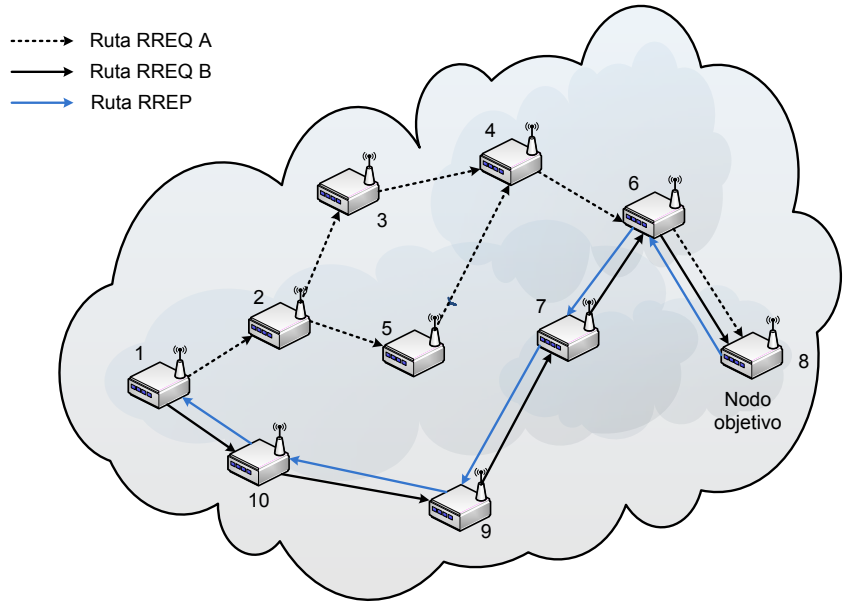


Figura II.5: Flujo típico de mensajes RREQ y RREP. De las dos rutas disponibles (línea punteada y contínua negra), el sistema elige la segunda, más corta (en línea continua azul).

II.2.2.2. OLSR

El protocolo OLSR, Optimized Link State Routing, se describe con detalle en el RFC 3626 [27] y, a diferencia del AODV, mantiene la información sobre las rutas activas constantemente actualizadas [28].

Esto permite que no haya un retardo inicial en el inicio de las transmisiones, a cambio de un flujo constante de pequeños mensajes (que, en cualquier caso, son de unos pocos bytes y completamente configurables).

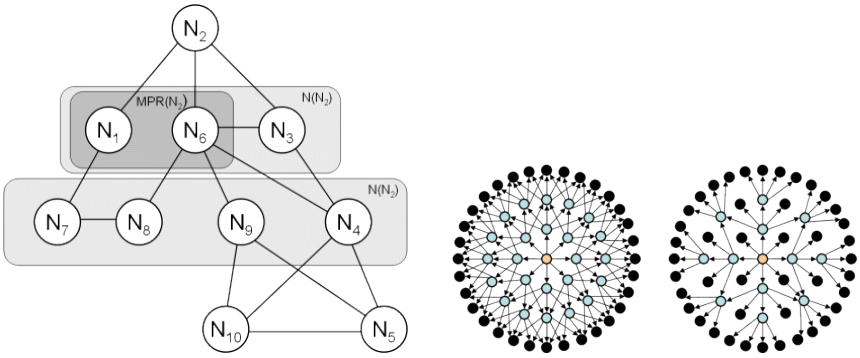
Estos mensajes tratan de mantener actualizada la información sobre los nodos vecinos, e incluyen información tal como direcciones IP o números de secuencia. Tras recibir esta información, cada nodo construye y mantiene su propia tabla de rutas, de forma que pueda calcular, con el algoritmo de rutas más cortas, el camino más eficiente a cualquier otro nodo de la red.

La información almacenada en esta tabla sólo se actualiza cuando:

- Se detecta un cambio en la vecindad del nodo.
- Expira una ruta y debe actualizarse su información.
- Se detecta un camino mejor (más corto) al mismo nodo de destino.

Otra de las características más importantes del protocolo OLSR es el uso de heurísticas específicamente diseñadas para la reducción de mensajes de difusión duplicados. Esta técnica recibe el nombre de *multi-point relays*, o MPR, y obtiene resultados muy destacables [29, 30].

La idea central de su funcionamiento consiste en elegir una serie de nodos específicos como "retransmisores" autorizados, de forma que sólo dichos nodos son los



(a) Esquema de conexiones entre nodos de la red. El conjunto MPR aparece sombreado en gris oscuro, formado por los nodos N_1 y N_6

(b) Comparación del flujo de mensajes originado por un único mensaje de difusión original. A la derecha, sólo los nodos en verde están autorizados para la retransmisión

Figura II.6: Técnica de MPR, que reduce de forma significativa el número de retransmisiones, perteneciente al protocolo OLSR.

que se encargan de retransmitir la información recibida por difusión, en lugar de todos los nodos, como se hace en los protocolos tradicionales. De esta forma, se reduce considerablemente la carga de la red y la cantidad de mensajes retransmitidos, ahorrando energía y tiempo de proceso en los nodos. En la Figura II.6 se ilustra este comportamiento.

Por diversos motivos, enumerados a continuación, los autores creen que el protocolo OLSR es superior al AODV en los escenarios planteados [31] y será, por tanto, el protocolo de elección en este trabajo de tesis:

- El retardo inicial en el establecimiento de las rutas (que puede llegar a varios segundos), puede ser inadmisible en ciertos entornos. Por ejemplo, en escenarios donde se desee proporcionar unos niveles de calidad de servicio (QoS) altos [14, 32], como en la transmisión de video y audio, que necesitan baja latencia.

  La situación empeora cuando la tasa de introducción o salida de nodos de la red se eleva, pues cada una de estas situaciones implica una nueva ejecución del algoritmo de establecimiento de rutas.

- Por otro lado, el protocolo OLSR está más maduro, desde el punto de vista de la especificación del estándar.

II.3. ESTADO DEL ARTE

Una vez establecidos los conceptos y cuestiones básicas sobre los diferentes tipos de redes ad-hoc existentes, en esta sección se analizarán brevemente los esquemas de seguridad más significativos planteados hasta la fecha, desglosándolos en cuestiones relativas a la privacidad de la información, su autenticación o su integridad.

II.3.1. Seguridad en redes MANets

Como se ha visto en las secciones anteriores, no existe un acuerdo claro en la comunidad investigadora sobre una definición general y consistente de lo que es una red ad hoc ni de sus características esenciales. Esto hace, obviamente, que tampoco sea fácil especificar los requisitos necesarios de seguridad ni diseñar soluciones.

De hecho, puede considerarse que, debibo a estas razones y al amplísimo espectro de posibles aplicaciones de esta tecnología, no es posible encontrar en la literatura de este campo un modelo de seguridad genérico, aplicable a todos los escenarios [33], a pesar de la multitud de ejemplos que reclaman lo contrario [34, 35, 36, 37].

Todos ellos presentan modelos globales de seguridad para redes ad hoc en sus títulos, pero en la práctica no consiguen sus objetivos por diversos motivos. Por ejemplo, la referencia [34] se ciñe a un tipo muy concreto de aplicación de las redes ad hoc, mientras que [35] únicamente trata el problema de los ataques de red. Por otro lado, en [36] sólo se trata, de forma incompleta, el problema de la securización del encaminamiento. Por último, la referencia [37] únicamente se encarga de proponer un método para asegurar la privacidad de las comunicaciones. Estos ejemplos, cuya calidad no se cuestiona ni se entra a valorar en absoluto, se citan aquí para ilustrar la dificultad de encontrar un modelo global de seguridad aplicable a todos los escenarios posibles.

Como se apuntó en la sección que detallaba los objetivos de esta tesis, que tiene un enfoque principalmente criptográfico, el esfuerzo principal se ha centrado en realizar aportaciones en los mecanismos de cifrado y autenticación. Para situar estos nuevos mecanismos en contexto, a continuación se analizará el estado del arte actual en estos dos ámbitos concretos.

II.3.2. Estado del arte en esquemas de cifrado

Los protocolos tradicionales utilizados para proteger las comunicaciones en Internet, como IPSec, SSL o SSH, no resultan adecuados en los entornos que estamos considerando, debido a su alta carga computacional y al incremento en el tamaño de los paquetes transmitidos que suponen.

Por estas razones, los primeros esquemas propuestos específicamente para redes ad-hoc móviles, como SNEP basado en el uso de RC5 o [38] desarrollado en el año 2002, trataban de solucionar estas limitaciones. SNEP, sin embargo, nunca fue especificado de forma detallada o implementado de forma pública. Otras propuestas algo más recientes, como TinySec [39] han corrido mejor suerte, pero siguen estando basadas en cifradores simétricos existentes.

En la actualidad, debido a la constante mejora en las capacidades de microcontroladores y microprocesadores, se considera que el hardware actual es capaz de utilizar primitivas criptográficas simétricas, asimétricas y funciones hash. En este sentido, el reciente estándar IEEE 802.15.4 [40] proporciona soporte hardware para ejecutar algoritmos como AES-128. En dispositivos que no cuenten con esta ayuda, la implementación en software de referencia de AES-128 ocupa 8kB de ROM y 300 bytes de RAM. En [41] puede encontrarse una excelente recopilación del rendimiento software de éste y otros cifradores en este tipo de plataformas.

Durante los últimos años, los nodos típicos de las redes ad-hoc móviles han sido habitualmente considerados como dispositivos demasiado restringidos para soportar criptografía, pero esta suposición ha cambiado últimamente. Haciendo uso de los esquemas de curvas elípticas (ECC), más eficientes que su contrapartida basada en el problema de la factorización, es posible obtener primitivas para cifrar datos (ECIES), firmar y verificar firmas (ECDSA) o negociar claves (ECDH).

En cualquier caso, sigue siendo innegable que sólo los nodos más caros y de alta gama disponen de estas capacidades y que, incluso en estos casos, los requerimientos computacionales y de memoria de estos algoritmos siguen siendo altos: una firma utilizando ECDSA necesita aproximadamente 2 segundos para completarse, requiriendo un total de 17kB de memoria ROM y 1.5kB de RAM [42].

Como resulta lógico, los requisitos para las funciones hash y HMAC son menores. Por ejemplo, la función SHA-1, esencial en el mecanismo de autenticación que propondremos en el Capítulo V, puede implementarse en sólo 3kB de ROM.

Desfortunadamente, tanto el estándar 802.15.4 como el soporte de curvas elípticas son soluciones 'elitistas', en el sentido de que encarecen considerablemente el coste de los nodos que las implementan. Por esta razón, no resultan adecuados en entornos donde el número de nodos es muy alto y provocaría costes prohibitivos. En conclusión, parece claro que, aún a día de hoy, sigue existiendo la necesidad de aportar primitivas criptográfivas, en este caso de cifrado, que no requieran de soporte hardware y puedan correr sin modificación en todo tipo de plataformas existentes, especialmente en aquellas de menor coste y capacidad computacional.

II.3.3. Taxonomía de los esquemas de autenticación de dispositivos

En esta sección se analizarán los modelos generales de autenticación y protocolos existentes para el establecimiento seguros de claves en redes ad hoc. Todos los modelos se resumen en la Figura II.7, agrupados por el esquema de cifrado utilizado, junto con algunos artículos de referencia que fueron los primeros en introducir el modelo.

II.3.3.1. Sin autenticación

Aunque resulte sorprendente, existen ciertos protocolos introducidos para su uso en redes ad hoc que no contemplan en absoluto opciones de autenticación o de privacidad. Por ejemplo, el proyecto Piconet [43] fue inicialmente diseñado para estudiar las redes móviles de dispositivos empotrados, haciendo énfasis en la conectividad de los mismos y dejando de lado toda medida de seguridad.

Otros sistemas introducidos para ser utilizados en *redes de uso personal* (PAN en inglés), como HomeRF [44] o IrDA, tampoco tienen mecanismos de seguridad definidos. Afortunadamente el HomeRF Working Group ha propuesto el protocolo *Wireless Access Protocol* (SWAP) como una especificación industrial abierta para la interconexión segura de dispositivos personales. Sin embargo, como resulta desgraciadamente, las primeras versiones tienen graves errores de diseño, como utilizar claves simétricas, para su uso con el algoritmo de cifrado Blowfish, de sólo 56 bits de longitud, que las hace propensan a sufrir ataques por fuerza bruta.

Se pretende que la siguiente versión de la especificación utilice claves de 128 bits, consideradas seguras a día de hoy, aunque todos los dispositivos de un usuario compartirán la misma clave, que inicialmente vendrá especificada en el proceso de fábricación, con el consiguiente riesgo de ataque si el usuario, como es habitual, no la cambia por una más segura.

Por otro lado, los protocolos IrDA están basados en el uso de radiación infrarroja, en lugar de ondas de radio. Esto proporciona mayor ancho de banda, a cambio de

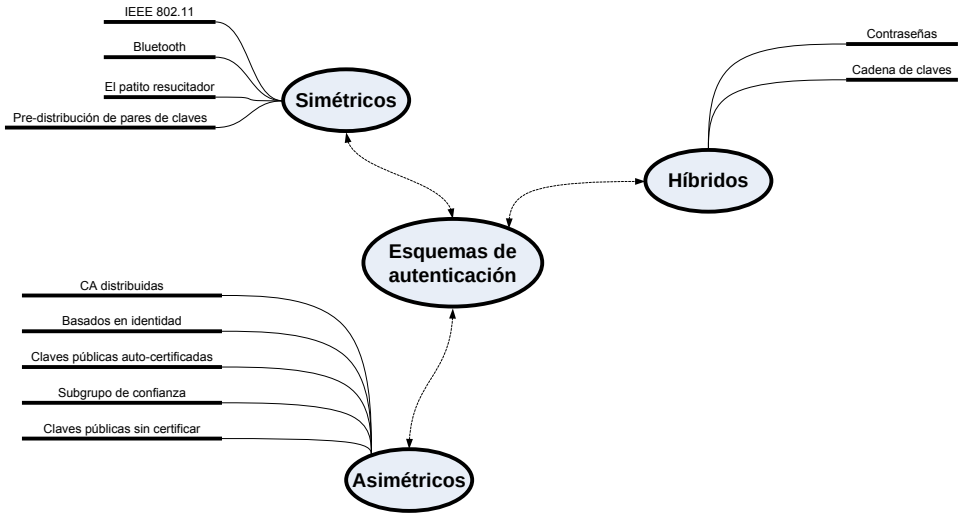


Figura II.7: Modelos de autenticación existentes para redes ad-hoc

necesitar visión directa entre los dispositivos a comunicar y una menor distancia de transmisión. Por tanto, puede decirse que la tecnología IrDA está recomendada para PANs, con el fin de conectar éstos con un ordenador. A pesar de que no se considera la implementación de ningún tipo de seguridad en estos protocolos, el extremedamente corto rango de comunicaciones y la necesidad de línea de visión directa puede proporcionar una suerte de autenticación "física".

II.3.3.2. Soluciones simétricas

En este tipo de soluciones un secreto debe ser compartido entre todos los dispositivos que participan en la comunicación. La mayoría de los esquemas actuales que utilizan esta aproximación hacen uso de un canal seguro preexistente para la transmisión de dicho secreto. A grandes rasgos, estos esquemas pueden ser clasificados en las siguientes categorías:

- *Modelo IEEE 802.11*: propuesto en 1997 para uso en redes inalámbrica de ámbito local (WLANs) [45]. Su principal desventaja es que necesita una gestión de claves "fuera de banda", pues necesita un canal seguro previo para el intercambio de claves. Evidentemente, este requisito no es admisible en todos los escenarios que se pretenden abordar en este trabajo de tesis.

- *Modelo Bluetooth*: este protocolo fue denominado como IEEE 802.15 [45] e introducido para uso en PANs. A pesar de que su algoritmo de cifrado, E_0, no ha sufrido un criptoanálisis definitivo (aunque sí algunos ataques de alta complejidad algorítmica, como [46]), su seguridad se considera insuficiente [47]. Otro gran inconveniente del que adolece Bluetooth es que no escala adecuadamente, debido a la necesidad de que el usuario teclee manualmente el secreto en cada dispositivo participante en la comunicación.

- *Modelo "El patito resucitador"*: detrás del cómico nombre propuesto por Stajano y Anderson, se esconde un modelo serio [48], que resuelve de forma imaginativa el problema del intercambio de claves. Los autores proponen la necesidad de

un contacto físico entre dispositivos, de forma que, una vez conectados, éstos no respondan a nuevas peticiones que no provengan del dispositivo con el que se vincularon inicialmente [3]. Sin embargo, este requisito es inviable en ciertos escenarios, como una simple red de sensores.

- *Modelo de pre-distribución de claves*: el último gran modelo, y quizás el más importante de los presentados, es el que propone diferentes formas de realizar una distribución previa de las claves simétricas, con el fin de no utilizar una única clave en toda la red. Este hecho es especialmente importante en las redes de sensores, que disponen de una seguridad física débil, y cuyo material de claves puede ser fácilmente recuperado.

  La dos grandes propuestas en este área son las de Gligor, que propone un protocolo de pre-distribución probabilístico [49], y la de Liu y Ning, que utilizan la idea de que una red de sensores suele ser estática, de forma que sus nodos no se mueven en absoluto o lo hacen muy lentamente [50]. Así, proponen un esquema que se basa en la localización de dichos nodos y utiliza la misma para la generación de las claves de autenticación y cifrado. Esta idea ha seguido siendo revisada, con algunas propuestas recientes interesantes [51].

II.3.3.3. Soluciones asimétricas

En contraposición a la categoría anterior, las soluciones asimétricas están basadas en el uso de esquemas asimétricos o de *clave pública*, que se utilizan para la autenticación de las partes y el establecimiento de claves de sesión entre ellas. Pueden distinguirse cuatro grandes grupos en este tipo de esquemas, que dependen del uso de una CA y certificados:

- Con la presencia de una CA y el uso de certificados.
- Con la presencia de una CA y sin el uso de certificados.
- Sin la presencia de una CA, pero con el uso de certificados.
- Sin el uso de CA ni certificados.

A continuación describiremos propuestas presentes en la literatura que encajan en alguna de las categorias anteriores:

- *Modelo de CA distribuida*: la idea central de este modelo, propuesto por Zhou y Hass [36], es repartir la responsabilidad de una CA central entre los nodos de la red. De esta forma se pretende evitar la grave vulnerabilidad que supone que la CA esté representada por un único nodo, debido a su limita protección física antimanipulación y relativa facilidad de compromiso.
- *Modelo basado en identidad*: utiliza la idea de *criptografía basada en identidad*, introducida por Shamir en 1984 [52]. El concepto esencial de este tipo de criptografía es utilizar identidades que puedan ser tratadas por un humano, como nombres o direcciones de correo electrónico, como claves públicas. De esta forma, las identidades son auto-certificadas; por ejemplo, la clave pública y certificado de Alicia

[3] De este hecho proviene el nombre del modelo: existe la creencia, no confirmada científicamente, de que ciertos animales consideran como un progenitor al primer ser vivo, incluso personas, que ven tras su nacimiento. Este vínculo se mantiene para el resto de su vida, al estilo de la propuesto realizada en este modelo

puede ser P_A=*alicia@uc33m.es*. Khalili propone en [53] un protocolo para autenticación y gestión de claves en redes ad-hoc que utiliza el concepto de criptografía basada en identidad recién expuesto.

- *Modelo auto-organizado*: este modelo explota la propiedad de auto-organización que resulta característica y única de las redes ad-hoc. Así, los nodos de la red distribuyen y auto-certifican sus propios certificados. El modelo, introducido por Hubaux y Buttán [54], asume la existencia de ciertas relaciones de confianza entre algunos nodos, al estilo de la jerarquía PGP.

CAPÍTULO III

REGISTROS DE DESPLAZAMIENTO DEFINIDOS SOBRE CUERPOS COMPUESTOS

III.1. INTRODUCCIÓN

En el capítulo anterior se presentaron y analizaron las restricciones inherentes al escenario contemplado en este trabajo, sobre todo en lo relativo a capacidad de proceso y energía disponible en los nodos de las redes consideradas. Estas restricciones provocan que el diseño de nuevos algoritmos para el cifrado de la información se convierta en una tarea compleja y polifacética.

Estas limitaciones afectan a muchos aspectos del diseño de un nuevo cifrador pero, sin duda, lo hacen en mayor medida sobre el tipo de cifrador a diseñar: cifrado en flujo o en bloque. En principio esta elección parece tener un candidato claro: los *algoritmos en flujo*, dada su mayor simplicidad y menor necesidad de recursos frente a los cifradores en bloque [1].

Por otro lado y debido a su extrema sencillez, se ha decidido utilizar algunos elementos y principios de funcionamiento, como registros de desplazamiento y una función de decimación, de un conocido generador de secuencia cifrante conocido como *generador shrinking*, diseñado por Coppersmith, Krawczyk y Mansour [2] en 1994.

Este generador hace uso de los denominados *registros de desplazamiento*, que constituyen uno de los pilares sobre los que se asienta este trabajo de tesis. Estas estructuras son muy conocidas y utilizadas en muchísimos ámbitos, resultando básicas en criptografía. Sus principales características y el porqué de su elección para formar parte del nuevo cifrador se analizan a continuación.

III.1.1. Registros de desplazamiento

Tradicionalmente este tipo de estructuras ha sido utilizado en numerosas aplicaciones, debido a las buenas propiedades estadísticas de las secuencias por ellos generadas [3]. Sus usos incluyen entre otros las *comunicaciones*, como los sistemas de TV digital y la señal de GPS, o la *criptografía*, como base para desarrollar cifradores en flujo (por ejemplo los algoritmos A5, utilizados en telefonía móvil).

Habitualmente estas estructuras funcionan sobre el cuerpo binario $GF(2)$, que proporciona un bit de salida en cada pulso de reloj. De hecho todo el estudio matemático, incluyendo las tablas habitualmente utilizadas para el diseño de registros de desplazamiento, se encuentran definidas sobre este cuerpo [4].

Como veremos en este trabajo, esta aproximación puede ser adecuada en las implementaciones hardware pero no en las software. Dado que el tamaño de los registros en los ordenadores actuales es como mínimo de 8 bits en las máquinas menos potentes y de hasta 64 bits en las de mayor potencia, realizar implementaciones orientadas al bit es claramente ineficiente.

El objetivo es, por tanto, *utilizar cuerpos finitos más apropiados para la arquitectura de los actuales microprocesadores*. En este sentido, las elecciones naturales son los cuerpos de Galois extendidos, con 2^n elementos, donde n está relacionado con el tamaño de los registros del propio microprocesador, que normalmente serán bytes o palabras de 16 ó 32 bits.

De esta forma, los elementos del cuerpo *encajan* perfectamente en una unidad de almacenamiento, de forma que puede ser manipulada de forma eficiente en software. Además, la tasa de generación de secuencia de salida también se ve incrementada: hasta 8, 16 ó 32 bits por cada pulso de reloj.

Desafortunadamente, las operaciones de multiplicación y división sobre cuerpos de Galois extendidos son computacionalmente caras [5], especialmente cuando se comparan con la suma binaria en $GF(2)$, que se implementa fácilmente con una simple operación XOR. Por tanto para acelerar en lo posible estas operaciones la mayoría de las implementaciones software que trabajan en el cuerpo $GF(2^8)$ utilizan tablas de búsqueda precomputadas [6, 7].

Por otro lado, el cuerpo de estructura adecuada y tamaño inmediatamente superior a $GF(2^8)$ es $GF(2^{16})$. Sin embargo, utilizar dicho cuerpo tiene un impacto muy significativo en las operaciones aritméticas como, por ejemplo, la multiplicación. En este caso, una tabla de búsqueda completa para la operación multiplicación ocuparía 8 GB, muy por encima de la capacidad de memoria de la mayoría de sistemas. En entornos con memoria limitada, como redes de sensores o MANets, el problema se agrava aún más.

En este capítulo se propone el uso de técnicas que realizan las operaciones de multiplicación y división en un cuerpo extensión de $GF(2^n)$. Así, los elementos del cuerpo $GF(2^k)$ se representan en términos de elementos de un subcuerpo $GF(2^n)$, donde $k = n \cdot m$. El cuerpo compuesto $GF((2^n)^m)$ puede entenderse como una representación de una extensión de grado m del cuerpo $GF(2^n)$, denominado *cuerpo base*.

Por otro lado, el segundo gran bloque temático de este capítulo consiste en un estudio exhaustivo sobre ciertas técnicas de implementación eficiente en software y, sobre todo, cómo se adaptan éstas a los LFSRs.

Con este estudio se pretende extraer diversas conclusiones y responder a ciertas preguntas, como qué técnica es más adecuada para su uso en LFSRs, qué grado máximo de mejora cabe esperar de cada una de ellas y cómo les afectan, por ejemplo, la capacidad de la plataforma hardware sobre la que se ejecutan.

Estructura del capítulo En la Figura III.1 pueden encontrarse esquematizados los dos grandes enfoques propuestos a la hora de mejorar el rendimiento de los LFSRs, que conforman el presente capítulo. El resto del mismo se organiza de la siguiente forma.

En la sección III.2 se introducen y describen brevemente los registros de desplazamiento extendidos, así como las variaciones en su funcionamiento interno respecto a los registros tradicionales. Estas diferencias radican esencialmente en la función de realimentación, que ahora implica multiplicaciones sobre cuerpos extendidos. Por esta razón, en la sección III.4 se introducen los diversos métodos de realizar multiplicaciones sobre estos cuerpos, analizándose los resultados obtenidos en el apartado III.4.1.

A continuación, en la sección IV.4.1.1, se describen diversas técnicas para la implementación eficiente de estos registros en software. Una detallada comparativa entre los diversos métodos y el análisis de los resultados obtenidos puede encontrarse en el apartado IV.4.3. Finalmente, en la sección III.7 se lleva a cabo un exhaustivo análisis estadístico de las secuencias de salida generadas por este tipo de registro, que asegura las buenas propiedades criptográficas del mismo.

El capítulo se completa con un análisis de todos los resultados obtenidos y unas conclusiones finales en la sección IV.8.

Figura III.1: Esquema de las mejoras a la implementación de LFSRs presentadas en este capítulo

III.2. REGISTROS DE DESPLAZAMIENTO EN $GF(2^n)^m$

Tradicionalmente los LFSRs, componentes básicos de muchos cifradores en flujo, se han definido sobre cuerpos de Galois de dimensión 2, es decir $GF(2)$.

Sin embargo, nada impide a estos registros trabajar en cuerpos de mayor tamaño, aprovechando la estructura subyacente del procesador en el que se ejecutan y resultando, de esta forma, mucho más eficientes en sus implementaciones software. Esta es precisamente y a grandes rasgos la propuesta hecha en este capítulo: *definir los LFSR sobre cuerpos compuestos de Galois, $GF(2^n)^m$, en lugar de los tradicionales $GF(2^n)$*.

La validez de toda esta propuesta se fundamenta en el hecho de que un LFSR definido sobre $GF(2^n)^m$ es matemáticamente equivalente a n registros de desplazamiento paralelos sobre $GF(2)$ de longitud $n \cdot m$, cada uno de ellos con la misma relación de recurrencia lineal pero diferentes estados iniciales [8]. Y lo que resulta más importante, *las propiedades fundamentales de las secuencias generadas por los LFSRs se preservan*, tal y como se demostrará en la sección III.7, de forma que siguen siendo aptos para su uso criptográfico.

Otro aspecto a destacar es que el estudio teórico exhaustivo realizado sobre los registros definidos sobre $GF(2)$ sigue siendo válido y puede aplicarse directamente a nuestro caso [3]. Los cuerpos algebraicos compuestos forman por tanto una pieza esencial en la definición de los nuevos registros de desplazamiento, que denominaremos en lo sucesivo *registros extendidos*.

III.2.1. Registros de desplazamiento extendidos

Los registros de desplazamiento extendidos se definen prácticamente de igual manera que los tradicionales. A continuación se exponen algunos breves conceptos básicos al respecto.

III.1 Definición. Un *registro de desplazamiento lineal extendido* (LFSR) de longitud l es una máquina de estados finita que opera sobre un cuerpo finito $\mathbb{F}_q$, de característica p, donde $q = p^e$ para algún $e \geq 1$. En nuestro caso, $q = 2^n$ y, por tanto, $\mathbb{F}_q = GF(2^n)$.

El LFSR cuenta con l celdas de memoria $r_0, r_1, \ldots, r_{l-1}$, cada una de ellas conteniendo elementos de $GF(2^n)$. En un instante de tiempo cualquiera t el contenido del registro conforma su *estado*, denotado $S_t = (s_{t+l-1}, s_{t+l-2}, \ldots, s_t)$. El estado en el instante de tiempo cero, S_0, se denomina *estado inicial* del LFSR.

En cada unidad de tiempo, o pulso de reloj en las implementaciones hardware, el registro actualiza su estado. Así, el estado S_{t+1} se deriva del estado S_t de la siguiente forma:

1. El contenido de la celda r_0 se convierte en la salida de la unidad de tiempo t y entra a formar parte de la *secuencia cifrante*.
2. El contenido de la celda r_i se mueve a r_{i-1} para cada i, $1 \leq i \leq l-1$.
3. El nuevo contenido de la celda r_{l-1} es el denominado *elemento de realimentación*, que se calcula de acuerdo a la siguiente *función de realimentación*:

$$s_{t+l} = c_l \sum_{i=0}^{l-1} c_i s_{t+i},\ c_i \in GF(2^n),$$

donde las constantes $c_0, c_1, ..., c_l$ se denominan *coeficientes de realimentación* y la conexión en c_l se llama *posición de realimentación*.

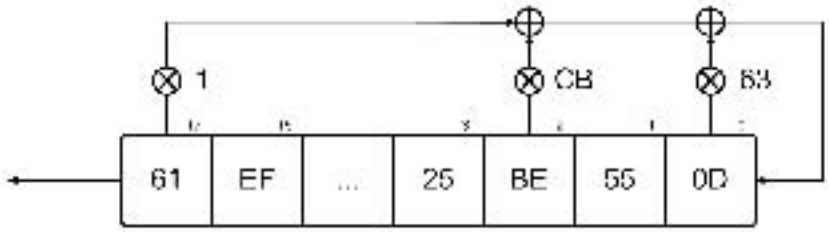


Figura III.2: LFSR extendido, definido sobre $GF(2^8)$. Su función de realimentación es $s_{t+18} = 1 \otimes s_{t+17} \oplus CB \otimes s_{t+2} \oplus 63 \otimes s_t$.

Estas constantes forman el polinomio generador o *polinomio característico* del LFSR:

$$p(x) = c_0 + c_1 x + c_2 x^2 + ... + c_{l-1} x^{l-1} + x^l = \sum_{i=0}^{l} c_i x^i, \quad c_i \in GF(2^n).$$

Como es sabido, si este polinomio es primitivo, entonces el periodo T de la secuencia generada por el LFSR es máximo de valor $T = 2^l - 1$. En la Figura III.2 puede encontrarse un ejemplo de un LFSR extendido de 17 etapas definido sobre $GF(2^8)$.

Como hemos mencionado, en esta nueva arquitectura los contenidos de las celdas, s_t, y los coeficientes de realimentación no son ya bits, o elementos de $GF(2)$, sino elementos de $GF(2^n)$. Este hecho afectará sobre todo a las operaciones de suma y multiplicación que deben realizarse sobre ellos, tal y como se describe en la próxima sección.

III.2.2. **Aritmética en** $GF(2^n)$

Tal y como se analizó en la sección II.1.2 (pág. 26 del Capítulo II), los elementos del cuerpo $GF(2^n)$ se representan habitualmente como polinomios de grado menor que n con coeficientes en $GF(2)$. De esta forma, por ejemplo, el elemento a de $GF(2^4)$, notado $a = 0111$, puede representarse por el polinomio $a(x) = x^2 + x + 1$.

Con la representación polinomial las operaciones aritméticas se llevan a cabo haciendo uso de un polinomio irreducible R (no factorizable en factores no triviales) de grado n. Este polinomio se define sobre un cuerpo binario de la siguiente forma:

$$R(x) = x^n + r_{n-1} x^{n-1} + ... + r_1 + 1, \quad r_i \in GF(2)$$

En este escenario, la operación suma sigue realizándose de la misma forma que en $GF(2)$, es decir con una simple operación XOR o suma módulo 2. La multiplicación, por contra, exige que el resultado sea reducido módulo R. En las siguientes secciones se analizan estas operaciones básicas con mayor profundidad.

III.2.2.1. Operación suma

La operación suma en este cuerpo sigue siendo simplemente la suma módulo dos, u operación XOR, para cada par de elementos. Con la representación polinomial, el proceso es tan sencillo como sumar los polinomios en la forma habitual y reducir sus coeficientes según la característica del cuerpo $GF(2)$.

n	R	Representación hexadecimal
8	$x^8 + x^6 + x^3 + x^2 + 1$	0x14D
16	$x^{16} + x^{14} + x^{12} + x^7 + x^6 + x^4 + x^2 + x + 1$	0x150D7
32	$x^{32} + (x^{24} + x^{16} + x^8 + 1)(x^6 + x^5 + x^2 + 1)$	0x165656565

Cuadro III.1: Polinomios de reducción R utilizados en esta tesis

III.1 Ejemplo. *Un sencillo ejemplo permite ilustrar el proceso. La suma de dos elementos de $GF(2^8)$, representados como sendos polinomios p_1 y p_2 (de grado menor que 8), quedaría de la siguiente forma:*

$$\begin{aligned} p_1 + p_2 &= (x^6 + x^3 + x^2 + 1) + (x^7 + x^6 + x^5 + x^2 + 1) = \\ &= x^7 + \cancel{2x^6} + x^5 + x^3 + \cancel{2x^2} + \cancel{2} = \\ &= x^7 + x^5 + x^3. \end{aligned}$$

III.2.2.2. Operación de multiplicación

La operación de multiplicación resulta conceptualmente igual de sencilla, pero su complejidad computacional es sensiblemente superior. En esencia, consiste en llevar a cabo la multiplicación habitual entre dos polinomios, pero reduciendo el polinomio resultante módulo R. Así, para multiplicar dos elementos $p_1(x), p_2(x) \in GF(2^n)$, se calcula $p_1(x) \cdot p_2(x)$ en la forma habitual, reduciendo el resultado módulo un polinomio irreducible R de grado n definido en $GF(2)$.

La elección de R altera evidentemente la forma en la que los elementos del cuerpo son representados, pero no afecta a las operaciones matemáticas subyacentes. Los polinomios R elegidos en este trabajo para diferentes valores de n son los especificados en el Cuadro III.1.

III.2 Ejemplo. *Se muestra a continuación un ejemplo de multiplicación en $GF(2^8)$. Para ello se hace uso de los polinomios p_1 y p_2 del ejemplo anterior y uno de los polinomios de reducción, $R = x^8 + x^6 + x^3 + x^2 + 1$, listados en el Cuadro III.1.*

$$\begin{aligned} p_1 \cdot p_2 &= (x^6 + x^3 + x^2 + 1) \cdot (x^7 + x^6 + x^5 + x^2 + 1) = \\ &= x^{13} + x^{12} + x^{11} + \cancel{x^8} + \cancel{x^6} + x^{10} + \cancel{x^9} + \cancel{x^8} + \\ &\quad + \cancel{x^8} + x^3 + \cancel{x^9} + x^8 + \cancel{x^7} + x^4 + \cancel{x^2} + \cancel{x^7} + \cancel{x^6} + \cancel{x^5} + \cancel{x^2} + 1 = \\ &= x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^4 + x^3 + 1 \end{aligned}$$

En esto punto, es necesario reducir el resultado módulo R, de forma que $x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^4+x^3+1$ módulo $x^8+x^6+x^3+x^2+1$ = 11110100011001 módulo 101001101 = 1101001 = $x^6+x^5+x^3+1$. Para que se comprenda con facilidad el resultado, se detallará el proceso de reducción modular a través de la división paso a paso:

$$\begin{array}{rl} 11110100011001 & \\ 101001101 & \oplus \\ \hline 1010010111001 & \\ 101001101 & \oplus \\ \hline 1101001 & \end{array}$$

Aunque es posible llevar a cabo las operaciones de multiplicación en estos cuerpos utilizando el método anterior (multiplicación de polinomios y reducción módulo un polinomio irreducible), éste resulta muy poco eficiente.

Por esta razón, estos cálculos se realizan normalmente haciendo uso de unas *tablas pre-computadas* de resultados. Para cuerpos de Galois pequeños, de dimensión menor que o igual a 8, es posible calcular de antemano todos los posibles productos entre diferentes elementos del cuerpo y almacenar los resultados en una tabla. Claramente, este método puede implicar grandes cantidades de memoria en función del cuerpo subyacente utilizado. Éste y otros métodos de multiplicación eficiente serán analizados en detalle en la sección III.4.

Por otro lado, es importante destacar que la multiplicación por un coeficiente '0' implica simplemente ignorar el correspondiente elemento del registro, mientras que la multiplicación por 1 no supone ningún cálculo, pues la salida de la operación es la propia entrada.

Por este motivo, se procurará que los coeficientes de realimentación sean '0' ó '1' siempre que sea posible, pues esto permite una implementación mucho más eficiente. Cuando sea necesario el uso de otros coeficientes c_i no binarios, entonces la multiplicación puede implementarse de forma diferente según el cuerpo $GF(2^n)$ sobre el que estemos trabajando.

III.3. CONSIDERACIONES DE DISEÑO

Analizadas brevemente las operaciones aritméticas básicas en $GF(2^n)$, en este apartado se realizarán algunas consideraciones importantes que deben tenerse en cuenta en el diseño de un LFSR extendido, como aspectos relacionados con su *rendimiento* o su *seguridad*.

III.3.1. Rendimiento

El aspecto más influyente en el rendimiento final del registro es, sin duda, la elección de la dimensión del cuerpo $GF(2^n)$, con n=8, 16 ó 32, sobre el que funcionará el LFSR.

Al incrementar el valor de n, el rendimiento global del LFSR aumentará, pues se incrementará proporcionalmente el número de bits generados por el registro en cada salida. Sin embargo, también aumenta el coste computacional de las operaciones implicadas, especialmente el de la multiplicación, para la obtención de dicha salida.

Otro factor muy importante a tener en cuenta está relacionado con la posición y número de realimentaciones, que influirán directamente en el rendimiento y, de forma secundaria, en la seguridad.

En primer lugar, para conseguir un LFSR de periodo máximo sobre $GF(2^n)^m$, las posiciones de realimentación deberían ser primas entre sí, es decir no compartir divisores comunes. Por otra parte, el número de realimentaciones debería ser pequeño para minimizar el número de operaciones implicadas y maximizar así el rendimiento. Este problema se acentúa en la arquitectura propuesta, donde las multiplicaciones son computacionalmente más costosas.

Sin embargo, un número demasiado reducido de realimentaciones también puede llevar a un fácil criptoanálisis de las secuencias producidas por el LFSR, cuando éste se utilice como pieza básica de un cifrador. Todos estos aspectos se analizará a continuación.

III.3.2. Seguridad

Unos de los objetivos de este trabajo es generar LFSRs que sean adecuados para su uso criptográfico, aunque es bien sabido que un LFSR es una estructura lineal y, por tanto, no puede utilizarse aisladamente como generador de secuencia cifrante. Para ello es necesario introducir un comportamiento no lineal en su funcionamiento, para lo que existen diversas estrategias [8, 9] que quedan fuera del ámbito de este trabajo.

Sin embargo sí existen conceptos importantes que deben ser tenidos en cuenta en su diseño, como el de que las posiciones de las realimentaciones deben formar un *conjunto de diferencias completamente positivas (Full Positive Diference Set)*.

Un FPDS es una elección, para las diferencias entre las posiciones de realimentación de un registro de desplazamiento, que asegura que no existen parejas de posiciones que se utilicen para actualizar más de un bit en cada iteración.

Se ha demostrado que los LFSR que cumplen esta condición son mucho más resistentes a diversos tipos de ataques, incluyendo ataques por correlación condicional y ataques por inversión [10]. También se ha demostrado una fortaleza superior contra ataques del tipo *guess-and-determine* como se muestra en [11].

III.3.2.1. FPDS

III.2 Definición. (FDPS) Un conjunto G forma un *conjunto de diferencias completamente positivas* si todas las diferencias positivas entre los elementos de G son diferentes.

n	m	Longitud de clave efectiva (bits)	Periodo del LFSR resultante
8	17	136	$2^{136} - 1$
16	9	144	$2^{144} - 1$
32	5	160	$2^{160} - 1$

Cuadro III.2: Valores de los parámetros n y m adecuados para el uso de LFSRs extendidos en aplicaciones criptográficas

Aplicado a los LFSRs, esto significa que las posiciones de las realimentaciones deben estar situadas de forman que constituyan un FPDS.

III.3 Ejemplo. *Sean dos conjuntos $G_0 = \{0, 2, 4, 17\}$ y $G_1 = \{0, 2, 15\}$. El FPDS del primer conjunto es $\Gamma_0 = \{\underline{2}, 4, \underline{2}, 17, 15, 13\}$. Como puede observarse, la diferencia 2 se repite una vez, lo que abriría la puerta a ciertos ataques criptográficos. Sin embargo, todas las diferencias de $\Gamma_1 = \{2, 15, 13\}$ son distintas entre sí, por lo que este LFSR resultará más seguro que el primero.*

III.3.2.2. Longitud del registro

Por último, queda elegir una longitud adecuada del registro, acorde con el propósito para el que vaya a ser utilizado. Para su uso en criptografía, la longitud de clave recomendada actualmente es como mínimo de 128 bits, para así evitar ataques por fuerza bruta.

En principio, existen claras restricciones para el valor de n, la dimensión del cuerpo base, dado que queremos que coincida con el tamaño de palabra de los microprocesadores. Las posibilidades de elección para m, dimensión de la extensión del cuerpo base, son pues limitadas.

Si consideráramos, por ejemplo, una plataforma de redes de sensores, donde los dispositivos son microcontroladores de 8 bits, n quedaría fijado a $n = 8$ y, en ese caso, un valor adecuado para m sería 17. De esta forma obtendríamos una longitud del registro efectiva de 136 bits, lo que permitiría alojar claves simétricas de 128 bits.

En otros cuerpos, como $GF(2^{16})$ y $GF(2^{32})$, con $n = 16$ y $n = 32$ respectivamente, el valor de m podría reducirse proporcionalmente. El Cuadro III.2 muestra unas indicaciones sobre los distintos valores de n y m para uso criptográfico.

III.3.3. **Funciones de realimentación**

Haciendo uso de los criterios expuestos en las secciones anteriores y con el fin de que sirvieran como referencia en el estudio, experimentación y posterior análisis del diseño de los LFSRs extendidos, se ha elegido un conjunto de polinomios primitivos, P_8, P_{16} y P_{32}, definidos sobre los cuerpos $GF(2^8)$, $GF(2^{16})$ y $GF(2^{32})$ respectivamente.

Las funciones de realimentación correspondientes a dichos polinomios primitivos son de la forma:

$$s_{t+17} = s_{t+15} \oplus \alpha s_{t+2} \oplus \beta s_t,$$

donde $\oplus$ denota suma sobre el cuerpo $GF(2^n)$, tal y como ha sido definida en la sección III.2.2.1, la multiplicación se realiza también sobre $GF(2^n)$, con $\alpha, \beta \in GF(2^n)$

Polinomio	Función de realimentación
P_8	$s_{t+17} = s_{t+15} \oplus C6 \otimes s_{t+2} \oplus 67 \otimes s_t$
P_{16}	$s_{t+17} = s_{t+15} \oplus 19B7 \otimes s_{t+2} \oplus 13C \otimes s_t$
P_{32}	$s_{t+17} = s_{t+15} \oplus F21DA317 \otimes s_{t+2} \oplus E28C895D \otimes s_t$

Cuadro III.3: Funciones de realimentación utilizadas en este trabajo

y $\alpha, \beta \neq 0$. Los valores concretos de α y β, así como las funciones de realimentación escogidas se muestran en el Cuadro III.3.

Finalmente, el polinomio característico del LFSR que se utilizará como base de estudio es:

$$p(x) = x^{17} \oplus x^{15} \oplus \alpha x^2 + \beta.$$

III.4. IMPLEMENTACIÓN EFICIENTE DE LA MULTIPLICACIÓN EN $GF(2^n)$

Como hemos visto, al ampliar el cuerpo algebraico sobre el que se apoya el funcionamiento de los nuevos registros de desplazamiento, sus operaciones internas también lo hacen. La más importante de todas es la *función de realimentación* que implica diversas sumas y multiplicaciones.

En el cuerpo tradicional $GF(2)$ estas operaciones resultan extremadamente sencillas, pero en el cuerpo $GF(2^n)$ las multiplicaciones implicadas en la realimentación se convierten en uno de los puntos críticos del esquema propuesto. A diferencia del tradicional cuerpo $GF(2)$, las operaciones de multiplicación en $GF(2^n)$ son muchos más costosas computacionalmente, por lo que podría darse el caso, en último extremo, de que los beneficios obtenidos por su uso no compensaran su mayor coste computacional.

En los siguientes apartados se estudiará en detalle esta cuestión, analizando las diferentes estrategias existentes a la hora de implementar esta operación, sus ventajas e inconvenientes y las peculiaridades de cada una de ellas.

Se estudiarán en concreto tres métodos, dos basados en el uso de tablas precomputadas, que hemos denominado MULTTABLE y LOGTABLE, y el algoritmo de multiplicación general, SHIFT, cuando los métodos anteriores no sean aplicables. Finalmente, se analiza un cuarto método, SPLITW8, que puede considerarse una optimización del algoritmo anterior. Para llevar a cabo esta evaluación, se ha escrito un software específico en lenguaje C y se ha contado con el apoyo de unas librerías públicas de libre uso, rápidas, eficientes y ampliamente utilizadas [12].

III.4.0.1. Método MULTTABLE

Cuando la dimensión n del cuerpo $GF(2^n)$ es pequeña, el procedimiento más rápido para llevar a cabo la multiplicación de dos elementos es precomputar los resultados y almacenar los mismos en una tabla de tamaño adecuado. Por ejemplo, en la Figura III.3 puede encontrarse las tablas de resultados correspondientes a las operaciones de suma y multiplicación para el cuerpo $GF(2^3)$.

En general, la tabla de resultados tiene un tamaño de $2^{(2n+2)}$ bytes, de forma que este método sólo resulta aplicable para valores de n razonablemente pequeños. En el Cuadro III.4 se muestra cómo crece el tamaño de dicha tabla para distintos valores de n. Como puede apreciarse en la práctica, el único valor razonable es $n = 8$. En este caso el tamaño de la tabla es de 256 KB. Para $n = 16$, la tabla ocuparía ya 16 GB, y una cantidad astronómica, unos 64 000 petabytes (!), para $n = 32$.

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Figura III.3: Tablas de suma (izq.) y multiplicación (dcha.) para $GF(2^3)$

n	Tamaño de la tabla resultante	Tamaño en bytes
8	256 KB	2^{18}
10	4096 KB	2^{22}
12	64 MB	2^{26}
14	1 GB	2^{30}
16	16 GB	$2^{34} \approx 10^{10}$
..	..	..
32	64 EB	$2^{66} \approx 10^{18}$

Cuadro III.4: Tamaños de la tabla de resultados precomputados

III.4.0.2. Método LOGTABLE

Cuando el método anterior no puede utilizarse, el siguiente método en cuanto a eficiencia para realizar estas multiplicaciones es hacer uso de un mecanismo ingenioso: tablas de logaritmos y logaritmos inversos, tal y como se describe en [13].

Utilizando estas tablas y algo de aritmética básica, es posible multiplicar dos elementos sumando sus correspondientes logaritmos y tomando luego el logaritmo inverso, para así producir el resultado final. En efecto, $a \times b = ilogTable[(logTable[a] + logTable[b])$, de forma que reducimos la operación multiplicación a tres búsquedas en una tabla y una operación suma (mucho más eficiente). En el Listado 4 puede encontrarse el código correspondiente a este proceso.

La primera de las tablas necesarias, que se denominará *logTable*, se define para valores entre 1 y $2^n - 1$, conteniendo para cada valor su logaritmo en el cuerpo correspondiente. Por otro lado la segunda tabla, llamada *invLogTable*, se define para los índices 0 a $2^n - 1$ y, de forma recíproca, realiza una correspondencia entre estos valores y su logaritmo inverso. Claramente, $logTable[invLogTable[i]] = i$, y $invLogTable[logTable[i]] = i$.

Este tipo de tablas ocupan $2^{(n+2)}$ bytes cada una, en lugar de los $2^{(2n+2)}$ del método anterior. De esta forma, ahora para $n = 8$ y $n = 16$, sólo son necesarios 2KB y 0.5MB, respectivamente. Sin embargo, el crecimiento exponencial es implacable y para $n = 32$ el tamaño necesario ya se dispara a 32GB, fuera del alcance de la mayoría de los computadores modernos.

III.4.0.3. Métod SHIFT

El método LOGTABLE, aunque supone una mejora muy importante respecto al presentado en primer lugar, tiene sin embargo unas necesidades de memoria poco realistas para $n = 32$. En este caso, cuando las tablas no son prácticas, siempre queda el recurso de la multiplicación de propósito general.

Ésta se implementa de manera eficiente convirtiendo el segundo factor, b, en una matriz de bits de dimensión $n \times n$ y multiplicándola por un vector de bits correspondiente al primer factor, a, para así obtener el vector producto [14].

Aunque es significativamente más lento que los dos métodos anteriores, es un procedimiento que funciona siempre, para cualquier tamaño de n y necesita unos requisitos de memoria mínimos.

Algoritmo 4 Generación de las tablas necesarias en el método LOGTABLE

```
#define NW (1 << w) /* En otras palabras, NW es igual a 2 elevado
a w- sima  potencia */

int mult(int a, int b) {
int sum_log;

if (a == 0 || b == 0)
   return 0;

sum_log = gflog[a] + gflog[b];

if (sum_log >= NW-1)
   sum_log -= NW-1;

return gfilog[sum_log];

}
```

III.4.0.4. Método SPLIPTW8

Por último, para el caso especial de $n = 32$, puede utilizarse un método particular únicamente válido para este valor de n.

El proceso comienza creando siete tablas de 256 KB cada una. Estas tablas se utilizan para multiplicar los números de 32 bits, dividiéndolos en cuatro partes de 8 bits cada una, y realizando entonces 16 multiplicaciones y sumas binarias para obtener el resultado final.

Haciendo uso de esta optimización, el algoritmo es 7 veces más rápido que utilizando el método SHIFT para el caso concreto de $n = 32$.

III.4.0.5. Arquitecturas analizadas

En los sucesivos apartados se analizará el diferente rendimiento de los métodos recién expuestos, con el fin de poder elegir posteriormente el más adecuado para cada escenario.

Para alcanzar este objetivo, las pruebas se han llevado a cabo en diferentes arquitecturas con el fin de obtener los resultados más realistas posibles. El Cuadro III.5 muestra las características de cada una de estas arquitecturas utilizadas también en el resto del trabajo.

La elección de estas plataformas se ha realizado procurando que resultasen lo más representativas posibles. Por esta razón se han incluido desde dispositivos con una exigua capacidad de cómputo, como microcontroladores de 8 bits, pasando por computadores ya desfasados, como un Pentium III, hasta ordenadores de última generación, como un Intel Core 2 o un AMD Athlon.

III.4.1. **Análisis de los resultados y Conclusiones**

Los resultados, en millones de multiplicaciones por segundo, se encuentran resumidos en el Cuadro III.6, señalándose en cada caso el método más eficiente (un guión

[1]Cada núcleo tiene una memoria caché privada de nivel 1, L1, dividida, a su vez, en una sección de instrucciones y otra de datos. La caché de nivel 2, de mayor tamaño, se comparte entre los núcleos.

Arquitectura	Frecuencia CPU	Cache L1I/L1D/L2[1]	Memoria RAM
AMD Athlon 64 Dual Core Processor 4200+	2.2Ghz	64KB/512KB	2GB
Intel Core 2 Duo	2GHz	32KB/32Kb/4MB	1GB
Intel Pentium III	450	—	192M
Microcontrolador ATmega168	16MHz	—	14KB

Cuadro III.5: Lista de arquitecturas utilizadas durante la evaluación de las propuestas realizadas, ordenadas de mayor a menor capacidad de proceso.

Arquitectura	Método			
ATmega168	MULTTABLE	LOGTABLE	SHIFT	SPLITW8
$GF(2^8)$	-	-	0.00079	-
$GF(2^{16})$	-	-	-	-
$GF(2^{32})$	-	-	-	-
PentiumIII	MULTTABLE	LOGTABLE	SHIFT	SPLITW8
$GF(2^8)$	31.73	$27.77/_{0.87}$	$1.39/_{0.04}$	-
$GF(2^{16})$	-	4.53	$0.55/_{0.12}$	-
$GF(2^{32})$	-	-	0.18	$0.87/_{4.83}$
DualCore	MULTTABLE	LOGTABLE	SHIFT	SPLITW8
$GF(2^8)$	110.03	$94.11/_{0.85}$	$4.40/_{0.03}$	-
$GF(2^{16})$	-	60.25	$0.55/_{0.02}$	-
$GF(2^{32})$	-	-	0.47	$8.03/_{17.085}$
Athlon	MULTTABLE	LOGTABLE	SHIFT	SPLITW8
$GF(2^8)$	89.62	$78.76/_{0.87}$	$5.52/_{0.06}$	-
$GF(2^{16})$	-	29.05	$2.25/_{0.07}$	-
$GF(2^{32})$	-	-	0.82	$3.41/_{4.15}$

Cuadro III.6: Resultados de los diferentes métodos del multiplicación en diversas arquitecturas (en millones de multiplicaciones por segundo).

indica que el método no es aplicable para esa combinación de tamaño de n y arquitectura). Por otro lado, los subíndices evalúan cómo se comporta el rendimiento de cada método para un cuerpo dado respecto al método más rápido. Este resulta ser MULTTABLE en todos los casos y es el que se ha considerado como referencia.

Así, por ejemplo, considerando la arquitectura *DualCore* y el cuerpo extendido $GF(2^8)$ puede observarse que el método MULTTABLE permite llevar a cabo algo más de 110 millones de multiplicaciones por segundo, mientras que el método LOGTABLE únicamente alcanza los 94.11 millones. El subíndice que aparece en este último método, 0.85, corresponde al cociente entre los dos valores anteriores y proporciona un ratio entre el rendimiento de los dos métodos. De esta forma, si el valor es menor que 1, el segundo método será más lento; concretamente 0.85 veces en este caso. De la misma forma, si el valor es mayor que 1, el segundo método será más rápido en la proporción indicada.

III.4.1.1. Arquitectura ATMega168

La primera de las plataformas analizadas corresponde a un microcontrolador de 8 bits, de gama baja o media, típico de una red de sensores. Éste cuenta con escasos recursos computacionales, tanto de CPU como de memoria, por lo que la elección del método más adecuado no resulta difícil ya que tan sólo uno de ellos es aplicable en la práctica: el método SHIFT. Los métodos MULTTABLE y LOGTABLE, ambos basados en el uso de tablas, quedan claramente fuera del alcance de un dispositivo con apenas 14 KB de memoria disponible.

En resumen, en este tipo de arquitecturas sólo puede hacerse uso del método SHIFT que, aunque es el más lento de todos, consigue realizar casi 800 multiplicaciones por segundo lo que supone una cifra nada despreciable en un arquitectura con recursos computacionales tan limitados.

III.4.1.2. Arquitecturas Intel y Athlon

En las arquitecturas PC puede observarse que los resultados absolutos son, como cabría esperar, proporcionales a su capacidad computacional concreta. Lógicamente la plataforma *DualCore*, mucho más moderna, puede realizar aproximadamente tres veces más operaciones por segundo que el *PentiumIII*.

A pesar de ello, puede observarse que las diferencias de rendimiento entre los distintos métodos se mantienen entre arquitecturas. En otras palabras, la proporción de cuánto es mejor o peor un método respecto a otro se mantiene entre las diferentes plataformas. Este hecho es fácil de comprobar observando que, por ejemplo, el coeficiente de evolución del método LOGTABLE toma los valores 0.87, 0.85 y 0.87 en las diferentes arquitecturas, lo que supone una desviación típica de sólo $\sigma = 0.011$. Por otro lado, el método SHIFT presenta una variación en este coeficiente algo mayor, de 0.03, aunque también éste puede considerarse un valor muy consistente.

La principal conclusión importante que puede extraerse de este hecho es que *las notables diferencias en las arquitecturas, tanto en capacidad computacional como memoria RAM, no tiene una influencia determinante en el rendimiento de los diferentes métodos*, que presentan comportamientos muy estables. Evidentemente, el rendimiento absoluto es muy diferente entre arquitecturas, pero "escala" de la misma forma en cada una de ellas.

Otro hecho significativo que se desprende del análisis de los datos presentados es que, como puede observarse, el rendimiento decrece rápidamente conforme aumenta la dimensión del cuerpo; de forma que, por ejemplo, la multiplicación en $GF(2^{32})$ es unas 10 veces más lenta que en $GF(2^8)$ en todas las arquitecturas. En la Figura III.4 puede encontrarse una gráfica comparativa del rendimiento de cada método en cada arquitectura.

Conclusiones Las conclusiones finales que pueden extraerse de los experimentos realizados resultan, a la vista de los datos, bastante claras. Trabajando en el cuerpo $GF(2^8)$ el método más eficiente para la multiplicación de elementos en $GF(2^n)$ resulta sin duda MULTTABLE, que utiliza tablas precomputadas de 256KB. El tamaño de estas tablas, sin embargo, las hace prohibitivas cuando pasamos al cuerpo $GF(2^{16})$. En este caso deben utilizarse otro tipo de tablas, las utilizadas por el método LOGTABLE. Éstas, sin embargo, requieren pasos adicionales, lo que provoca que dicho método vea decrementado ligeramente su rendimiento.

Finalmente, en $GF(2^{32})$, el método más adecuado es SPLITW8, que implementa una mejora sobre el método general SHIFT, aún cuando requiera una memoria total de 1792KB para almacenar las tablas necesarias.

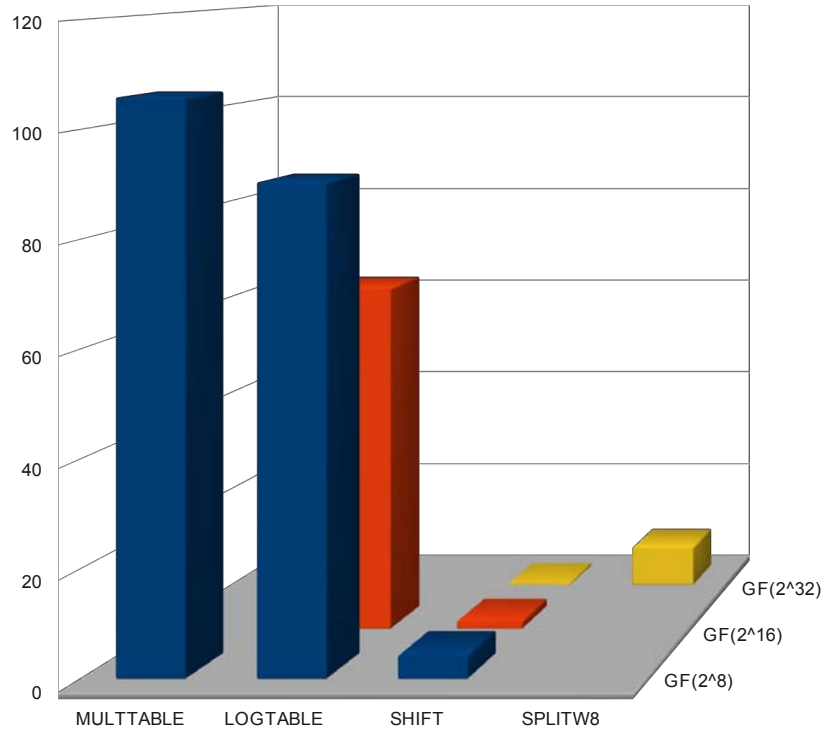


Figura III.4: Comparativa de los métodos de multiplicación en el cuerpo algebraico $GF(2^n)$

III.5. IMPLEMENTACIÓN EFICIENTE DE LFSRS EN SOFTWARE

Como se ha comentado con anterioridad, los LFSRs son estructuras especialmente difíciles de implementar de forma eficiente en software. La razón principal es el desplazamiento necesario, en cada pulso de reloj, del contenido de cada una de sus celdas.

Este desplazamiento ocurre de forma simultánea en las implementaciones hardware, de forma que todo el proceso puede llevarse a cabo en un único pulso de reloj. No ocurre lo mismo, sin embargo, en las versiones software donde el proceso se convierte en iterativo y costoso. En este caso se necesita para un registro de l etapas un total de $l - 1$ desplazamientos.

Este efecto se ve agravado cuando el LFSR está definido, como ocurre habitualmente, sobre el cuerpo binario $GF(2)$. En este caso el rendimiento se ve fuertemente afectado por dos factores fundamentales:

- Dado que la unidad de trabajo es el bit y que un microprocesador trabaja como unidad mínima con bytes, se debe utilizar un byte para almacenar cada bit. Desestimando el evidente desaprovechamiento de memoria, este hecho afecta asimismo al rendimiento, pues deben transferirse en cada desplazamiento 8 bits de un lugar a otro de la memoria en vez de uno sólo.

- Por la misma razón, el acceso a cada bit del registro es muy costoso, pues deben utilizarse máscaras y desplazamientos internos dentro de cada palabra que afectan de forma muy significativa al rendimiento final.

Un ejemplo de la implementación de un LFSR tradicional, que hace uso de la librería mencionada en la sección III.4, puede verse a continuación:

Algoritmo 5 Implementación tradicional de un LFSR

```
1  void shift()
2  {
3      int i=0;
4
5      feedBackByte_R1 = 0;
6
7      // Itera sobre los coeficientes de realimentacion
8      for (i=0; i < num_coefficients_R1; i++)
9         feedBackByte_R1 = (feedBackByte_R1^
10                 galois_single_multiply(val_coefficients_R1[i],
11                    contents_R1[coefficients_R1[i]],galoisField));
12
13     // Obtiene el byte de salida
14     generatedByte_R1 = contents_R1[0];
15
16     // Desplaza los contenidos del registro
17     for (i=1 ; i < R1_LENGTH ; i++)
18        contents_R1[i-1] = contents_R1[i];
19
20     // Se incluye el byte de realimentacion
21     contents_R1[R1_LENGTH -1] = feedBackByte_R1;
22
23     return generatedByte_R1;
24 }
```

Como se ha comentado en la sección anterior, las operaciones de multiplicación necesarias para trabajar en un nuevo cuerpo algebraico base suponen la mayor parte del coste computacional del funcionamiento de estos registros. Por esta razón, es muy importante optimizar al máximo la implementación software de estas operaciones. Asimismo, los registros de desplazamiento son estructuras con una implementación software de gran dificultad, debido a su inherente naturaleza paralela. Por tanto, es también esencial utilizar cualquier técnica de implementación en software que ayude a mejor el rendimiento final.

En las siguientes secciones se describirán y analizarán tres de estas técnicas de implementación: *búfers circulares*, *ventanas deslizantes* y *desdoblamiento de bucles*. Para ello estas técnicas han sido implementadas con la misma librería utilizada en la sección anterior.

III.5.1. Búfer circular

Esta técnica es conocida y utilizada ampliamente en otros ámbitos de la algorítmica [15, 16]. Sin embargo, aunque parece especialmente adecuada para su aplicación a registros de desplazamiento, no ha sido habitualmente utilizada para este propósito. A los autores les consta únicamente que el cifrador Dragon [17], recientemente presentado en el proyecto eSTREAM, la haya incorporado de forma inherente a su diseño.

Los beneficios de esta técnica son obvios para los LFSR, pues permite evitar los costosos desplazamientos necesarios, $L-1$, en un registro de L etapas. Su funcionamiento es muy sencillo y consiste, básicamente, en utilizar varios punteros, que se mueven a lo largo del registro, en lugar de mover los datos en sí. Cuando éstos llegan al final del búfer, vuelven al principio, creando la ilusión de un *búfer circular*.

Los punteros necesarios para el uso de esta técnica aplicada a LFSR son los siguientes:

- *P_INICIO*: Indica la posición de inicio del registro. Ésta corresponde a la celda que genera la salida en cada ciclo y, según la notación utilizada en la sección III.2, correspondería a la celda s_{t+l}.
- *P_FIN*: Indica el final del registro, es decir, la celda que recibe la realimentación en cada ciclo, s_0.
- P_FeedBack$_i$: Son una serie de punteros, uno por cada realimentación del registro. Cada uno de ellos almacena la posición de cada realimentación, que, como el resto de punteros, también se mueven en cada ciclo.

En el Algoritmo 6 puede encontrarse un esquema de funcionamiento de esta técnica de implementación.

Como puede apreciarse, actualizar los diferentes índices necesarios para el funcionamiento del algoritmo implica el uso de aritmética módulo l, la longitud del registro. Si l es una potencia de 2, esta operación es muy eficiente en los actuales microprocesadores, pero no lo es tanto en caso contrario. Para evitar este inconveniente, puede utilizarse una variante de esta técnica, que se presenta a continuación.

III.5.2. Ventana deslizante

La técnica de la *ventana deslizante*, como método genérico, se aplica en diferentes ámbitos incluso en criptografía [18], pero es especialmente conocida por su uso en el protocolo TCP/IP, donde se encarga del control del flujo de datos [19].

Algoritmo 6 Implementación de un LFSR utilizando un búfer circular

```
byte LFSR_CircularBuffer()
{

    unsigned int i=0;
    unsigned int feedBackByte = 0;

    // Itera sobre los coeficientes de realimentación
    for (i=0; i < num_coefficients; i++){
           feedBackByte = (feedBackByte ^
                  galois_single_multiply(val_coefficients[i],
                     contents[coefficients[i]],galoisField));
        coefficients[i] = (++coefficients[i])  % R_LENGTH;
    }

    // Obtiene el byte de salida
    generatedByte = contents[punteroInicio];

    // Se avanzan los punteros modulo la longitud del registro
    punteroInicio = (++punteroInicio  % R_LENGTH);
    punteroFin = (++punteroFin  % R_LENGTH);

    // Se incluye el byte de realimentación
    contents[punteroFin] = feedBackByte;

    return generatedByte;
}
```

Utilizando los mismos principios de la técnica anterior es posible, sin embargo, evitar alguno de sus inconvenientes haciendo uso de un búfer de longitud doble de la necesaria. El esquema de funcionamiento se convierte, entonces, en el que sigue:

- Cuando se escribe un nuevo valor en el búfer, se hace simultáneamente en dos lugares diferentes. El puntero comienza en la mitad del búfer de longitud doble y cuando alcanza el final vuelve al medio de nuevo.
- De esta forma, se puede acceder a todos los $k-l$ valores intermedios en posiciones fijas a la derecha del puntero.

En el Listado 7 se encuentra un esquema del funcionamiento de este método y, en la Figura III.5, una comparativa con la técnica del búfer circular. Como ventaja adicional, este esquema permite que el registro puede tener una longitud arbitraria, sin restricciones, y a pesar de ello resultar muy eficiente en su funcionamiento. El tiempo de actualización de sus contenidos permanece siempre constante y pequeño, independientemente del tamaño del búfer, cualidad ésta que resulta especialmente importante para su uso en aplicaciones criptográficas.

III.5.3. Desdoblamiento de bucle

Por último, la técnica de *desdoblamiento de bucle*[2] (*Loop unrolling* o *loop unwinding* en inglés) es un método de optimización muy utilizado en diversos ámbitos de la informática, que pretende mejorar el tiempo de ejecución de aquellas porciones de código

[2]Resulta especialmente difícil encontrar referencias al nombre de esta técnica en castellano. Incluso en textos académicos la técnica suele citarse exclusivamente en inglés.

Algoritmo 7 Implementación de un LFSR utilizando la técnica de ventana deslizante

```
byte LFSR_SWINDOW(void)
{
    int i=0, j=0;
    uint8_t feedBackByte = 0, generatedByte = 0;
    int punteroK = R1_LENGTH;

    feedBackByte = 0;

    // Itera sobre los coeficientes de realimentación
    for (i=0; i < num_coefficients; i++)
        feedBackByte = (feedBackByte ^
              contents[punteroK - R1_LENGTH + coefficients[i]]);

    // Obtiene el byte de salida
    generatedByte = contents[punteroK-R1_LENGTH];

    // Se incluye el byte de realimentación
    contents[punteroK-R1_LENGTH] = contents[punteroK] = feedBackByte;

    if (++punteroK == 2*R1_LENGTH)
        punteroK = R1_LENGTH;

    return generatedByte;
}
```

que contengan bucles [20][21]. Dado que la implementación de un registro de desplazamiento en software es básicamente la repetición del cuerpo de un bucle, esta técnica parece, a priori, encajar de forma natural con nuestros objetivos.

El desdoblamiento de bucle funciona replicando el cuerpo original del bucle principal varias veces, ajustando el código que se encarga de la comprobación del fin del bucle y eliminando las instrucciones de salto que resulten redundantes. Aunque pueda parecer ilógico a primera vista, esta técnica consigue en la mayoría de los casos que el código optimizado, a pesar de ser más largo, requiera menos tiempo de ejecución. En el Listado 8 puede encontrarse un ejemplo sencillo que ilustra el funcionamiento básico de esta técnica.

La eficacia de este método de optimización se debe fundamentalmente a dos razones básicas:

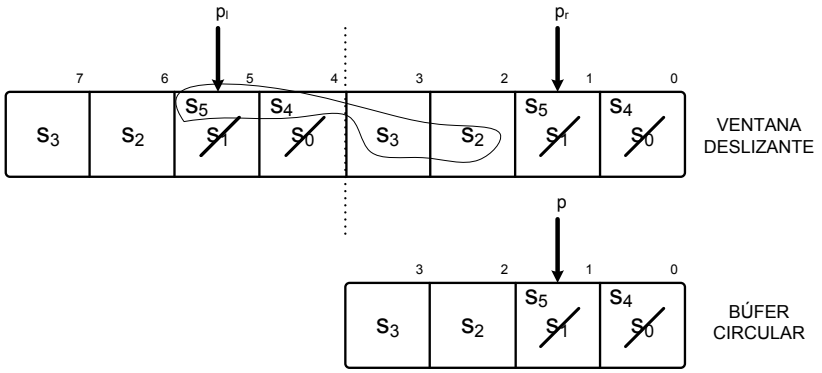


Figura III.5: Comparativa de los métodos de ventana deslizante y búfers circulares

Algoritmo 8 Técnica de desdoblamiento de bucle aplicada a un ejemplo sencillo

```
1
2  // Código sin optimizar        // Código optimizado
3  for (int x = 0; x < 100; x++)       for (int x = 0; x < 50; x++)
4   {                   {
5       delete(x);                    delete(x);
6   }                       delete(x+1);
7                           delete(x+2);
8                           delete(x+3);
9                           delete(x+4);
10                      }
```

- El número de instrucciones de saltos se reduce y, en consecuencia, la variable índice que controla el bucle se modifica un número menor de veces.
- En casi cualquier arquitectura moderna diseñada con un alto grado de paralelismo, esta técnica puede obtener mejores prestaciones de ejecución para las instrucciones del interior del bucle.

El segundo de los aspectos citado es, sin duda, el más importante. En el ejemplo anterior el código sin optimizar no puede explotar el *pipeline* del microprocesador, pues una instrucción de salto bloquea habitualmente este proceso, incluso en aquellas arquitecturas que utilicen *predicción de salto* [22]. Sin embargo, el código optimizado puede incluir las instrucciones de las líneas 5 a la 9 simultáneamente en las diferentes etapas del *pipeline*, aumentado de esta forma el paralelismo y rendimiento de la ejecución.

Aunque es difícil de cuantificar y depende en gran medida de cada caso concreto, en general esta técnica permite mejoras en el tiempo de ejecución que oscilan entre el 10 y el 30 % [23, 24].

III.6. ANÁLISIS DEL RENDIMIENTO

Tras analizar los aspectos subyacentes más importantes, como los diferentes métodos de multiplicación y técnicas para la implementación eficiente en software, en este apartado se unifican todos estos aspectos individuales y se analiza el rendimiento global de un LFSR diseñado mediante las diversas técnicas propuestas en esta tesis. Concretamente, se trata de un LFSR de 17 etapas, cuyos polinomios de realimentación correspondientes son los definidos en la sección III.3.3.

El rendimiento de la implementación software de estos métodos ha sido analizado en las plataformas anteriormente presentadas en el Cuadro III.5, con el fin de estudiar su comportamiento en entornos reales. Este análisis ha sido dividido en dos grandes apartados, en función de en qué parte de la ejecución de cada método se hace un mayor énfasis: *microanálisis* y *macroanálisis*.

El primero de ellos mide, a muy bajo nivel, cuáles son los factores clave como fallos de memoria caché, utilización de los buses o CPI[3] en la ejecución de un LFSR implementado en software. Por su parte, el macroanálisis se centra en las métricas de rendimiento usuales como tiempo de ejecución, flujo de generación de secuencia cifrante y otras.

Finalmente, se ha analizado el comportamiento de todas las técnicas anteriores en cuatro cuerpos algebraicos diferentes: el tradicional cuerpo binario $GF(2)$ y sus cuerpos extendidos $GF(2^8)$, $GF(2^{16})$ y $GF(2^{32})$. La operación de multiplicación se ha implementado utilizando el método más rápido en cada caso, según se analizó en III.4.

III.6.1. **Microanálisis**

El microanálisis del rendimiento incluye aspectos de bajo nivel, muy cercanos a la estructura hardware del computador y al sistema operativo. Algunos de estos aspectos incluyen medir el número de ciclos de CPU utilizados (y, en consecuencia, el CPI), los fallos en los accesos a memoria, la utilización de los buses del sistema, etc...

Estos datos permitirán averiguar si la implementación realizada no está completamente optimizada y tiene algunos cuellos de botella que disminuyan su rendimiento. Estos datos serán obtenidos con el mecanismo conocido como *unidades de monitorización del rendimiento* (*Performance Monitoring Units*, PMUs).

III.6.1.1. Unidades de monitorización del rendimiento

Hoy en día, todos los grandes procesadores modernos incluyen un conjunto de monitores hardware de rendimiento que capturan información de la microarquitectura de muy bajo nivel. Pueden, de esta forma, proporcionar información más exacta sobre cómo el hardware está siendo utilizado por los diferentes programas que corren en ellos [25].

Curiosamente, aunque estos monitores han existido desde hace varios años, es solo recientemente cuando se han empezado a utilizar de forma masiva, debido a diferentes causas (falta de buena documentación por parte del fabricante, interfaces con el SO escasos y defectuosos y falta de herramientas de calidad para su uso).

Especialmente importante para este trabajo es medir, con mucha precisión y con una sobrecarga mínima, el tráfico existente entre el núcleo del procesador y el subsistema de memoria. Este y otros datos, que se denominan *eventos*, se miden utilizando *contadores de rendimiento*, que no son más que registros hardware especiales del procesador dedicados exclusivamente a esta tarea.

[3] *Clocks per Instructions Retired*. Se analizará con detalle en la sección III.6.1.2.

Para obtener todos estos valores se han utilizado dos herramientas de análisis: la *Intel VTune profiling tool* [26] y el interfaz *perfmon2* [27]. Estas herramientas nos permitirán obtener los datos necesarios y realizar un microanálisis pormenorizado de la ejecución de un LFSR en software, a la vez que nos permitirá determinar los parámetros más importantes que influyen en su rendimiento.

Recolección de los datos Para realizar una toma de datos correcta y fiable, es importante respetar escrupolosamente los siguientes aspectos:

- Las medidas deben realizarse siempre con una carga en el sistema similar y reproducible. De esta forma, el código bajo análisis tendrá un flujo de ejecución similar cada vez. Para conseguir esto, hay que asegurarse de que no existen procesos innecesarios en ejecución y que el proceso analizado tiene a su disposición la mayoría del tiempo de CPU. En este trabajo, para asegurar al máximo las condiciones anteriores, el SO[4] fue arrancando en el nivel de ejecución 3, sin entorno gráfico.
- El código bajo análisis debe haber sido compilado utilizando las máximas opciones de optimización permitidas por el compilador. En entornos UNIX, la opción -O3 proporciona estas características, incluyendo un uso agresivo del *pipeline* (que, como se ha analizado anteriormente, aporta beneficios claros en el caso de los LFSR).

III.6.1.2. Utilización de CPU

El uso de CPU suele medirse utilizando la métrica estándar CPI (*Clocks per Instructions Retired*) [28]. Este valor se calcula de forma sencilla dividiendo el número de ciclos de reloj en un periodo de tiempo de ejecución del código analizado por el número de instrucciones que han finalizado durante el mismo:

$$CPI = \frac{\text{Número de ciclos de reloj}}{\text{Número de instrucciones}}$$

El CPI es habitualmente el primero de los ratios que se comprueba cuando se desea determinar dónde es necesario centrar los esfuerzos de optimización en un código determinado. Valores bajos de CPI suponen buenos ratios, ya que indican que el código se ejecuta eficientemente. Por el contrario, altos valores de CPI implican que se están utilizando muchos más ciclos de reloj de los que deberían. Normalmente los ratios CPI observados en programas de uso común oscilan entre 0.75, que supone un valor excelente, y 4, que es indicativo de un rendimiento realmente pobre.

III.6.1.3. Subsistema de memoria caché

Uno de los aspectos más importantes en el análisis del rendimiento es la influencia de los accesos a memoria y, más concretamente, a la memoria caché. En función del procesador, pueden existir múltiples niveles de memoria caché antes de que sea necesario acceder a la memoria principal.

En esta tesis se ha supuesto una estructura de este subsistema de memoria como la descrita en la figura III.6, con dos niveles de caché de nivel 1 y una de nivel 2. Esta última, la memoria L2, es compartida entre instrucciones y datos, mientras que L1 dispone de dos áreas diferenciadas: L1D para datos y L1I para instrucciones. Por supuesto, como es habitual, la memoria L1 suele ser mucho más rápida que L2.

[4] Linux Ubuntu 9.04

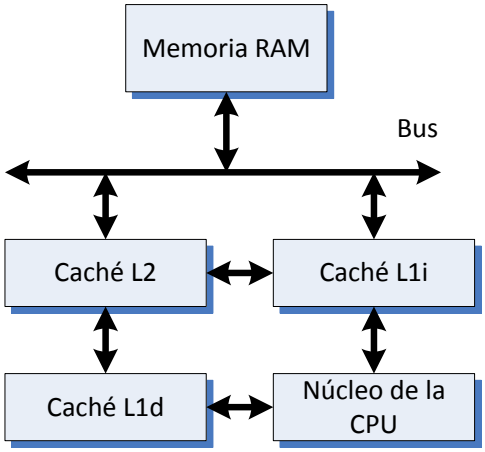


Figura III.6: Estructura de memoria caché considerada en este trabajo de tesis

La importancia de esta métrica radica en el hecho de que un valor alto de fallos en L1 indica que el código a menudo no encuentra los datos que necesita en este nivel y debe "pagar" la penalización que supone el acceso a L2, una memoria de acceso más lento que la anterior.

Por otro lado, una tasa alta de fallos en L2 indica que el código tiene un conjunto residente de datos mayor que la propia L2. Evitar este problema es una de las razones por los que los métodos de multiplicación propuestas tratan, siempre que sea posible, de crear tablas que puedan ser alojadas completamente en este nivel de memoria.

III.6.1.4. Utilización de los buses del sistema

El ratio de utilización de buses está directamente relacionado con el anterior, ya que mide el porcentaje de ciclos de bus que se utilizan para la transferencia de datos de cualquier tipo. Claramente, una alta utilización de estos buses indica que existe un flujo de tráfico muy alto entre el procesador y la memoria principal.

Dado que el acceso a los buses del sistema es una operación muy lenta comparada con la transferencia entre niveles de memoria caché, un código con una mala puntuación en esta métrica tendrá obviamente un bajo rendimiento.

III.6.1.5. Resultados y Conclusiones del microanálisis

Los resultados obtenidos por las implementaciones realizadas en este trabajo de tesis han sido excelentes. El valor CPI ha oscilado siempre entre 0.64 y 0.95, para el caso mejor y peor, respectivamente. Este dato demuestra que las implementaciones están altamente optimizadas y son, por tanto, útiles para medir el rendimiento real de las diferentes configuraciones de LFSR planteadas.

En cuanto al uso de memoria, durante la ejecución de los experimentos se pueden diferenciar claramente dos etapas.

En la primera de ellas, transitoria, se observa una tasa de utilización de los buses muy alta, que va paulatinamente decreciendo hasta estabilizarse a un valor muy bajo. Esto se debe, como es lógico, a una serie de fallos en la memoria caché producidos por la carga del programa en memoria principal, la creación de las tablas de multiplicación, etc... En la Figura III.7 puede observarse la evolución de la tasa de fallos durante este

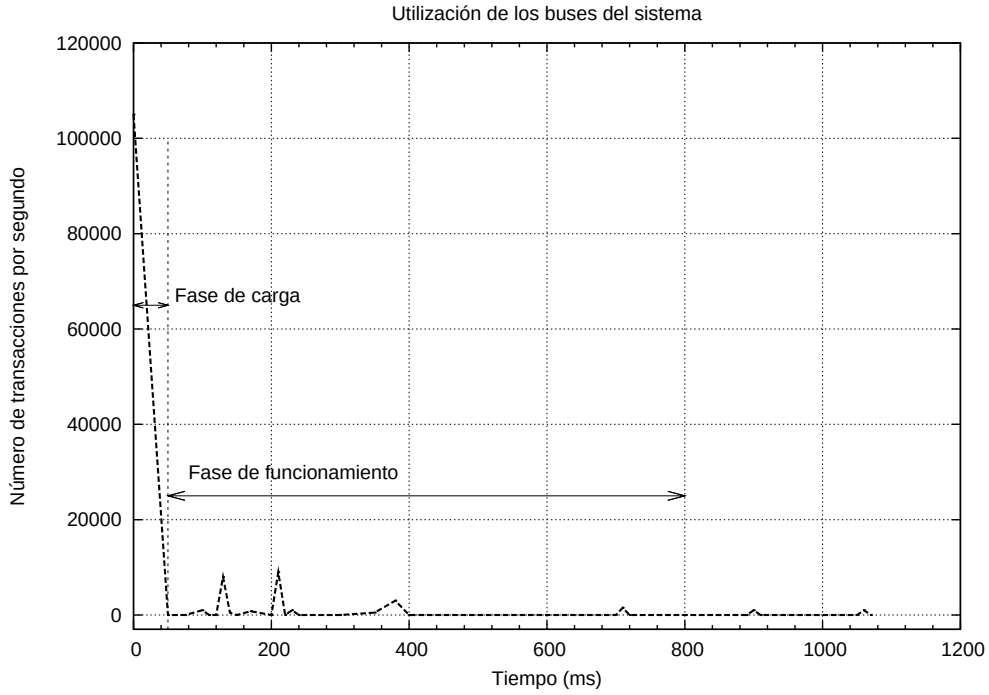


Figura III.7: Evolución de la utilización de los buses del sistema. El eje de abscisas indica el tiempo, en milisegundos, transcurrido desde el inicio de la ejecución del LFSR. En el eje de ordenadas pueden encontrarse el número de transferencias por segundo en los buses.

periodo, etiquetado como '*Fase de carga*', que se extiende durante los primeros 50 ms aproximadamente.

En el segundo periodo, que dura desde los 50 ms en adelante ('*Fase de funcionamiento*'), es cuando realmente comienza la ejecución del LFSR. En ese momento, todos los datos necesarios están ya en memoria caché y puede observarse que se producen muy pocos fallos en L2, lo que implica que la mayoría de la actividad de la caché ocurre en L1 y L2.

Por otro lado, también se han medido otros parámetros menores, como el tiempo medio de los accesos a memoria o el número de fallos en la predicciones de salto, aunque sus valores concretos no se muestran puesto que son prácticamente cero en todos los casos. Ambos parámetros parecen, de esta forma, confirmar las conclusiones pues unos valores tan bajos, que pueden considerarse cero en la práctica, corresponden a una alta eficiencia del programa analizado, en todas las configuraciones.

Finalmente, en la Figura III.8 puede encontrarse, de forma gráfica, un resumen de todos los datos aportados en estos experimentos. En ella se compara, además, el comportamiento de estos LFSRs extendidos con otros algoritmos de referencia, como AES funcionando en modo "cifrado en flujo" (AES-CTR) [29, 30] y RC4 [31]. De esta comparativa pueden extraerse interesantes conclusiones.

La primera es que el CPI de todas las implementaciones es sensiblemente menor que el de AES-CTR. Este hecho resulta lógico hasta cierto punto, pues aunque AES sea probablemente uno de los algoritmos más revisados y optimizados de la historia de la criptografía, no deja de ser un cifrador en bloque, mucho más "pesado" y complejo que

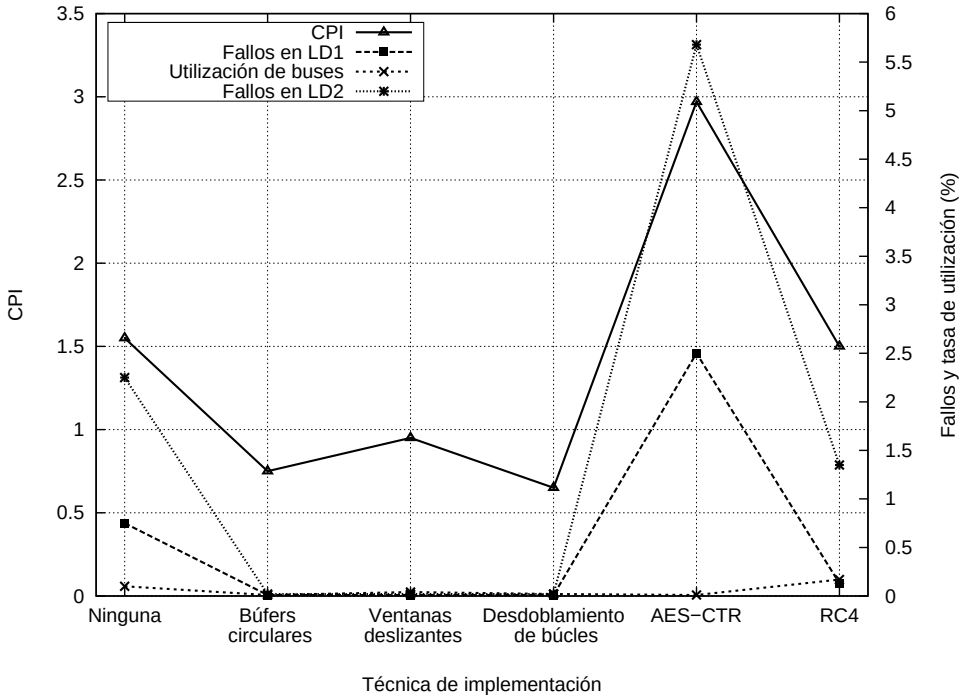


Figura III.8: Comparativa de diversos parámetros correspondientes al microanálisis entre las implementaciones realizadas y otros algoritmos de referencia, como AES-CTR y RC4.

un LFSR bien optimizado.

Por otro lado, el CPI es también inferior, aunque en menor medida, al de RC4. Este dato sí resulta más significativo, pues RC4 es un cifrador en flujo, con el que sí podemos comparar directamente los resultados obtenidos.

III.6.2. Macroanálisis

A diferencia del estudio anterior, el macroanálisis que se presenta a continuación hace uso de las métricas habitualmente utilizadas en la valoración de la eficiencia de una implementación, como el tiempo de ejecución total o la capacidad máxima de generación de datos de salida. En nuestro caso, los experimentos consistieron en evaluar las métricas anteriores durante la generación de una secuencia cifrante de 10^7 bytes por parte de un LFSR construido según las técnicas expuestas hasta el momento.

Dado que existen muchos factores que influyen en el rendimiento final, como el cuerpo base elegido o las diferentes mejoras a la implementación estudiadas en las sección IV.4.1.1, se han utilizado 16 configuraciones diferentes, con el fin de poder distinguir qué proporción del rendimiento final corresponde a cada uno de los factores bajo estudio.

Para cada una de estas configuraciones se ha calculado la *tasa de generación de secuencia cifrante*, medida en Mbytes por segundo, y un *factor de mejora*. Este factor mide el incremento observado en el rendimiento de cada configuración respecto al que tendría un LFSR tradicional, definido sobre $GF(2)$, y ejecutado en exactamente las mismas condiciones. De esta forma, un factor de mejora de 2 significa que el modelo propuesto

es 2 veces más rápido que las implementaciones existentes.

Por último, cada medida han sido llevada a cabo realizando un total de 10 ejecuciones de cada configuración, descartando el valor máximo y mínimo, y calculado la media aritmética del resto de valores.

III.6.2.1. Factores de mejora

Con el fin de analizar con precisión el comportamiento de cada una de las mejoras de implementación propuestas, se han definido varios factores de mejora cada uno de ellos con un objetivo diferente:

- Factor de mejora α: mide la mejora obtenida a través del uso, exclusivamente, de cuerpos extendidos. Se calcula comparando los tiempos necesarios para generar la secuencia de prueba en $GF(2)$ y en cada cuerpo.
- Factor de mejora β: mide la mejora debida a utilizar, únicamente, las diversas técnicas de mejora de implementación: búfers circulares, ventanas deslizantes y desdoblamiento de bucles. Este factor se calcula, por tanto, comparando los tiempos invertidos por la configuración en el cuerpo correspondiente sin mejoras y con cada una de las mejoras de implementación.
- Factor de mejora γ: indica la mejora global obtenida por el uso combinado de las mejoras de implementación y de los cuerpos extendidos. Es, claramente, el mejor indicador, pues refleja la mejora total que es posible obtener utilizando las técnicas expuestas. De forma intuitiva, y teniendo en cuenta las inherentes e inevitables imprecisiones en la medida, se debe cumplir que $\gamma \approx \alpha + \beta$.

III.4 Ejemplo. *Veamos con un ejemplo cómo se han calculado los diferentes factores de mejora. Utilizando los datos que pueden encontrarse en las Tablas III.7 y III.8, se observa que: para la arquitectura Dual Core los tiempos que necesita para generar la secuencia de prueba con la configuración tradicional, con ventanas deslizantes en $GF(2)$, sin mejoras en $GF(2^8)$ y con ventanas deslizantes en $GF(2^8)$, son respectivamente* 1.58, 0.47, 0.56 *y* 0.44 *segundos.*
Así, el factor α que mide la relación entre los tiempos invertidos por la configuración tradicional y la que utiliza cuerpos extendidos sería $\alpha_{GF(2^8)} = 1.58/0.56 = 2.82$.
Por otro lado, con el uso de ventanas deslizantes, trabajando en el mismo cuerpo, sólo son necesarios 0.44 segundos, por lo que el factor de mejora β sería, por tanto, $\beta = 0.56/0.44 = 1.27$.
Finalmente, el tiempo invertido utilizando ventanas deslizantes y un cuerpo extendido, $GF(2^8)$ en este caso, es de 0.44 *segundos. Por tanto, el último factor de mejora, $\gamma_{GF(2^8)}$ es de $\gamma_{GF(2^8)} = 1.58/0.44 = 3.59$.*

III.6.2.2. Resultados y Conclusiones del macroanálisis

El primero de los aspectos que ha sido estudiado es la evolución del factor α o, lo que es lo mismo, cómo varía la mejora en el rendimiento que proporciona el uso de cuerpos extendidos. Para obtener resultados más significativos, se ha realizado este análisis en diversas plataformas hardware, estudiando también cómo afecta ésta al rendimiento final.

Factor α Los valores obtenidos pueden encontrarse resumidos en el Cuadro III.7. En ella se encuentran los resultados para cada arquitectura hardware, clasificados según el cuerpo base que ha sido utilizado, junto con la tasa de generación y factor de mejora α obtenidos.

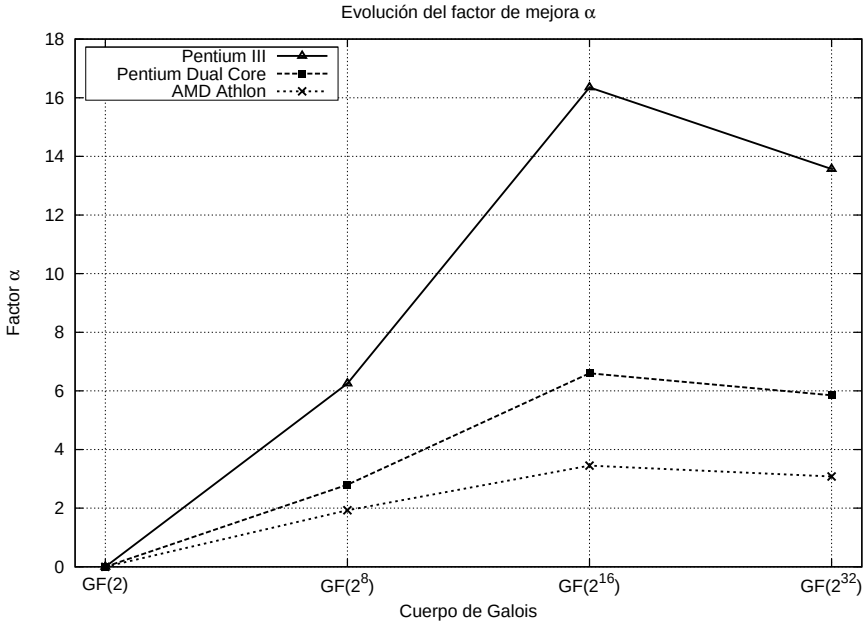


Figura III.9: Evolución del factor α frente al tamaño del cuerpo extendido base

La primera de las conclusiones que puede extraerse de los datos es que, tal y como se esperaba, conforme el tamaño del cuerpo aumenta, el rendimiento también lo hace en todas las arquitecturas. Sin embargo, un hecho significativo y llamativo es que el beneficio que puede obtenerse es distinto en las diferentes arquitecturas. Tal y como puede observarse en la Figura III.9, la arquitectura que obtiene mayor beneficio, con un factor α máximo de 16.35 para $GF(2^{16})$, es la de Pentium III que constituye la de menor complejidad y capacidad de cómputo. En otras palabras, puede decirse que las "peores" plataformas se benefician más del uso de cuerpos extendidos que las arquitecturas más modernas.

La razón de este comportamiento es que las plataformas como Pentium III cuentan con un menor número de estructuras de optimización hardware internas, tales como módulos de predicción de salto o pipeline de mayor número de etapas. Por tanto, el beneficio del uso de los cuerpos extendidos es más notable y no queda "difuminado" e incluido en el aumento de rendimiento proporcionado, de forma intrínseca, por las técnicas de optimización hardware. En cierta manera, podríamos decir que lo que se consigue es *emular* por software las estructuras anteriores.

En el resto de plataformas, más modernas y por tanto quizás más significativas, el aumento del rendimiento es también muy notable. Por ejemplo, en el cuerpo $GF(2^8)$ el LFSR resultante es casi 3 veces más rápido que su equivalente tradicional sobre $GF(2)$. Este buen resultado es posible, en gran medida, gracias a que las operaciones de multiplicación se llevan a cabo utilizando tablas de pequeño tamaño, que caben completamente en memoria RAM y cuyo acceso es muy rápido.

Sin embargo, conforme aumenta el tamaño del cuerpo, las operaciones matemáticas involucradas se vuelven cada vez más costosas. Por esta razón, se produce un curioso fenómeno: el factor de mejora se incrementa de forma lineal hasta $GF(2^{16})$ y, entonces, en lugar de seguir aumentando como cabría esperar, decrece ligeramente para $GF(2^{32})$.

La razón de este comportamiento se debe a que, en ese punto, el compromiso entre

el aumento de rendimiento y el coste computacional de las operaciones de multiplicación se rompe. En dicho punto, el coste de cada operación empieza a ser mayor que el beneficio obtenido y, por tanto, deja de compensar su uso respecto a $GF(2^{16})$.

Esto no significa, por supuesto, que trabajar en el cuerpo $GF(2^{32})$ no sea rentable respecto a hacerlo en el tradicional $GF(2)$, sino únicamente que la tendencia observada hasta el momento en el rendimiento, conforme el tamaño del cuerpo aumenta, comienza a decrecer. En cualquier caso, trabajar en $GF(2^{32})$ resulta entre 3 y 13 veces más eficiente que en el cuerpo binario.

Resumen A la vista de los experimentos realizados y los datos recogidos es posible obtener dos importantes conclusiones en cuanto al uso de cuerpos extendidos en LFSRs. La primera es que el uso de éstos cuerpos proporciona importantísimas mejoras respecto a los métodos tradicionales, que pueden llegar a factores 14, o del 1300 %[5] en arquitecturas con una capacidad de cómputo modesta. En arquitecturas más modernas y potentes, el incremento es algo menor pero alcanza todavía cifras muy significativas, del orden del 600 %.

La segunda de las conclusiones es que, en contra de lo cabría esperar, el cuerpo más eficiente resulta ser $GF(2^{16})$ en lugar de $GF(2^{32})$, aunque éste último proporciona un mayor número de bits por cada salida.

La explicación de este fenómeno radica en el hecho de que conforme aumenta la dimensión del cuerpo las operaciones matemáticas involucradas se vuelven cada vez más costosas. Cuando el tamaño de los elementos del cuerpo alcanza los 32 bits, el coste de cada operación comienza a superar el beneficio obtenido y, por tanto, decrece su rendimiento respecto a $GF(2^{16})$.

De acuerdo a los datos, puede concluirse, finalmente, que el cuerpo extendido más eficiente para la implementación de LFSRs en software es $GF(2^{16})$.

Técnicas de implementación: factores β y γ El uso de cuerpos extendidos con el fin de aumentar el rendimiento no excluye, por supuesto, la utilización de otras técnicas de mejora. En esta sección se analiza el impacto en el rendimiento debido exclusivamente al uso de estas técnicas y, finalmente, al uso combinado de cuerpos extendidos y técnicas de mejora, que se medirán con los factores β y γ respectivamente.

La primera de las técnicas, los búfers circulares, obtiene un factor de mejora β máximo de 7.7 para el cuerpo $GF(2^{16})$. Las técnicas de ventanas deslizantes y desdoblamiento de bucles incrementan este valor hasta 8.25 y 10.15, respectivamente. Estos valores, junto con los correspondientes para el resto de combinaciones de técnica-cuerpo, pueden encontrarse en el Cuadro III.8.

Sin embargo, se comprueba cómo el aumento del tamaño del cuerpo y el coste de las operaciones implicadas hace que las técnicas de mejora tengan cada vez menos efecto en el rendimiento, como puede observase claramente en la Figura III.10 (a). Así, en $GF(2)$, la técnica de desdoblamiento de bucles obtiene un factor β de casi 9, mientras que la misma técnica, en $GF(2^{32})$, apenas tiene una mejora de $\beta = 1.28$ (22 %) respecto a no utilizar dicha técnica. En general ninguna de las técnicas supera un factor de mejora de 2 para ninguno de los cuerpos extendidos.

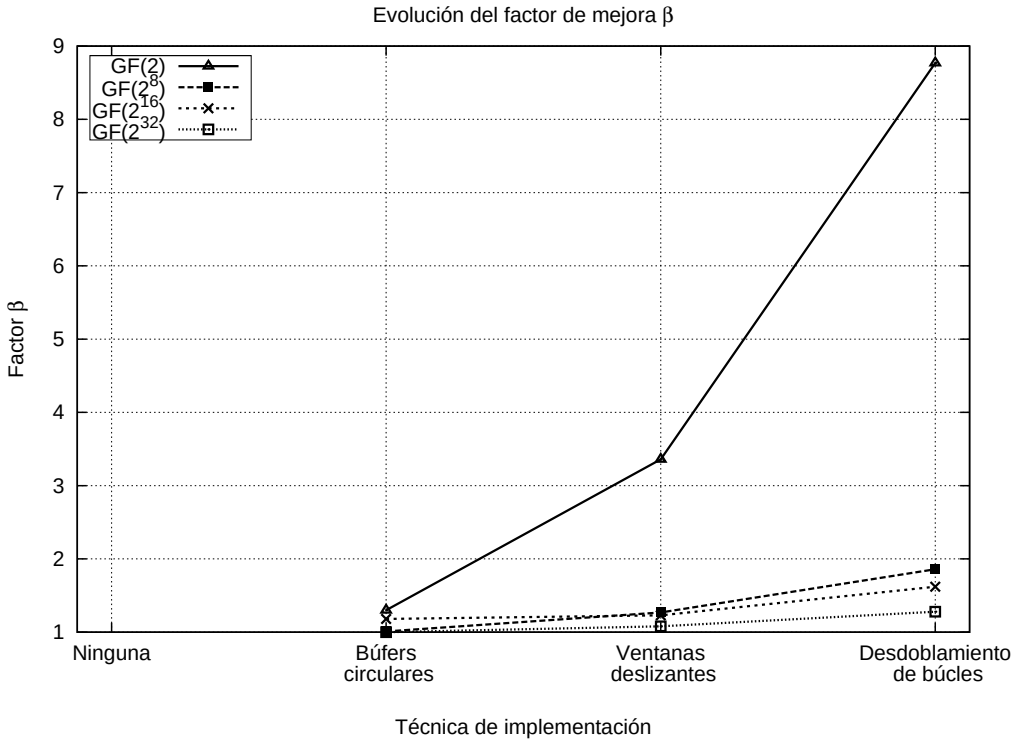
Podemos concluir, por tanto, que las técnicas de mejora son muchas más efectivas en $GF(2)$ y que van perdiendo importancia conforme aumenta el tamaño del cuerpo. Por esta razón, en $GF(2^{32})$ es posible que no compense asumir el coste de implementación de ninguna técnica de mejora, dado que el máximo beneficio obtenible será de 1.2 o de un 21 %, que se obtiene con el desdoblamiento de bucles.

[5] La relación entre el factor de mejora α y la misma mejora expresada en porcentaje es $\% = (\alpha - 1) \times 100$. Así, un factor α=2 implica una mejora del $(2 - 1)100 = 100\,\%$.

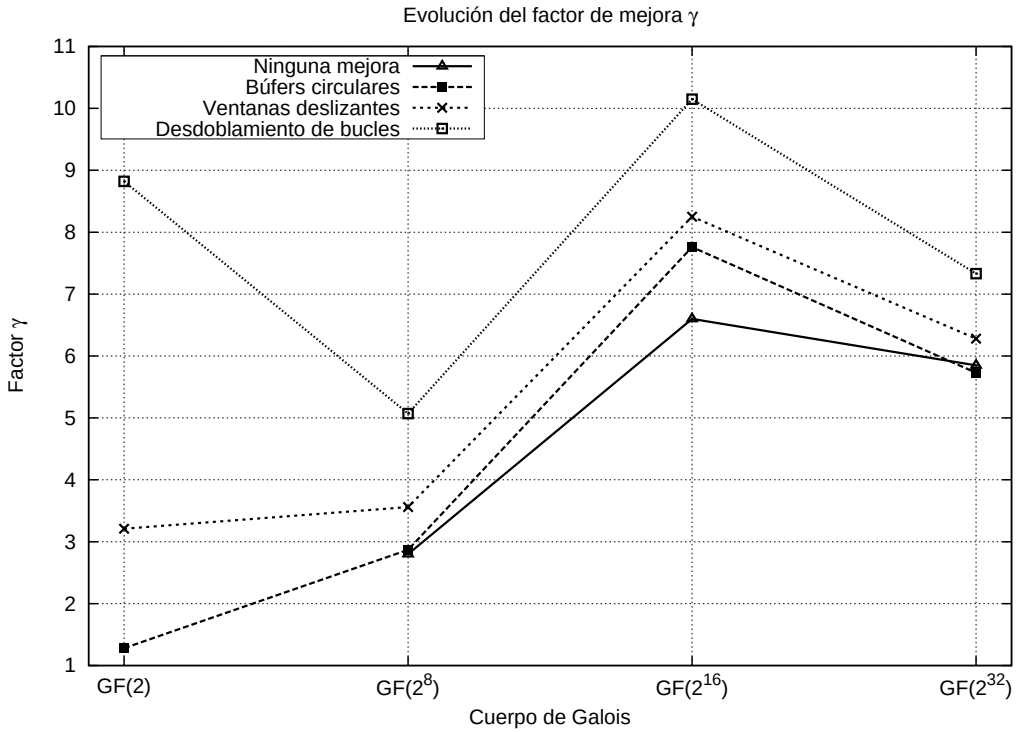
	$GF(2)$		
Arquitectura	Segundos	Rendimiento (Mbytes/s)	Factor de mejora α
ATmega168	4450	0.002	—
Pentium III	24.37	0.41	—
Dual Core	1.58	6.32	—
Athlon	1.14	8.77	—
	$GF(2^8)$		
Arquitectura	Segundos	Rendimiento (Mbytes/s)	Factor de mejora α
ATmega168	1650	0.006	3
Pentium III	3.89	2.57	6.25
Dual Core	0.56	21.27	2.80
Athlon	0.59	16.94	1.93
	$GF(2^{16})$		
Arquitectura	Segundos	Rendimiento (Mbytes/s)	Factor de mejora α
ATmega168	—	—	—
Pentium III	1.49	6.71	16.35
Dual Core	0.24	41.66	6.60
Athlon	0.33	30.30	3.45
	$GF(2^{32})$		
Arquitectura	Segundos	Rendimiento (Mbytes/s)	Factor de mejora α
ATmega168	—	—	—
Pentium III	1.79	5.58	13.57
Dual Core	0.27	37.03	5.85
Athlon	0.37	27.02	3.08

Cuadro III.7: Resultados numéricos del factor α. Este factor mide el incremento en el rendimiento producido por el uso de cuerpos extendidos.

Los resultados numéricos de los experimentos realizados concluyen, por tanto, que la forma más eficiente de implementar registros de desplazamiento en software es trabajar en el cuerpo extendido $GF(2^{16})$, en lugar del cuerpo binario tradicional, y utilizar la técnica de implementación de desdoblamiento de bucles, como muestra la Figura III.10 (b). De esta forma es posible conseguir mejoras en el rendimiento espectaculares, con un factor de mejora γ de 10.15, que supone un incremento de más del 900 %.



(a) Evolución del factor β en función de la técnica de mejora utilizada.



(b) Evolución del factor γ en función del tamaño del cuerpo extendido utilizado.

Figura III.10: Evolución de los factores β y γ en la arquitectura Athlon

Técnica de mejora		Cuerpo			
		$GF(2)$	$GF(2^8)$	$GF(2^{16})$	$GF(2^{32})$
Ninguna	Tasa de salida (MB/s)	6.32	21.27	41.66	37.03
	Factor de mejora β	-	-	-	-
	Factor de mejora γ	-	2.80	6.60	5.74
Búfers circulares	Tasa de salida (MB/s)	9.70	22.72	58.82	43.47
	Factor de mejora β	1.30	1.01	1.18	1.00
	Factor de mejora γ	1.28	2.87	7.76	5.73
Ventanas deslizantes	Tasa de salida (MB/s)	24.39	27.02	62.5	47.61
	Factor de mejora β	3.36	1.27	1.23	1.08
	Factor de mejora γ	3.21	3.56	8.25	6.28
Desdoblamiento de bucles	Tasa de salida (MB/s)	66.67	38.46	76.92	55.55
	Factor de mejora β	8.87	1.86	1.62	1.28
	Factor de mejora γ	8.82	5.07	10.15	7.33

Cuadro III.8: Resultados numéricos: factores β y γ. El factor γ representa el incremento en rendimiento máximo que, finalmente, puede obtenerse con los métodos introducidos en este trabajo de tesis.

III.7. TESTS ESTADÍSTICOS

Una vez analizado el rendimiento de los nuevos registros de desplazamiento, en esta sección comprobaremos que las excelentes propiedades estadísticas que caracterizan a los tradicionales registros definidos en $GF(2)$ se mantienen en $GF(2^n)$. Para ello se ha aplicado una exhaustiva batería de pruebas, pertenecientes a una serie de tests estadísticos ampliamente reconocidos y utilizados.

Debido a que es imposible proporcionar una prueba matemática concluyente de que un generador funciona realmente de forma aleatoria, estos tests ayudan a detectar ciertas debilidades que éste podría presentar.

Metodología A grandes rasgos, este proceso se lleva a cabo tomando varias secuencias proporcionadas por el generador y convirtiéndolas en entrada de los distintos tests, cada uno de los cuales está diseñado para comprobar si cierto aspecto que una secuencia aleatoria sí exhibiría está presente en la secuencia de entrada. Por ejemplo, uno de los aspectos básicos de una secuencia binaria podría ser que el número de 0's y 1's fuera aproximadamente el mismo.

Los tests existentes en la comunidad científica se han ido generalizando con el tiempo, incluyendo cada vez más pruebas y abarcando más aspectos de las secuencias analizadas. Para incluir también una perspectiva histórica, estos tests serán presentados de forma cronológica y, por tanto, de menor a mayor complejidad y efectividad, como puede observarse en la Figura III.11.

El estudio comienza con una breve exposición de los conceptos estadísticos básicos necesarios para comprender el funcionamiento y las conclusiones de los tests posteriores. A continuación se desarrollará una descripción de los *postulados de Golomb*, que fueron el primer intento de establecer una serie de condiciones necesarias para medir la bondad de una secuencia pseudoaleatoria periódica, para después presentar algunos de los tests estadísticos básicos, como el test de frecuencias, el de series o el de rachas, entre otros.

En los apartados III.7.4, III.7.5 y III.7.6 se describen elconjunto de tests utilizados y, finalmente, en la sección III.7.7 se presenta un resumen de los resultados obtenidos.

III.7.1. Fundamentos estadísticos

III.3 Definición. (Variable aleatoria continua) Se dice que X es una *variable aleatoria continua* si es el resultado de un experimento y puede tomar el valor de cualquier número real. Una *función de densidad de probabilidad* de una variable aleatoria continua X es una función $f(x)$ que puede ser integrada y cumple:

1. $f(x) \geq 0$ para todo $x \in \mathbb{R}$.
2. $\int_{-\infty}^{\infty} f(x)dx = 1$
3. Para todo $a, b \in \mathbb{R}$, se verifica que $P(a < X \leq b) = \int_a^b f(x)dx$.

III.4 Definición. (Distribución normal) Una variable aleatoria X tiene una *de media* μ *y varianza* σ^2 si su función de densidad de probabilidad está definida por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}}, \quad -\infty < x < \infty$$

Así, se dice que X es $\mathcal{N}(\mu, \sigma)$. Si X es $\mathcal{N}(0, 1)$, entonces se dice que X tiene una *distribución normal estándar*.

La distribución normal posee ciertas propiedades importantes que conviene destacar:

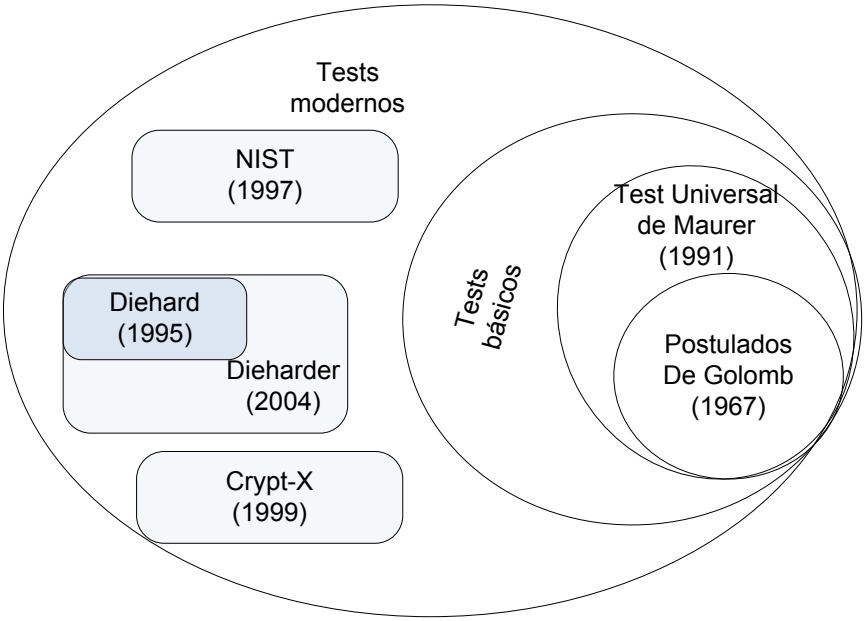


Figura III.11: Tests estadísticos

1. Tiene una única moda, que coincide con su media y su mediana.
2. La curva normal es asintótica al eje de abscisas. Por ello, cualquier valor entre $-\infty$ y ∞ es teóricamente posible. El área total bajo la curva es, por tanto, igual a 1.
3. Es simétrica con respecto a su media. Según esto, para este tipo de variables existe una probabilidad de un 50 % de observar un dato mayor que la media, y un 50 % de observar un dato menor.
4. La distancia entre la línea trazada en la media y el punto de inflexión de la curva es igual a una desviación típica. Cuanto mayor sea ésta, más aplanada será la curva de la densidad.
5. El área bajo la curva comprendido entre los valores situados aproximadamente a dos desviaciones estándar de la media es igual a 0.95. En concreto, existe un 95 % de posibilidades de observar un valor comprendido en ese intervalo.
6. La forma de la campana de Gauss depende de los parámetros μ y σ (véase Figura III.12). La media indica la posición de la campana, de modo que para diferentes valores la gráfica se desplaza a lo largo del eje horizontal. Por otra parte, la desviación estándar determina el grado de apuntamiento de la curva. Cuanto mayor sea el valor de σ, más se dispersarán los datos en torno a la media y la curva será más plana. Un valor pequeño de este parámetro indica, por tanto, una gran probabilidad de obtener datos cercanos al valor medio de la distribución.

Como se deduce de este último apartado, no existe una única distribución normal, sino una familia de distribuciones con una forma común, diferenciadas por los valores de su media y su varianza. De entre todas ellas, la más utilizada es la distribución normal estándar, que corresponde a una distribución de media 0 y varianza 1.

III.1 Proposición. *Si la variable aleatoria X es $\mathcal{N}(\mu, \sigma)$*, entonces la variable $Z = (X - \mu)/\sigma$ es $\mathcal{N}(0, 1)$.

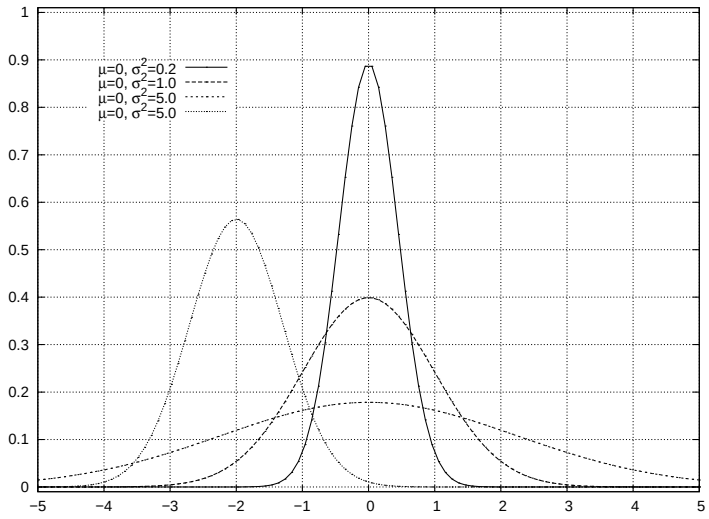


Figura III.12: Distribución normal, con diferentes medias y varianzas.

III.5 Definición. (Distribución χ^2) Una variable aleatoria X tiene una distribución χ^2 con v grados de libertad si su función de densidad de probabilidad está definida por:

$$f(x) = \begin{cases} \frac{1}{\Gamma(v/2)2^{v/2}} x^{\frac{v}{2}-1} e^{-\frac{x}{2}} & , \ 0 \leq x < \infty \\ 0 & x < 0 \end{cases}$$

donde Γ es la función gamma de Euler. Esta función se define de la siguiente forma:

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx, \qquad t > 0$$

La media y la varianza de esta distribución son $\mu = v$ y $\sigma^2 = 2v$, respectivamente.

III.7.1.1. Contraste de hipótesis

El *contraste de hipótesis*, también denominado test de hipótesis o prueba de significación, es una técnica de inferencia estadística para juzgar si una propiedad que se supone cumple una población estadística es compatible con lo observado en una muestra de dicha población.

En nuestro caso, la propiedad que se desea comprobar es la aleatoriedad de las secuencias producidas por el generador propuesto en esta tesis.

III.6 Definición. (Hipótesis estadística) Una *hipótesis estadística* H es una afirmación sobre la distribución de una o más variables aleatorias.

III.7 Definición. (Contraste de hipótesis) Un *contraste de hipótesis* es un procedimiento, basado en los valores observados de dicha variable, que lleva a la aceptación o rechazo de la hipótesis planteada H. Es importante señalar que el test sólo proporciona una medida de la fuerza de la evidencia dada por los datos contra la hipótesis, por lo que la conclusión es de naturaleza probabilística, no definitiva.

Por último, el *nivel de significación α del contraste de hipótesis H* es la probabilidad de rechazar la hipótesis H cuando ésta es cierta.

La hipótesis que se desea comprobar se denomina *hipótesis nula*[6], y suele denotarse como H_0. El nombre de "nula" indica que H_0 representa la hipótesis que se presupone cierta a menos que los datos indiquen su falsedad, y puede entenderse por tanto en el sentido de "neutra".

Como se ha comentado, la hipótesis H_0 nunca puede ser considera absolutamente probada, aunque sí puede ser fácilmente rechazada por los datos. Por ejemplo, la hipótesis de que dos poblaciones tienen la misma media podría ser rechazada cuando ambas difirieran mucho, pero no podría ser "demostrada" mediante muestreo, puesto que siempre cabría la posibilidad de que las medias difirieran en una cantidad lo suficientemente pequeña para que no pudiera ser detectada, aunque la muestra fuera muy grande.

De manera explícita o implícita, la hipótesis nula se enfrenta a otra denominada *hipótesis alternativa* y que se denota H_1. En los casos en los que no se especifica H_1 de manera explícita puede considerarse definida implícitamente como "H_0 es falsa".

Desde el punto de vista de esta tesis, *H_0 es la hipótesis de que el LFSR planteado, definido sobre $GF(2^n)$, produce secuencias aleatorias*. H_1 es, por tanto, la hipótesis de que el generador no produce dichas secuencias y no puede considerarse apto para su uso en aplicaciones criptográficas.

III.5 Ejemplo. *En la distribución $B(1;p)$ el campo de variación del parámetro p es el intervalo (0,1). Una hipótesis nula puede ser la pertenencia de p al intervalo $\Theta_0 = (0, 0.3]$ y la alternativa a $\Theta_1 = (0.3, 1)$, es decir, $H_0 = \{0 < p \leq 0.3\}$, $H_1 = \{0.3 < p < 1\}$.*

Si los subconjuntos Θ_0 y Θ_1 se componen de un único elemento (θ_0 y θ_1, respectivamente) las hipótesis correspondientes se denominan *simples* y, en caso contrario, *compuestas*. En la hipótesis simple, la distribución de probabilidad queda perfectamente determinada y es única.

III.7.1.2. Nivel de significación

La solución dada al problema del contraste de dos hipótesis implica la posibilidad de acertar o fracasar en la elección, al no saber con certeza cuál es la verdadera. La situación queda reflejada en el siguiente cuadro:

		Decisión	
		Aceptar H_0	Rechazar H_0
Hipótesis cierta	H_0	Correcta	Errónea
	H_1	Errónea	Correcta

Expresado de otra forma:

- Si la hipótesis nula es cierta y se acepta, la decisión es, evidentemente, correcta.
- Si la hipótesis nula es cierta y se rechaza, la decisión es errónea y se comete un *error de tipo I*, o de primera especie.
- Si la hipótesis alternativa es cierta y se acepta, la decisión es correcta.

[6] El término 'hipótesis nula' fue introducido por Fisher en su exposición sobre el caso de la dama y el té con leche para representar la hipótesis defendida por este investigador: la posibilidad nula de que la dama pudiera distinguir el orden en el que se vertieron el té y la leche.

- Si la hipótesis alternativa es cierta y se rechaza, la decisión es, de nuevo, errónea y se comete un *error de tipo II*, o de segunda especie.

La probabilidad de cometer el error de tipo I se llama *nivel de significación* del contraste, o tamaño de la región crítica, y se denota por α.

Por otro lado, la probabilidad de cometer el error de tipo II no tiene un nombre particular, aunque se suele representar por β. Habitualmente se utiliza su complementario a la unidad (1-β), que sí recibe el nombre de *potencia del contraste*, y corresponde a la probabilidad de rechazar la hipótesis nula siendo falsa (o, equivalentemente, aceptar la hipótesis alternativa siendo cierta).

Es importante señalar que si el nivel de significación del contraste de hipótesis es demasiado elevado, existe entonces el peligro de que puedan aceptarse secuencias que no tienen realmente las características deseadas. Por otro lado, si el nivel de significación es demasiado bajo, el resultado sería que podrían rechazarse secuencias perfectamente válidas (incluso aquellas producidas por un generador realmente aleatorio, como los basados en fenómenos físicos). En general, *los niveles utilizados en criptografía oscilan entre 0.001 y 0.05*.

El método que seguiremos puede resumirse en los siguientes pasos:

1. Enunciar la hipótesis. Estableciendo la hipótesis nula en términos de igualdad:

$$H_0 : \theta = \theta_0$$

   Por otro lado, establecer la hipótesis alternativa, lo que puede realizarse de tres maneras, dependiendo del problema analizado:

$$H_1 : \theta \neq \theta_0, \qquad \theta < \theta_0, \qquad \theta > \theta_0$$

   En el primer caso se habla de *contraste bilateral o de dos colas*, y en los otros dos de *lateral* (derecho en el 2º caso, o izquierdo en el 3º) o *de una cola*.

2. Elegir un nivel de significación α y construir la zona de aceptación. A la zona de rechazo la llamaremos *región crítica*, y su área es el nivel de significación.

3. Elegir un estadístico de contraste[7]. Verificar la hipótesis extrayendo una muestra cuyo tamaño se ha decidido en el paso anterior y obteniendo de ella el correspondiente estadístico.

4. Decidir si el valor calculado en la muestra cae dentro de la zona de aceptación, en cuyo caso se acepta la hipótesis, o se rechaza en caso contrario.

[7] Un *estadístico* es una función que opera sobre los elementos de una muestra aleatoria. Por ejemplo, podría definirse un estadístico que midiera el número de 1's en una secuencia binaria.

III.6 Ejemplo. *Este ejemplo ilustra de forma sencilla el procedimiento anterior. Partimos de la siguiente situación: "Un estudio trata de establecer el efecto del estrés sobre la presión arterial. La hipótesis de partida es que la presión sistólica media en varones jóvenes estresados es mayor que 18 cm de Hg. Se estudia una muestra de 36 sujetos y se obtiene que la media de la presión es $\mu = 18.5$ cm de Hg, con una varianza de $\sigma = 3.6$. Con un nivel de significación de $\alpha = 0,05$, ¿qué puede decirse sobre la hipótesis de partida?".*

***Solución**: Siguiendo los pasos del procedimiento anterior, puede establecerse que:*

1. *La hipótesis nula, aquella que pretendemos rechazar, es $H_0 : \mu = 18$, y la hipótesis alternativa: $H_1 : \mu > 18$, que se trata de un contraste lateral derecho.*
2. *El nivel de significación viene dado en el enunciado del problema, y es $\alpha = 0.05$.*
3. *El estadístico que utilizaremos para el contraste es el conocido t-Student:*

$$T = \frac{\mu - H_0}{S/\sqrt{n}}$$

donde la región crítica se define como $T > t_\alpha$. Si el contraste hubiera sido lateral izquierdo, la región crítica se hubiera definido como $T <_{1-\alpha}$. En nuestro caso, el valor obtenido es $t_{(36)0.05} = 1.69$.

4. *Calculamos el valor de t en la muestra, obteniendo:*

$$\frac{18.5 - 18}{3.6/\sqrt{36}} = 0.833$$

y, como no es mayor que 1.69, la hipótesis no está en la región crítica y, por tanto, H_0 no es rechazada.

En esta tesis, el procedimiento de contraste de hipótesis se convertirá en el siguiente:

1. La hipótesis nula H_0 será que el generador propuesto produce secuencias pseudoaleatorias de calidad criptográfica.
2. El nivel de significación será de $\alpha = 0.01$, valor típico utilizado a menudo en esta clase de pruebas, en las que se pretende comprobar la validez para uso criptográfico. Nótese que se trata de un valor más exigente que los utilizados habitualmente en otras áreas científicas, donde un nivel de significación habitual se sitúa en torno a $\alpha = 0.05$.
3. La toma de muestras y el cálculo del correspondiente estadístico se hará utilizando una serie de baterías de tests de reconocido prestigioso y amplio uso, como los tests Diehard, NIST, ENT y Crypt-X.
4. La verificación y decisión final se hará en base a los resultados obtenidos en el punto anterior.

III.7.1.3. P-valores

En este punto se continúan estudiando conceptos estadísticos necesarios para la correcta interpretación de las pruebas realizadas posteriormente.

III.8 Definición. (p-valor) Se define como *p-valor* [32], también conocido como *valor crítico*, a la probabilidad, bajo la hipótesis nula, de que el azar proporcione un resutado más discrepante del que ha dado la muestra.

En este caso, "más discrepante" significa que la distribución bajo la hipótesis nula diese un valor más lejano de cumplir la hipótesis nula del que han proporcionado los datos.

Un p-valor puede entenderse también como el menor nivel de significación para el que se rechazaría la hipótesis nula, H_0. O, más formalmente, el p-valor de un experimento es una variable aleatoria tal que su distribución bajo la hipótesis nula es uniforme en el intervalo $[0, 1)$.

III.7 Ejemplo. *Consideremos que se lleva a cabo un experimento para determinar si el lanzamiento de una moneda es correcto, es decir, tiene un 50 % de probabilidades tanto de salir cara como cruz. En este caso, la hipótesis nula H_0 sería que la moneda no está trucada y que, por tanto, cualquier desviación de la media puede ser achacado al azar. Los resultados del experimento son que la moneda produce 14 caras en un total de 20 tiradas.*
El p-valor del experimento sería, por tanto, la probabilidad, que denominaremos p_0, de que una moneda perfecta produjera al menos 14 caras más la probabilidad, p_1, de que produjera 6 caras o menos. Dado que el lanzamiento de una moneda sigue una distribución binomial, la probabilidad de que se produzcan al menos 14 caras es de 0.0577. *Por simetría, $p_1 = p_0$, por lo que, finalmente, el p-valor sería* $0.0577 \times 2 = 0.1154$.

En general, la hipótesis nula sería rechazada si el p-valor es menor o igual que el nivel de significación, α. En el ejemplo anterior, el p-valor es mayor que el valor de significación típico, $\alpha = 0.05$, por lo que puede considerarse que la observación es consistente con la hipótesis y que, por tanto, que se produzcan 14 caras en 20 lanzamientos pueden achacarse a una simple casualidad. Podría decirse que la desviación observada es suficientemente pequeña y que no es "estadísticamente significativa con un nivel (de significación) del 5 %".

Un inconveniente claro de este método es que la decisión de lo que es "estadísticamente significativo" no deja de ser una elección arbitraria. El valor de $\alpha = 0.05$, es ampliamente utilizado en Estadística pero, por ejemplo, en criptografía y en particular en esta tesis, se utilizarán valores más bajos ($\alpha = 0.01$), que resultarán más exigentes con los experimentos y resultados.

III.7.1.4. Test de Kolmogorov-Smirnov

Parece lógico que cada uno de estos métodos se complemente con procedimientos de análisis que cuantifiquen de un modo más exacto las desviaciones de la distribución normal. Existen distintos tests estadísticos que podemos utilizar para este propósito. Uno de los más extendidos en la práctica es el llamado test de *Kolmogorov-Smirnov*.

Éste test se basa en la idea de comparar la función de distribución acumulada de los datos observados con la de una distribución normal, midiendo la máxima distancia entre ambas curvas. Como en cualquier test de contraste de hipótesis, la hipótesis nula se rechaza cuando el valor del estadístico supera un cierto valor crítico.

Existen otros tests, como el de Anderson-Darling o el conocido χ^2, que también podrían utilizarse. Sin embargo, el test de Kolmogorov-Smirnov otorga un peso menor a las observaciones extremas y por la tanto es menos sensible a las desviaciones que normalmente se producen en estos tramos.

El procedimiento de aplicación de este test es el siguiente:

1. Obtener la función de distribución empírica. Para ello se calcula:

$$F_n(x_i) = \frac{N_i}{n},$$

donde N_i es la frecuencia absoluta acumulada, es decir, el número de observaciones acumuladas menores o iguales a x_i, $N_i = n_1 + \ldots + n_i$. Si la hipótesis nula es que

la muestra procede de una población con función de distribución $F(x)$, las diferencias entre $F_n(x)$ y $F(x)$, para una muestra de tamaño suficientemente grande, es de esperar que no sean significativas.
2. Se define el estadístico D, que se calcula como la mayor discrepancia vertical entre $F_n(x)$ y $F(x)$, es decir: $D_n = max \mid F_n(x_i) - F(x_i) \mid$.
3. Se fija el valor de significación, α.
4. Si $D_n < D_{n,\alpha}$ se acepta la hipótesis. Para un número de observaciones grande ($n > 100$) el valor crítico de $D_{n,\alpha}$ se puede aproximar por:

$$\sqrt{-ln(\alpha/2)/2n}$$

El estadístico de Kolmogorov-Smirnov es el elegido por la batería de tests utilizada en esta trabajo de tesis.

III.7.1.5. Aleatoriedad

Los tests estadísticos que se analizarán a continuación tienen como finalidad comprobar la aleatoriedad de las secuencias producidas por la nueva estructura de LFSR propuesta en este trabajo de tesis. Dicha aleatoriedad se compone, en esencia, de los siguientes aspectos:

- *Uniformidad*: En cualquier punto del proceso de generación de la secuencia, la probabilidad de aparición de un 0 debe ser igual a la de un 1; esto es, la probabilidad de cada uno debe ser 1/2. Por tanto, el número esperado de ceros (o unos) es $n/2$, donde n es la longitud de la secuencia considerada.

- *Escalabilidad*: Cualquier test aplicable a una secuencia debe ser aplicable también a cualquier subsecuencia de la misma. De esta forma, si una secuencia es aleatoria, cualquier subsecuencia debe serlo también.

- *Consistencia*: El comportamiento de un generador de secuencia debe ser consistente, es decir, no debe estar influido por el uso de diferentes valores iniciales o *semillas*. Por tanto, no es adecuado comprobar el comportamiento de un PRNG haciendo uso de una única semilla.

III.7.1.6. Resumen de conceptos

En el Cuadro III.9 se resumen algunos de los conceptos presentados, esenciales para interpretar correctamente los resultados de los tests estadísticos que serán presentados en las siguientes secciones.

III.7.2. **Postulados de Golomb**

Solomon W. Golomb (1932-) es un ingeniero y matemático norteamericano[8] que formuló las primeras condiciones necesarias, que no suficientes, que debe cumplir toda secuencia pseudoaleatoria para ser considerada de calidad. Estas condiciones, conocidas como *postulados de Golomb*, fueron formuladas en 1967 en su famoso libro *Shift Regiters Sequences* [3].

[8] A modo de curiosidad, es interesante señalar que Golomb fue uno de las primeras personalidades académicas relevantes en realizar el test de inteligencia Mega IQ, diseñado por Ronald K. Hoeflin. Su puntación en el test, considerado el más elitista del mundo, fue de 176, lo que supone una inteligencia sólo poseída por una de cada millón de personas. Un rápido cálculo permite estimar que, estadísticamente, sólo unas 6 000 personas en todo el mundo eran tan inteligentes como Golomb en el momento en el que éste realizó el test.

Test estadístico	Función que actúa sobre unos datos (una cadena binaria en nuestro caso), y que se utiliza para decidir si rechazar o no la hipótesis nula, H_0.
Hipótesis nula, H_0	Una afirmación sobre determinada condición o propiedad de la secuencia. Para los propósitos de este trabajo de tesis, la hipótesis nula H_0 sostiene que la secuencia en cuestión es aleatoria.
Test Kolmogorov-Smirnov	Un estadístico que se utilizará para determinar si un conjunto de datos se ajusta a cierta distribución de probabilidad.
Nivel de significación, α	La probabilidad de rechazar erróneamente la hipótesis nula; o lo que es lo mismo, la probabilidad de concluir que H_0 es falsa cuando es, en realidad, verdadera. Otros términos para referirse a este concepto son error tipo I o error α.
p-valor	Probabilidad, bajo la hipótesis nula, de que el azar proporcione un resultado más discrepante del que ha dado la muestra.
Entropía	Una medida del "desorden" de un sistema cerrado. Aplicada a una cierta cantidad de información, la entropía asociada a una variable aleatoria X con probabilidades $p_i, ..., p_n$ se define como $H(X) = \sum_{i=1}^{n} p_i log p_i$

Cuadro III.9: Resumen de los conceptos estadísticos más importantes.

Antes de describirlos, se introducen una serie de conceptos necesarios para su comprensión.

III.9 Definición. Sea $s = s_0, s_1, \ldots$ una secuencia infinita. La subsecuencia de los primeros n términos de s se denota por $s^n = s_0, \ldots, s_{n-1}$.

III.10 Definición. La secuencia $s = s_0, s_1, \ldots, s_n$ se denomina *periódica de periodo N* si $s_i = s_{i+N}$ para todo $i \geq 0$. Si s es una secuencia de periodo N, entonces el ciclo de s es la subsecuencia s^N.

III.11 Definición. (Racha) Sea s una secuencia. Una *racha* de s es una subsecuencia de s consistente en la repetición de un mismo símbolo (0's o 1's en el caso de una secuencia binaria). En este caso, una racha de 0's se denomina *hueco* y una racha de 1's se llama *bloque*.

III.12 Definición. (Autocorrelación) Sea $s = s_0, \ldots, s_n$ una secuencia periódica de periodo N. La función de autocorrelación de s es la función $C(t)$ definida en el dominio de los enteros como:

$$C(t) = \frac{1}{N} \sum_{i=0}^{N-1} (2s_i - 1)(2_{i+t} - 1), \qquad 0 \leq t \leq N - 1$$

La función de autocorrelación mide la similitud entre la secuencia s y ella misma desplazada t posiciones. Si s es una secuencia aleatoria de periodo N, el valor $| N \cdot C(t) |$ debería mantenerse pequeño para todos los posibles valores de t, $0 < t < N$.

III.13 Definición. (Postulados de Golomb) Sea s una secuencia periódica de periodo N. Los postulados de aleatoriedad de Golomb establecen que:

1. En el ciclo s^Nde s, el número de 1's y 0's pueden diferir, como máximo, en uno.
2. En un ciclo s^N de s, al menos la mitad de las rachas deben tener longitud 1, al menos la cuarta parte longitud 2, al menos la octava parte longitud 3, etc... Más aún, para cada una de estas longitudes, el número de huecos y bloques debería estar equilibrado.
3. La función de autocorrelación $C(t)$ debe ser bivaluada y cumplir que, para algún entero K:

$$N{\cdot}C(t) = \sum_{i=0}^{N-1}(2s_i - 1)(2s_{i+t} - 1) = \begin{cases} N, & t = 0 \\ K, & 1 \leq t \leq N-1 \end{cases}$$

III.14 Definición. (PN-secuencia) Una secuencia binaria que cumpla los postulados de Golomb se denomina *pn-secuencia*.

III.8 Ejemplo. *Considérese la secuencia periódica s de periodo* $N = 15$ *y ciclo:*

$$s^{15} = \{0,1,1,0,0,1,0,0,0,1,1,1,1,0,1\}$$

Puede comprobarse fácilmente que cumple los postulados de Golomb anteriormente citados y que, por tanto, puede considerarse una pn-secuencia:

1. *El número de 0's en* s^{15} *es 7, y el número de 1's es 8. Como difieren en uno, este postulado se cumple.*
2. s^{15} *tiene 8 rachas. Hay 4 de longitud 1 (2 huecos y 2 bloques), 2 rachas de longitud 2 (1 hueco y un bloque), una racha de longitud 3 (1 hueco) y una de longitud 4 (1 bloque).*
3. *Finalmente, la función de autocorrelación toma efectivamente dos valores:* $C(0) = 1$ *y* $C(t) = \frac{-1}{15}$ *para cada* $1 \leq t \leq 14$.

III.7.3. Tests estadísticos básicos

III.7.3.1. Test de frecuencias

El objetivo de este test es comprobar si el número de ceros y unos de una secuencia s están equilibrados, es decir, es aproximadamente el mismo. Se designa por n_0y $n_1$el número de ceros y unos respectivamente. El estadístico que se utiliza es, por tanto:

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

siendo $n = n_0 + n_1$, que sigue una distribución χ^2 con 1 grado de libertad. La aproximación es suficientemente buena si $n \geq 10$, aunque se recomiendan valores mucho mayores, como $n \gg 10000$.

III.7.3.2. Test de series

Este test sirve para determinar si el número de veces que pueden encontrarse las tuplas 00, 01, 10 y 11, como subsecuencias de s, es aproximadamente el mismo. Se designan por n_{00}, n_{01}, n_{10} y n_{11} a estos números. El estadístico utilizado en este caso es:

$$X_2 = \frac{4}{n-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{n}(n_0^2 + n_1^2) + 1$$

donde ahora se verifica que $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$. El estadístico X_2 sigue una distribución χ^2 con dos grados de libertad si $n \geq 21$.

III.7.3.3. Test del póquer

En este caso se considera un entero positivo m tal que:

$$k = \lfloor \frac{n}{m} \rfloor \geq 5 \cdot 2^m$$

A continuación se divide la secuencia s en k partes de tamaño m, y se etiqueta como n_i al número de ocurrencias del tipo i de la secuencia de longitud m, $1 \leq i \leq 2^m$. Este test determina si cada una de las secuencias de longitud m aparece aproximadamente el mismo número de veces. El estadístico utilizado en este caso es:

$$X_3 = \frac{2^m}{k} \sum_{i=1}^{2^m} n_i^2 - k$$

que sigue aproximadamente una distribución χ^2con $2^m - 1$ grados de libertad.

III.7.3.4. Test de rachas

El objetivo de este test es comprobar si el número de rachas de distintas longitudes en la secuencia s es como se espera que sea en una secuencia realmente aleatoria.

El número de huecos (o bloques) de longitud i en una secuencia aleatoria de longitud n es $e_i = (n - i + 3)/2^{i+2}$.

Se considera k igual al mayor entero i para el que $e_i \geq 5$, y se denota por B_i y H_i al número de bloques y huecos de longitud i en s, para $1 \leq i \leq k$. El estadístico utilizado es, entonces:

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(H_i - e_i)^2}{e_i}$$

que sigue, aproximadamente, una distribución χ^2 con $2k - 2$ grados de libertad.

III.7.3.5. Test de autocorrelación

Finalmente, este test comprueba las correlaciones entre s y la misma s desplazada. Para ello se considera un entero d, $1 \leq d \leq \lfloor n/2 \rfloor$. El número de bits en s que no son iguales a sus $d-$cambios es $A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$. El estadístico queda:

$$X_5 = \frac{2(A(d) - \frac{n-d}{2})}{\sqrt{n-2}}$$

que sigue, aproximadamente, una distribución $\mathcal{N}(0, 1)$, si $n - d \geq 10$.

III.7.4. **Test universal de Maurer**

Este test fue presentando en 1992 por el profesor Ueli Maurer, del *Swiss Federal Institute of Technology*, en Zurich [33, 34]. Recibe el nombre de *universal* porque es sensible a varios defectos típicos en los generadores, y que pueden no ser detectados con otros tests de forma individual. Por ejemplo, el test de Maurer es sensible a todas las características analizadas por los cinco tests básicos presentados en el apartado anterior, por lo que, en cierta manera, son un subconjunto de éste.

Por otro lado, en general, es más lento que otros algoritmos especializados, en el sentido de que necesita analizar mayor longitud de secuencia (por ejemplo, hasta 29

veces más para detectar una desviación significativa que un simple análisis de frecuencias).

Sin embargo, esto no tiene porqué ser un problema en la práctica, pues el generador debería ser capaz de proporcionar grandes cantidades de secuencia rápidamente. Por otra parte, una vez generada dicha secuencia, el test en sí mismo es muy eficiente, por lo que este requisito no supone una desventaja importante respecto a otros tests.

Concretamente, la longitud de la secuencia necesaria es n, donde n es del orden de $10 \cdot 2^L + 1000 \cdot 2^L$ con $6 \leq L \leq 16$. Los primeros $Q = 10 \cdot 2^L$ bloques de L bits sirven como bloques de inicialización, mientras que los últimos $K = \lfloor n/L \rfloor - Q$ bloques de longitud L son los que realmente se comprueban. El tamaño de Q asegura que, con una alta probabilidad, todas las posibles cadenas de L-bits están representadas en los bloques de inicialización.

El test de Maurer mide, básicamente, la entropía por bit de la secuencia de entrada, que es, según el autor, *"la forma correcta de medir la calidad de una fuente de bits de aplicación criptográfica"*. La entropía tiene relación directa con la compresibilidad, por lo que el test en esencia mide esta característica. Si una secuencia es compresible de una manera significativa, no puede ser considerada aleatoria.

El estadístico utilizado por Maurer es el siguiente:

$$X_m = \frac{1}{K} \sum_{i=Q+1}^{Q+K} log_2 t_i$$

donde t_i denota el número de índices desde la ocurrencia anterior del símbolo i-ésimo. En otras palabras, el test consiste en inspeccionar la secuencia buscando coincidencias con el bloque de test de L bits y anotando la distancia entre coincidencias.

Bajo la hipótesis nula de que el generador produce secuencias aleatorias, la esperanza de X_m viene dada por la siguiente expresión:

$$E(X_m) = 2^{-L} \sum_{i=1}^{\infty} (1 - 2^{-L})^{i-1} log_2 i$$

La varianza, σ, puede ser calculada de forma aproximada:

$$Var(X_m) = \frac{c(L,K)}{K} Var(log_2 G)$$

donde G denota una variable aleatoria distribuida geométricamente, con parámetro $1 - 2^{-L}$, y $c(L,K)$ tiene el valor aproximado:

$$c(L,K) \approx 0.7 - \frac{0.8}{L} + (1.6 + \frac{12.8}{L}) K^{-4/L}$$

Sin embargo, Coron and Naccache [35], aunque confirmaron esta aproximación, advirtieron que "la inexactitud de esta aproximación puede hacer que el test sea hasta 2.67 veces más permisivo de lo que teóricamente debería ser".

Para secuencias aleatorias, el estadístico $Z_m = (X_m - \mu)/\sigma$ debería ajustarse a una distribución $\mathcal{N}(0,1)$.

III.7.5. Diehard

Este conjunto de tests [36, 37], desarrollados y publicados en 1995 por el profesor George Marsaglia de la Universidad de Florida State, está considerado como el más importante en la comprobación de aleatoriedad.

La razón esencial es que parece presentar una mayor sensibilidad para detectar defectos estadísticos pequeños en las secuencias de bits que los tests tradicionales, como χ^2, desviaciones o los tests de correlación y entropía.

Por esta razón los tests de Diehard están considerados como *muy exigentes*. De hecho, varios generadores comerciales de números pseudo-aleatorios, que se venden bajo el reclamo de *perfecta aleatoriedad*, fallan en uno o varios de los tests de Diehard. Por otro lado existen tipos completos de generadores, como los basados en congruencias lineales (a menudo utilizados en software), que tampoco pasan el test de Diehard.

El test de Diehard original estaba compuesto por una batería de 15 pruebas individuales, aunque este número ha ido incrementándose hasta las 18 actuales.

Como una evolución de la suite Diehard, puede encontrarse el conjunto de tests *Dieharder*, escrito por Robert G. Brown del departamento de Física de la Universidad de Duke [38].

En palabras del propio autor, "*Dieharder trata de englobar todos los tests de aleatoriedad existentes, tales como el propio Dierhard, el desarrollado por el NIST FIPS o el de Donald Knuth*". Cuenta con una estructura flexible y extensible, de forma que resulta sencillo añadir nuevos tests al conjunto.

III.7.5.1. Batería de tests

A continuación se describen brevemente los tests más significativos que componen la batería, describiendo su funcionamiento y qué característica de la secuencia pretenden comprobar[9]:

- *Birthdays*: Elige puntos aleatorios en un intervalo grande. El espacio entre ellos debería estar distribuido de acuerdo a una distribución de Poisson. El nombre de la prueba se basa en la conocida paradoja del cumpleaños[10].

- *32x32 Binary Rank* y *6x8 Binary Rank*: Selecciona sólo algunos bits de algunos número aleatorios para formar una matriz binaria. Seguidamente, calcula el rango de la matriz y, finalmente, comprueba si los rangos de las sucesivas matrices están distribuidos uniformemente.

- *Count the 1s (stream)*: Cuenta el número de bits a 1 en cada byte de la secuencia. Luego convierte este número en "letras" para, finalmente, contar el número de veces que aparece cada palabra de cinco letras y comprobar si están uniformemente distribuidas.

- *Parking Lot*: Utilizando la secuencia a comprobar, coloca círculos de radio unidad en una rejilla 100 x 100. Si el círculo se solapa con otro, empieza de nuevo. Tras 12 000 intentos, el número de círculos correctamente *aparcados* debería seguir una distribución normal.

- *Minimum Distance (2D Spheres)*: Utilizando la secuencia a comprobar, coloca 8 000 puntos en una rejilla de 10 000 x 10 000. Calcula la distancia mínima entre cada

[9] Se ha mantenido el nombre en inglés de las pruebas, pues podrían considerarse, en cierta manera, nombres propios, por no tener interés su traducción y para facilitar la identificación de las pruebas en los resultados proporcionados por las mismas.

[10] La paradoja del cumpleaños hace referencia al sorprendente resultado matemático que establece que la probabilidad de encontrar a dos personas que celebren su cumpleaños el mismo día es del 50.7 % en un grupo de tan sólo 23 personas. Por supuesto, esto no es una paradoja estricta, sino que se refiere a que el resultado contradice la intuición. La mayoría de la gente apostaría que la probabilidad es mucho más baja, y que es necesario un número mucho mayor de personas para que se alcance la probabilidad real.

pareja de puntos. El cuadrado de esta distancia debería estar exponencialmente distribuida con cierta media.

- *3D Spheres*: Distribuye de forma aleatoria 4 000 puntos en el interior de un cubo de lado 1 000. Centra una esfera en cada punto, cuyo radio es la distancia mínima a otro punto. El volumen de la esfera más pequeña debería estar exponencialmente distribuida con cierta media.
- *Squeeze*: Multiplica el número 231 por números en coma flotante en el rango $[0, 1)$ hasta alcanzar 1. Repite este proceso 100 000 veces. El número de multiplicaciones necesarias debería seguir cierta distribución conocida.
- *Sums*: Genera una secuencia larga de números en coma flotante en el rango $[0, 1)$. Suma secuencias de 100 números consecutivos. Dichas sumas deberían estar distribuidas de forma normal, con cierta media y desviación típica.
- *Runs*: Genera una secuencia larga de números en coma flotante en el rango $[0, 1)$. Cuenta las rachas de números que crecen y decrecen. El resultado debería ajustarse a cierta distribución.
- *Craps*: Juega 200 000 partidas de cara o cruz. De nuevo, el resultado debería ajustarse a cierta distribución.

III.7.5.2. Resultados

El nivel de significación α utilizado es el adecuado para aplicaciones criptográficas, que se sitúa en el rango $\alpha \in [0.001, 0.01]$, lo que da un nivel de confianza en el intervalo $[99.9\,\%, 99\,\%]$. Para interpretar correctamente los resultados, hay que tener en cuenta que la mayoría de los tests producen como resultado un $p - valor$, que debería ser uniforme en el intervalo $[0, 1)$.

La batería Dieharder, como la del NIST que se analizará en el siguiente apartado, se basan en el método de *contraste de hipótesis*, que se estudió brevemente en la sección III.7.1.1. El método concreto que utilizan puede resumirse en los siguientes pasos:

- Establecer la hipótesis nula. Por ejemplo, que la secuencia binaria bajo análisis es aleatoria.
- Calcular el estadístico.
- Calcular el *p-valor*.
- Comparar el $p - valor$ obtenido con el nivel de significación establecido. Como se ha comentado, en esta tesis utilizaremos un nivel de significación $\alpha = 0.01$. Finalmente, el test será considerado como válido si el $p - valor$ es mayor que α. En caso contrario, se determinará que la secuencia analizada no ha superado el test satisfactoriamente.

En resumen, puede considerarse que la secuencia analizada es aleatoria con un nivel de significación del 99 % si el $p - valor$ obtenido por ella es mayor que 0.01 y menor que 0.99.

Tests Diehard La batería Diehard necesita como entrada para cada uno de sus tests individuales un fichero binario con, al menos, 80 millones de bits. Teniendo esto en cuenta y para la obtención de resultados se han utilizado 10 secuencias de 100 millones de bits cada una, utilizando diferentes valores iniciales en el LFSR y promediando los valores obtenidos.

Número del test	Nombre del test	p-valor	Resultado
1	*Birthdays*	0.244780	Satisfactorio
2	*OPERM5*[11]	0.234550	Posiblemente débil
3	*32x32 Binary Rank*	0.584027	Satisfactorio
4	*6x8 Binary Rank*	0.752711	Satisfactorio
5	*Bitstream*	0.205423	Satisfactorio
6	*OPSO*	0.680878	Satisfactorio
7	*OQSO*	0.513476	Satisfactorio
8	*DNA*	0.714685	Satisfactorio
9	*Count the 1s (stream)*	0.986580	Satisfactorio
10	*Count the 1s (bytes)*	0.447169	Satisfactorio
11	*Parking Lot*	0.551586	Satisfactorio
12	*Minimum Distance (2D Spheres)*	0.698957	Satisfactorio
13	*3D Spheres (minimum distance)*	0.871600	Satisfactorio
14	*Squeeze*	0.061987	Satisfactorio
15	*Sums*	0.197436	Satisfactorio
16	*Runs*	0.273377	Satisfactorio
17	*Craps*	0.807377	Satisfactorio
18	*Marsaglia and Tsang GCD*	0.924185	Satisfactorio
19	*Marsaglia and Tsang Gorilla*	0.073101	Satisfactorio

Cuadro III.10: Resultados de la batería de tests Dierharder

Tests Dieharder Por otro lado, la batería Dieharder proporciona un total de 218 p-valores para sus 18 tests. Aunque esta cantidad puede ser, en principio, suficiente para una realizar una buena valoración de los resultados, se estimó conveniente aumentar dicha cantidad, a través de la modificación del código fuente de los tests, para que se mostraran p-valores que anteriormente no aparecían. De esta forma se consiguió elevar los datos hasta 2 000, lo que resulta, sin duda, estadísticamente más significativo.

Este hecho facilita también la comprobación del requisito de distribución uniforme para los p-valores, que puede valorarse de forma sencilla a través de una representación gráfica de estos valore para todos los tests sobre el intervalo $[0, 1)$. En el Cuadro III.10 pueden encontrarse los resultados de la ejecución del test. Puede observarse que las excelentes propiedades estadísticas de los LFSR se preservan perfectamente, dado que todos los tests proporcionan un resultado plenamente satisfactorio. En la Figura III.13 puede observarse, además, la distribución de p-valores que, como hemos comentado, debe ser uniforme en el intervalo $[0, 1)$.

[11] Este test es incluido para respetar la completitud del test original Diehard. Sin embargo, según el autor de Dieharder no debe ser usado, pues contiene errores importantes. Según sus propias palabras: *"¡Atención! Este test falla en TODOS los generadores de secuencia conocidos, incluyendo aquellos cuya fortaleza está sobradamente demostrada. ¡No utilice este test para verificar generadores!. Es muy probable que contenga errores de implementación o fallos de diseño que producen un estadístico no válido."*

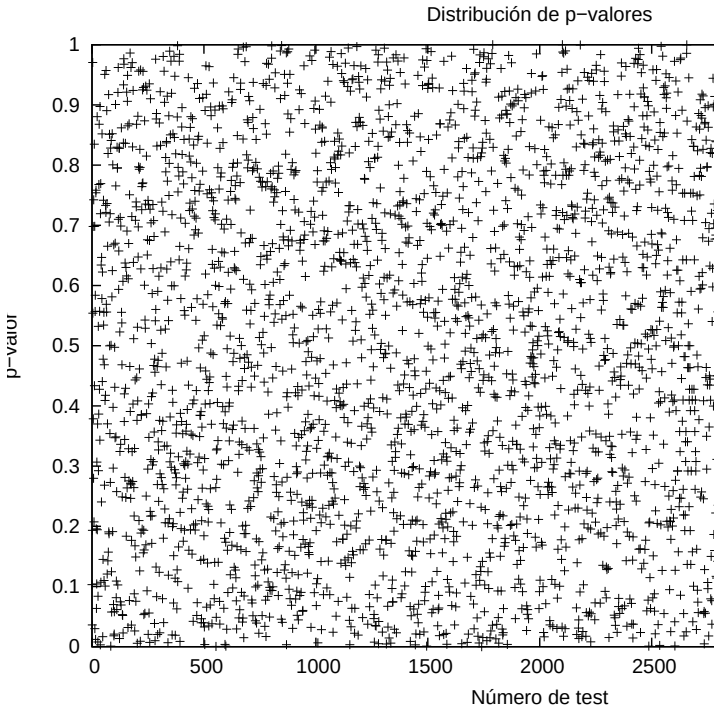


Figura III.13: Distribución de p-valores en los diferentes tests de la batería Dieharder

III.7.6. NIST

El conjunto de tests del NIST, el *National Institute of Standards and Technology*, es el resultado de la colaboración entre distintas divisiones del organismo [32]. Se empezó a elaborar en 1997 y es, sin duda, el otro gran referente en tests de aleatoriedad junto con el Diehard.

A continuación se resumen los tests del conjunto, con una breve descripción sobre su funcionalidad.

- *Frequency*: Trata de descubrir un desequilibrio entre el número de ceros y de unos.
- *Cumulative Sums*: Trata de detectar demasiados ceros o unos al inicio de la secuencia.
- *Longest Runs Of Ones*: Busca desviaciones de la distribución normal de la longitud de las secuencias de unos (denominadas *rachas*).
- *Runs*: Busca desviaciones de la distribución normal en el número de rachas.
- *Rank*: Detecta desviaciones en la distribución normal de subsecuencias periódicas.
- *Spectral*: Análisis espectral de la secuencia.
- *Non-overlapping Template Matchings*: Detecta demasiadas subsecuencias no-periódicas.
- *Overlapping Template Matchings*: Busca demasiadas ocurrencias de rachas de m-bits.

- *Maurer's Universal Statistical*: Mide la "compresibilidad" de la secuencia, según lo expuesto en la sección III.7.4.
- *Random Excursions*: Desviación significativa del número de visitas de un camino aleatorio[12]a un estado concreto.
- *Random Excursions Variant*: Desviación significativa del número de visitas, a través de múltiples caminos aleatorios, a un estado concreto.
- *Approximate Entropy*: Detección de una distribución no uniforme de palabras de longitud m. Valores pequeños de este estadístico implican una fuerte regularidad.
- *Serial*: Similar al estadístico anterior.
- *Lempel-Ziv Complexity*: Detecta si puede ser comprimida más que una secuencia realmente aleatoria.
- *Linear Complexity*: Complejidad lineal de la secuencia.

En esta batería de tests, el resultado proporcionado por cada uno de ellos es también uno o más p-valores. Estos tests requieren unas secuencias de entrada de una longitud bastante elevada. Para no encontrar ningún problema en este sentido y comprobar adecuadamente la consistencia del generador, tal como se analizó en la sección III.7.1.5, se generan para estas pruebas mil secuencias, con sus correspondientes valores iniciales, de 1 millón de bits de longitud cada una.

Una vez realizados los tests, con el fin de interpretar correctamente el resultado de los mismos, se sigue las indicaciones proporcionadas por el NIST en la documentación oficial de la suite [39]. En esencia deben de verificarse dos condiciones:

- Un porcentaje mínimo de secuencias debe pasar cada test.
- Los p-valores obtenidos deben encontrarse, de nuevo, uniformemente distribuidos en el intervalo $[0, 1)$.

En el Cuadro III.11 puede observarse el resultado proporcionado por cada test individual, junto con la media y varianza de los p-valores obtenidos. Como corresponde a un test pasado satisfactoriamente, la media de estos debe estar cercana a 0.5, puesto que éstos deben estar uniformemente distribuidos en el intervalo $[0, 1)$.

Es importante también que esta distribución tenga una varianza muy pequeña, alrededor de 0.08, lo que indica que no existen un número excesivo de valores extremos, demasiado cercanos a 0 o a 1.

Condición 1: Proporción de secuencias que pasan cada test En los resultados proporcionados por esta suite estadística, cada uno de los tests incluye un valor denominado *proporción*, que hace referencia al número de secuencias que pasan satisfactoriamente el test en cuestión. La proporción se calcula dividiendo el número total de secuencias cuyo p-valor es mayor que el nivel de significación, α, por el número total de secuencias, 1000 en nuestros experimentos.

El NIST especifica un rango concreto para los valores aceptables de la proporción de secuencias válidas. Este rango se calcula utilizando la siguiente expresión:

$$p \pm 3\sqrt{p(1-p)/m}$$

donde $p = 1 - \alpha$ y m el número de secuencias comprobadas. Como ya se ha comentado, en estos experimentos se utilizaron 1000 secuencias (m=1000) con 1 millón

Número del test	Nombre del test	Media de p-valores	Varianza de p-valores	Resultado
1	*Frequency*	0.5026	0.0807	Satisfactorio
2	*Cumulative Sums*	0.5095	0.0828	Satisfactorio
3	*Longest Runs Of Ones*	0.5026	0.0823	Satisfactorio
4	*Runs*	0.5012	0.0859	Satisfactorio
5	*Rank*	0.4957	0.0809	Satisfactorio
6	*Spectral*	0.5978	0.0657	Satisfactorio
7	*Non-overlapping Template Matchings*	0.5018	0.0830	Satisfactorio
8	*Overlapping Template Matchings*	0.4975	0.08198	Satisfactorio
9	*Maurer's Universal Statistical*	-	-	Satisfactorio
10	*Random Excursions*	0.4966	0.03553	Satisfactorio
11	*Random Excursions Variant*	-	-	Satisfactorio
12	*Approximate Entropy*	0.4673	0.0817	Satisfactorio
13	*Serial*	0.5004	0.0843	Satisfactorio
14	*Lempel-Ziv Complexity*	0.4889	0.0784	Satisfactorio
15	*Linear Complexity*	0.5148	0.0811	Satisfactorio

Cuadro III.11: Resultados de la batería de tests NIST

de bits por secuencia. Según la fórmula anterior, el rango de proporción aceptable es $[0.9805, 0.9994]$. En la Figura III.14 puede observarse la proporción de secuencias válidas para cada uno de los 15 tests. Como todos los valores están incluidos en el rango especificado, es posible concluir que las secuencias generadas por este LFSR extendido pueden considerarse aleatorias.

Condición 2: Examen de la uniformidad de los p-valores La comprobación de la uniformidad de los p-valores en el intervalo $[0, 1)$ puede hacerse con una simple inspección visual. Por supuesto, una forma mucho más rigurosa consiste en utilizar la distribución χ^2 (véase III.7.1). Para ello, para cada test, se calcula el valor de este estadístico, que es de la forma:

$$\chi^2 = \sum \frac{(C_i - m/10)^2}{m/10}$$

donde C_i es el número de p-valores en el sub-intervalo $[(i-1)/10, i/10)$, con $i = 1, ..., 10$, y m es el número de secuencias comprobadas. Una vez obtenidos estos valores, se calcula a su vez un $p - valor_t$ para cada test con la siguiente expresión:

$$p - valor_t = igamma(9/2, \chi^2/2, 9/2)$$

donde *igamma* corresponde a la *función gamma incompleta superior*. La *función gamma*, denotada como $\Gamma(z)$, es una función que extiende el concepto de factorial a los números

Test	1	2	3	4	5	6	7	8
$p_t - valor$	0.432	0.546	0.325	0.033	0.356	0.681	0.174	0.392
Conclusión	✓	✓	✓	✓	✓	✓	✓	✓
Test	9	10	11	12	13	14	15	16
$p_t - valor$	0.277	0.186	0.447	0.303	0.035	0.184	0.714	0.107
Conclusión	✓	✓	✓	✓	✓	✓	✓	✓

Cuadro III.12: Uniformidad de los p-valores (suite NIST).

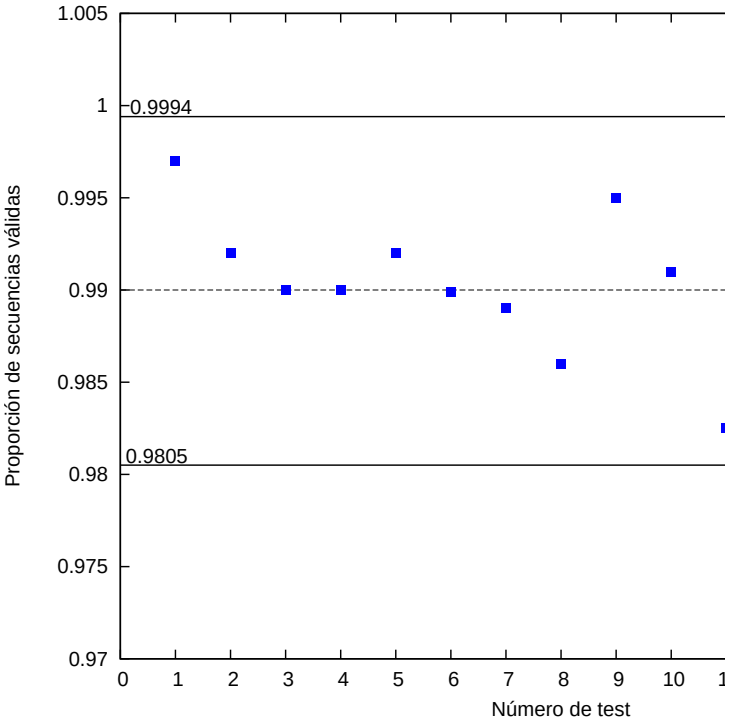


Figura III.14: Proporción de secuencias válidas en los tests NIST.

complejos. En su representación integral se define con la expresión:

$$\Gamma(z) = \int_x^\infty t^{z-1}e^{-t}dt$$

En la representación anterior, los límites de integración superior e inferior están fijados. Las funciones gamma incompleta superior $\gamma(a,x)$ e inferior $\Gamma(a,x)$ se obtienen modificando los límites de integración superior o inferior respectivamente:

$$\gamma(a,x) = \int_x^\infty t^{a-1}e^{-t}dt\Gamma(a,x) = \int_0^x t^{a-1}e^{-t}dt$$

Si cada $p - valor_t \geq 0.0001$ entonces el conjunto de p-valores inicial puede considerarse uniformemente distribuido, cumpliendo la condición impuesta, y, por tanto, la secuencia analizada resulta finalmente aleatoria. La Tabla III.12 muestra los $p - valores_t$ para cada uno de los tests, donde puede comprobarse que la distribución es uniforme.

III.7.7. Conclusiones del análisis estadístico

Como se ha ido comprobando en la presentación y análisis de cada batería de tests, el resultado final es claro. Las secuencias generadas por los nuevos registros de desplazamiento, definidos sobre $GF(2^n)$, mantienen las excelentes propiedades estadísticas de sus predecesores binarios.

De esta forma se demuestra que pueden utilizarse para el amplio abanico de sus aplicaciones tradicionales, incluidas las criptográficas. Por supuesto, nuestro interés radica en la justificación que este análisis proporciona para poder utilizar estos LFSRs extendidos como parte del nuevo cifrador en flujo que se presentará en el siguiente capítulo.

III.8. CONCLUSIONES

En este capítulo se ha propuesto el uso de cuerpos algebraicos extendidos, de dimensión $n > 2$, como base para la construcción de LFSRs mucho más eficientes. Aunque tradicionalmente estas estructuras se han definido sobre cuerpos de Galois binarios, $GF(2)$, nada impide hacerlo sobre cuerpos de dimensión superior. De esta forma, se pretende aprovechar de forma más efectiva la estructura subyacente del procesador en el que se ejecutan y obtener así implementaciones software mucho más eficientes.

En este sentido, la primera de las aportaciones realizadas en este capítulo ha sido un estudio exhaustivo de dichas implementaciones software, haciendo uso de diversas técnicas. El análisis ha sido realizado utilizando dos familias de parámetros, con el fin de obtener resultados más detallados y precisos.

La primera de las familias, denominada *microanálisis*, hace uso de parámetros como el CPI, el número de fallos en memoria caché y la tasa de utilización de los buses del sistema. La segunda familia, por otro lado, el *macroanálisis*, utiliza métricas conocidas, como el tiempo de ejecución total. Los objetivos perseguidos por este análisis son, a grandes rasgos, los siguientes:

- Evaluar con precisión el impacto del uso de cuerpos extendidos en el diseño y construcción de implementaciones software de LFSRs.

- Presentar y evaluar otras importantes técnicas de implementación, comunes en otros ámbitos de la algorítmica, pero cuyo rendimiento en LFSRs tampoco había sido evaluado.

Los resultados numéricos muestran como estas técnicas permiten obtener incrementos en el rendimiento final realmente significativos. Comparados con los métodos tradicionales, es posible conseguir factores de mejora de hasta 10.15, que supone un aumento en la eficiencia de más de un 900 %.

El estudio ha revelado también otros hechos relevantes y llamativos. El primero es que, tal y como se esperaba, el rendimiento obtenido por el LFSR aumenta conforme lo hace el tamaño del cuerpo base $GF(2^n)$ utilizado. Sin embargo, debido a diversos factores, esta mejora no es lineal, como podría indicar la intuición, sino que sufre un pequeño decremento en $GF(2^{32})$ respecto a $GF(2^{16})$.

Las razones que explican este fenómeno están relacionadas con el equilibrio entre beneficio proporcionado por el cuerpo base y el coste computacional de sus operaciones. Básicamente, cuando el tamaño del cuerpo aumenta, también lo hace el coste de sus operaciones de multiplicación y esto repercute, directamente, en la función de realimentación del LFSR, cuyo cálculo se vuelve más costoso. De esta forma, la ventaja de obtener en cada ciclo de funcionamiento del LFSR una salida de mayor tamaño, 8, 16 y 32 bits respectivamente, se pierde gradualmente. El límite, de acuerdo a los datos obtenidos, parece estar en $GF(2^{16})$ que es el cuerpo que mejor rendimiento obtuvo en los experimentos realizados.

Por tanto, a pesar de que los cuerpos extendidos han sido propuestos para su uso en LFSR, este trabajo demuestra que su uso no siempre resulta rentable. Cuerpos mayores que $GF(2^{16})$ parecen no resultar eficaces, pues el coste de sus operaciones internas es tan alto, que provoca que se pierdan sus ventajas.

Por otro lado, se ha mostrado cómo otras técnicas de mejora, como búfers circulares o desdoblamiento de bucles se adaptan perfectamente a la implementación de LFSRs, mejorando muy significativamente la tasa de generación de salida de los mismos. Los resultados concluyen, también, que estas técnicas resultan más eficaces en cuerpos de

pequeño tamaño, como $GF(2^8)$, $GF(2^{16})$, incluso $GF(2)$ y que van perdiendo eficacia para cuerpos de mayor tamaño.

Los experimentos realizados concluyen, por tanto, que *la forma más eficiente de implementar registros de desplazamiento en software es trabajar en el cuerpo extendido* $GF(2^{16})$, *combinando esta técnica con la de desdoblamiento de bucles*.

Por último, se ha llevado a cabo un estudio estadístico profundo de las secuencias generadas por estos nuevos registros de desplazamiento, con el fin de comprobar que siguen manteniendo las excelentes propiedades exhibidas por sus contrapartidas binarias. Para ello se han utilizado diversas baterías de tests estándar, utilizadas habitualmente para estos propósitos por multitud de trabajos científicos en diversos ámbitos no sólo criptográficos. Los resultados son concluyentes y muestran, sin lugar a dudas, que las secuencias generadas pueden considerarse pseudoaleatorias y son válidas, por tanto, para el gran número de aplicaciones típicas de los LFSRs.

Este último resultado justifica el uso de estos registros como parte del diseño de un nuevo cifrador en flujo, que será presentado en el siguiente capítulo.

CAPÍTULO IV

CIFRADO DE LAS COMUNICACIONES

IV.1. INTRODUCCIÓN

Como se ha analizado en los primeros capítulos de esta tesis, la protección de la privacidad de las comunicaciones de una red ad-hoc es un pilar fundamental en su seguridad, teniendo en cuenta el carácter inalámbrico de las mismas.

Este tipo de redes poseen, además, otras características peculiares, la mayoría de las cuales afectan directamente al proceso de cifrado de las comunicaciones y que, por tanto, deben ser tenidas en cuenta de forma especial:

- **Restricciones energéticas y de cómputo**: Estas restricciones, que han sido ya discutidas en detalle en el Capítulo II, afectan directamente a la capacidad de los dispositivos de utilizar ciertos esquemas de cifrado, como la clave pública, que queda fuera de las posibilidades de los mismos. Son necesarios esquemas ligeros, simples y con bajos requerimientos de memoria.

- **Comunicaciones no fiables**: Las comunicaciones en una red ad-hoc no resultan confiables, dada su naturaleza inalámbrica. Esto afecta directamente al proceso de cifrado, que debe ser diseñado de forma robusta para que pueda gestionar la pérdida de paquetes que, sin duda, se producirán en las transmisiones en mayor o menor medida.

- **Transmisión multimedia**: Debido a su inherente naturaleza de *difusión* [1], las redes ad-hoc están especialmente bien dotadas para ser utilizadas en aplicaciones de transmisión de contenido multimedia, que son uno de los servicios más demandados hoy en día.

  Esta clase de aplicaciones tienen unos altos requisitos de *calidad de servicio*, que aseguran un cierto nivel mínimo de calidad en las comunicaciones [2]. Además, estos requisitos de calidad suelen elevarse aún más cuando es necesario asegurar la confidencialidad de la información transmitida.

Estas razones convierten el diseño y desarrollo de una propuesta para garantizar la privacidad en este tipo de redes en un verdadero desafío. En una primera aproximación al problema, teniendo en cuenta los requisitos recién expuestos, podría pensarse en el uso de *cifradores en flujo* como candidatos a soluciones del problema de la privacidad en este tipo de redes.

Cifradores en flujo A pesar de todo el trabajo realizado en torno a ellos, no existe un consenso unánime en la comunidad criptológica a la hora de encontrar una definición clara y precisa de lo que se entiende por un cifrador en flujo. Un acuerdo de mínimos podría llevar a una definición común como la siguiente.

IV.1 Definición. (**Cifrador en flujo** - Definición tradicional) Un *cifrador en flujo* es un cifrador simétrico en el que los dígitos del texto en claro se cifran de uno en uno, y en el que el cifrado de dígitos consecutivos varia durante el proceso.

Aunque esta definición es correcta, no establece una distinción clara entre los cifradores en bloque y en flujo. Es cierto que esta frontera resulta cada vez más difusa debido, entre otros motivos, a ciertos modos de operación de los cifradores en bloque como el *modo contador* de AES [3]. Sin embargo, siguen existiendo rasgos diferenciadores claros como, por ejemplo, la existencia de una memoria interna en los cifradores en flujo. Podemos, en este punto, reescribir la definición anterior de una forma más precisa.

IV.2 Definición. (**Cifrador en flujo** - Definición alternativa) Un *cifrador en flujo* es un cifrador simétrico con *memoria interna* que recibe los dígitos del texto en claro, definidos sobre un alfabeto $\mathcal{A}$, de uno en uno y que, en paralelo, produce el texto cifrado sobre el mismo alfabeto.

Aunque tradicionalmente el alfabeto $\mathcal{A}$ se ha definido sobre elementos de $GF(2)$, es decir bits, nada impide hacerlo sobre estructuras de 8, 16, 32 o, incluso, 64 bits. En cuanto a su modo de operación y, a pesar de que existen ejemplos de lo contrario como Helix [4], la mayoría de los cifradores en flujo pueden clasificarse en dos grandes grupos: *síncronos* y *auto-sincronizantes*.

Muy brevemente, la diferencia principal entre ellos radica en la forma en la que el estado interno del cifrador se actualiza en cada ciclo de operación. Los cifradores síncronos lo hacen de forma *independiente* del texto en claro mientras que, por el contrario, los auto-sincronizantes utilizan parte del texto en claro ya procesado para dicha actualización.

Proyecto eSTREAM A pesar de sus innegables e inherentes ventajas, y de su larguísima tradición de uso, sobre todo en entornos militares, lo cierto es que los esquemas de cifrado en flujo parecen haber caído en un cierto estado de desuso y abandono en los últimos años. Parte de la comunidad criptológica considera que han sido eclipsados por la creciente popularidad y difusión de sus esquemas equivalentes de cifrado en bloque.

A esta situación ha contribuido, sin duda, el hecho de que exista un estándar claro, público y muy difundido de cifrado en bloque, conocido como AES [5, 6], y, por el contrario, no pueda encontrarse un equivalente en cifrado en flujo. Afortunadamente, para remediar esta situación, a principios de 2004 se puso en marcha el proyecto eSTREAM, cuyo origen puede encontrarse en una conferencia impartida por Adi Shamir en la *RSA Data Security Conference* de ese mismo año.

En ella, Shamir ponía en duda la necesidad de la continuidad de los esquemas de cifrado en flujo, dada la notoriedad, difusión y excelente rendimiento proporcionados por AES, incluso en el modo contador o CTR, que puede hacer las veces de un cifrado en flujo [3]. El mismo Shamir proponía contraargumentos a su tesis, indicando que los cifradores en flujo tendrían que demostrar su superioridad en dos áreas indiscutibles:

- En aquellos escenarios donde fuera necesario *una tasa de cifrado excepcionalmente alta* en las implementaciones software.
- Allá donde se necesite *un consumo de recursos muy bajo* en las implementaciones hardware.

El proyecto eSTREAM recibió en su primera fase 34 propuestas, distribuidas en dos grandes grupos: aquellas diseñadas con un perfil más adecuado para su implementación hardware y otras para ser utilizadas en software. No hay que olvidar que

el objetivo del proyecto era demostrar la teórica superioridad de los cifradores en flujo sobre los de bloque, al menos en aquellas áreas enunciadas por Shamir. Por este motivo, ya durante la fase preliminar del mismo sólo se admitieron aquellos algoritmos que demostraran ser más rápidos que AES funcionando en modo contador.

Finalmente el proyecto finalizó en abril de 2008, con un conjunto total de 4 cifradores ganadores aptos para su implementación software, y otros 3 orientados al hardware [7, 8].

Parece claro que, sobre la base de las conclusiones de este proyecto, los *cifradores en flujo* constituyen sin duda una excelente opción sobre la que basar el diseño de una solución al problema de la privacidad en redes ad-hoc. Particularmente, este tipo de cifradores poseen una serie de características que encajan perfectamente en el marco de los requisitos de estas redes móviles:

- Como se ha visto, los cifradores en flujo son habitualmente los esquemas de cifrado más *rápidos*. Este hecho cobra especial importancia en la transmisión multimedia, donde habitualmente se necesitan altas tasas de transferencia, entre 10kbps y 500kbps [9], y no es admisible que el cifrado se convierta en el cuello de botella del sistema.

- Son, por otro lado, conceptualmente muy *simples*. Al margen de las ventajas evidentes que esto supone, sobre todo a la hora de su implementación hardware, esta característica se traduce directamente en esquemas *energéticamente eficientes*, lo que resulta especialmente significativo en estos entornos.

Necesidad de una nueva solución A pesar de las ventajas expuestas, muchos de los algoritmos ganadores del proyecto eSTREAM, tanto en sus vertientes hardware como software, resultan demasiado complejos para ser implementados en algunos de los dispositivos de baja capacidad de cómputo considerados en este trabajo de tesis.

Por esta razón se decidió diseñar un nuevo algoritmo de cifrado en flujo, que hemos denominado *LFSRe* (LFSR extendido), basado en el uso de los registros de desplazamientos extendidos, presentados en el capítulo anterior, y los principios del *generador shrinking*. Aunque LFSRe haga uso conceptualmente de algunas ideas de éste generador, ambos cifradores cuentan con importantes diferencias. Las principales aportaciones realizadas a la idea original pueden resumirse de la siguiente forma:

- El generador shrinking original trabaja sobre el cuerpo binario $GF(2)$. Por contra, la propuesta de este trabajo utiliza los registros de desplazamientos anteriormente presentados, que se definen sobre cuerpos algebraicos extendidos $GF(2^n)$, donde n puede definirse en función del tamaño de palabra del microprocesador donde vaya a ejecutarse el cifrador.

- Como consecuencia del cambio anterior, los antiguos criterios de decimación dejan de ser válidos. Es necesario, pues, definir un conjunto de nuevas funciones, que serán descritas y analizadas detalladamente en la sección IV.3, evaluando las ventajas e inconvenientes de cada una.

- Se ha añadido un módulo de carga y gestión del *vector de inicialización*, concepto del que el diseño original carecía por lo que no podía ser utilizado en las modernas redes de conmutación de paquetes tipo Internet. Este módulo será presentado y analizado en la sección IV.5.

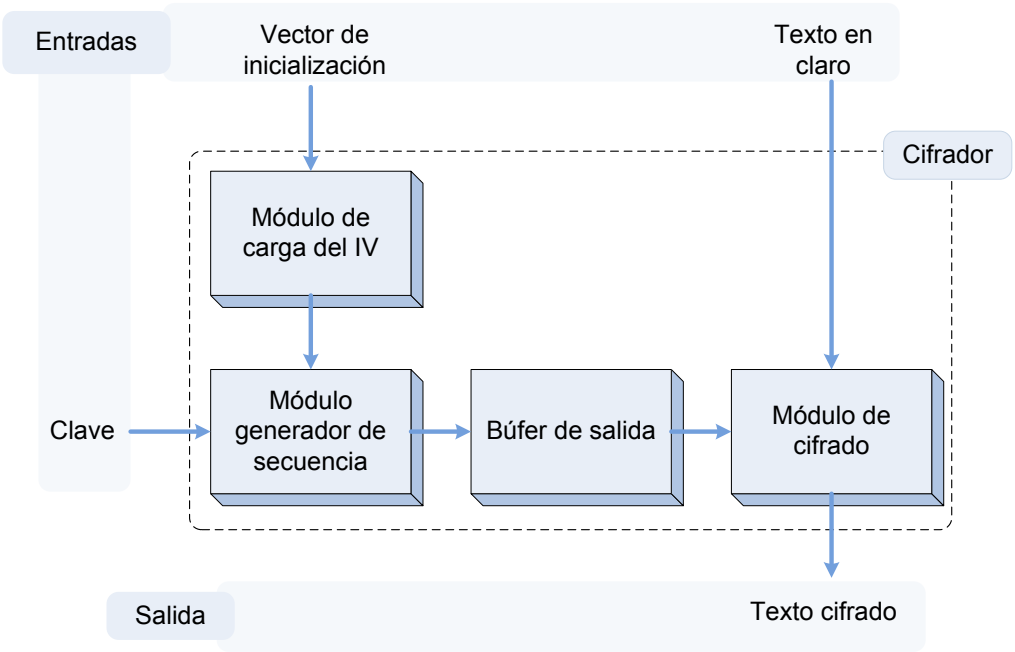


Figura IV.1: Diagrama de bloques del cifrador presentado.

- Si bien la necesidad del uso de un búfer de salida, que ocultase la salida irregular del generador shrinking, fue ya apuntada por los autores originales, las implicaciones del mismo no fueron convenientemente exploradas. En la sección IV.4 se abordará esta cuestión, analizando detalladamente todos los aspectos relevantes.

Las aportaciones anteriores se materializan finalmente en un nuevo diseño de cifrador en flujo, que a grandes rasgos, consta de cuatro bloques esquemáticos, que se resumen a continuación y en la Figura IV.1:

- El **módulo de generación de secuencia cifrante**, que utiliza nuevas funciones no lineales como criterios de decimación para generar una secuencia pseudoaleatoria de salida.
- El **módulo de cifrado** es sencillo y rápido, habitual en este tipo de cifradores, tal y como se describió en la sección anterior.
- El **búfer de salida** oculta la salida irregular del módulo de generación de secuencia. Si este paso no se realiza de forma cuidados, se abre la puerta a criptoanálisis elementales; en efecto, un atacante únicamente debe monitorizar la salida del generador para poder recuperar completa o parcialmente los contenidos de los registros que lo constituyen.
- El **módulo de carga del vector de inicialización (IV)** resulta necesario para poder utilizar un cifrador en flujo en una red cuya unidad de información transmitida o recibida no es un flujo sin un paquete de datos.

Cada uno de estos módulos será analizado detalladamente en las secciones sucesivas. Se comenzará por el *módulo de generación de secuencia cifrante*, que será descrito en la sección IV.2. A continuación en la sección IV.3, se presentarán los nuevos *criterios*

de decimación comparando sus diferentes características y eligiendo aquel que resulte óptimo.

Todos los problemas asociados con el imprescindible *búfer de salida* son presentados y resueltos en la sección IV.4 y sus subapartados. Por otro lado, el *módulo de carga y gestión del vector de inicialización* se describe en IV.5. El último de los módulos del cifrador, el *módulo de cifrado*, se presenta en la sección IV.6. Dado que éste resulta extremadamente simple, en el mismo apartado se describe también el diseño final del cifrador, donde se muestran todos los módulos constituyentes y cómo estos interaccionan entre sí.

A continuación, en la sección IV.7 puede encontrarse un *estudio comparativo del rendimiento del cifrador*, donde se incluyen otros algoritmos considerados de referencia, tanto de flujo como de bloque. Un *resumen de los resultados obtenidos*, junto con las *conclusiones* que pueden extraerse de los mismos, ponen fin al capítulo en la sección IV.8.

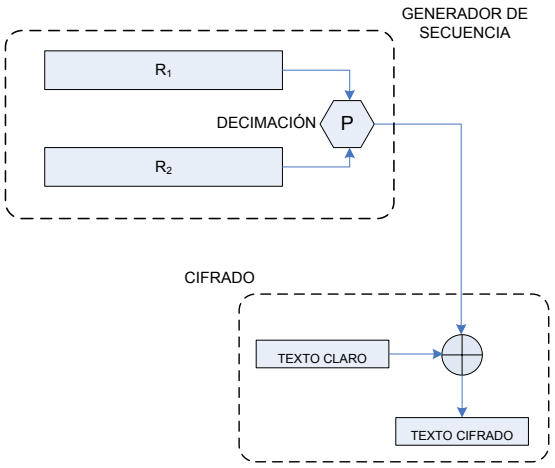


Figura IV.2: Generador shrinking tradicional, definido sobre el cuerpo binario

IV.2. MÓDULO DE GENERACIÓN DE SECUENCIA CIFRANTE

Este es el primero de los módulos que forman parte del cifrador y, sin duda, uno de los más importantes. Su objetivo es proporcionar una secuencia pseudoaleatoria de longitud arbitraria, con excelentes propiedades estadísticas, que será utilizada posteriormente en el proceso de cifrado.

Este módulo está parcialmente basado en las ideas de funcionamiento del conocido generador *shrinking*, que se describe brevemente a continuación.

IV.2.1. El generador shrinking

El generador *shrinking* es un generador de secuencia cifrante muy conocido, presentado por Coppersmith, Krawczyk, y Mansour [10] en 1994. Se trata de un generador basado en la interacción no lineal entre las salidas de dos LFSRs. A pesar de ser extremadamente simple, rápido y escalable, cuenta con una excelente resistencia al criptoanálisis.

IV.2.1.1. Funcionamiento

Su funcionamiento es, como se ha comentado, muy sencillo. El generador consta de dos registros de desplazamiento, R_1, denominado *registro de control*, que controla o decima la secuencia producida por R_2. Denotaremos con L_1 y L_2 sus respectivas longitudes y con P_1 y P_2 sus correspondientes polinomios característicos [11]. En la Figura IV.2 puede encontrarse un esquema del generador.

La secuencia producida por R_1, cuyos bits denotaremos como a_i, controla los bits de la secuencia producida por R_2, b_i, que se incluyen en la *secuencia de salida*, c_j, de acuerdo a la siguiente *regla de decimación P*:

1. SI $a_i = 1$ ENTONCES $c_j = b_i$.
2. SI $a_i = 0$ ENTONCES b_i es *descartado*.

Más formalmente, la secuencia de salida c_j está constituida por $c_j = b_{i_j}$, donde, para cada $j \geq 0$, i_j es la posición del j-ésimo bit con valor '1' en la secuencia producida por R_1.

IV.2.1.2. Propiedades estadísticas

El generador *shrinking* produce unas secuencias de salida con unas propiedades estadísticas óptimas, que satisfacen todos los criterios deseables sobre periodo de máxima longitud posible o alta complejidad lineal.

Como se demuestra en [10], el periodo de la secuencia de salida es:

$$T = (2^{L_2} - 1)\, 2^{(L_1 - 1)} \tag{IV.1}$$

siempre que los polinomios característicos P_1 y P_2 sean primitivos y se cumpla que $mcd(L_1, L_2) = 1$. Por otro lado, la complejidad lineal de la secuencia producida, denotada por LC, satisface la siguiente desigualdad:

$$L_2 2^{(L_1 - 2)} < LC \leq L_2 2^{(L_1 - 1)} \tag{IV.2}$$

Un simple cálculo, basado en el hecho de que cada estado de R_2 coincide una vez con cada estado de R_1, permite calcular fácilmente el número de 1's en la secuencia de salida. Este número es constante e igual a:

$$No.\ 1's = 2^{(L_2 - 1)}\, 2^{(L_1 - 1)} \tag{IV.3}$$

Comparando las ecuaciones anteriores IV.1 y IV.3, puede deducirse que la secuencia de salida es una secuencia cuasi-equilibrada. Además, cuenta con unas propiedades estadísticas muy deseables para su uso criptográfico [10], por lo que este generador es adecuado para uso en criptosistemas y generadores de secuencia.

IV.2.1.3. Criptoanálisis

Sorprendentemente, a pesar de su simplicidad, el generador *shrinking* ha permanecido inmune y resistente a los muchos intentos de criptonálisis realizados hasta la fecha [12, 13]. De hecho, a día de hoy, el mejor ataque conocido, que presupone el conocimiento de P_1 y P_2, aunque no el contenido inicial de los registros, necesita de aproximadamente $O(2^{L_1} \cdot L_2^3)$ operaciones. Por el contrario, si los polinomios se mantienen en secreto, la complejidad del ataque anterior crece hasta las $O(2^{2L_1} \cdot L_1 \cdot L_2)$ operaciones.

En este último supuesto, considerando que $L_1 \approx l$ y $L_2 \approx l$, el nivel de seguridad proporcionado por el generador es equivalente a 2^{2l} operaciones. Por tanto, si se consideran longitudes del orden de $L_1 \approx 64$ y $L_2 \approx 64$, el generador puede considerarse seguro contra todos los ataques conocidos hasta el momento. Ésta constituye, sin duda, otra de las razones para la elección de este esquema como base para el diseño del nuevo cifrador.

IV.2.2. **Polinomios primitivos y funciones de realimentación**

El algoritmo de cifrado que presentamos, LFSRe, es un cifrador *flexible*, debido a que hace uso de registros de desplazamiento extendidos, que pueden ser adaptados a las plataformas hardware consideradas y a los diferentes niveles de seguridad exigidos.

Configuración	Función de realimentación
P_8	$s_{t+17} = s_{t+15} \oplus C6 \otimes s_{t+2} \oplus 67 \otimes s_t$
P_{16}	$s_{t+17} = s_{t+15} \oplus 19B7 \otimes s_{t+2} \oplus 13C \otimes s_t$
P_{32}	$s_{t+17} = s_{t+15} \oplus F21DA317 \otimes s_{t+2} \oplus E28C895D \otimes s_t$

Cuadro IV.1: Funciones de realimentación utilizadas en este trabajo de tesis

Como consecuencia, LFSRe no tiene una especificación única, como suele ser el caso en los algoritmos de cifrado. No sólo puede variar la longitud de sus claves, a través de la longitud de sus registros internos, sino también el tamaño de palabra de los mismos.

LFSRe puede definirse, por tanto, utilizando diferentes *configuraciones*, que denotaremos P_8, P_{16} y P_{32}, según hagan uso de distintos cuerpos algebraicos definidos sobre $GF(2^8)$, $GF(2^{16})$ y $GF(2^{32})$ respectivamente. Si bien la seguridad del cifrador no se ve afectada, puesto que todas sus configuraciones son seguras, sí existen importantes diferencias en el rendimiento de cada una de ellas, como se analizó con detalle en el Capítulo III. Las distintas configuraciones presentadas se recogen en el Cuadro IV.1.

Teniendo en cuenta los resultados sobre rendimiento obtenidos con anterioridad, nuestro cifrador se definirá utilizando P_{16} siempre que sea posible. Sin embargo, ciertas arquitecturas, como la mayoría de dispositivos en las redes de sensores, utilizan un tamaño de palabra de 8 bits, por lo que en estos casos se deberá utilizar P_8.

Funciones de realimentación Cada una de estas configuraciones lleva asociado un polinomio primitivo de referencia, cuyas funciones de realimentación correspondientes son de la forma:

$$s_{t+17} = s_{t+15} \oplus \alpha s_{t+2} \oplus \beta s_t,$$

donde $\oplus$ denota suma sobre el cuerpo $GF(2^n)$, la multiplicación se realiza también sobre $GF(2^n)$, con $\alpha, \beta \in GF(2^n)$ y $\alpha, \beta \neq 0$. Los valores concretos de α y β, así como las funciones de realimentación escogidas, se resumen en el Cuadro IV.1.

IV.3. NUEVOS CRITERIOS DE DECIMACIÓN

Como se ha analizado a lo largo del capítulo anterior, una de las propuestas realizadas en esta tesis es sustituir los registros de desplazamientos tradicionales, R_1 y R_2, definidos sobre $GF(2)$, por equivalentes sobre $GF(2^n)$, con n=8, 16 y 32. De esta forma, la salida producida por los mismos no será binaria sino que, en cada ciclo, se obtendrán 8, 16 y 32 bits, respectivamente.

Como resulta obvio, éste es un cambio muy significativo, que afecta profundamente al funcionamiento interno del sistema. De hecho, puede considerarse que nos encontramos ante un nuevo generador, puesto que éste debe ser redefinido en su totalidad y su seguridad debe ser revisada.

Entre los aspectos más importantes que deben ser rediseñados se encuentra el *criterio o función de decimación*, pilar esencial del cifrador, y que ahora tendrá que trabajar sobre el cuerpo extendido $GF(2^n)$. A continuación se analizan todos los aspectos relacionados con el mismo.

IV.3.1. Función de decimación

Puesto que un registro de desplazamiento es una estructura completamente lineal, sus secuencias de salida son fácilmente predecibles y su criptoanálisis, por tanto, resulta trivial. En efecto, haciendo uso del algoritmo de Berlekamp-Massey un atacante sólo necesita capturar unos pocos bits de la secuencia cifrante (concretamente $2L$ bits, donde L es la longitud del registro que en este caso coincide con la complejidad lineal de la secuencia) para reconstruir el LFSR mínimo que la genera completamente.

Por tanto, un cifrador en flujo que únicamente realice una operación XOR entre la secuencia cifrante y el texto en claro a cifrar resulta vulnerable a un *ataque por texto claro conocido*. Dado que el atacante conoce los textos claros $m_1, m_2, \ldots, m_n$ correspondientes a los textos cifrados $c_1, c_2, \ldots, c_n$, los bits originales de la secuencia cifrante pueden reconstruirse, simplemente, calculando $m_i \oplus c_i, 1 \leq i \leq n$.

Puesto que la resistencia a este tipo de ataques es imprescindible en cualquier cifrador moderno, resulta necesario utilizar diversas estrategias para "romper" esta linealidad:

1. *Combinación no-lineal de la salida de varios LFSRs*: varios registros trabajan en paralelo y su salida se combina utilizando una función no-lineal f. Es el enfoque utilizado por, entre otros, el algoritmo A5/1 [14, 15]. En general, estos generadores pueden atacarse utilizando los *ataques de correlación*.
2. *Filtrado no-lineal de los contenidos de un único LFSR*: se hace uso de una función no-lineal f con n entradas, donde n es el número de etapas del LFSR, y una única salida. Normalmente estos filtros hacen uso de las denominadas *cajas S* (S-boxes), que cuentan con los mismos requisitos que las utilizadas en los cifradores en bloque. Por ejemplo, el cifrador Turing [16] hace uso de la caja S de Skipjack, un conocido cifrador en bloque, como parte de su filtrado no-lineal.
3. *Generadores de reloj controlado*: uno o varios LFSRs controlan el reloj de otros LFSRs, que son lo que realmente producen la secuencia de salida, a través de la denominada *función de decimación*. En esta categoría se incluyen, entre otros, el generador que presentamos.

Las *funciones de decimación* se definen de forma más rigurosa tal y como sigue.

IV.3 Definición. Una *función de decimación simple* F_D es una aplicación definida como $F_D : GF(2^n) \rightarrow GF(2)$. Por otro lado, una *función de decimación compuesta* F_D es una aplicación definida como $F_D : GF(2^n) \times GF(2^n) \rightarrow GF(2)$.

IV.1 Ejemplo. *Aunque, como se analizará en la siguiente sección, existen una serie de requisitos deseables para la funciones de decimación, en principio éstas pueden tomar cualquier forma en su dominio de definición.*
Por ejemplo, una función F_D podría comprobar si un número entero es impar. La definición de esta función quedaría:

$$F_D^{impar} : \mathbb{Z} \rightarrow GF(2)$$

$$F_D^{impar}(x) = \begin{cases} 1, & si\ x \bmod 2 \neq 0 \\ 0, & en\ otro\ caso \end{cases}$$

A la vista de la definición anterior, el comportamiento del generador de secuencia se puede definir de forma muy sencilla. La función F_D se irá aplicando iterativamente a cada elemento producido por R_1. Si la salida de F_D es 1, el elemento generador por R_2 en ese mismo ciclo pasa a la secuencia de salida. Si es 0, dicho elemento se descarta.

En el caso de que F_D sea compuesta y, por tanto, reciba como parámetros los elementos de R_1 y R_2 simultaneámente, el funcionamiento es análogo.

IV.3.2. Requisitos de los criterios de decimación

Dado que ha cambiado el dominio de definición de las funciones de decimación, las reglas habituales de $GF(2)$ no pueden ser directamente aplicadas a $GF(2^n)$.

Deben crearse, por tanto, nuevos criterios que, dado que van a ser utilizados para uso criptográfico, resulte imprescindible que cumplan una serie de requisitos analizados a continuación.

Equilibrados Un criterio de decimación debe producir, en término medio, *el descarte de la mitad* de los elementos de la secuencia producida por R_1. Si se desecha un número menor, el criterio se vuelve obviamente más eficiente, pero también facilita el criptoanálisis del sistema. Por el contrario, si se eliminan más de la mitad de la salida producida, se obtiene un mayor nivel de seguridad, aunque se degrada en exceso el rendimiento final.

Estables El número de elementos descartados no debe depender de los datos de entrada, de forma que la bondad del criterio no decaiga si la secuencia sobre la que se aplica no es de buena calidad estadística.

Eficientes Los criterios de decimación deben ser rápidos y eficientes en su implementación software. Concretamente, se tratará de evitar todos aquellos criterios que realicen operaciones a nivel de bit, por dos razones básicas:

- *Tamaño de los elementos*: el tamaño de los elementos del cuerpo algebraico subyacente ha cambiado, por lo que las operaciones "nativas" deberían ser a nivel de byte y superiores.
- *Eficiencia de implementación*: La unidad mínima de almacenamiento en un computador es el byte. Por esta razón, el acceso a un bit concreto es una operación costosa, puesto que implica el uso de máscaras y varias instrucciones máquina. Sin embargo, los criterios definidos a nivel de byte resultan más eficientes, puesto que aprovechan mejor el juego de instrucciones del microprocesador, haciendo uso de una única instrucción, o un número mínimo de ellas.

Por otro lado, en función de cuántos registros tengan en cuenta en sus decisiones, los criterios serán clasificados en *simples*, cuando impliquen sólo a R_1, o *compuestos*, si utilizan también a R_2.

A continuación se describirán, analizando su comportamiento y puntos fuertes y débiles, los nuevos criterios de decimación aportados en este trabajo, que se han denominado MITAD, MENOR, PAR y NO-LINEAL.

IV.3.3. Criterio MITAD

El primero de los criterios presentados, que podría considerarse la "traducción" del criterio tradicional del generador *shrinking* al nuevo diseño, es el denominado *criterio MITAD*.

Su definición, de acuerdo a la definición dada en la sección IV.3 para el cuerpo $GF(2^n)$ es la siguiente:

$$F_D^{MITAD}(x) = \begin{cases} 1, & \text{si } x < 2^{n-1} \\ 0, & \text{en otro caso} \end{cases} \quad , \quad x \in GF(2^n)$$

Sean $a_i, b_i, c_i \in GF(2^n)$ los elementos de las secuencias producidas por los registros R_1, R_2 y la secuencia de salida, respectivamente. Entonces, haciendo uso de este criterio, el funcionamiento del generador queda finalmente:

- SI $a_i < 2^{n-1}$ ENTONCES $c_i = b_i$
- SI $a_1 \geq 2^{n-1}$ ENTONCES se descarta b_i

Así, por ejemplo, para el caso de $GF(2^8)$ el criterio toma la siguiente forma:

- SI $a_i < 128$ ENTONCES $c_i = b_i$
- SI $a_i \geq 128$ ENTONCES se descarta b_i

Para los cuerpos $GF(2^{16})$ y $GF(2^{32})$, los valores umbral para el criterio serán, claramente, $2^{15} = 32\,768$ y $2^{31} = 2\,147\,483\,648$.

Como puede verse, éste puede clasificarse como un criterio *simple*, pues sólo tiene en cuenta el valor de R_1 y *equilibrado*, siempre que la secuencia producida por R_1 también lo sea. En el caso de que no lo sea, la secuencia de salida del generador no será de calidad, pudiéndose producir casos extremos, en el que todos los valores de R_1 sean mayores que 2^{n-1} y el generador descarte todos los valores y no produzca ninguna salida.

Éste, sin embargo, es un problema que no debería producirse pues los registros de desplazamiento producirán buenas secuencias, desde el punto de vista estadístico, sea cual sea el contenido inicial de sus registros (exceptuando el caso de todos los valores a cero).

Por otro lado, es también un criterio muy *eficiente*, pues trabaja a nivel de byte, medias palabras (16 bits) o palabras (32 bits) y su implementación en software puede realizarse en dos únicas instrucciones. En el Listado 9 puede encontrarse el código ensamblador correspondiente. Como puede observarse, es el criterio más eficiente posible, pues consta únicamente de dos instrucciones máquina, una de comparación y otra de salto.

Por último, en el Cuadro IV.2 pueden encontrarse otras características de este criterio, incluyendo los tiempos invertidos en procesar una serie de secuencias de diferente longitud.

Algoritmo 9 Código ensamblador correspondiente al criterio de decimación MITAD, para el caso $GF(2^8)$

```
1 cmpl   $0x80,-0x4(ebp)   <<Compara el valor de a_i con 128>>
2 jg     80483de <main+0x62>  <<Instrucción de salto>>
```

Longitud secuencia (bytes)	Número descartes	Tiempo
10^6	498 584	0.406s
10^7	4 998 626	2.711s
10^8	49 998 399	18.945s

Cuadro IV.2: Resultados para el criterio MITAD

IV.3.4. Criterio MENOR

El siguiente de los criterios creados se ha denominado *criterio MENOR*, haciendo referencia a que compara los contenidos de ambos registros de la siguiente forma:

- SI $a_i < b_i$ ENTONCES $c_i = b_i$
- SI $a_i \geq b_i$ ENTONCES se descarta b_i

El criterio resulta, por tanto, *compuesto*, puesto que tiene en cuenta en su decisión el contenido de ambos registros, y *equilibrado*, pues descarta, en la ejecución a largo plazo, la mitad de los elementos generados por R_1. Los resultados de los experimentos realizados para comprobar estas características pueden encontrarse en el Cuadro IV.3.

El criterio MENOR tiene, sin embargo, una implementación software algo menos eficiente que MITAD, pues implica el uso de tres instrucciones en lugar de dos, como puede observarse en el listado ensamblador correspondiente (Listado 10). Aunque la diferencia no resulta especialmente significativa (ligeramente superior a un 7 % de incremento), este hecho se refleja en los tiempos invertidos por este criterio para generar la secuencia de prueba, algo superiores a los de MITAD.

IV.3.5. Criterio PAR

El siguiente de los criterios presentados ha sido denominado *criterio PAR*. Pretende hacer uso del hecho de que, en término medio, la mitad de los elementos de la secuencia generada por los LFSR serán pares y la otra mitad impares.

Longitud secuencia (bytes)	Número descartes	Tiempo
10^6	496 631	0.437s
10^7	4 978 821	3.671s
10^8	49 807 252	19.821s

Cuadro IV.3: Resultados para el criterio MENOR

Algoritmo 10 Código ensamblador correspondiente al criterio de decimación MENOR

```
1 mov    -0x4(ebp),eax      <<Pone en EAX el valor de R_1>>
2 cmp    -0x8(ebp),eax      <<Lo compara con el valor de R_2>>
3 jge    80483e0 <main+0x64>  <<Comprueba si R_1 es menor o mayor que R_2>>
```

Longitud secuencia (bytes)	Número descartes	Tiempo
10^6	500 286	0.487s
10^7	5 001 594	3.952s
10^8	50 003 343	20.626s

Cuadro IV.4: Resultados para el criterio PAR

Como consecuencia, la definición del criterio toma la siguiente forma:

- SI a_i es PAR ENTONCES $c_i = b_i$
- SI a_i es IMPAR ENTONCES se descarta b_i

Esta definición puede llevar asociado a priori un pequeño inconveniente. La razón radica en el hecho de que un computador obtiene la paridad de un número comprobando si su bit menos significativo es 1 (en cuyo caso es par) o 0 (en cuyo caso es impar).

Pero, como se analizó en la sección IV.3, uno de los requisitos deseables de estos nuevos criterios es que no realicen operaciones a nivel de bit, sino a nivel de byte como mínimo, para aprovechar la arquitectura interna de los microprocesadores. La comprobación de paridad podría, por tanto, hacer que este criterio viera disminuido su rendimiento respecto al de los anteriores, pues implica el uso de máscaras, con el fin de poder operar a nivel de bit.

Efectivamente, como puede observarse en el Listado 11, el número de instrucciones necesarias crece hasta cuatro. Aunque no es, desde luego, una degradación de rendimiento muy importante, es suficiente para hacer crecer ligeramente los tiempos invertidos por este criterio, como puede observarse en el Cuadro IV.4.

Algoritmo 11 Código ensamblador correspondiente al criterio de decimación PAR

```
1 mov    -0x4(ebp),eax      << Pone en EAX el valor de R_1 >>
2 and    $0x1,eax      << Realiza una máscara: todos
3                     los bits a cero excepto el más significativo >>
4 test   eax,eax        << Comprueba si EAX es cero (y por tanto, su bit más
  significativo también lo era; es decir, es o no par) >>
5 jne    80483e2 <main+0x66>  << Toma la decisión >>
```

En cuanto a su clasificación, este criterio es *simple*, puesto que sólo R_1 está involucrado en la toma de decisión. Por otro lado, el criterio presenta también, como todos los analizados hasta el momento, una buena *estabilidad* y resulta correctamente *equilibrado*.

Longitud secuencia (bytes)	Número descartes	Tiempo
10^6	749 669	0.571s
10^7	7 498 460	4.882s
10^8	74 993 343	25.586s

Cuadro IV.5: Resultados para el criterio NO-LINEAL

IV.3.6. Criterio NO-LINEAL

Por último y a diferencia de los anteriores, se presenta un criterio con comportamiento no lineal. Los sistemas no lineales son aquellos que no satisfacen el *principio de superposición* o, de manera más informal, aquellos que no pueden ser descritos fácilmente como la suma de sus componentes[1].

En este caso el criterio de decimación se define de la siguiente forma:

- SI $(a_i * b_i)$ mód $GF(2^n)$ es PAR ENTONCES $c_i = b_i$
- SI $(a_i * b_i)$ mód $GF(2^n)$ es IMPAR ENTONCES se descarta b_i

Sin embargo, los datos experimentales confirman que, debido a su naturaleza no lineal, este criterio no cumple de forma adecuada el requisito de resultar equilibrado. Como puede observase en la Tabla IV.5, la función de decimación desecha alrededor del 75 % de los elementos de entrada, un porcentaje que está muy por encima del deseado 50 %.

Aunque en sí mismo este hecho no afecta a la calidad criptográfica de la salida, sí lo hace al rendimiento final del generador, que se verá reducido sensiblemente, ya que se debe ejecutar durante un 25 % más de tiempo que el resto de criterios para producir la misma cantidad de secuencia de salida.

Por tanto, a pesar de que la operaciones de multiplicación y reducción modular implicadas son eficientemente implementadas, como puede verse en el Listado 12, el criterio resulta, lógicamente, el más lento de todos los presentados.

Algoritmo 12 Código ensamblador correspondiente al criterio de decimación NO-LINEAL

```
1 mov    0x1c(%esp),%edx      << Pone en EDX el valor de R_2 >>
2 mov    0x18(%esp),%eax      << Pone en EAX el valor de R_1 >>
3 imul   %edx,%eax        << Multiplica ambos valores >>
4 and    $0x1,%eax        << Comprueba si el resultado es par >>
5 test   %eax,%eax
6 jne    8048538 <main+0xa4>
```

[1] Estos sistemas, cuya salida no es proporcional a la entrada, son de mucho interés para los físicos, pues la mayoría de los sistemas físicos presentes en la Naturaleza se comportan de forma no lineal. Como ejemplo paradigmático de estos sistemas suele citarse al tiempo metereológico, donde pequeños cambios en alguna de las variables implicadas tienen grandes e impredecibles consecuencias, de forma que pueda analizarse el comportamiento de este tipo de decimación.

Criterio	Tipo	Equilibrado (%)	Tiempo por decisión
MITAD	Simple	✓ (49.85 %)	0.406μs
MENOR	Compuesto	✓ (49.66 %)	0.437μs
PAR	Simple	✓ (50.02 %)	0.487μs
NO-LINEAL	Compuesto	✗ (74.96 %)	0.571μs

Cuadro IV.6: Tabla resumen de datos experimentales para los diferentes criterios presentados

IV.3.7. Conclusiones

En esta sección se han presentado una serie de nuevos criterios de decimación, adaptados a los nuevos registros de desplazamiento utilizados en el diseño final del cifrador, comprobando cuáles de ellos cumplen, y en qué medida, los requisitos impuestos en la sección IV.3.2.

En principio, todos los criterios, excepto aquel diseñado mediante una función no lineal, resultan aptos para su uso, pues presentan buenas características de *estabilidad, equilibrio y eficiencia*.

Es, sin embargo, en este último aspecto donde podemos encontrar las diferencias más significativas. Éstas radican en la complejidad de la definición de cada criterio y, como consecuencia, en los diferentes rendimientos proporcionados por cada uno.

En este sentido, el claro ganador es el criterio MITAD, que necesita únicamente 2 instrucciones máquina, que hace que no parezca posible encontrar un criterio aún más sencillo con menor coste computacional y que, por supuesto, siga siendo funcional.

Aunque las diferencia en el rendimiento entre criterios es mínima en una decisión individual, ésta puede hacerse significativa conforme aumenta la cantidad de secuencia de salida generada. En la Figura IV.3 pueden encontrarse los tiempos invertidos por cada criterios en la generación de secuencias de salida de diferentes longitudes (10^6, 10^7 y 10^8 elementos).

Por último, los datos experimentales obtenidos para cada uno de los criterios se resumen en el Cuadro IV.6.

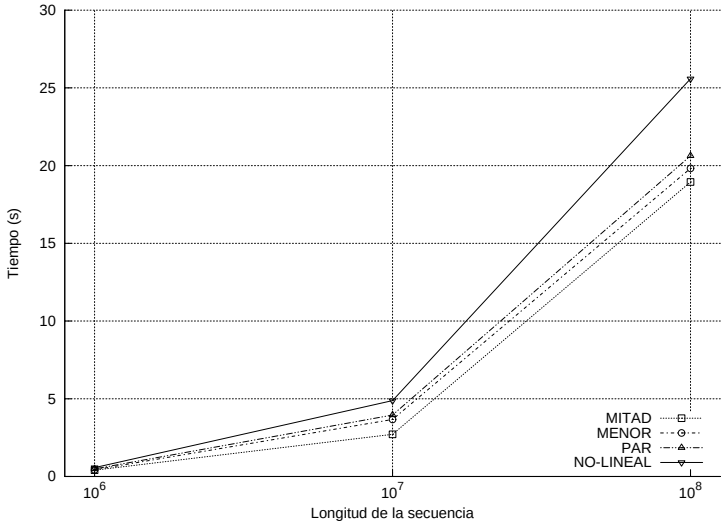


Figura IV.3: Comparación del rendimiento de cada criterio de decimación analizado

IV.4. BÚFER DE SALIDA Y TÉCNICAS PARA LA GESTIÓN DE LA DECIMACIÓN IRREGULAR

Debido al uso de las funciones de decimación, el módulo de generación de secuencia produce su salida "a saltos", de forma irregular, presentando un número elevado de ciclos de reloj en los que no se generan elementos.

Por este motivo, tanto este generador, como todos aquellos que proporcionen de esta forma su salida, pueden resultar vulnerables a un criptoanálisis trivial, bajo determinadas circunstancias, que se analizan a continuación.

Criptoanálisis de generadores con salida irregular Para llevar a cabo este ataque, un atacante debe ser capaz de monitorizar la salida del generador y tener acceso a su señal de reloj interna. Bajo estas condiciones, es posible recuperar total o parcialmente los contenidos de los registros R_1 y R_2.

En el caso de que el generador sea definido sobre el cuerpo tradicional $GF(2)$, el ataque es directo y definitivo. En efecto, si en un ciclo de reloj determinado el cifrador no produce salida, el atacante puede inferir de inmediato que el bit correspondiente de la secuencia generada por R_1 era un 0. Por el contrario, si se produce salida, entonces el bit correspondiente de R_1 era 1 y el de R_2 el que se entrega como salida.

De esta forma resulta trivial reconstruir, casi en su totalidad, los contenidos iniciales de los registros, excepto unos pocos bits de R_2, que pueden, sin embargo, obtenerse inmediatamente con algunos intentos por prueba y error (en término medio, unos $L/2$ intentos, siendo L la longitud de R_2).

Si, por el contrario, el generador de secuencia ha sido definido utilizando cuerpos extendidos, haciendo uso de algunos de los criterios presentados en la sección anterior, el criptoanálisis se hace algo más difícil, aunque no imposible.

En efecto, si se utiliza, por ejemplo, el criterio de decimación MITAD, el atacante, al observar un ciclo que no produce salida, sólo puede inferir que el valor correspondiente de R_1 es menor que 2^{n-1}, siendo n el tamaño del cuerpo utilizado. Este hecho, innegablemente, dificulta el ataque, sobre todo para valores de n de 16 y 32 bits, pero en última instancia no lo hace imposible. El resto de criterios de decimación, como MENOR, PAR Y NO-LINEAL, implican ataques de complejidades algorítmicas similares.

Fallo del sistema El hecho de que, en un determinado momento, el sistema no sea capaz de proporcionar salida debido al uso del proceso de decimación, se define como un *fallo o falta* del sistema. Evidentemente, esta situación debe evitarse a toda costa, pues tiene efectos indeseables, como la reducción de la velocidad a la que puede operar el sistema o el descenso drástico de su nivel de seguridad, abriendo la puerta al criptoanálisis recién mencionado.

Para evitar este inconveniente, que no es exclusivo del generador shrinking, sino de cualquier generador de secuencia con salida irregular, se han propuesto dos soluciones básicas en la literatura, que fueron mencionadas inicialmente por los autores originales Coppersmith, Krawczyk y Mansour:

- La primera consiste en aumentar la velocidad a la que el sistema genera secuencia, manteniendo o reduciendo la velocidad a la que éste la "consume". De esta forma, se espera que la secuencia no se agote y el sistema cuente siempre con elementos disponibles para ser utilizados en el cifrado.

- La segunda solución consiste en el uso de un *búfer de salida*, que "oculte" el proceso de decimación al atacante o criptoanalista y proporcione una tasa de salida regular.

Evidentemente, con el fin de reducir el coste de la implementación, tanto en hardware como en software, y, sobre todo para obtener el rendimiento máximo en la producción de secuencia cifrante, el objetivo es minimizar en lo posible tanto el tamaño del búfer B como el ratio al que funciona el sistema.

Métodos correctivos Sin embargo, las propuestas anteriores no fueron desarrolladas ni analizadas con suficiente detalle por los propios autores. En [17][2] el problema se aborda con mayor profundidad pero, aún en ese caso, quedan algunos aspectos por aclarar.

Por este motivo, en las secciones IV.4.1 y IV.4.2, se presentarán las soluciones originales aportadas por estos autores y se avanzará en el análisis de diversos aspectos relativos al ratio de funcionamiento del sistema y al búfer de salida. Por ejemplo, se determinará el tamaño mínimo que debe poseer éste para que nunca se agote, o que lo haga con una probabilidad arbitrariamente baja.

Por otro lado, existen ciertos aspectos relativos al funcionamiento del generador que han sido completamente ignorados hasta la fecha. El más importantes de ellos puede considerarse, sin duda, la posibilidad de que, durante el funcionamiento normal del sistema, el búfer de salida alcance su máxima capacidad y, en consecuencia, se desborde.

En este sentido, demostraremos en las siguientes secciones que con los ratios de funcionamiento propuestos por lo autores, el búfer se desbordará inevitablemente en periodos de tiempo relativamente pequeños. Esta es una situación que los autores del generador shrinking no contemplan en la propuesta original y, en consecuencia, no mencionan cómo gestionar. Con el fin de remediar esta situación, en esta tesis se aportarán dos nuevas soluciones, que pueden resumirse brevemente, de la siguiente forma:

- *Pausa de generación*: consiste en detener el módulo de generación de secuencia durante el tiempo necesario para evitar que el búfer se desborde.
- La *gestión dinámica del ratio* adapta el ratio de funcionamiento del sistema, reduciendo o aumentando su valor, a la ocupación del búfer en cada momento, con el fin de evitar el agotamiento o desbordamiento del mismo.

Estas soluciones serán descritas en profundidad en la sección IV.4.4.

IV.4.1. Ratio de funcionamiento

Como se ha comentado, el uso de una función de decimación, componente imprescindible del primer módulo, provoca que éste entregue secuencia, por término medio, a la mitad del ritmo con el que el módulo de cifrado la consume. Por esta razón, un sistema así definido sufrirá frecuentes faltas, situaciones en las que no podrá entregar secuencia cifrante.

Para evitarlo, la primera de las medidas correctivas que proponen los autores del generador shrinking consiste en definir un *ratio de funcionamiento* entre ambos módulos, que "adapte" las diferentes velocidades de funcionamiento de ambos.

[2] La referencia es, en realidad, un informe técnico interno de IBM, institución donde trabajaban ambos autores en la fecha de publicación. Curiosamente, aunque es citado por multitud de artículos, no está disponible actualmente en Internet y es necesario pedirlo directamente a los autores.

Resulta necesario, por tanto, el uso de *dos señales de reloj diferentes*, una para cada una de los módulos implicados, con el fin de tratar de evitar que se produzcan faltas y, simultáneamente, maximizar en lo posible el rendimiento del sistema.

Ratios a, b y α A continuación se presentan una serie de definiciones básicas respecto a estos relojes.

IV.4 Definición. (*Ratio a*) Se define como *ratio a*, o *ratio del módulo generador*, al número de pulsos que el generador de secuencia utiliza por unidad de tiempo. En cada uno de ellos, puede producirse o no una salida.

Nótese que este ratio hace referencia al número de pulsos que alimentan al generador, *no al número de elementos que éste, finalmente, genere*. En cada uno de estos pulsos, el generador tiene una probabilidad 1/2 de generar un elemento de salida, y la misma de no generarlo. De esta forma, el número esperado de elementos generados tras n pulsos es $n/2$.

IV.5 Definición. (Ratio *b*) Se define como *ratio b*, o *ratio del módulo de cifrado*, al número de elementos que el módulo de cifrado consume (cifra) por unidad de tiempo. A diferencia del ratio *a*, el ratio *b* sí consume con seguridad al menos un elemento por cada unidad de tiempo.

IV.6 Definición. (Ratio α) Finalmente, se define como *ratio* α, o *ratio de funcionamiento del cifrador*, al ritmo al que el sistema es capaz de suministrar secuencia cifrante sin producir faltas. Obviamente, el ratio α será igual a la relación entre a y b, de forma que $\alpha = a : b$.

IV.2 Ejemplo. *Imagínese un sistema capaz de ejecutar seis veces la rutina del generador de secuencia por unidad de tiempo. Su ratio a será, por tanto,* $a = 6$. *En cada una de estas ejecuciones, el sistema tendrá una posibilidad de 1/2 de generar o no secuencia.*
En esta misma unidad de tiempo, el sistema consume, de forma constante, 3 *caracteres de la secuencia generada en el proceso anterior, por lo que su ratio b será* $b = 3$. *De esta forma, el ratio de funcionamiento global del sitema será* $\alpha = a : b = 6 : 3 = 2$, *lo que significa que el módulo de generación de secuencia funciona dos veces más rápido que el de cifrado.*

En otras palabras, el ratio α proporciona una medida de cuán más rápido es el módulo de generación de secuencia respecto al de cifrado. De forma intuitiva, este hecho implica que, cuanto mayor sea el valor de α, mayor será la diferencia de velocidad entre ambos módulos y, por tanto, menor las probabilidad de que se produzca una falta. Aunque, como analizaremos en la sección IV.4.3.3, aumenta entonces la probabilidad de que el búfer de salida llegue a su máxima capacidad y se desborde. En la Figura IV.4 se resumen los diferentes ratios y se muestra un ejemplo concreto.

IV.4.1.1. Implementación hardware y software

Desafortunadamente, el uso de dos "ritmos" o relojes diferentes plantea una serie de cuestiones y dificultades que es necesario resolver, y que cobran especial importancia en la implementación el sistema.

Implementación hardware Cuando dicha implementación se lleva a cabo en hardware, ésta apenas presenta inconvenientes. La necesidad de dos señales de reloj puede solucionarse fácilmente utilizando *circuitos multiplicadores* [18], muy comunes, por ejemplo, en el diseño de microprocesadores.

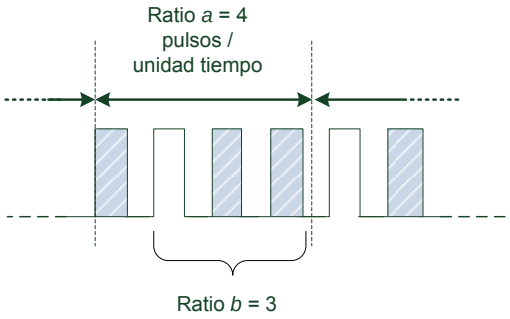


Figura IV.4: La figura muestra un sistema con ratio $\alpha = 4 : 3$. Los pulsos sombreados indican aquellos en los que se produce salida. Aunque $a = 4$, en la unidad de tiempo mostrada como ejemplo se producen sólo 3 caracteres de secuencia cifrante. Dado que $b = 3$, también se consumen 3, de forma que el tamaño del búfer en esa unidad de tiempo queda intacto.

Un circuito multiplicador aplicado a una señal de reloj eleva, o reduce, la frecuencia de la misma en un factor predeterminado. De esta forma, la señal de reloj necesaria para cada módulo puede obtenerse a partir de una señal original con una frecuencia f_θ arbitraria.

Dado que el módulo de cifrado necesita la señal de reloj de menor frecuencia, éste podría alimentarse del reloj original, de forma que $f_{cifrado} = f_\theta$. Por otro lado, haciendo uso del multiplicador, la señal de reloj de la que haría uso el módulo generador de secuencia sería $f_{generador} = f_\theta \cdot \alpha$.

Implementación software Sin embargo, el término "señal de reloj" pierde su significado original cuando se aborda la implementación software del sistema. Por supuesto, no hay relación directa entre los ciclos de reloj del microprocesador que ejecuta la implementación del cifrador y los ciclos "lógicos" del mismo.

Resulta necesario definir, por tanto, un procedimiento para simular estos dos ritmos asíncronos. La idea es sencilla y se basa en utilizar dos bucles consecutivos. El primero de ellos contendrá la implementación del módulo de generación de secuencia, y simulará su ratio de funcionamiento a.

El segundo, por otro lado, hará lo propio con el módulo de cifrado y el ratio b. Así, por ejemplo, un ratio de $\alpha = 2.5$, se implementaría utilizando estos dos bucles, que repetirán las porciones de código necesarias el número adecuado de veces. Lógicamente, dado que $\alpha = 2.5 = 25/10 = 5/2$, el primer bucle debería ejecutarse 5 veces y 2 el segundo. El resultado final puede encontrarse en el Listado 13.

Algoritmo 13 Simulación del ratio de funcionamiento α en software

```
1  for (a=0; a<5; a++)
2    <<Código del generador de secuencia>>
3  for (b=0; b<2; b++)
4    <<Código del módulo de cifrado>>
```

IV.4.1.2. Consideraciones sobre el ratio α

Considérese un generador de secuencia definido con un ratio $\alpha = a : b$, de forma que, en cada intervalo de tiempo, se genera en promedio $a/2$ elementos y, de éstos, consume b elementos en el proceso de cifrado.

En este escenario, pueden deducirse las siguientes observaciones:

- Si $a/2 < b$, o, lo que es lo mismo, $a < 2b$, el sistema no será capaz de suministrar secuencia suficientemente rápido y producirá faltas de forma constante.
- Si $a = 2b$, el sistema se mantendrá en cierto equilibrio, pues, en término medio, el número de elementos generados y consumidos es el mismo, y sólo se producirán faltas de forma ocasional.
- Si $a > 2b$, el sistema genera secuencia más rápidamente de la que ésta se consume, por lo que las faltas en el suministro de elementos para la secuencia cifrante final serán poco numerosas.

A la vista de los datos anteriores, parece claro que, para evitar faltas del sistema, el diseño final deberá hacer uso de ratios $\alpha > 2$ (esta cuestión será analizada en mayor profundidad en las secciones IV.4.3 y IV.4.4). En ese caso, el generador produce más elementos de los que se consumen en el proceso de cifrado. Surge, por tanto, una pregunta natural: ¿qué ocurre con los elementos no utilizados?. En este punto, planteamos varias alternativas:

- Dichos elementos, simplemente, se descartan. Esto supone un claro desperdicio de tiempo de computación y, lo que es aún peor, implica una pérdida de rendimiento importante.
- Estos elementos pueden almacenarse, para su uso posterior, cuando haya ciclos en los que salida del generador de secuencia sea menor que la necesitada por el módulo de cifrado.

Esta última opción, mucho más recomendable y eficiente, refuerza la hipótesis que venimos planteando de que el uso de la contramedida del aumento del ratio de funcionamiento está inevitablemente vinculada al uso de un búfer de salida, que almacene los elementos sobrantes, y que ambas medidas se necesitan mutuamente.

Un búfer infinito Para ilustrar este punto, obsérvese la Figura IV.5, en la que puede encontrarse una simulación realizada para mostrar el comportamiento de un módulo generador funcionando a diferentes ratios α. La gráfica muestra cómo evolucionaría la cantidad de elementos almacenados en un búfer imaginario, de tamaño teórico infinito, colocado a la salida del generador de secuencia.

Como puede observarse, el ratio $\alpha = 1$ conduciría inevitablemente al agotamiento de dicho búfer en unas pocas iteraciones. Por el contrario, utilizando un ratio $\alpha = 3$, el comportamiento del sistema sería el opuesto: el tamaño del búfer crecería sin límites y, sea cual sea éste en la práctica, terminará por desbordarse irremediablemente. Por último, la situación de equilibrio se produce alrededor de $\alpha = 2$, donde el sistema podría funcionar de forma estable, es decir, sin producir faltas y sin requerir tamaños de búfer excesivamente grandes.

La pregunta obvia que surge a continuación es: ¿cuál es la combinación adecuada de ratio α y tamaño de búfer para que el sistema sea estable?. Esta cuestión será abordaba a continuación.

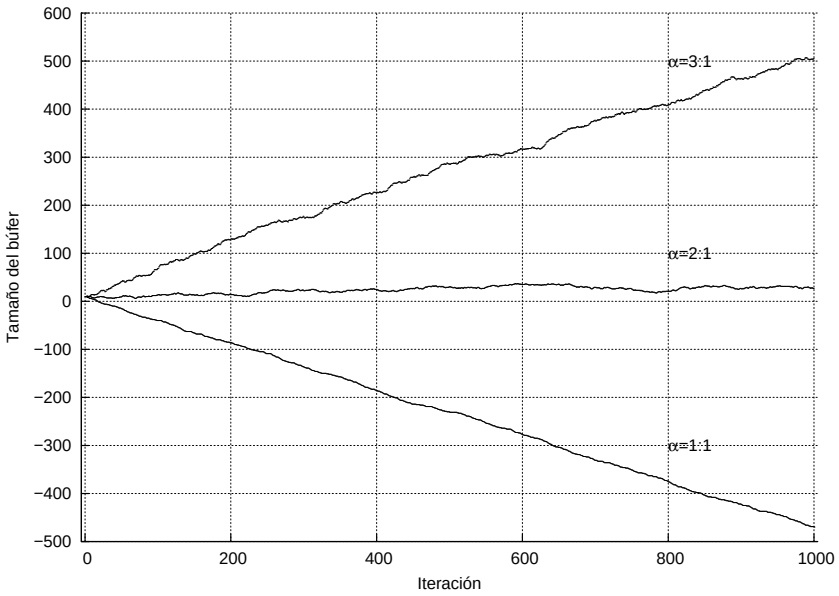


Figura IV.5: Simulación del efecto de ratio de funcionamiento α. El experimento se ha llevado a cabo utilizando un generador definido sobre el cuerpo $GF(2^8)$, funcionando a ratios $\alpha = 1$ (1:1), $\alpha = 2$ (2:1) y $\alpha = 3$ (3:1), y generando una secuencia cifrante de 10^6 elementos.

IV.4.2. Análisis del tamaño del búfer de salida

Como hemos comentado hasta el momento, utilizar un búfer de salida *resulta imprescindible en la implementación de todo generador de secuencia con salida irregular*, por dos razones básicas.

En primer lugar, este búfer es necesario para ocultar la decimación irregular del generador y evitar de esta forma un criptoanálisis trivial. Por otro lado, se utilizará también para almacenar el excedente de secuencia cifrante, que se genera como consecuencia del incremento del ratio de funcionamiento del módulo de generación.

El búfer debe situarse a la salida del generador de secuencia, de forma que pueda almacenar los elementos producidos por éste, y a la entrada del módulo de cifrado, al que proporciona, en cada ciclo de reloj, los elementos previamente almacenados.

De esta forma, se produce un desacoplamiento entre el módulo de generación y el de cifrado, que sirve para adaptar sus diferentes velocidades de funcionamiento. En la Figura IV.6 puede observarse un esquema del cifrador que incorpora uno de estos búferes.

Dado que el búfer resulta imprescindible, surge entonces de forma natural la siguiente pregunta: ¿cuál debe ser el tamaño mínimo del mismo para un correcto funcionamiento del sistema? Es necesario, por tanto, antes de abordar los detalles de su implementación, analizar con mayor profundidad la relación entre estos dos aspectos fundamentales, ratio de funcionamiento y tamaño del búfer correspondiente, encontrando respuesta a las siguientes cuestiones:

Problema 1 Dado un ratio de funcionamiento de α, ¿cuál es el tamaño mínimo B del búfer que asegura que la probabilidad de una falta, que denotaremos por $M(B, \alpha)$, puede hacerse arbitrariamente pequeña?

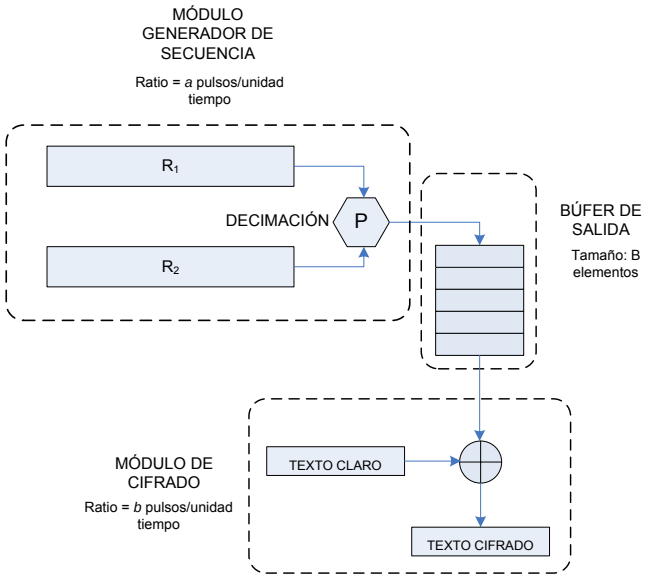


Figura IV.6: Esquema de cifrador con búfer intermedio entre el módulo de generación y el de cifrado

Problema 2 Dado que el tamaño del búfer utilizado es B, ¿cuál es el ratio mínimo de funcionamiento que garantiza que la probabilidad de falta $M(B, \alpha)$ puede hacerse arbitrariamente pequeña?

Para llevar a cabo este análisis, modelaremos el sistema realizando las siguientes suposiciones:

- El búfer tiene capacidad para albergar un máximo de B elementos, el tamaño de cada uno de los cuales dependerá del cuerpo subyacente que se utilice (bytes, medias palabras o palabras). En cualquier caso, esta decisión no tiene efecto sobre el análisis que se desarrollará a continuación.
- El eje temporal del sistema está divido en unidades de tiempo de igual duración, que denominaremos *intervalos*, sin que sea relevante la magnitud concreta de los mismos.
- En cada uno de estos intervalos, el sistema ejecuta a veces la rutina de generación de secuencia, con una probabilidad de 1/2 de generar un elemento e incorporándolo al búfer.
- Asimismo, durante cada uno de los intervalos, el sistema retira del búfer b elementos de forma sistemática.

En la Figura IV.7 puede encontrarse un esquema del modelo del sistema que se utilizará para los cálculos posteriores.

La modelización matemática se llevará a cabo utilizando las denominadas *cadenas de Markov* [19, 20], que se postulan como un candidato natural para afrontar este tipo de problemas. El análisis comenzará obteniendo, en la siguiente sección, un resultado que será necesario para los cálculos posteriores. A continuación, en la sección IV.4.2.2 se introducirán una serie mínima de conceptos acerca de las cadenas de Markov y,

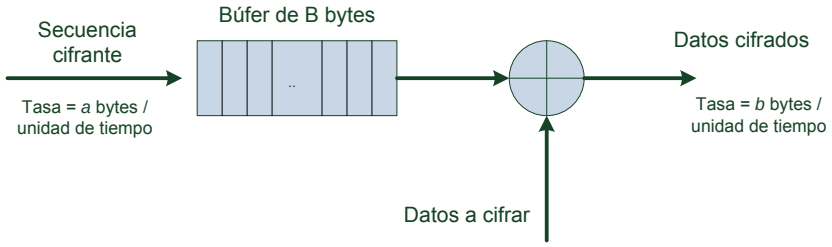


Figura IV.7: Modelo del sistema utilizado en la derivación del tamaño adecuado del búfer en función del ratio de funcionamiento

finalmente, se presentan los cálculos realizados, juntos con los resultados obtenidos y su correspondiente análisis.

IV.4.2.1. Cálculos preliminares

Comenzaremos el análisis calculando la probabilidad de que el módulo generador produzca exactamente k elementos en un intervalo de tiempo dado. El módulo generador, que funciona a un ratio a, tiene una probabilidad de $1/2$ de generar un elemento en cada uno de estos intervalos.

Este funcionamiento recuerda inmediatamente al de un *proceso de Bernoulli*, que se define de la siguiente forma.

IV.7 Definición. Un *proceso de Bernoulli* es una secuencia de n experimentos tales que:

1. Cada experimento tiene dos posibles salidas, que se denominan *éxito* y *fracaso*.
2. La probabilidad de éxito p es la misma en cada experimento, y ésta no se ve afectada por el conocimiento del resultado de experimentos previos (se trata de sucesos independientes). La probabilidad de fracaso viene dada por $q = 1 - p$.

IV.3 Ejemplo. *Ejemplos de procesos de Bernoulli:*

1. *El lanzamiento de una moneda. Las dos posibles salidas son cara y cruz, que podemos asociar, arbitrariamente, a éxito y fracaso.*
2. *El lanzamiento de un dado, en el que se decide que éxito es obtener un 6 y fracaso cualquier otro número.*
3. *Un sondeo de opinión, en el que los sujetos son elegidos al azar y se les pregunta si votarán 'sí' o 'no' en cierto referendum político.*

En nuestro caso, la probabilidad de éxito p corresponde a que el generador produzca salida en determiando ciclo de reloj, lo que ocurre con probabilidad $1/2$. Exactamente igual ocurre si no se obtiene salida, por lo que $q = p = 1/2$.

IV.1 Teorema. *Realizados n experimentos de un proceso de Bernoulli con probabilidad p de éxito, la probabilidad de que haya exactamente j éxitos es:*

$$b(n, p, j) = \binom{n}{j} p^j q^{n-j}$$

IV.4 Ejemplo. *Se lanza una moneda al aire seis veces. ¿Cuál es la probabilidad de que se obtengan exactamente 3 caras? La respuesta, completamente anti-intuitiva, es:*

$$b(6,0'5,3) = \binom{6}{3}\left(\frac{1}{2}\right)^3\left(\frac{1}{2}\right)^3 = 20 \times \frac{1}{64} = 0'3125$$

Si la probabilidad de fracaso, $q = p$, entonces la fórmula queda:

$$b(n,p,j) = \binom{n}{j}p^j p^{n-j} = \binom{n}{j}p^n$$

Teniendo en cuenta que un intervalo de tiempo está divido en a ciclos, queda finalmente que la probabilidad de que el sistema genere exactamente k elementos en un intervalo de tiempo, que denominaremos P_k, es:

$$P_k = b(a,1/2,k) = \binom{a}{k}\left(\frac{1}{2}\right)^a \tag{IV.4}$$

Una derivación alternativa, con la que llegaremos al mismo resultado anterior, consiste en utilizar una secuencia binaria, donde un bit '1' significa que, en ese ciclo de reloj el generador produjo un elemento y un bit '0' significa que no lo hizo, y contar el número de combinaciones que tienen exactamente k bits a '1'.

Este número equivale a la cantidad de combinaciones con repetición de a elementos tomados de k en k:

$$\binom{a}{k}$$

Dado que cada una de estas secuencias es equiprobable, con probabilidad $1/2$, el resultado final es que la probabilidad de que se produzcan exactamente k elementos en un intervalo de tiempo es:

$$\left(\frac{1}{2}\right)^a\binom{a}{k} \tag{IV.5}$$

Que, por supuesto, coincide con la expresión IV.4.

IV.4.2.2. Conceptos básicos sobre cadenas de Markov

En 1907, A. A. Markov comenzó el estudio de un nuevo e importante tipo de proceso estadístico, en el que la salida de un experimento puede afectar al siguiente. Este tipo de procesos se conocen hoy como *cadenas de Markov* [20].

Cadenas de Markov Una cadena de Markov consta de una serie de *estados*, $S = \{s_1, s_2, \ldots, s_r\}$. El proceso comienza en uno de estos estados y transita sucesivamente de un estado a otro. Si la cadena está en un momento dado en el estado s_i, entonces transita al estado s_j con probabilidad p_{ij}. Estas probabilidades se denominan *probabilidades de transición* y suelen escribirse en forma de matriz.

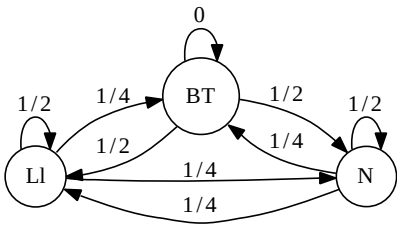


Figura IV.8: Diagrama de transiciones de la cadena de Markov definida en el ejemplo IV.5

IV.5 Ejemplo. *De acuerdo a su autor, J.R.R. Tolkien, el fantástico mundo de la Tierra Media fue bendecido con muchos dones, pero entre ellos no se encontraba el del buen tiempo. De hecho, sus habitantes nunca disfrutaban de dos días seguidos de buen tiempo (BT). Si un día lucía el sol, al día siguiente tendrían lluvia (Ll) o nieve (N) con seguridad e igual probabilidad. Si llovía o nevaba, tendrían un 50 % de posibilidades de continuar igual al día siguiente.*
Con esta información, es posible formar una cadena de Markov $\boldsymbol{P}$ *que modelice el desagradable clima de la Tierra Media:*

$$\boldsymbol{P} = \begin{array}{c} \\ Ll \\ BT \\ N \end{array} \begin{array}{c} \begin{array}{ccc} Ll & BT & N \end{array} \\ \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array}$$

Es muy común también representar la cadena utilizando un diagrama de transiciones, como puede observarse en la Figura IV.8.

Matriz de transición Se aborda ahora una cuestión esencial en el desarrollo, que consiste en calcular la probabilidad de que, dada una cadena en un estado s_i arbitrario, transite a un estado s_j en exactamente n pasos. Este importante concepto se denota por $p_{ij}^{(n)}$ y puede calcularse utilizando el siguiente resultado.

IV.2 Teorema. *Sea* $\boldsymbol{P}$ *la matriz de transición de una cadena de Markov. La entrada* ij*-ésima,* $p_{ij}^{(n)}$*, de la* n*-ésima potencia de la matriz* $\boldsymbol{P}$*, denotado* $\boldsymbol{P}^n$*, proporciona la probabilidad de que la cadena, empezando en el estado* s_i*, transite al estado* s_j *en* n *pasos.*

IV.6 Ejemplo. *Si se calculan las sucesivas potencias de la matriz de transiciones del ejemplo anterior (IV.5), se observa que las predicciones sobre el tiempo son independientes del tiempo del día actual:*

$$\boldsymbol{P}^6 = \begin{array}{c} \\ Ll \\ BT \\ N \end{array} \begin{array}{c} \begin{array}{ccc} Ll & BT & N \end{array} \\ \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.2 & 0.4 \end{pmatrix} \end{array}$$

Como puede observarse, las predicciones para los tres tipos de tiempo, Ll, BT y N, son 0.4, 0.2 *y* 0.4 *independientemente del estado en el que comenzara la cadena. Éste es un ejemplo de un tipo de cadena de Markov muy importante denominada cadena regular, que será estudiada con más detalle a continuación.*

Tipos de cadenas de Markov Las cadenas de Markov pueden clasificarse en diversos tipos, en función de las propiedades de su matriz de transición. A continuación describiremos las denominadas cadenas *ergódicas* y *regulares*, que resultan de vital importancia en los cálculos posteriores.

IV.8 Definición. Una cadena de Markov se considera *ergódica o irreducible* si es posible transitar desde cualquier estado a cualquier otro (no necesariamente en una iteración).

IV.9 Definición. Una cadena de Markov es *regular* si alguna potencia de la matriz de transición tiene sólo elementos positivos. O, en otras palabras, para algún n es posible transitar desde cualquier estado a cualquier otro en exactametne n pasos.

Como la segunda de las restricciones es más fuerte, resulta claro que toda cadena regular es ergódica. Pero no toda cadena ergódica es necesariamente regular, como puede observarse en el siguiente ejemplo.

IV.7 Ejemplo. *Considérese la matriz de transición $\boldsymbol{P}$ de una cadena de Markov definida como:*

$$\boldsymbol{P} = \begin{matrix} \\ A \\ B \end{matrix} \begin{matrix} \begin{matrix} A & B \end{matrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{matrix}$$

Resulta claro que es posible transitar libremente entre cualesquiera estados, luego la cadena es ergódica. Sin embargo, si n es impar, entonces no es posible moverse del estado A al estado A en n pasos y, si n es par, no se puede transitar del estado A al B en n pasos. Por tanto, luego la cadena no es regular.

Distribución estacionaria El siguiente teorema es también esencial en el desarrollo posterior, pues permitirá calcular las probabilidades de falta del generador de secuencia y el tamaño de búfer adecuado en función del ratio de funcionamiento.

IV.3 Teorema. ***(Teorema Fundamental del Límite para cadena regulares)*** *Sea $\boldsymbol{P}$ la matriz de transición de una cadena regular. Entonces, conforme $n \to \infty$, las sucesivas potencias $\boldsymbol{P}^n$ tienden a una matriz límite $\boldsymbol{W}$ cuyas filas son idénticas e iguales a un vector $\boldsymbol{w}$. El vector $\boldsymbol{w}$, denominado vector fijo, es estrictamente positivo, es decir, todas sus componentes son positivas y suman 1.*

Más aún, el vector fijo se alcanza desde cualquier estado inicial, como demuestra el siguiente resultado.

IV.4 Teorema. *Sea $\boldsymbol{P}$ la matriz de transición de una cadena regular con vector fijo $\boldsymbol{w}$. Entonces para cualquier vector de probabilidad $\boldsymbol{u}$ (estado inicial), se cumple que $\boldsymbol{u}\boldsymbol{P}^n \to \boldsymbol{w}$ conforme $n \to \infty$.*

Otros resultados y definiciones importantes que serán necesarias más adelante son las siguientes.

IV.5 Teorema. *Sea $\boldsymbol{P}$ una matriz de transición regular, y sea*

$$\boldsymbol{W} = \lim_{n\to\infty} \boldsymbol{P}^n,$$

sea $\boldsymbol{w}$ la fila común de $\boldsymbol{W}$, y sea $\boldsymbol{c}$ un vector columna cuyos componentes son todos 1. Entonce se cumple que:

1. ***wP=w**, y que cualquier vector fila **v** tal que **vP=v** es un múltiplo de **w**.*
2. ***Pc=c**, y que cualquier vector columna **x** tal que **Px=x** es un múltiplo de **c**.*

IV.10 Definición. Un vector fila **w** con la propiedad **wP=w** se denomina *vector fila fijo* de **P**. De igual forma, un vector columna **x** tal que **Px=x** se llama *vector columna fijo* de **P**.

IV.8 Ejemplo. *Aplicando el Teorema IV.3, resulta fácil encontrar el vector límite w partiendo del hecho de que*

$$w_1 + w_2 + w_3 = 1$$

y que

$$(w_1, w_2, w_3)\begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} = (w_1, w_2, w_3)$$

La expresión anterior permite plantear las siguientes cuatro ecuaciones con tres incógnitas:

$$\begin{aligned} w_1 + w_2 + w_3 &= 1, \\ (1/2)w_1 + (1/2)w_2 + (1/4)w_3 &= w_1, \\ (1/4)w_1 + (1/4)w_3 &= w_2, \\ (1/4)w_1 + (1/2)w_2 + (1/2)w_3 &= w_3, \end{aligned}$$

El teorema anterior garantiza que este sistema de ecuaciones tiene solución única. Resolviéndolo, se obtiene la solución

$$\boldsymbol{w} = (0.4, 0.2, 0.4)$$

que, por supuesto, coincide con lo predicho por $\boldsymbol{P}^6$, dado en el ejemplo IV.6.

Para calcular el vector fijo se utiliza habitualmente el siguiente procedimiento. Se asume que el valor de un estado cualquiera, $\mathbf{w}_i$, es 1, y se usan entonces el resto de ecuaciones. Dado que estas ecuaciones tendrán una solución única, puede obtenerse **w** a partir de esa solución dividiendo cada una de sus componentes por la suma de estas mismas componentes.

IV.9 Ejemplo. ***Continuación del ejemplo IV.8.** Se fija arbitrariamente el valor de $w_1 = 1$, y se plantean entonces el resto de ecuaciones lineales que se obtienen de **wP=w**:*

$$\begin{aligned} (1/2) + (1/2)w_2 + (1/4)w_3 &= 1, \\ (1/4) + (1/4)w_3 &= w_2, \end{aligned}$$

Resolviendo se obtiene:

$$(w_1, w_2, w_3) = (1, 1/2, 1)$$

Dividiendo ahora este vector por la suma de sus componentes, se obtiene la solución final:

$$\boldsymbol{w} = (1/(5/2), (1/2)/(5/2), 1/(5/2)) = (0'4, 0'2, 0'4)$$

IV.4.2.3. Estudio del tamaño del búfer de salida utilizando la teoría de cadenas de Markov

En este punto, presentadas ya las herramientas matemáticas básicas necesarias, se tratará de responder a las dos preguntas planteadas en IV.4.2. Para ello se utilizará la teoría de cadenas de Markov para modelizar un búfer de salida, y analizar cómo evolucionaría su ocupación cuando se modifican los ratios de funcionamiento α y los tamaños del propio búfer.

De esta forma, se encontrarán las relaciones entre ambos parámetros y cómo el cambio en el valor de uno de ellos afecta al otro. El objetivo final es encontrar aquella combinación de ratio de funcionamiento y tamaño de búfer que minimiza el valor de ambos y, al mismo tiempo, sigue siendo segura y eficiente.

Variables estadísticas El análisis comienza definiendo un conjunto de *variables estadísticas*. La primera de ellas se denota como B_n, y representa el número de elementos presentes en el búfer al final del intervalo de tiempo n. Por otro lado, E_n representa el número de elementos obtenidos del generador de secuencia y, finalmente, L_n el número de elementos introducidos o eliminados del búfer al final de dicho intervalo de tiempo n.

De la definición de estas variables resultan claras las siguientes relaciones:

- La variación en el tamaño del búfer al final de cada intervalo de tiempo resulta de restar el número de elementos que se toman para el cifrado, b, de los que han sido calculados en el generador de secuencia. Luego se cumple que $L_n = E_n - b$.
- El tamaño del búfer se incrementará o decrementará en cada intervalo de tiempo n en L_n, luego $B_n = B_{n-1} + L_n$.

IV.10 Ejemplo. *Así, supóngase que el sistema a analizar se encuentra en un instante de tiempo k, en el que su búfer tiene 3 elementos, $B_{k-1} = 3$, (de 5 elementos de tamaño máximo). Por otro lado, el sistema funciona con un ratio $\alpha = 3:2$. En dicho intervalo de tiempo el generador de secuencia proporciona 3 elementos, de forma que las variables estadísticas anteriores tomarían los valores $E_k = 2$, $L_k = 3 - 2 = 1$ y $B_k = 4$.*

El conjunto de variables y otros parámetros que participan en la modelización del sistema pueden encontrarse resumidos en el Cuadro IV.7.

Cadena del espacio de estados Para modelizar un búfer de tamaño B comenzaremos definiendo una cadena de Markov $X = \{X_n, n = 0, 1, 2, ...\}$ en el espacio de estados $\{0, 1, 2, \ldots, B\}$. De esta forma, por ejemplo, $X_n = 3$, significa que, en el instante n, la cadena está en el estado 3; es decir, el búfer simulado tiene en ese momento 3 elementos.

La definición de la cadena X no estaría completa sin especificar las probabilidades de transición entre sus estados. Para ilustrar el cálculo de estas probabilidades, utilizaremos un ejemplo concreto, que se presenta a continuación.

IV.11 Ejemplo. *Sea $X_3^{3:2}$ la cadena que modela el funcionamiento de un generador con ratio $\alpha = 3:2$ y tamaño de búfer $B = 3$. Sus probabilidades de transición pueden encontrarse entonces en la Figura IV.9. En ella la columna Secuencia simula la salida producida por el generador durante tres ciclos, donde un '1' simboliza que se ha producido un elemento en dicho ciclo y un '0' que no. Así, la secuencia '101' significa que el generador produjo 2 elementos, en los ciclos primero y tercero.*

Símbolo	Descripción
B	Tamaño máximo del búfer
a	En cada uno de los intervalos de tiempo, se ejecuta a veces la rutina de generación de secuencia, con una probabilidad de 1/2 de generar en cada ejecución un elemento e incorporándolo al búfer
b	El sistema retira del búfer b elementos de forma sistemática en cada intervalo de tiempo
α	Relación entre las dos cantidades anteriores, $\alpha = a : b$
E_n	Número de elementos proporcionados por el generador de secuencia al final del intervalo de tiempo n
L_n	Número de elementos introducidos o eliminados del búfer al final del intervalo n. Claramente, $L_n = E_n - b$
B_n	Tamaño (número de elementos) del búfer al final del intervalo de tiempo n. Claramente, $B_n = B_{n-1} + L_n$
$M(B, \alpha)$	Probabilidad de que el búfer se agote y se produzca una falta del sistema. Es decir, que un intervalo tiempo el búfer no tengo elementos que suministrar al módulo de cifrado

Cuadro IV.7: Resumen de los conceptos más significativos del modelo del búfer de salida del sistema

En la misma figura pueden encontrarse también las distribuciones de las variables E_n (número de elementos generados al final del ciclo) y L_n (variación en el tamaño del búfer al final del ciclo). Puede observase entonces que, dado que $b = 2$, el tamaño del búfer se decrementará en 2 elementos sólo cuando el generador no haya producido ningún elemento en ese ciclo, es decir, $E_n = 0$. Este caso corresponde a la primera fila de la tabla, cuyos elementos han sido resaltados.

De las ochos combinaciones posibles, éste es el único caso en el que $L_n = -2$ por lo que, claramente, la probabilidad de que se presente esta situación en cada ciclo del generador es $Pr(L_n = -2) = 1/8$. El resto de probabilidades de transición pueden calcularse de forma análoga a partir de los valores de las variables E_n y L_n.

Finalmente, para calcular la *matriz de transición* de la cadena $X_n^{3:2}$ basta con aplicar el correspondiente valor de L_n a cada posible combinación de índices i y j. Por ejemplo, el elemento (0,0) de la matriz de transición hace referencia a la probabilidad de transitar

Secuencia			E_n	L_n
0	0	0	0	-2
0	0	1	1	-1
0	1	0	1	-1
0	1	1	2	0
1	0	0	1	-1
1	0	1	2	0
1	1	0	2	0
1	1	1	3	1

Pr(L_n=-2)=1/8
Pr(L_n=-1)=3/8
Pr(L_n=0)=3/8
Pr(L_n=1)=1/8

Figura IV.9: Por ejemplo, sólo tres [(0, 0), (0, 1), (1, 0)] de las ocho posibles combinaciones conducen a que el tamaño del búfer se decremente en un elemento, por lo que Pr(L_n=-1)=3/8.

$$\mathbf{P} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 7/8 & 1/8 & 1/8 & 0 \\ 4/8 & 3/8 & 1/8 & 0 \\ 1/8 & 3/8 & 3/8 & 1/8 \\ 0 & 1/8 & 3/8 & 4/8 \end{pmatrix} \end{array}$$

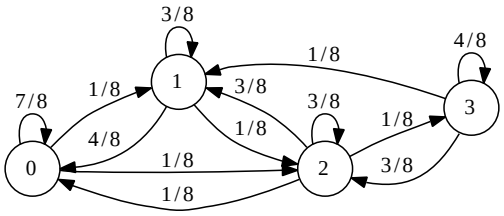


Figura IV.10: Matriz y diagrama de transiciones para la cadena de Markov **P** que modela la evolución del tamaño del búfer de salida B.

del estado 0 al estado 0, es decir, permanecer en dicho estado tras un ciclo de reloj. El estado 0 corresponde a un búfer con 0 elementos, o vacío. Por tanto, la entrada (0,0) de la matriz de transición corresponde a la probabilidad de que un búfer vacío en el ciclo n permanezca en ese estado en el ciclo $n+1$.

Para calcular esta probabilidad, basta con tener en cuenta todos aquellos valores de L_n compatibles con el estado final. En esta caso, dado que el estado final es el mismo, los valores obvios son $L_n = 0$. A ellos hay que sumar aquellos que reducirían la ocupación del búfer por debajo de cero, si eso fuera posible. Esta situación corresponde a una falta del sistema. Según los valores de la Figura IV.9, estos valores son $L_n = -1$ y $L_n = -2$. Por tanto, si la cadena está en el estado $X_n^{3:2} = 0$, es decir, el búfer está vacío en el instante n, la probabilidad de permanecer en dicho estado al instante siguiente es:

$$P(0,0) = Pr\{X_{n+1} = 0 | X_n = 0\} = Pr(L_n = -2) + Pr(L_n = 1) + Pr(L_n = 0) = 1/8 + 3/8 + 3/8 = 7/8$$

Finalmente, en la Figura IV.10 puede encontrarse la matriz de transición resultante y, a su derecha, el diagrama de transiciones que muestra la representación gráfica de estas probabilidades.

Probabilidades de transición En este punto estamos en disposición de formalizar y generalizar el ejemplo anterior, y obtener la *matriz de transición* de la cadena X para cualquier combinación de ratio α y tamaño B.

Para ello se hará uso de la expresión IV.6 que se dedujo en la sección IV.4.2.1. Esta expresión hace referencia a la probabilidad de que el sistema genere exactamente k elementos durante a ciclos de reloj. Esta probabilidad, que denotaremos P_k^a, puede calcularse de la siguiente forma:

$$P_k^a = b(a, 1/2, k) = \binom{a}{k}\left(\frac{1}{2}\right)^a \tag{IV.6}$$

Haciendo uso de esta expresión, finalmente se obtiene que las probabilidades de transición para una cadena de Markov X que modeliza la evolución de la ocupación de un búfer de tamaño B, que recibe los elementos de un generador funcionando a un ratio $\alpha = a : b$ se definen de la siguiente forma:

$$P(i,j) = \begin{cases} \left(\frac{1}{2}\right)^a \sum_{k=0}^{b-i} \binom{a}{k} & \text{si } j = 0 \\ \left(\frac{1}{2}\right)^a \sum_{k=B+b-i}^{a} \binom{a}{k} & \text{si } j = B \\ 0 & \text{si } -b > j-i > a-b \\ \left(\frac{1}{2}\right)^a \binom{a}{j+b-i} & \text{resto de casos} \end{cases} \tag{IV.7}$$

La expresión anterior también puede escribirse en forma matricial:

$$\mathbf{P} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ B-1 \\ B \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & \dots & B-1 & B \end{array} \\ \begin{pmatrix} 2^{-a}\sum_{k=0}^{b}\binom{a}{k} & 2^{-a}\binom{a}{b+1} & 2^{-a}\binom{a}{b+2} & \dots & 2^{-a}\binom{a}{B-b-1} & 2^{-a}\sum_{k=B+b}^{a}\binom{a}{k} \\ 2^{-a}\sum_{k=B-b-1}^{a}\binom{a}{k} & 2^{-a}\binom{a}{b} & 2^{-a}\binom{a}{b+1} & \dots & 2^{-a}\binom{a}{B-b-2} & 2^{-a}\sum_{k=B+b-1}^{a}\binom{a}{k} \\ 2^{-a}\sum_{k=B-b-2}^{a}\binom{a}{k} & 2^{-a}\binom{a}{b-1} & 2^{-a}\binom{a}{b} & \dots & 2^{-a}\binom{a}{B-b-3} & 2^{-a}\sum_{k=B+b-2}^{a}\binom{a}{k} \\ \vdots & \dots & \dots & \ddots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 2^{-a}\binom{a}{b} & 2^{-a}\sum_{k=b+1}^{a}\binom{a}{k} \\ 0 & 0 & 0 & \dots & 2^{-a}\binom{a}{b-1} & 2^{-a}\sum_{k=b}^{a}\binom{a}{k} \end{pmatrix} \end{array} \tag{IV.8}$$

Analizando la matriz P anterior, puede deducirse fácilmente que la cadena X es ergódica. En efecto, basta observar que las posiciones $(i, i-1)$ de P, que permite transitar de cualquier estado al anterior y $(i, i+1)$, que permite pasar de cualquier estado al posterior, siempre están definidas con probabilidades positivas. Salvo, por supuesto, para los casos límites de búfer vacío y lleno, $(i, 0)$ y (i, B), respectivamente.

Como resultado, la cadena X definida por P es **ergódica** y cuenta, por tanto, con una **distribución estacionaria**. Nótese que la propiedad de ergocidad no requiere que se pueda transitar de cualquier estado a cualquier estado *en un único paso*, sino que haya un "camino" de estados que lo conecten, con probabilidades positivas en cada estado intermedio.

Distribución estacionaria Sea $w = (w_0, w_1, \dots, w_B)$ el vector fijo que define las probabilidades estacionarias de X. De acuerdo con el Teorema IV.3, el vector w puede calcularse resolviendo el siguiente sistema de ecuaciones:

$$\begin{cases} w_0 + w_1 + \ldots + w_B = 1 \\ w_0 = w_0 \cdot P(0,0) + w_1 \cdot P(0,1) + \ldots + w_B \cdot P(0,B) \\ \vdots \\ w_B = w_0 \cdot P(B,0) + w_1 \cdot P(B,1) + \ldots + w_B \cdot P(B,B) \end{cases} \tag{IV.9}$$

Sin embargo, el cálculo anterior resulta engorroso y tedioso, incluso utilizando programas específicos, pues es necesario resolver un sistema de $(B+1)\times(B+1)$ ecuaciones.

En este caso suele utilizarse un método más eficiente y sencillo, derivado de la Definición IV.10. Según esta definición, el vector fijo w puede también calcularse resolviendo el siguiente sistema de ecuaciones matriciales:

$$\begin{aligned} wP &= w \\ w\mathbf{1} &= \mathbf{1} \end{aligned}$$

donde $\mathbf{1}$ es el vector columna unidad. Sea R la matriz que se obtiene de $I - P$, y sustituyendo su última columna por el vector columna unidad $\mathbf{1}$. Entonces, el sistema de ecuaciones anterior es equivalente a $wR = (0, 0, \ldots, 1)$ y, por tanto:

$$w = (0, 0, \ldots, 1)R^{-1}$$

La operación anterior anula todas las columnas de R^{-1} excepto la última, de la que se obtiene finalmente el vector fijo w. En el Listado 14 (pág. 152) puede encontrarse el algoritmo que calcula la distribución estacionaria w de una matriz de transición P.

Método alternativo Haciendo uso del Teorema IV.4, puede derivarse un método diferente para calcular la distribución estacionaria de P.

Éste consiste en elegir arbitrariamente un vector w_0 cuyas componentes sumen 1. Luego, de forma iterativa, y teniendo en cuenta que $w_i \to w$ conforme $i \to \infty$, se calcula el vector $w_i = w_{i-1}P$, hasta que, en una iteración dada j, la distancia entre los vectores w_j y w_{j-1} es menor que algún parámetro predeterminado ϵ.

Este método tiene la ventaja, respecto al presentado anteriormente, de que su implementación resulta más sencilla. Sin embargo, resulta más costoso computacionalmente, pues el número de iteraciones necesario, y las multiplicaciones asociadas, puede hacerse alto si el valor del parámetro ϵ es muy bajo.

IV.4.3. Análisis de los resultados

Una vez obtenida la distribución estacionaria w con cualquiera de los dos métodos presentados, sólo resta utilizarla para calcular la probabilidad de que búfer de salida se agote y, por tanto, se produzca una falta o, por el contrario, de que éste se llene y no pueda admitir más elementos.

Antes de llevar a cabo este cálculo, se presenta el siguiente teorema, esencial en la teoría de cadenas de Markov, y que resulta imprescindible en el razonamiento posterior.

IV.6 Teorema. ***(Ley de los Números Grandes para cadena ergódicas)*** *Sea $H_n^{(n)}$ la proporción de veces en n pasos que una cadena ergódica está en el estado s_j. Entonces, para cualquier $\epsilon > 0$,*

$$P\left(|H_n^{(n)} - w_j| > \epsilon\right) \to 0,$$

independientemente del estado de inicio s_i.

Este teorema proporciona una nueva interpretación del significado del vector fijo. Por ejemplo, este teorema predice que teniendo en cuenta el vector fijo obtenido en el Ejemplo IV.5, $w = (0'4, 0'2, 0'4)$, a largo plazo y en término medio, la cadena pasará el 40 % del tiempo en el estado 0 (**Ll**uvia), el 20 % en el estado 1 (**B**uen **T**iempo) y el 40 % restante en el 2 (**N**ieve).

En general, según esta Ley, a largo plazo la cadena pasará una fracción w_j del tiempo en el estado s_j. Por tanto, comenzando en cualquier estado, la cadena siempre alcanzará cualquier otro estado s_j; de hecho, lo hará un número infinito de veces.

Aplicando este punto de vista a la modelización del búfer, el vector fijo contiene la proporción del tiempo que la cadena pasa en cada estado. El estado w_0 corresponde a un búfer vacío, por lo que el sistema pasará el w_0 % del tiempo "agotado". Idéntico razonamiento sirve para el estado de búfer lleno, w_B.

Como consecuencia de la Ley anterior, *se demuestra que la probabilidad de que el búfer se agote, o de que no se llene, nunca puede hacerse nula*. Sin embargo, también se mostrará que, ajustando de forma adecuada los parámetros del tamaño del búfer y el ratio de funcionamiento, puede conseguirse que esta probabilidad se arbitrariamente pequeña de forma que, a efectos prácticos, pueda considerarse despreciable.

IV.4.3.1. Agotamiento del búfer

Teniendo en cuenta lo dicho anteriormente, la probabilidad final que buscamos desde el inicio de este análisis corresponde, simplemente, a la primera componente del vector fijo, w_0. Por tanto, *la probabilidad de falta para un sistema con tamaño de búfer B y ratio de funcionamiento α viene dada por $M(B, \alpha) = w_0$.*

Por otro lado, la probabilidad de que el búfer no se agote corresponde a "la proporción del tiempo que la cadena *no pasa* en el estado s_0". O lo que es lo mismo, dado que las componentes de w suman 1, esta probabilidad es igual a:

$$\sum_{i=1}^{B} w_i = 1 - M(B, \alpha)$$

De esta forma, resolviendo numéricamente la expresión anterior, utilizando por ejemplo el algoritmo presentado en el Listado 14 (pág 152), puede darse respuesta, por fin, a los dos problemas planteados en la sección IV.4.2, a saber:

- **Problema 1**: Partiendo de que el sistema funciona con un ratio de funcionamiento α, ¿cuál es el tamaño mínimo B del búfer que asegura que la probabilidad de una falta puede hacerse menor que un parámetro ϵ, arbitrariamente pequeño?
- **Problema 2**: Dado que el tamaño del búfer utilizado es B, ¿cuál es el ratio mínimo de funcionamiento que garantiza que la probabilidad de falta pueda, igualmente, hacerse arbitrariamente pequeña?

Las respuestas serán detalladas a continuación.

Probabilidad de agotamiento La solución numérica, y la obtención del vector fijo w, proporciona, por tanto, las probabilidades de falta, o agotamiento del búfer, para diferentes tamaños de éste y ratios de funcionamiento α. Los datos que pueden encontrarse en el Cuadro IV.8.

Evidentemente, para valores de $\alpha = 1 : 1$, que equivale a la definición teórica original del generador shrinking, el sistema fallará con probabilidad 1, es decir, $M(B, \alpha =$

Tamaño búfer B	Ratio α							
	1.0	1.25	1.33	1.6	1.66	2.0	2.5	3.0
8	1.0	0.9624	0.9126	0.7639	0.6315	0.1111	$7.9516 \cdot 10^{-4}$	$7.7786 \cdot 10^{-6}$
16	1.0	0.9624	0.9126	0.7639	0.6312	0.0588	$1.1060 \cdot 10^{-6}$	$7.5023 \cdot 10^{-11}$
24	1.0	0.9624	0.9126	0.7639	0.6312	0.0400	$1.5391 \cdot 10^{-9}$	$7.2359 \cdot 10^{-16}$
32	1.0	0.9624	0.9126	0.7639	0.6312	0.0303	$2.1418 \cdot 10^{-12}$	$6.9790 \cdot 10^{-21}$

Cuadro IV.8: Probabilidades de falta (agotamiento del búfer) para distintos valores del tamaño de búfer B y ratio de funcionamiento α.

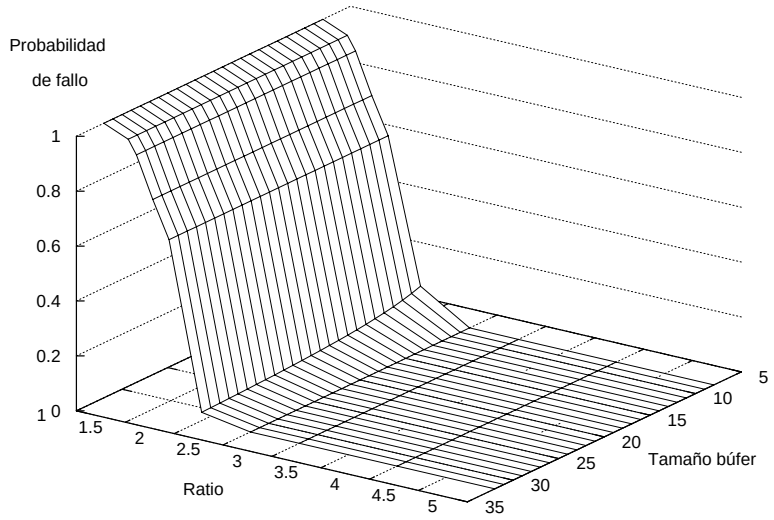


Figura IV.11: Relación entre el tamaño de búfer de salida, ratio de funcionamiento y probabilidad de falta.

1) = 1. Por supuesto, éste es un resultado ya conocido, y reconocido por los propios autores, pero el método que hemos presentado permite obtener una justificación formal del mismo.

Los resultados, sin embargo, se vuelven más interesantes conforme aumenta el ratio α. En ese caso, se observa claramente cómo desciende gradualmente la probabilidad de falta. De hecho, lo hace de forma abrupta a partir de un ratio de 1.25. Este efecto puede observarse más claramente en la Figura IV.11, que muestra de forma gráfica la relación entre tamaño de búfer, ratio y probabilidad de falta.

Como cabría esperar, en general, si aumenta el tamaño del búfer, también disminuye la probabilidad de falta. Sin embargo, este no es el caso para valores de $\alpha < 2$, donde un aumento del tamaño del búfer no reduce la probabilidad de falta, o lo hace en una cantidad despreciable.

La razón estriba en el hecho de que, para esos ratios de funcionamiento, las probabilidades de transición desde cualquier estado hacia el estado s_0, que modeliza el agotamiento del búfer, se hacen gradualmente mayores conforme α disminuye. Por esta razón, sea cual sea el tamaño de búfer, finalmente el estado s_0 terminará siendo alcanzado.

Sin embargo, a pesar del efecto recién mencionado, la principal conclusión de los

datos obtenidos es que *el factor determinante de la probabilidad de falta del sistema es el ratio de funcionamiento*. Este parámetro tiene mucha mayor influencia sobre $M(B, \alpha)$ que el tamaño del búfer, de forma que pueden obtenerse probabilidades de falta muy pequeñas con un tamaño de búfer también muy reducido, de tan sólo unas decenas de elementos.

Dado que este tamaño de búfer es perfectamente asumible en cualquier plataforma hardware y software actual, incluso en aquellas que cuentan con una capacidad de cómputo y almacenamiento mínimas, cabría preguntarse si no es posible aumentar el rendimiento del sistema reduciendo el ratio de funcionamiento, aún a costa de tener que aumentar el tamaño del búfer.

Sin embargo, como hemos visto anteriormente, llevar a cabo ese ajuste no es posible, pues para valores de $\alpha < 2$, la probabilidad de falta aumentará de forma exponencial, y este es el punto clave, ***independientemente*** del tamaño del búfer B.

Conclusiones A modo de resumen, podemos remarcar que el ratio de funcionamiento mínimo para un sistema viable en la práctica es $\alpha = 2.5$. Incluso para este ratio, solo son necesarios tamaños de búfer de unos pocos elementos, alrededor de 32, para conseguir probabilidades de falta del orden de $M(32, 2.5) = 2 \cdot 10^{-12}$. Por otro lado, también es posible variar el tamaño del búfer de salida para ajustar la probabilidad de fallo. De hecho, para el ratio anterior, $\alpha = 2.5$, doblar el tamaño del búfer reduce M en un factor de 1 000 aproximadamente.

IV.4.3.2. Estrategias para gestionar una falta

A pesar de que, como se ha visto, las probabilidades de falta pueden hacerse arbitrariamente pequeñas, éstas nunca pueden hacerse cero, por lo que, eventualmente, se producirá algún falta durante el funcionamiento del sistema.

Los autores del generador shrinking, Coppersmith, Krawczyk y Mansour, ya dieron algunas pautas sobre cómo gestionar esta situación en su trabajo seminal [10]. Estos métodos son básicamente los presentados a continuación.

Rellenar los faltas con valores arbitrarios Por ejemplo, con valores '0' y '1' de forma alternativa, si el generador está definido sobre $GF(2)$. En el caso del sistema presentado en este trabajo, se propone comenzar en un valor arbitrario i del cuerpo $GF(2^n)$ correspondiente e incrementar este valor en cada falta, módulo el orden del cuerpo. De esta forma, en el primer falta la salida del sistema sería $i \equiv$ mód 2^n, $i + 1 \equiv$ mód 2^n en el segundo, etc...

Si suponemos que los parámetros del sistema han sido elegidos de forma cuidadosa y que, por tanto, la tasa de faltas es muy pequeña, este método no tiene porqué reducir sensiblemente la calidad de la secuencia cifrante producida. Por supuesto, si la tasa de faltas fuera alta, la secuencia contendría un alto número de elementos "predecibles" y, por tanto, su complejidad lineal caería drásticamente.

Por otro lado, la operación de reducción modular incrementa la complejidad de la implementación, tanto software como hardware, y aunque se espera que se utilice de forma excepcional, es costosa computacionalmente, lo que implicaría una ligera penalización en el rendimiento.

Almacenamiento de elementos descartados Este método, que puede resultar a priori algo más complejo que el anterior, consiste en disponer un segundo búfer donde almacenar algunos de los elementos que se van descartando durante el funcionamiento normal del sistema, con el fin de utilizar en caso de falta.

Este enfoque tiene el problema de que necesita este segundo búfer, aunque si bien éste podría ser de unos pocos elementos y, por tanto, ocupar pocos bytes de almacenamiento. Por contra, proporciona, en principio, una solución más elegante y segura que el método anterior.

Ambos métodos pueden aplicarse por igual al caso del generador shrinking extendido, pero por las razones anteriormente expuestas, los autores recomiendan el uso del segundo método.

IV.4.3.3. Desbordamiento del búfer

Como resulta lógico en un búfer de tamaño finito, existen también la posibilidad contraria al agotamiento, es decir, que éste se llene o desborde. De hecho, teniendo en cuenta el ratio α necesario para que no se produzcan faltas en el sistema de forma sistemática, la situación de desbordamiento se producirá a menudo.

En esta sección trataremos, como ya hicimos en el caso de agotamiento de búfer, de caracterizar esta situación, calcular la probabilidad de que ésta ocurra y analizar las consecuencias cuando se produce.

Probabilidad de desbordamiento De forma análoga al procedimiento utilizado para calcular la probabilidad de falta, la situación de desbordamiento se produce cuando la cadena de Markov utilizada para la modelización y simulación del sistema transita al estado s_B.

Por tanto, la probabilidad de desbordamiento para un sistema con un tamaño de búfer B y un ratio α, que denotaremos por $F(B, \alpha)$, corresponde a la última componente del vector fijo, o w_B. Utilizando los mismos métodos que los utilizados para el cálculo de la probabilidad de falta $M(B, \alpha)$, los valores resultantes pueden encontrarse resumidos en el Cuadro IV.9

Como era presumible, se observa claramente cómo aumenta la probabilidad de desbordamiento conforme lo hace el ratio de funcionamiento, puesto que se generan más elementos en cada ciclo del generador. F empieza a ser considerable a partir de un ratio $\alpha = 2$.

Índice de descarte Con el fin de facilitar el análisis de los resultados, hemos definido una nueva magnitud, que denominaremos *índice de descarte* y denotaremos $D(B, \alpha)$. Este índice pretende dar una estimación de la proporción de elementos generados que no pueden ser almacenados en el búfer porque éste ha alcanzado su máxima capacidad y que, por tanto, deben ser desechados.

La expresión para calcular este índice es la siguiente:

$$D(B, \alpha) = \frac{\text{Núm. elementos descartados}}{\text{Núm. iteraciones}}$$

De la definición anterior se desprende que si $D(M, \alpha)$ es menor que 1, el sistema descarta menos elementos de los que genera. Por contra, si es mayor que 1, se están desechando un mayor número de elementos de los que, finalmente, el generador de secuencia entrega al módulo de cifrado. Claramente esta situación debe evitarse en la medida de lo posible, y se deberá tratar de diseñar un sistema con un índice de descarte tan bajo como sea posible.

Los valores calculados para $D(B, \alpha)$ en el sistema que presentamos pueden encontrarse en la parte baja de el Cuadro IV.9 y representados de forma gráfica en la Figura

Probabilidad de desbordamiento, $F(B,\alpha)$							
	Ratio α						
Tamaño búfer B	1.0	1.5	2.0	2.5	3.0	4.0	5.0
8	0.000	$7.778650 \cdot 10^{-6}$	0.111111	0.631525	0.763933	0.912621	0.962419
16	0.000	$7.502397 \cdot 10^{-11}$	0.058826	0.631525	0.763932	0.912621	0.962419
24	0.000	$7.274135 \cdot 10^{-16}$	0.040000	0.631525	0.763932	0.912621	0.962419
32	0.000	$3.823595 \cdot 10^{-18}$	0.030303	0.631525	0.763932	0.912621	0.962419
Índice de descarte, $D(B,\alpha)$							
	Ratio α						
Tamaño búfer B	1.0	1.5	2.0	2.5	3.0	4.0	5.0
8	0.0000	0.0000	0.034000	0.491000	0.500000	0.993000	1.488000
16	0.0000	0.0000	0.003000	0.488000	0.481000	0.977000	1.485000
24	0.0000	0.0000	0.000000	0.438000	0.533000	0.889000	1.437000
32	0.0000	0.0000	0.023000	0.413000	0.454000	0.925000	1.407000

Cuadro IV.9: Índice de descarte para distintos tamaños de búfer B y ratios de funcionamiento α.

IV.12. Del análisis de estos datos se desprende que la probabilidad de desbordamiento y el índice de descarte están íntimamente relacionados y presentan una correlación directa.

Por otro lado, ocurre de nuevo que, en el caso del agotamiento del búfer, el factor determinante que produce el desbordamiento del búfer reside en el ratio de funcionamiento y no el tamaño del mismo.

Se trata pues, de encontrar un equilibrio entre la tendencia del sistema a agotar y desbordar el búfer, situaciones que se caracterizan con la probabilidad de falta y el índice de descarte, respectivamente. Si, como se ha visto en la sección anterior, se incrementa el ratio de funcionamiento α con el fin de reducir la probabilidad de falta, inevitablemente aumentará la probabilidad de desbordamiento y, con ella, el índice de descarte.

Puntos de equilibrio Para encontrar la situación de equilibrio óptima pueden analizarse los datos mostrados en la Figura IV.13. En ella se muestran los parámetros $M(B,\alpha)$ y $D(B,\alpha)$ en los ejes Y izquierdo y derecho respectivamente, con sus escalas correspondientes. Pueden observase dos puntos de interés, resaltados con dos pequeños círculos y etiquetados como puntos A y B, situados en los puntos de abcisa $\alpha = 2.11$ y $\alpha = 2.5$, respectivamente.

Ambos puntos conforman *puntos de equilibrio del sistema*, o lugares donde ambas curvas se cruzan y, por tanto, no es posible reducir el valor de alguno de los dos parámetros sin incrementar el del otro. Además, ambas magnitudes tienen un comportamiento prácticamente exponencial alrededor de este punto, por lo que pequeños incrementos o decrementos de algún parámetro implicarían un gran cambio en el recíproco.

Sin embargo, es fácil observar como, en el punto de equilibrio A, la probabilidad de falta $M(B, 2.11) = 0.1$, que resulta inaceptablemente elevado. Es necesario, pues, renunciar a este punto como valor óptimo del sistema y evaluar el siguiente punto candidato.

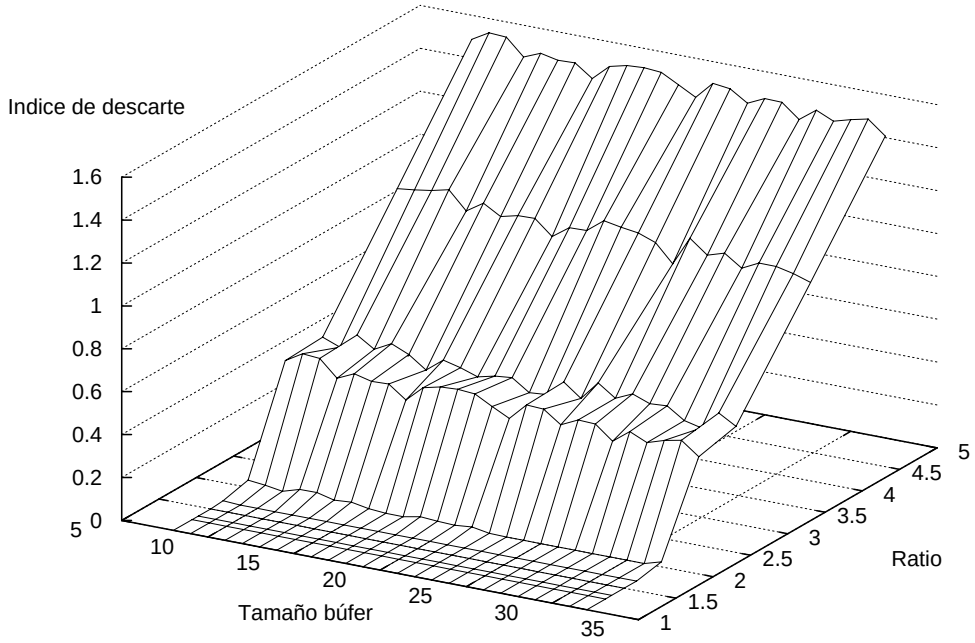


Figura IV.12: Representación gráfica del índice de descarte, $D(B, \alpha)$, para los distintos valores del tamaño de búfer B y ratio de funcionamiento α

El punto B se encuentra en la abscisa $\alpha = 2.5$ y en él la probabilidad de falta comienza a hacerse suficientemente pequeña. Los valores concretos de esta probabilidad son, por ejemplo, de $7.9516 \cdot 10^{-4}$ para un tamaño de búfer $B = 8$ y de $2.1418 \cdot 10^{-12}$ para $B = 32$. El resto de los datos pueden encontrarse en el Cuadro IV.8. Por supuesto, de los valores anteriores, sólo el segundo, que determina un tamaño de búfer mínimo de $B = 32$ elementos resulta aceptable.

A la luz de estos datos, parece claro que el ratio de funcionamiento mínimo del sistema debe situarse por encima de $\alpha = 2.5$, de forma que la probabilidad de falta tome valores aceptablemente bajos. Por otro lado, con el fin de que el búfer se desborde el menor número de veces y, en consecuencia, se desperdicien el menor número posible de elementos, el tamaño mínimo del búfer debe ser superior a $B = 32$ elementos.

Propiedades estadísticas de la secuencia de salida Otro punto interesante que debe ser tratado es el de las consecuencias que, para la secuencia de salida producida por el sistema, tiene el hecho de que el búfer se llene. Como se ha descrito con anterioridad, cuando se produce esta situación, regida por el índice de descarte, el sistema no puede almacenar más elementos y, simplemente, debe descartarlos.

Como consecuencia, la secuencia de salida finalmente producida por el sistema no es exactamente la obtenida del diseño teórico del generador shrinking, sino una que es nuevamente "decimada" en función del tamaño del búfer.

Resulta obvio, sin embargo, que las propiedades estadísticas [10], prácticamente perfectas, de la secuencia *shrunken* se mantienen intactas en este caso. Estas propiedades incluyen, entre otras, una baja correlación entre los elementos de la secuencia, el mismo número de ocurrencias de cada elemento y una distribución uniforme de las subsecuencias [21].

En efecto, una secuencia de propiedades estadísticas perfectas de la que se eliminen

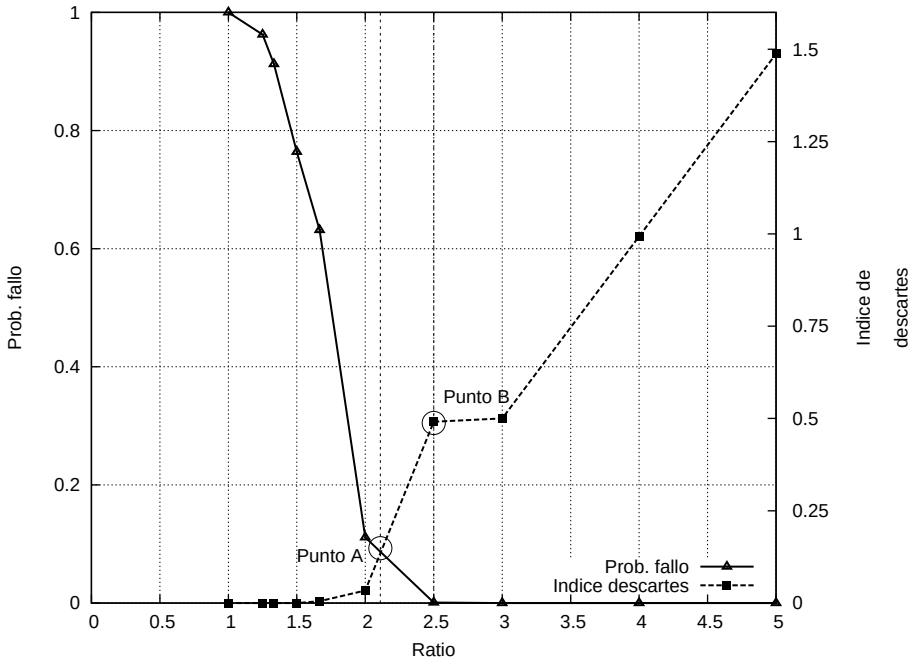


Figura IV.13: Probabilidades de fallo e índice de descarte combinados para un sistema con tamaño de búfer $B = 8$.

algunos elementos, sea cual sea el criterio utilizado, siempre resulta en otra secuencia estadísticamente perfecta. Puede decirse que se obtiene, simplemente, una secuencia estadísticamente perfecta de menor longitud.

El resultado anterior puede demostrarse utilizando un sencillo razonamiento de reducción al absurdo. En efecto, si existiera una transformación que, dada una secuencia de entrada estadísticamente perfecta, produjera una secuencia de salida que no preservara dichas propiedades, significaría que existirían porciones de la secuencia de entrada "privilegiadas" respecto a otras. Este hecho contradice la hipótesis de partida, luego no puede existir tal transformación.

IV.4.4. Contramedidas para el desbordamiento del búfer

Tal y como se ha visto en las secciones anteriores, el desbordamiento del búfer del sistema, sea cual sea el tamaño de éste, es una consecuencia inevitable si se quiere evitar la situación de falta, mucho más peligrosa desde el punto de vista de seguridad y rendimiento global del sistema.

Es importante señalar que otros cifradores que producen también una salida irregular ignoran completamente el problema. Desde el propio generador shrinking, hasta el más reciente Decim [22], candidato del proyecto eSTREAM. En palabras de los propios autores:

> [...] If the ABSG outputs one bit when the buffer is full, then the newly computed bit is not added into the queue, i.e. it is dropped. [...]

Esta estrategia de obviar el problema y, simplemente, descartar el elemento generado produce una evidente pérdida de rendimiento y desperdicio de recursos en-

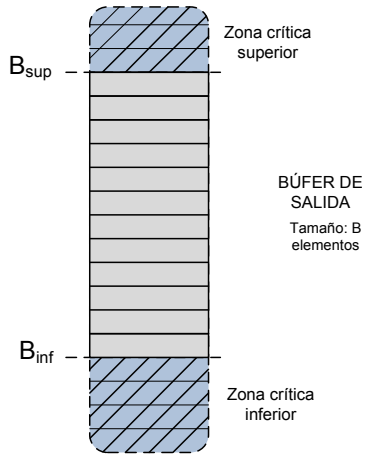


Figura IV.14: Valos umbrales superior e inferior, B_{sup} y B_{inf} respectivamente.

ergéticos. Aunque este hecho puede pasar prácticamente desapercibido en las plataformas más potentes, puede suponer un gasto significativo en las más humildes, habitualmente alimentadas con baterías.

Para evitar este problema se presentan dos nuevas soluciones, que hemos denominado *ajuste dinámico del ratio* y *pausa de generación*, que impiden en ambos casos dicho desbordamiento y sus consecuencias.

IV.4.4.1. Ajuste dinámico del ratio de funcionamiento

La primera de la soluciones resulta conceptualmente muy sencilla: dado que el módulo de generación de secuencia funciona más rapidamente que el módulo de cifrado, un remedio simple consiste en ajustar esta diferencia de velocidad.

De esta forma, reduciendo o aumentando el valor del ratio de funcionamiento α en función de la ocupación del búfer del sistema en cada momento, es posible evitar el agotamiento o desbordamiento del mismo. Durante este tiempo, por supuesto, el módulo de cifrado continuaría funcionando normalmente.

Valores umbrales Los niveles de ocupación a los que deben dispararse el ajuste dinámico del ratio α, se definen como *valores o niveles umbrales superior e inferior*, y los denotaremos como B_{sup} y B_{inf}.

Alcanzar el valor B_{sup} indica que el búfer está cerca de su límite máximo de ocupación y que, por tanto, debe rebajarse el ritmo de producción de elementos. B_{inf}, por el contrario, muestra que el búfer sufre riesgo de vaciarse y producir una falta del sistema. En la Figura IV.14 puede encontrarse una descripción gráfica de estos valores.

En general, los valores óptimos para el parámetro B_{sup} dependen del ratio de funcionamiento α y del tamaño de búfer B. De forma intuitiva, cuanto mayor sea el valor de α, menores pueden ser tanto B_{inf} como B_{sup}, puesto que la probabilidad de falta será menor y, por tanto, el valor de B_{inf} podrá acercarse más a cero sin posibilidad de que, finalmente, lo alcance.

Por otro lado, otro aspecto que debe ser tenido en cuenta es que los valores óptimos de estos parámetros deben ser tales que permitan minimizar el tiempo total que el módulo de generación de secuencia está en funcionamiento. De esta forma, en dispositivos embebidos o de baja capacidad de cómputo se reduce considerablemente el consumo total debido al proceso de cifrado.

Tomando en consideración los valores aconsejados para el ratio de funcionamiento α y tamaño de búfer B, $\alpha = 2.5$ y $B = 32$ elementos, respectivamente, cuya justificación puede encontrarse en la sección IV.4.3, los valores óptimos para B_{sup} y B_{inf} se sitúan en torno al 80 % y 20 %, respectivamente.

Funcionamiento El incremento o decremento del ratio debe hacerse dentro de unos límites pues, de lo contrario, el ratio crecería sin límite o se haría cero cuando el búfer estuve cerca de agotarse o desbordarse, respectivamente.

Este límite superior e inferior para el ratio α se denotarán como α_{max} y α_{min} y, de acuerdo a los experimentos realizados, sus valores óptimos oscilan entre 6 para α_{max} y 1 para α_{min}.

Presentados los elementos necesarios, el funcionamiento de la solución propuesta puede resumirse finalmente de la siguiente forma:

1. El sistema funciona normalmente hasta que se alcanza uno de los dos umbrales B_{sup} o B_{inf}.
2. Si se trata del umbral B_{sup}, el búfer está próximo a desbordarse. Por tanto, el sistema deberá reducir el ratio α, nunca por debajo de α_{min}, para rebajar el ritmo de producción de secuencia y permitir que el búfer reduzca su ocupación.
3. Por otro lado, si se alcanza el umbral B_{inf}, el agotamiento del búfer resulta probable. En consecuencia, se aumentará el ratio α, no por encima de αmax, con el fin de aumentar la ocupación del búfer y evitar una falta del sistema.
4. En cualquiera de las situaciones anteriores, el módulo de cifrado sigue funcionando con normalidad, sin alterar su ritmo de trabajo, con el fin de producir una salida regular.

De acuerdo a estas pautas, el ratio α varia de forma dinámica durante el funcionamiento del sistema y, por tanto, la ocupación del búfer oscila entre B_{sup} e B_{inf}. En la Figura IV.16 puede encontrarse cómo evoluciona esta ocupación a lo largo del tiempo.

Ajuste del ratio α Por otro lado, el ajuste del ratio α debe realizarse de forma gradual, para evitar un posible *efecto rebote*, que provoque problemas de desbordamientos o agotamientos abruptos.

Sin embargo, incluso cuando el ajuste se realiza de forma conveniente, esta propuesta adolece de ciertos inconvenientes, sobre todo cuando ésta se utiliza en una implementación hardware del sistema.

En ese caso, sería necesario *modular* la señal de reloj del módulo de generación de secuencia, incluyendo valores no enteros, como 2.25 o 2.5. Esto implica un coste adicional en complejidad del diseño, que debería incluir un módulo especial para gestionar estos cambios en el ratio de funcionamiento.

Por otro parte, también provocaría que el módulo de generación y el de cifrado desincronizasen sus relojes y funcionasen, por tanto, de forma asíncrona. Para evitar estos inconvenientes se propone utilizar en las implementaciones hardware, exclusivamente, ratios α de valor entero,

Afortunadamente, el ajuste dinámico de α resulta más sencillo cuando la implementación del cifrador se lleva a cabo en software. En ese caso, basta con ajustar convenientemente el número de iteraciones de los bucles que gobiernan los ratios a y b, tal y como se describió en la sección IV.4.1.1. De acuerdo a estas sencillas directrices, el valor de α se ajusta de forma efectiva a la ocupación del búfer. En la Figura IV.15 puede encontrarse cómo evoluciona el valor del ratio durante una ejecución real del sistema.

Por último, en el Listado 15 puede encontrarse cómo quedaría la implementación final de este método de ajuste.

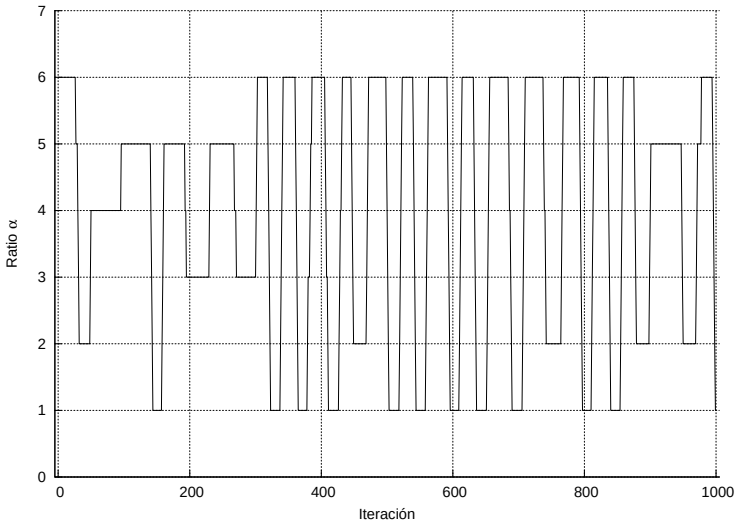


Figura IV.15: Evolución del valor del ratio α durante 1 000 ciclos del generador ($\alpha_{max} = 6, \alpha_{min} = 1$)

IV.4.4.2. Pausa de generación

El segundo método que presentamos puede considerarse, en cierta manera, una simplificación de la solución anterior. En este caso, la idea consiste en hacer que el generador funcione únicamente cuando sea necesario para evitar el agotamiento del búfer. Podría verse como un caso particular del ajuste dinámico del ratio, en el que el ritmo se reduce hasta hacerlo cero en el caso de que el búfer estuviese próximo a desbordarse.

El objetivo de este método es proporcionar tiempo al búfer de salida para que reduzca su nivel de ocupación, de forma que, finalmente, no se produzcan descartes de elementos. Una vez que la ocupación del búfer baje a niveles intermedios, el módulo de generación puede reactivarse, de forma que tampoco se produzcan, por supuesto, faltas en el sistema. De esta forma, el módulo de generación funcionaría en un régimen discontinuo, alternando su activación o desactivación en función del nivel de ocupación.

Valores umbrales En esta solución son también necesarios los dos valores umbrales definidos para la propuesta anterior aunque, en este caso, toman significados ligeramente diferentes.

B_{sup} hace referencia ahora al valor de ocupación del búfer que, una vez alcanzado, provocaría la desactivación total del módulo de generación de secuencia. Por otro lado, B_{inf} indica el nivel al que deberá reactivarse dicho módulo, con el fin de que no se produzcan faltas.

Debido a su modo de funcionamiento, el valor óptimo para B_{sup} puede ser más cercano, en esta caso, al 100 %. En efecto, dado que la generación de elementos se interrumpe completamente y, por tanto, existe la seguridad de que el búfer no se desbordará en los próximos ciclos, el valor de B_{sup} puede ser tan alto como un 99 %. Por otro lado, el valor adecuado de B_{inf} no sufre cambios y se sitúa, de nuevo, en torno al 20 %.

Sin embargo, como se analizará a continuación, el nivel B_{sup} nunca se alcanza en la práctica, debido al modo de funcionamiento del generador.

Funcionamiento A grandes rasgos, el generador sólo funciona cuando la ocupación del búfer baja por debajo del nivel crítico B_{inf} y, por tanto, corre riesgo de agotarse. Cuando la ocupación se recupera y sale de la zona crítica, el generador de nuevo, se para. De esta forma el generador funciona de forma óptima, generando elementos únicamente cuando éstos son estrictamente necesarios.

La operación de esta nueva solución se resume a continuación:

1. En cada ciclo de funcionamiento se comprueba si se ha alcanzado uno de los dos umbrales B_{sup} o B_{inf}.
2. Si se trata del umbral B_{sup}, el búfer está cerca de su máxima ocupación. En ese instante, se desactiva el módulo de generación de secuencia, con el fin de permitir la recuperación del búfer. A pesar de que esta situación no se presenta en la práctica, esta comprobación debe estar presente en cualquier caso.
3. Si se ha alcanzado la zona crítica por debajo de B_{inf}, el búfer está próximo a agotarse. En consecuencia, durante este ciclo, se activará el funcionamiento del módulo de generación.
4. Por supuesto, mientras tanto el módulo de cifrado sigue funcionando con normalidad.

De esta forma, el nivel de ocupación del búfer irá variando entre los límites B_{sup} y B_{inf}, conforme el módulo de generación se active o desactive. Como consecuencia, el nivel de ocupación del búfer se mantiene siempre en torno al nivel B_{inf}, con poca variación respecto al mismo. En la Figura IV.16 puede encontrarse una ejecución real del sistema utilizando este método.

Dado que esta propuesta, conceptualmente más sencilla, no necesita de un ajuste dinámico del ratio, la implementación del bucle principal del módulo de generación resulta también más simple, como puede observarse en el Listado 16.

IV.4.4.3. Análisis de los resultados obtenidos

A lo largo de las secciones anteriores se han presentado dos métodos eficaces y noveles para evitar el desbordamiento del búfer de salida del sistema, que se producirá de forma inevitable si no se hace uso de éstos.

Es importante señalar, no obstante, que estos métodos pueden aplicarse a cualquier generador de secuencia con salida irregular, y no sólo al cifrador propuesto en esta tesis.

Aunque ambos métodos consiguen su objetivo, su funcionamiento difiere sensiblemente. El primero de ellos, la adaptación dinámica del ratio α, se basa en el ajuste del valor de dicho ratio en función del nivel de ocupación del búfer, incrementándolo o reduciéndolo según las necesidades. Como consecuencia, el nivel de ocupación oscila de forma notable, con grandes picos entre los valores B_{sup} y B_{inf}.

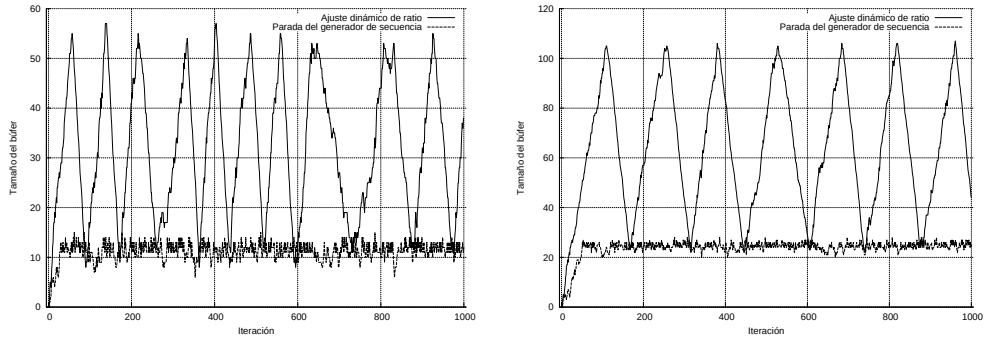
Por otro lado, el segundo método utiliza una estrategia diferente: hacer funcionar el generador sólo cuando existe riesgo de agotamiento del búfer. De esta forma, sólo se generan elementos cuando el nivel de ocupación desciende por debajo de B_{inf}. Este enfoque tiene la ventaja evidente de que minimiza el uso del generador, lo que se refleja en un ligero aumento del rendimiento del mismo y en una reducción muy considerable en la eficiencia energética, que se sitúa en torno al 20 %.

Otra ventaja de este segundo método es que, al disponer sólo de un valor umbral, su implementación, tanto en software como hardware, resulta también algo más simple y necesita menos memoria y recursos para su ejecución.

Asimismo, las oscilaciones en el nivel de ocupación del búfer son menos abruptas que en el primer método, reduciéndose progresivamente conforme aumenta el tamaño

Método	Ahorro en ciclos de reloj	Ocupación mínima y máxima del búfer
Adaptación dinámica de α	19.62 %	12.50 %-89.06 %
Parada del generador	18.80 %	12.50 %-20.31 %

Cuadro IV.10: Datos experimentales para ambos métodos anti-desbordamiento. Los parámetros utilizados fueron: tamaño del búfer $B = 64$, $\alpha_{max} = 6$ y α_{min} para el primer método y $\alpha = 2.5$ para el segundo.



(a) Tamaño del búfer $B = 64$ y $\alpha = 2'5$

(b) Tamaño del búfer $B = 128$ y $\alpha = 2'5$

Figura IV.16: Evolución del nivel de ocupación del búfer del sistema con el uso de las dos medidas anti-desbordamiento presentadas.

del búfer. Este efecto se produce en ambos métodos, y provoca comportamientos ligeramente erráticos cuando el tamaño del búfer es pequeño, del orden de 32 elementos o menos. Cuando el tamaño crece por encima de los 64 elementos el sistema alcanza estabilidad, y el efecto es inapreciable.

Por último, el Cuadro IV.10 resume los datos experimentales obtenidos sobre ambos métodos.

IV.4.5. Conclusiones

A la vista del análisis realizado y los datos experimentales obtenidos en las secciones previas, la primera de las conclusiones es que el diseño original del generador shrinking, tal y como se propuso por sus autores en [10], resulta impecable en su planteamiento teórico pero adolece de importantes inconvenientes cuando se lleva a la práctica.

El principal de estos inconvenientes es que debe utilizar un búfer conectado al módulo generador para ocultar su salida irregular. Con el fin de evitar su agotamiento, lo que introduciría debilidad criptográfica en la secuencia generada, los autores propusieron incrementar la velocidad de funcionamiento del módulo generador respecto al de cifrado.

Las consecuencias de este incremento habían permanecido inexploradas hasta la fecha. Los análisis realizados en este trabajo indican que, si no se toman medidas, el búfer se desbordará inevitablemente debido a dicho aumento de ratio. En otras palabras, *no resulta posible evitar el vaciado y desbordamiento del búfer simultáneamente*. O bien

se reduce el ratio para aumentar el rendimiento y evitar que se llene, con lo que se producen faltas, o bien se aumenta el ratio para evitar estos faltas y entonces, inevitablemente, el búfer se desbordará.

El desbordamiento del búfer, al igual que su agotamiento, debe evitarse a toda costa, pues acarrea una serie de inconvenientes muy importantes:

- *Se reduce notablemente el rendimiento*, pues se generan multitud de elementos que, posteriormente, son descartados y no se aprovechan en el módulo de cifrado. Su generación, por tanto, ha resultado inútil y completamente prescindible.
- Los elementos descartados provocan que *el periodo de la secuencia generada no sea exactamente el estimado de forma teórica*, pues hay una gran cantidad de elementos que no llegan a formar parte de la secuencia cifrante. Obviamente, el periodo real es menor que el teórico y, además, resulta difícil de calcular en la práctica, debido a la multitud de factores que afectan al valor final del mismo.
- Por último, el desbordamiento provoca que *la secuencia cifrante no sea exactamente la secuencia shrunken*, que ha sido bien estudiada y analizada, sino que ésta es nuevamente decimada, en función de parámetros como el tamaño del búfer o el ratio de funcionamiento. En cualquier caso, hemos demostrado que este hecho no debería, en principio, afectar a la seguridad criptográfica de la secuencia generada.

Afortunadamente, se han presentado nuevos métodos con los que sí resulta posible evitar estos problemas. Éstos consisten en medidas que evitan el desbordamiento del búfer, y se basan en la gestión del módulo de generación de secuencia.

Dado que su uso resulta imprescindible, los datos sobre el funcionamiento del búfer, tales como su tamaño o ratio de funcionamiento, se convierten en un parámetro más del sistema del cifrado. Aunque no es necesario que éstos se mantengan en secreto, sí deben intercambiarse entre ambos comunicantes antes de comenzar la comunicación. Esto no supone mayor problema, pues existen muchos algoritmos de cifrado con ciertos parámetros configurables, como la longitud de clave variable de AES [5].

Valores óptimos de los parámetros del sistema De acuerdo a los datos obtenidos en las secciones IV.4.1 y IV.4.2, puede concluirse que el ratio de funcionamiento mínimo para un sistema viable en la práctica es $\alpha = 2.5$. Combinando el valor de α con unos tamaños adecuados del búfer, pueden conseguirse probabilidades de agotamiento del mismo prácticamente nulas.

En cuanto este tamaño de búfer necesario, llama la atención que, incluso para el ratio recomendado, sólo es necesario un tamaño de búfer B de unos pocos elementos, alrededor de 32, para conseguir probabilidades de falta del orden de $M(32, 2.5) = 2 \cdot 10^{-12}$. Además, doblar el tamaño del búfer reduce dicha probabilidad de falta M en un factor de 1 000 aproximadamente.

Medidas anti-desbordamiento Para evitar el desbordamiento del búfer y los problemas asociados, se han propuesto y analizado dos medidas preventivas: el ajuste dinámico del ratio α y la pausa en la generación de elementos. Analizadas ventajas e inconvenientes de cada propuesta, se llega a la conclusión de que, aunque pueden utilizarse ambos métodos, resulta recomendable la segunda opción, debido a su mayor sencillez de funcionamiento e implementación, y su mayor eficiencia energética.

Gracias al uso de estas medidas, además de conseguir evitar las consecuencias indicadas anteriormente, es posible incrementar el rendimiento de las implementaciones en alrededor de un 20 %, gracias a que se optimiza al máximo el uso del generador de secuencia.

Periodo de la secuencia de salida Como se ha mencionado, resulta importante aclarar que que el periodo de la secuencia producida por un generador de secuencia real basado en funciones de decimación, **que no haga uso de las medidas anti-desbordamiento propuestas en este trabajo**, es siempre menor que el establecido teóricamente. Estas estimaciones únicamente resultan válidas para un sistema "sobre el papel", que funciona sin limitaciones de memoria ni tiempo y, en la práctica, sólo pueden considerarse como un límite superior.

En la práctica, la implementación de cualquier sistema que hago uso de funciones de decimación, como el generador shrinking o, en general, cualquier generador de secuencia con salida irregular se encontrará, ineludiblemente, con las dificultades analizadas en este trabajo.

Desgraciadamente, derivar las expresiones analíticas que determinen con exactitud el periodo de la secuencia de salida puede resultar imposible en la práctica. Este valor, en un sistema real, depende de multitud de factores, como el ratio de funcionamiento, el índice de descarte, del tamaño de búfer o del contenido inicial de los registros.

Afortunadamente, la solución para evitar esta dificultades es sencilla y consiste en utilizar alguna de las medidas para evitar el desbordamiento. De esta forma no se pierden elementos y se tiene la seguridad de que las estimaciones para el periodo de la secuencia resultantes son fiables.

Algoritmo 14 Algoritmo de cálculo de la distribución estacionaria w de X

```
...
static int B=32;
static int a=5;
static int b=2;

// Cálculo de la matriz de transición
public static Matrix getMarkovMatrix(){
    Matrix P = new Matrix(B+1,B+1);
    int i,j,x;
    double v=0;

    for (i=0; i<=B; i++){
        for (j=0; j<=B;j++){
            v=0;
            if (j==0){
                for (x=0; x<=b-i; x++){
                    v+=binomialCoefficient(a,x);
                }
            } else if (j==B){
                for (x=B+b-i; x<=a; x++){
                    v+=binomialCoefficient(a,x);
                }
            } else if ((j-i)>(a-b) || (j-i)<-b){
                v=0;
            }
            else{
                v=binomialCoefficient(a,j+b-i);
            }
            v*=Math.pow(0.5, a);
            P.set(i,j,v);
        }
    }

    return P;
}

public static void main(String[] args) {
    Matrix P, Q, R, S;

    P = getMarkovMatrix();
    Q = Matrix.identity(B+1, B+1).minus(P);
    Q.setMatrix(0, B, B, B, new Matrix(B+1,1,1.0));
    R=Q.inverse().getMatrix(B,B,0,B);
}
```

Algoritmo 15 Ajuste dinámico del ratio de funcionamiento α.

```
1  //  Est á el búfer cerca de desbordarse?. Si es así, reducir el ratio a.
2  if ((tama o_búfer > b_sup) && (ratio_a > ratio_a_min))
3    ratio_a--;
4
5  //  Est á el búfer cerca de agotarse?. Si es así, aumentar el ratio a.
6  if ((tama o_búfer < b_inf) && (ratio_a < ratio_a_max))
7    ratio_a++;
8
9  // Bucle principal del módulo de generación
10 for (x=0; x < ratio_a; x++)
11   <<código del generador de secuencia>>
12 for (j=0; j < b; j++)
13   <<código del módulo de cifrado>>
```

Algoritmo 16 Implementación del método de pausa en la generación de secuencia

```
1  // El m'odulo de generaci'on s'olo funciona si el el búfer est'a cerca de agotarse.
2  if (tama o_búfer < b_inf){
3    // Bucle principal del módulo de generación
4    for (x=0; x < a; x++)
5      <<código del generador de secuencia>>
6  }
7
8  for (j=0; j < b; j++)
9    <<código del módulo de cifrado>>
```

IV.5. MÓDULO DE CARGA DEL VECTOR DE INICIALIZACIÓN

Los algoritmos de cifrado en flujo se han utilizado tradicionalmente en entornos donde la información se procesa de acuerdo a ciertas pautas:

- Los datos a cifrar se generan de forma continua, con una cierta tasa fija o variable.
- El medio de transmisión suele ser "ruidoso", por lo que se producen a menudo pequeñas pérdidas de información, que no pueden ser recuperadas, y que deben ser gestionadas correctamente por el proceso de cifrado.

Ejemplos de estos entornos incluyen mayoritariamente el cifrado de la voz, con los originales secrófonos y la más reciente telefonía móvil [14].

Por otro lado, resulta inconcebible diseñar un nuevo algoritmo de cifrado en flujo que tenga dificultades para operar en Internet, dada su ubicua presencia en todos los aspectos de la vida moderna. Sin embargo, Internet es una *red de conmutación de paquetes*, muy diferente a las redes tradicionales de telefonía, por lo que su llegada obligó a los algoritmos en flujo a adaptarse al nuevo entorno, incluyendo, por ejemplo, un *mecanismo de sincronización*, que sea capaz de hacer frente a los problemas inherentes de este tipo de redes:

- *Pérdida de paquetes*: en determinadas circunstancias una pequeña parte de los paquetes transmitidos pueden perderse o corromperse, de forma que no alcancen su destino.
- *Distintas latencias*: la comunicación puede tener diferentes latencias entre las partes, dependiendo de la congestión de la red en cada momento.
- Paquetes *fuera de orden*: los paquetes pueden seguir diferentes rutas y llegar desordenados a su destino.

Aunque estos problemas son corregidos de forma transparente por el protocolo del nivel de transporte (TCP), es necesario que un algoritmo de cifrado robusto sea capaz de hacerles frente, pues existen multitud de escenarios donde el protocolo TCP no actúa o no resulta adecuado.

Cifradores en flujo síncronos Como se introdujo brevemente en la sección IV.1, los cifradores en flujo *síncronos*, como es el caso del algoritmo que presentamos en esta tesis, actualizan su estado interno independientemente del texto en claro que están procesando. Las consecuencias de este planteamiento son importantes:

- El generador de secuencia se convierte en una máquina finita de estados que actúa como un *generador de número aleatorios criptográficamente seguro*. De esta forma se convierte en una alternativa práctica al incondicionalmente seguro (pero nada práctico) cifrador de Vernam.
- No existe la *propagación de errores*: si un símbolo del texto en claro es modificado debido a errores en la transmisión, sólo ese dígito será descifrado incorrectamente, sin afectar a otros símbolos. Ésta es, sin duda, una característica muy deseable para, por ejemplo, las comunicaciones inalámbricas, donde estas modificaciones aleatorias en los datos intercambiados son comunes debido al ruido inherente del medio de transmisión.

- *Modelo de seguridad particular*: el hecho de que el texto en claro no afecta al estado interno del generador de secuencia hace que el análisis de la seguridad del mismo sea radicalmente distinto al de cifradores en flujo auto-sincronizantes o cifradores en bloque. En efecto, para estos dos últimos esquemas, el atacante puede hacer uso de los ataques de texto claro y cifrado escogido, donde puede influir en el estado interno del cifrador o en sus variables intermedias. Sin embargo, estos ataques se tornan inútiles para un cifrador en flujo síncrono, para el que el único ataque factible es el de texto en claro conocido.

Debido a estas buenas propiedades, los cifradores en flujo síncronos son mayoría. De hecho, en el reciente proyecto eSTREAM, 31 de los 34 candidatos aceptados para evaluación se engloban dentro de esta categoría.

Sin embargo, este tipo de cifradores adolecen de un inconveniente importante, que cobra máxima importancia en una red como Internet, pues está relacionado con la sincronización en la comunicación. Como resulta claro, emisor y receptor deben estar perfectamente sincronizados para que el proceso de descifrado funcione correctamente. De otra forma, en cuanto un símbolo se añada o elimine durante la transmisión, las partes en comunicación perderán su sincronía y el resto del mensaje se descifrará de forma incorrecta.

Para solucionar este problema se suele utilizar un *mecanismo de sincronización*, que funciona, a grandes rasgos, como sigue:

- El texto en claro se divide en pequeñas porciones denominadas *tramas*, de tamaño fijo, que se cifran consecutivamente, como se muestra en la Figura IV.17. De esta forma, una trama consiste habitualmente en un número de trama y un bloque de datos cifrados.

- Antes del cifrado de cada trama, el cifrador en el flujo se inicializa con la clave secreta y un *valor o vector de inicialización*, que habitualmente se deriva del contador de trama utilizando alguna función pública (como una función hash). Algo más formalmente, el cifrador se inicializa durante su puesto en marcha con estado inicial de s bits, S_0, y una clave secreta de kc bits, K_c. En la práctica, parámetros S_0 se deriva de K_c, de un vector de inicialización IV de iv bits y de una función $Init$ de la siguiente forma:
$$S_0 = Init(K_c, IV)$$
Es responsabilidad del usuario del cifrador asegurarse de que un par (K_c, IV) no se utiliza nunca para cifrar más de un texto en claro, porque, de otra forma, la seguridad del cifrador, por su naturaleza aditiva, se ve seriamente comprometida.

Un ejemplo típico de un cifrador en flujo síncrono es el algoritmo A5/1 [14], que aún se utiliza en la comunicación GSM en ciertos países europeos. El tamaño de cada trama es de 228 bits, y el vector de inicialización se deriva directamente del contador de trama. También el algoritmo E0, utilizado en el protocolo inalámbrico Bluetooth [23, 24] utiliza un mecanismo de sincronización.

Dificultades del IV A pesar de su aparente simplicidad, incorporar un vector de inicialización a un cifrador en flujo no es, sin embargo, tarea fácil. La razón principal estriba en el hecho de que se trata de un parámetro de entrada, público, que está bajo el control parcial de un atacante, lo que incrementa el número de ataques que el mismo puede llevar a cabo.

Por esta razón, recientemente se han descrito numerosos ataques exitosos contra el mecanismo de IV de cifradores en flujo:

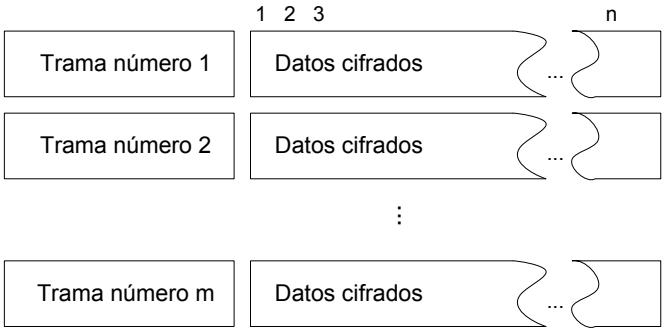


Figura IV.17: Uso de tramas para el mecanismo de sincronización.

- De los 34 cifradores aceptados en el proyecto eSTREAM, 10 fueron descartados por problemas en el proceso de inicialización del IV (véase, por ejemplo, [25, 22, 26]). Esto supone más de un sorprendente 25 % del total de propuestas.
- Otros cifradores, aparentemente fuertes desde el punto de vista académico tradicional, también han tenido problemas con su mecanismo de sincronización, como Turing [27] y Helix [28].
- Por último, incluso cifradores que son ampliamente utilizados actualmente, han sufrido los mismos inconvenientes, habitualmente con consecuencias desastrosas, que han permitido su completo criptoanálisis. Es el caso del estándar GSM [15], o el protocolo de comunicación inalámbrica WEP [29].

El resto de esta sección está organizada como sigue. Comienza detallando, en el apartado IV.5.1, los requisitos deseables de todo mecanismo de sincronización, analizando aspectos como la forma adecuada de generación del IV o la longitud óptima de éste. A continuación, en la sección IV.5.2 se presenta un *módulo de gestión del IV*, que incorpora dichos requisitos y que será incorporado al diseño final de cifrador. Por último, en los apartados IV.5.3 y IV.5.4 se evalúan la seguridad y rendimiento del módulo y, finalmente, se resumen brevemente los puntos más relevantes sobre esta cuestión y se presentan una serie de conclusiones.

IV.5.1. Requisitos del vector de inicialización (IV)

En este apartado se analizan los diversos requisitos que debe cumplir todo mecanismo de gestión del vector de inicialización para considerarse seguro. Estos requisitos incluyen cuestiones como la *seguridad semántica* de las comunicaciones, la *generación y longitud de los IVs* y otros aspectos.

IV.5.1.1. Seguridad semántica

La *seguridad semántica* hace referencia al hecho de que, en ocasiones, a pesar de que un texto se transmita cifrado, es posible "intuir" su contenido, sobre todo en entornos con poca variabilidad en los mensajes, como ocurre típicamente en los protocolos de comunicaciones.

Por ejemplo, el protocolo 802.11 de comunicaciones inalámbricas hace uso de unos paquetes de señalización, denominados paquetes de *beacon*, cuya longitud y contenidos no suele variar e, incluso, se envían a intervalos regulares. Como consecuencia, si todos ellos se cifran con la misma clave, un algoritmo en flujo tradicional, que no utilice un

IV, produciría el mismo texto cifrado. Por tanto, una vez que se identifica uno, todos los demás pueden ser también fácilmente detectados y, su contenido, intuido, aunque no se haya obtenido la clave de cifrado.

El uso de vectores de inicalización puede utilizarse para mitigar este problema y evitar así que cifrar dos veces el mismo texto en claro con la misma clave produzca el mismo texto cifrado. Por supuesto, deben utilizarse diferentes IV en cada proceso de cifrado. Cuando estos cifradores se utilizan en protocolos reales, el IV suele cambiarse por cada paquete enviado, por lo que, en la práctica, significa que el estado interno del cifrador se modifica constantemente.

IV.5.1.2. Longitud y generación de los IVs

La longitud del IV, y la forma en la que éste se genera, puede tener un gran impacto tanto en la seguridad como en el rendimiento final del cifrador.

Por una parte, si el IV es excesivamente largo se incrementará innecesariamente la longitud de cada paquete, dado que éste debe ser transmitido junto con los datos cifrados. Esto provocará, además, una significativa pérdida de rendimiento y un aumento en el consumo energético, que cobra especial importancia en redes MANet o de sensores.

Por otro lado, si el tamaño del IV es demasiado pequeño, aumenta de forma importante la probabilidad de que éste se repita, lo que acarreará importantes problemas de seguridad, como se ha apuntado brevemente en los párrafos anteriores. En efecto, si el mismo IV se utiliza para cifrar dos textos en claro diferentes, P_1 y P_2, a menudo es posible recuperar los textos en claro originales [30]. La razón es que resulta factible recuperar $P_1 \oplus P_2$, que proporciona mucha información sobre P_1 y P_2 cuando éstos tienen suficiente redundancia. Éste suele ser el caso en protocolos de comunicación, donde buena parte del contenido de los paquetes, como la cabecera, es fija y puede ser fácilmente reconstruida con muy poco información extra.

Llegados a este punto cabe preguntarse, ¿qué longitud mínima debería tener un IV?. Por supuesto, independientemente del modo en que se derive el IV para cada paquete, un IV de n bits se repetirá inevitablemente después de enviar $2^n + 1$ paquetes.

Sin embargo, utilizando malas estrategias para la generación del IV, es fácil que se produzcan repeticiones mucho antes de ese límite superior. Por ejemplo, si cada IV se elige de forma aleatoria, como hacían varios fabricantes en sus productos que implementaban el protocolo WEP[3], de forma estadística, por el mismo principio que la *paradoja del cumpleaños*, la primera repetición se producirá después de sólo aproximadamente $2^{(n/2)}$ iteraciones.

Por estas razones, se recomienda el uso de vectores de inicialización de, al menos, 64 bits. Éste es el caso de algunos cifradores modernos, como Decim, aunque otros, como Dragon, proponen tamaños de 256 bits, que se antojan algo excesivos, sobre todo teniendo en cuenta el incremento que se acarrea en la longitud de cada paquete y en el consumo energético.

En en este sentido, la estrategia más sencilla parece en este caso la más óptima. Nuestro cifrador utilizará, por tanto, un contador de 64 bits que hará las veces de IV y que se incrementará de forma secuencial con cada paquete cifrado. Este IV se incluirá en el paquete enviado, de forma que el receptor pueda recuperarlo y utilizarlo en el proceso de descifrado.

[3] La lista de vulnerabilidades de las implementaciones del protocolo WEP es vergonzosamente larga. Otra de las más importantes es que el tamaño elegido para el IV era excesivamente corto, de sólo 24 bits. De esta forma, tras enviar únicamente 16 millones de paquetes, que resulta una cantidad pequeña en una red con un volumen de tráfico muy alto, se producen inevitablemente repeticiones del IV.

De esta forma es posible cifrar un total de 2^{64} bits utilizando la misma clave. Suponiendo que cada paquete de datos tiene la longitud máxima permitida, alrededor de unos 1500 bytes en Internet, esta cantidad equivale al envío de más de $1'5 * 10^{15}$ paquetes, antes de que sea necesario reemplazar la clave.

IV.5.1.3. Seguridad del mecanismo de inicialización del IV

Comenzaremos por definir qué se entiende por que un mecanismo de inicialización de un cifrador en flujo, que carga el IV y establece el estado inicial del mismo, sea seguro.

Partiremos para ello de las recomendaciones de Rogaway [31], que propone trabajar en un escenario en el que el atacante tenga control sobre el IV. Esto significa que, además de poder elegir libremente el texto en claro, también puede elegir el IV que se utiliza en el cifrado, siempre que no utilice dos veces el mismo valor.

Este proceso se conoce con el nombre de *ataque de IV elegido*, y trata de explotar el hecho de que la mayoría de los cifradores en flujo pueden especificarse con una función iterativa relativamente sencilla. Como consecuencia, si el diseño no ha sido realizado con cuidado, algunos elementos de la secuencia cifrante pueden expresarse como una simple función booleana de la clave secreta K y el IV. En ese punto, el atacante puede recuperar de forma inmediata la clave K, puesto que sólo debe invertir la función que relaciona K, IV y la secuencia cifrante.

Sin embargo, en 2006 Wu y Preneel [28] propusieron un ataque sobre el cifrador Phelix que se basaba en la reutilización del vector de inicialización, rompiendo, en cierta manera, las reglas del juego del modelo anterior.

Como era previsible, este enfoque ha despertado una fuerte oposición entre algunos criptógrafos, como Bernstein [32], que argumentan que si los usuarios no respetan las precauciones y requisitos necesarios para utilizar un algoritmo, entonces no puede culparse a éste de las consecuencias. De hecho, como se ha analizado en las secciones anteriores, cualquier cifrador en flujo aditivo puede romperse fácilmente bajo la asunción de Wu y Preneel y permitiendo la reutilización del IV.

Del debate anterior podemos extraer la siguiente definición informal de la seguridad del mecanismo de inicialización de un cifrador en flujo.

IV.11 Definición. (Seguridad del IV) Asumiendo que el generador de secuencia de un cifrador en flujo es seguro, es una condición suficiente para la seguridad global del cifrador que la salida del módulo de inicialización del par $(clave, IV)$, es decir, el estado inicial, no pueda ser distinguido de una secuencia aleatoria.

En otras palabras, la definición anterior establece que un mecanismo de IV es seguro si "mezcla" adecuadamente los bits de K e IV, de forma que no pueda definirse una función lineal, que es fácilmente invertible, que los relacione con los bits de la secuencia cifrante.

Utilizaremos este criterio como punto de partida en el diseño de nuestro mecanismo de inicialización.

IV.5.1.4. Evaluación del mecanismo de inicialización del IV

¿Cómo podemos, pues, comprobar que la salida del cifrador, tras el proceso de mezcla del IV, no presenta correlaciones y no puede ser distinguida de una secuencia aleatoria?.

Tradicionalmente se han utilizado tests estadísticos para la evaluación de los cifradores en flujo, como nosotros mismos hemos hecho en el capítulo anterior, para la evaluación

de los nuevos registros de desplazamiento extendidos. Sin embargo, la introducción del vector de inicialización en este tipo de cifradores es relativamente reciente, y la metodología habitualmente con estos tests no resulta adecuada para la evaluación del mecanismo de inicialización del IV.

La razón estriba en que el enfoque habitual obvia que un cifrador en flujo equipado con un mecanismo de IV debería ser también inmune a un ataque de IV elegido, como se ha analizado en la sección anterior. Como parte del proyecto eSTREAM, Saarinen [33] ha proporcionado una batería de tests específicamente diseñados para verificar la eficacia del mecanismo de IV.

Estos tests, que serán analizados con mayor profundidad en la sección IV.5.3, tratan de encontrar subconjuntos de bits de entrada que resulten menos "mezclados" durante el proceso de inicialización que otros bits, lo que, habitualmente, resulta más probable al inicio o al final del vector que conforma el IV.

Sorprendemente, muchos de los cifradores presentados a eSTREAM obtuvieron pobres resultados en estos tests, como DECIM, hasta el punto de que dichos resultados provocaron un rediseño del cifrador por parte de los autores y la generación de una segunda versión mejorada del mismo, denominada DECIMv2.

IV.5.2. **Módulo de gestión del IV**

En esta sección se presenta finalmente un módulo de gestión del IV que cumple todos los criterios de seguridad anteriormente citados. Por simplicidad, éste hace uso de las propias estructuras del cifrador, como los registros de desplazamiento y la función de decimación, de forma que su diseño se mantiene simple y eficiente.

El mecanismo de inicialización del IV debe utilizarse en diversas situaciones:

- Durante la *fase de inicialización del cifrador*, antes de que éste esté en disposición de generar secuencia cifrante válida.
- Cada vez que sea necesario *reiniciar el estado del cifrador*, a través del IV, como sucede cuando éste se utiliza en redes de comunicaciones basadas en conmutación de paquetes.
- Cuando, por seguridad, la *vida útil* de la combinación {K,IV} se agote, y se necesario cargar un nuevo par.

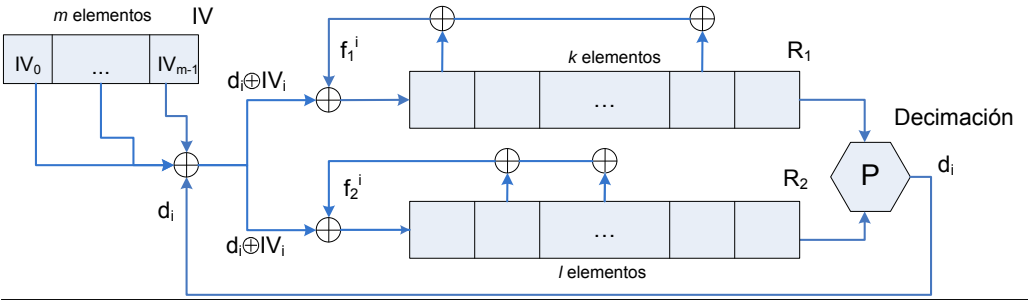
Este proceso ha sido diseñado para resultar eficiente, lo que resulta especialmente importante en las aplicaciones prácticas que requieren una reinicialización frecuente, como las transmisiones móviles e inalámbricas, que suelen utilizar el número de trama o paquete como IV. En la sección IV.5.4 puede encontrarse una comparativa de rendimiento con otros importantes cifradores.

IV.5.2.1. Inicialización del cifrador

Antes de que el cifrador esté en disposición de poder generar secuencia cifrante, la clave secreta K debe ser diversificada, "mezclando" K con el valor del IV.

Definamos la clave K como concatenación de dos cadenas $K_1 = u_0u_1 \dots u_{k-1}$ y $K_2 = v_0v_1 \dots v_{l-1}$, y el vector inicial como $IV = i_0u_1 \dots i_{m-1}$, siendo las longitudes de estas cadenas k, l y m, respectivamente[4].

[4] Aunque se trate exclusivamente de una cuestión de notación, hay que recordar que, aunque estas longitudes se han expresado en bits en este capítulo, tanto las claves como el IV están definidos sobre $GF(2^n)$ y, por tanto, son elementos de, como mínimo, 8 bits de longitud.



Entrada = $\{\mathbf{K}{=}K_1{-\!\!-}K_2,\mathbf{IV}\}$

1. Ambos registros, R_1 y R_2, inicialmente vacíos, se cargan con los valores correspondientes de K_1 y K_2.
2. Los m elementos del IV, $IV_0, \ldots IV_{m-1}$ se introducen en R_1 y R_2 de la siguiente forma:
   Se llevan a cabo m ciclos del cifrador. Así, en el ciclo i-ésimo, se obtendrán de R_1 y R_2 sendos elementos de realimentación, que denominaremos f_1^i y f_2^i, y un elemento de la secuencia cifrante, obtenido a través de la función de decimación, d_i.
   A continuación se calcula $d_i' = d_i \oplus IV_i$ y, posteriormente, tanto $d_i' \oplus f_1^i$ como $d_i' \oplus f_2^i$, que son los elementos que finalmente proporcionan la realimentación de R_1 y R_2.
3. En otras palabras, se realizan m ciclos del cifrador con la conexiones indicadas en la figura superior y con los elementos de realimentación $d_i \oplus IV_i \oplus f_1^i$ y $d_i \oplus IV_i \oplus f_2^i$ con $0 \leq i \leq m$ para R_1 y R_2 respectivamente

Salida = Nuevo estado para R_1 **y** R_2

Figura IV.18: Proceso de inicialización y carga del IV

La carga inicial del IV se realiza de acuerdo al proceso indicado en la Figura IV.18, siendo el número total de combinaciones (K, IV), o estados iniciales del cifrador, de 2^{k+l+m}. Por ejemplo, para una clave secreta típica de 128 bits y un IV de 64 bits, el número de estados es equivalente a $2^{128+64} = 2^{192}$.

Criterios de diseño Las principales características del diseño de este proceso son las siguientes:

- La imprescindible *no linealidad* del método se consigue a través de la propia función de decimación del módulo de cifrado. En la sección IV.5.3 se analizará la seguridad de este enfoque desde el punto de vista estadístico.

- Otra características más importantes de nuestro método de carga, que lo diferencia de otros, como el del Decim o Dragon, es que *permite acomodar IVs de distintas longitudes*. El proceso de carga permanece invariable sea cual sea la longitud del IV utilizado. Éste puede, incluso, variar de forma dinámica, para ajustarse a diferentes aplicaciones, como el cifrado en una red o de documentos que van a ser almacenados *offline*, sin necesidad de ajustar la estructura interna.

- Otro de los criterios esenciales de diseño ha sido la *eficiencia del proceso*, que permitiera su uso en escenarios de frecuente inicialización. Los datos acerca del rendimiento pueden encontrarse en la sección IV.5.4.

IV.5.3. Evaluación de la seguridad

Tal y como se introdujo brevemente en IV.5.1.4, la evaluación de la seguridad del mecanismo de carga del IV resulta difícil, debido a su novedad, y la consecuente falta de experiencia en la comunidad criptológica al respecto, y a la falta de pruebas específicas.

Para tratar de paliar este vacío, Saarinen [33] presentó en 2006 un conjunto de tests específicos, cuyo objetivo esencial es comprobar que la mezcla del IV con la clave K es "homogénea" y se realiza con una función no lineal, de forma que no sea fácilmente invertible.

Estos tests se basan en el hecho de que el funcionamiento de la mayoría de los cifradores en flujo puede ser expresado con una función de iteración sencilla y, por tanto, algunos de los elementos de la secuencia cifrante pueden escribirse como funciones booleanas de la clave K y el IV.

Estos tests sirven para comprobar la resistencia del cifrador a un ataque de IV elegido y, por tanto, mantienen constante la clave K y varían IV, de forma que el cifrador completo puede verse como una "caja negra" con un único parámetro de entrada. Para llevar a cabo este trabajo examinan la *forma normal algebraica* (ANF), en busca de anomalías como elementos redundantes o sesgos importantes.

A continuación se describen brevemente estas pruebas:

Monomios de grado d Basado en el *test de Möbius*, presentado por Filiol [34], esta prueba comprueba si una función booleana dada tiene el número esperado de monomios de grado d. Para $d = 0$, este test se denomina el *test afín*.

En pocas palabras, esta prueba cuenta el número de '1' de la función ANF en cada posición x con un peso de Hamming d. A continuación se aplica un test estadístico χ^2 tradicional para determinar si el número obtenido es excepcionalmente alto o bajo.

Cambio de bits Este test mide el efecto de cambiar uno de los bits de entrada de una función booleana. La prueba puede llevarse a cabo tanto sobre la función f original o sobre su forma normal, y puede utilizarse también el mismo test χ^2 con un grado de libertad que en el test anterior.

Resultados Realizados los tests anteriores sobre el mecanismo de carga del IV presentado, los resultados pueden considerarse muy satisfactorios, no observándose ninguna debilidad estadística que permitiera un ataque de IV escogido.

Es importante señalar que los tests se han realizado para un valor de $m \geq 8$, lo que implica que, para que el proceso de carga sea seguro, deben utilizarse longitudes de IV superiores a los $8 * 8 = 64$ bits.

Conforme decrece el valor de m, el nivel de seguridad también lo hace. Por ejemplo, para valores de $m \leq 5$ (40 bits), los tests pueden distinguir la secuencia de salida de una aleatoria con un margen de error muy pequeño, con una probabilidad $P < 2^{-8}$, aunque para ellos necesitan utilizar un total de 2^{18} IV diferentes.

En cualquier caso, no se recomienda el uso de IVs de longitud menor a 64 bits, porque eso aumentaría considerablemente la posibilidad de que se repitiera el mismo IV durante la vida útil de la clave K, lo que habilitaría el ataque descrito en IV.5.1.2.

IV.5.4. Evaluación del rendimiento del proceso de carga del IV

Por último, se estudia a continuación el rendimiento ofrecido por el mecanismo de carga del IV, comparándolo con los cuatro cifradores ganadores del proyecto eSTREAM en el perfil de software.

Cifrador	Rendimiento proceso carga (ciclos/byte)
Salsa20	34.90
Rabbit	292.38
SOSEMANUK	744.46
LFSRe	1225.33
HC-128	63349.83

Cuadro IV.11: Resultados de rendimiento del proceso de carga del *IV* presentado.

Los datos fueron obtenidos durante la carga de 10^6 IVs, utilizando el procedimiento habitual (10 ejecuciones, descartando los valores extremos mayor y menor y calculando la media aritmética del resto). En el Cuadro IV.11 pueden encontrarse los datos definitivos, expresados en ciclos/byte.

Como se desprende de dichos datos, el proceso de carga de *IV* de nuestro cifrador es muy competitivo, resultando sólo 1.6 veces más lento que uno de los ganadores del eSTREAM, SOSEMANUK, pero, simultáneamente, más de 51 veces más rápido que otro de los candidatos, HC-128. Como puede verse, la variabilidad en el rendimiento de los procesos de carga es altísima entre los distintos cifradores. Por ejemplo, HC-128 es más de 1800 veces más lento que el más rápido de los cifradores en este sentido, Salsa20. Por supuesto, todos los cifradores comparados son de perfil software.

El rendimiento máximo que puede alcanzar el mecanismo de carga está determinado, en última instancia, por la estructura interna del cifrador. Por tanto, resulta difícil poder mejorar sustancialmente la velocidad de carga del VI, debido a que los pilares básicos del diseño del cifrador son los registros de desplazamiento. Éstos son, por definición, de naturaleza lineal por lo que, a pesar del uso de una función de decimación, que introduce el componente no lineal, es necesario realizar un número mínimo de iteraciones para ocultar de forma efectiva la linealidad subyacente.

Factor de resincronización Como se analizado en las secciones previas, es necesario combinar la clave secreta K con un IV único para cada paquete con el fin de evitar la reutilización de la misma secuencia cifrante.

Resulta claro, por tanto, que este proceso de sincronización introduce una sobrecarga inevitable en el sistema, ya que el algoritmo de cifrado debe funcionar durante muchos ciclos de reloj antes de poder producir secuencia cifrante válida. Además, cuanto menor sea el paquete, y la secuencia cifrante necesaria para cifrarlo, mayor es la sobrecarga producida.

Con el fin de cuantificar esta sobrecarga, definiremos un *factor de resincronización* (FR), que es una función de la longitud del paquete, y que toma la siguiente forma:

$$FR = \frac{\text{Número de ciclos total}}{\text{Número de ciclos para la secuencia cifrante}}$$

Para que un cifrador en flujo resulte utilizable en aplicaciones prácticas, debe conseguir un buen equilibrio que lleve a un mecanismo de sincronización seguro y eficiente. En la Figura IV.19 puede encontrarse el FR de nuestro mecanismo, comparado con el otros tres importantes cifradores, como A5/1, E0 y RC4[5].

[5]Para RC4 hemos asumido que los primeros 256 bytes deben ser descartados. Este es un comportamiento común en muchas aplicaciones, que tiene como objetivo evitar ciertas debilidades de diseño.

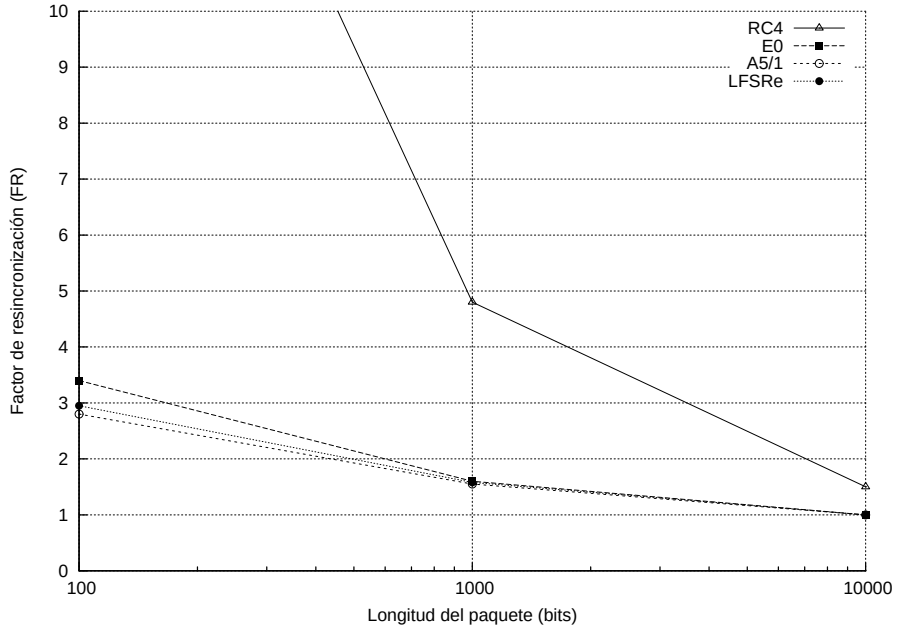


Figura IV.19: Factor de resincronización como función de la longitud del paquete a cifrar para RC4, E0, A5/1 y nuestro cifrador.

Del análisis de la figura pueden obtenerse la siguientes conclusiones:

- El comportamiento del mecanismo de sincronización de nuestro cifrador muestra un *buen rendimiento*. Resulta muy similar al de A5/1 y E0, para todo el rango de tamaños de paquetes.
- El *factor de resincronización es mayor cuanto menor sea la longitud del paquete a cifrar*, siendo realmente importante cuando éste es menor de aproximadamente 1 000 bits. Dado que se estima que la mitad de los paquetes que circulan por Internet son sólo de unos 512 bits de longitud, la eficiencia del mecanismo de sincronización cobra especial importancia en ese tramo.
- Existe una *correlación clara entre el tamaño del estado interno del cifrador y el FR*. Aquellos cifradores con mayor estado interno requerirán, habitualmente, un mayor tiempo para la resincronización.
- Las propiedades de fuerte no linealidad de un buen mecanismo de sincronización resultan muy *similares a las de un cifrador en bloque*. De hecho, SOSEMANUK utiliza un mecanismo de cifrado en bloque para su mecanismo de sincronización.

  Por tanto, antes de poder generar secuencia cifrante válida, un cifrador en flujo debe realizar prácticamente el mismo trabajo que un cifrador en bloque que tenga el mismo tamaño de estado interno. Por tanto, su rendimiento para paquetes pequeños, menores que su tamaño interno, será siempre menor que el de un cifrador en bloque.

  Resulta entonces llamativo el hecho de que los algoritmos Bluetooth y GSM, por ejemplo, utilicen cifradores en flujo para el cifrado de sus comunicaciones, que son siempre pequeñas (2745 bits, como máximo, y 224 bits, respectivamente). Por

otro lado, A5/1 necesita 414 ciclos de reloj para cifrar un paquete, de los que 186, casi la mitad, son empleados en el mecanismo de sincronización.

La razón de que se haya optado en estos casos por un cifrador en flujo, a pesar de su menor rendimiento para estos tamaños de datos a cifrar, puede radicar en el hecho de que, habitualmente, sus primitivas criptográficas básicas se implementan más eficientemente en hardware.

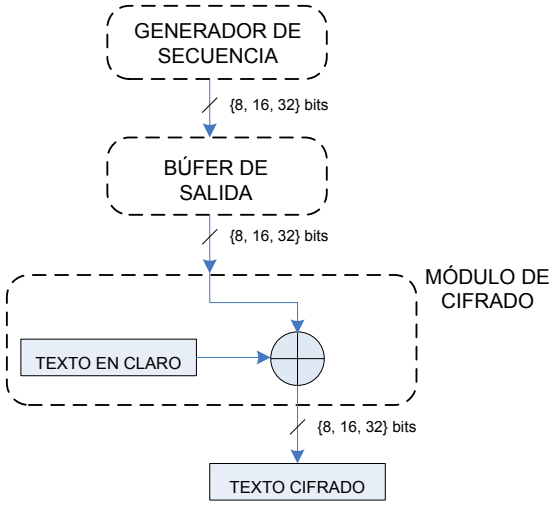


Figura IV.20: Módulo de cifrado de LFSRe.

IV.6. MÓDULO DE CIFRADO Y DISEÑO FINAL

El último de los bloques funcionales de nuestro cifrador está constituido por el *módulo de cifrado*. Dado que se trata de un cifrador aditivo, este módulo es realmente simple, tal y como puede observarse en la Figura IV.20.

El proceso final de cifrado consta únicamente de una operación XOR o suma binaria módulo 2 entre la secuencia cifrante, proveniente del búfer de salida del módulo generador, y los datos en claro. En función del cuerpo algebraico elegido para nuestro cifrador, ambos flujos de datos tendrán longitudes de 8, 16 o 32 bits.

Diseño final Este módulo de cifrado completa el diseño final del cifrador que hemos ido analizando en las secciones anteriores. A modo de resumen, este diseño puede encontrarse de forma gráfica en la Figura IV.21.

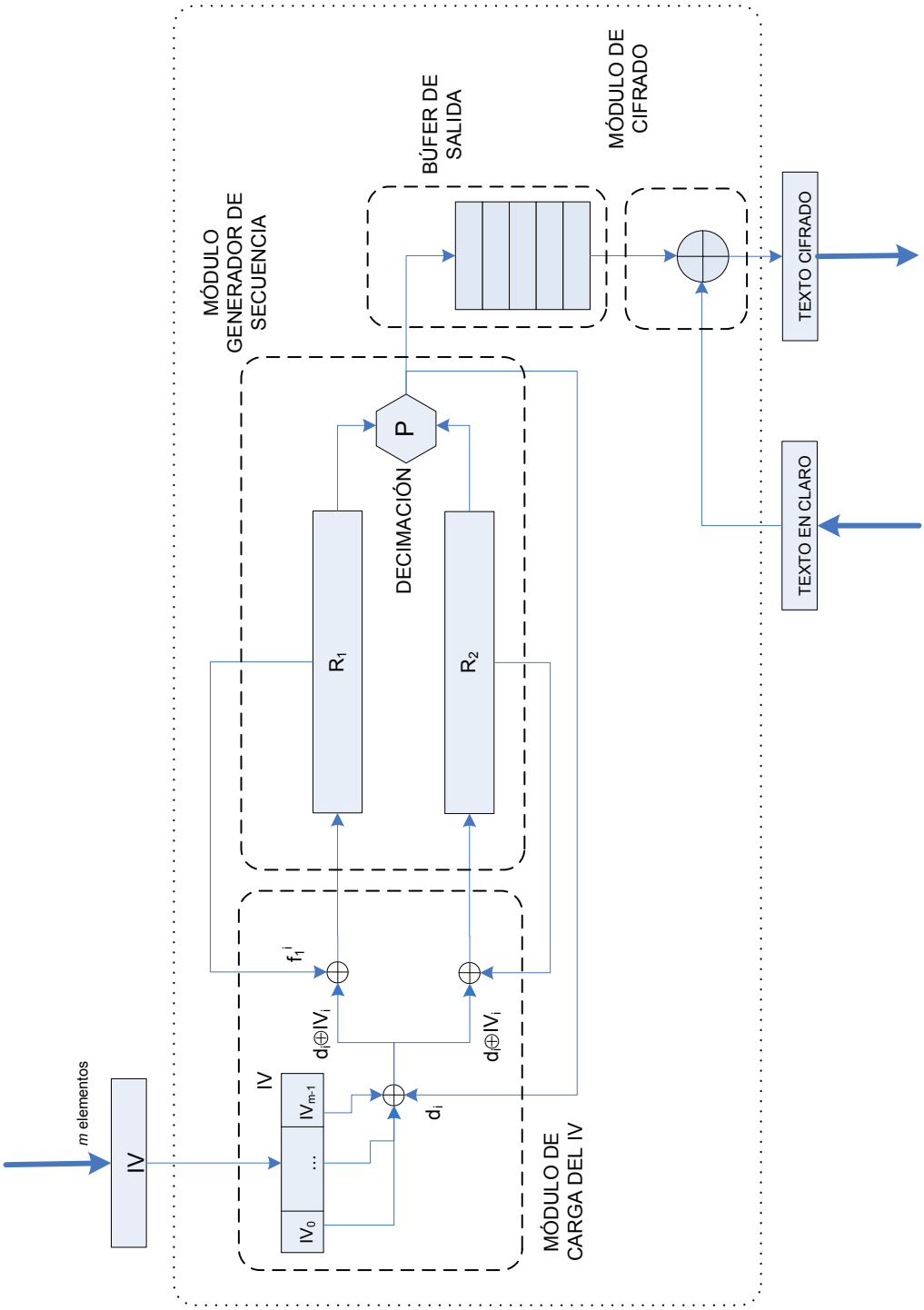


Figura IV.21: Diseño final del cifrador LFSRe.

IV.7. COMPARATIVA Y EVALUACIÓN DEL RENDIMIENTO

Finalmente, en esta sección, una vez completo el diseño del cifrador, se evalúa el rendimiento del mismo comparándolo con el de otros cifradores en flujo de vanguardia. Estos algoritmos son los candidatos del proyecto eSTREAM, ya mencionado, finalmente elegidos como ganadores en la categoría del perfil orientado a software.

También se incluyen otros algoritmos, como RC4, considerado una referencia para contrastar el rendimiento de nuevos algoritmos de cifrado en flujo, o AES-CTR, el modo de flujo del conocido estándar de cifrado en bloque. Por último, también se considera en la comparativa el algoritmo Dragon, que si bien no fue elegido ganador del proyecto, si alcanzó las últimas fases del mismo. La razón de su inclusión es que presenta una estructura interna similar a la de LFSRe, pues hace uso también de registros de desplazamiento, aunque con funciones de realimentación no lineales.

Metodología de evaluación Para llevar a cabo la evaluación del rendimiento, se ha hecho uso del entorno desarrollado para éste propósito por el equipo del proyecto eSTREAM[6]. Este entorno constituye la metodología de evaluación "oficial", por lo que los resultados entre algoritmos son directamente comparables.

Los resultados de la evaluación pueden encontrarse en el Cuadro IV.12. En ella se muestran, además de las longitudes de clave e *IV* utilizados por cada algoritmo, los siguientes parámetros:

- **Velocidad de cifrado en flujo**: mide la velocidad, en ciclos/byte, a la que el cifrador puede procesar un conjunto continuo de datos. No se mide el tiempo que se emplea en cargar la clave ni el IV.
- **Velocidad de cifrado en bloque (40, 576 y 1500 bytes)**: hace referencia a la velocidad de cifrado de multitud paquetes de 40, 576 y 1500 bytes de longitud, respectivamente. Las longitudes no son arbitrarias, sino que han sido elegidas debido a que constituyen las longitudes estadísticamente más frecuentes en la redes de comunicaciones habituales. En este caso, sí se incluye el tiempo de carga y mezcla de la clave e *IV*.
- **Carga IV**: constituye el tiempo empleado en la carga inicial del IV, que se lleva a cabo durante la inicialización del cifrador. Es más compleja y de mayor longitud que la *mezcla del IV*, que se realiza en cada paquete a cifrar, y que se incluye en los tiempos anteriores.

Los diferentes algoritmos aparecen ordenados por su velocidad de cifrado en flujo. Nótese que, a pesar de éste es un criterio importante, sin duda, constituye sólo uno de ellos. De hecho, en ciertos entornos, como en una red de conmutación de paquetes como Internet, las velocidades más significativas son las obtenidas en el cifrado de paquetes, y no el flujo continuo de datos.

Los datos muestran que el rendimiento obtenido por nuestro cifrador puede considerarse excelente, a pesar de que obtener la máxima velocidad no era un objetivo primordial en el diseño del mismo. Se ha pretendido, principalmente, diseñar un algoritmo muy sencillo, con una implementación simple y que haga uso de pocos recursos.

A pesar de ello, LFSRe resulta sólo un 31.5 % más lento que RC4 en el cifrado en flujo. Por el contrario, nuestro cifrador obtiene mejores resultados en el cifrado de paquetes, con un 453 % y 47 % de mejora para paquetes de 40 y 576 bytes, respectivamente.

[6]El entorno está disponible para distintos sistemas operativos y puede descargarse en la siguiente URL: http://www.ecrypt.eu.org/stream/perf/

Algoritmo	Clave	IV	Velocidad de cifrado				Carga IV
			Flujo	40 bytes	576 bytes	1500 bytes	
HC-128	128	128	2.86	587.00	43.19	18.43	23308.78
Salsa20	128	64	3.46	9.48	3.57	3.64	14.29
SOSEMANUK	128	64	5.81	52.37	12.52	9.62	1245.71
Rabbit	128	64	9.46	34.45	11.77	10.76	825.55
Dragon	128	128	11.37	74.09	26.07	23.23	1925.54
RC4	128	–	13.07	445.19	42.27	24.48	16737.80
LFSRe	128	128	17.19	80.45	28.73	24.54	1225.33
AES-CTR	256	128	26.67	37.65	27.09	26.91	59.04

Cuadro IV.12: Resultados experimentales del análisis de rendimiento de distintos cifradores en flujo. Los datos se muestran en ciclos/byte.

También respecto a otra gran referencia en el campo, AES-CTR, el modo más rápido de este estándar, LFSRe obtiene buenos resultados. Como cabría esperar, en el procesamiento de flujos continuos de datos, nuestro cifrador es un 35.5 % más veloz que AES-CTR. Las diferencias se ajustan conforme crece el tamaño del paquete, obteniendo, finalmente, rendimientos similares para paquetes de 1500 bytes.

En resumen, podemos concluir que el rendimiento mostrado por LFSRe, a pesar de su extrema sencillez de diseño, es excelente. Aunque inferior al de los algoritmos más rápidos del proyecto eSTREAM, nuestro cifrador supera claramente a RC4 y AES-CTR en la mayoría de escenarios.

IV.8. CONCLUSIONES

En este capítulo de la tesis se ha presentado un nuevo cifrador en flujo, denominado LFSRe, especialmente orientado a su uso en dispositivos con poca capacidad de cómputo, como los que constituyen las redes de sensores y redes ad-hoc en general.

Por esta razón, el cifrador hace uso de los registros de desplazamientos extendidos, presentados en el capítulo anterior, y los principios del *generador shrinking*. En este sentido, se han presentado y analizado unos nuevos *criterios de decimación*, evaluando los pros y contras de cada uno de ellos.

También se ha profundizado en ciertos aspectos relacionados con este generador en particular, y con los de salida irregular en general, que no habían sido convenientemente aclarados hasta la fecha. Entre ellos se encuentra el problema de su salida irregular, y la necesidad de un búfer de salida con el fin de ocultarla.

Hemos demostrado, a través de la modelización del búfer con cadenas de Markov, que utilizando el planteamiento original del generador shrinking, es imposible evitar, simultáneamente, el agotamiento y el desbordamiento de dicho búfer. Como resultado, se ha obtenido asimismo que el ratio α de funcionamiento mínimo del sistema debe ser superior a $\alpha = 2.5$, con el fin de que la probabilidad de falta tome valores aceptablemente bajos. Por otro lado, el tamaño mínimo del búfer debe ser superior a $B = 32$ elementos con el fin de que éste se desborde el menor número de veces posible.

Sin embargo, se han aportado dos formas nuevas de gestionar esta salida irregular, que hemos denominado *pausa* y *gestión dinámica del ratio*, que permiten evitar completamente los problemas anteriores.

Por otro lado, se ha añadido un módulo de carga y gestión del *vector de inicialización*, que permite que LFSRe, que es un cifrador en flujo, pueda ser utilizado en las modernas redes de conmutación de paquetes, como Internet. Se han analizado diversos aspectos de su funcionamiento, como las dificultades asociadas al uso del IV, o la longitud adecuada para el mismos.

Por último, se ha analizado el rendimiento que es capaz de proporcionar el cifrador en diferentes escenarios, desde flujos continuos de datos a paquetes de diversa longitud, obteniendo excelentes resultados, comparables a los de conocidas referencias en el campo, como RC4 o AES-CTR.

CAPÍTULO V

AUTENTICACIÓN DE LAS COMUNICACIONES

V.1. INTRODUCCIÓN

Analizada en los capítulos anteriores la protección de la privacidad de las comunicaciones, en el presente capítulo se aborda el otro gran pilar que conforma un esquema de seguridad completo: la autenticación.

Sim embargo, debido a su naturaleza única, proveer de mecanismos de seguridad a las redes ad-hoc en general, y a las de sensores en particular, resulta un gran desafío, pues se trabaja en escenarios muy restrictivos. Por ejemplo, éste último tipo de redes, que serán nuestro marco de trabajo principal en este capítulo, poseen unas caracterísiticacs diferenciadoras. Entre ellas pueden citarse la ausencia total de infraestructura previa de comunicaciones o de control administrativo [1] y sus bajos recursos computacionales, de ancho de banda y energéticos.

Particularmente, en lo que se refiere a la autenticación de la información, se suele hablar de seis grandes retos, la mayoría compartidos por el proceso de cifrado:

1. Comunicación inalámbrica y, por tanto, fácilmente interceptable.
2. Recursos limitados en cada nodo de la red.
3. Posibilidad de redes muy grandes y densas, lo que provoca una gran volumen de tráfico y colisiones en las comunicaciones.
4. Falta de estructuras fijas de comunicaciones, lo que dificulta la definición de los diferentes protocolos.
5. *La topología de la red es desconocida antes del despliegue de la misma.*
6. *Alto riegos de ataque y manipulación física en los nodos, de forma que se puede recuperar fácilmente las claves almacenadas en los mismos.*

De la lista anterior, las dos últimas características afectan directamente al proceso de autenticación. Sin embargo, además de las anteriores, existen dos importantes restricciones que serán las que verdaderamente marquen la dificultad de encontrar una solución al problema de la autenticación, que son el *consumo energético* y la *gestión de claves*.

Restricciones en el consumo energético Los algoritmos criptográficos utilizados por los protocolos que proporcionan seguridad a las comunicacoines tienen un coste energético apreciable que debe ser tenido en cuenta y analizado adecuadamente. Las limitaciones de potencia y energía de los nodos actuales de redes de sensores más habituales hacen que los esquemas de clave pública sean inviables en la práctica [2, 3]. En cualquier caso, los cifradores basados en clave secreta son entre dos y cuatro órdenes de magnitud más rápidos y eficientes que, por ejemplo, las firmas digitales [4], de forma que este tipo de esquemas se convierten en prácticamente la única opción viable y práctica para proteger este tipo de redes.

Restricciones en la gestión de claves A pesar de las múltiples propuestas que pueden encontrarse en la literatura para este propósito, hasta la fecha las únicas opciones realmente viables y prácticas para la distribución de claves en redes de gran tamaño (varios miles de nodos), es la *pre-distribución de claves* [5, 6].

Según este esquema, las claves se instalarían en los nodos durante el proceso de fabricación. Existen dos formas básicas de llevar este proceso a cabo: bien con una *única clave maestra* para toda la red, o bien con un *conjunto de* $n-1$ *claves simétricas*, siendo n el número total de nodos de la red, en la que cada nodo compartiría una clave privada única con el resto de nodos.

Desafortunadamente, ambos enfoques resultan inadecuados. La razón para el primero es que la captura de cualquier nodo comprometería la seguridad de toda la red, puesto que ésta comparte una única clave privada. Por otro lado, la segunda solución implica la pre-distribución y almacenamiento de $n-1$ claves en cada nodo, lo que hace un total de $n(n-1)/2$ claves en toda la red. Estos número se incrementan de forma muy rápida y, claramente, resulta inviable para red de gran tamaño.

V.1.1. Escenario de trabajo

Hemos comentado ya que las redes de sensores poseen muchas características que las hacen más vulnerables a ciertos ataques que los equipos de computación convencionales. Por ejemplo, su comunicación inalámbrica permite una fácil interceptación y alteración de los mensajes intercambiados en la red.

Además de estos ataques, comunes a culaquier red de comunicación inalámbrica, las redes de sensores son susceptibles a *ataques de consumo de recursos*. Este tipo de ataques trata de agotar los recursos de los nodos, como sus baterías o el propio ancho de banda, que suele ser muy limitado.

Con el fin de contextualizar exactamente nuestra aportación, a continuación se introduce el escenario de trabajo, que describe exactamente qué capacidades se suponen a un potencial atacante y las características más relevantes de los nodos que conforman la red:

- Los nodos no poseen equipamiento anti-manipulación física y, por tanto, se supone que el atacante tiene acceso directo y fácil a toda la información almacenada en la memoria, incluyendo las claves criptográficas. Esta restricción permite aplicar nuestro esquema en escenarios reales, con nodos comerciales existentes que pueden encontrarse fácilmente y resultan económicos.

- El atacante puede estar presente en cualquier momento del ciclo de vida de la red, incluyendo el despligue inicial. Tampoco existen restricciones sobre el número de atacantes o su localización física.

- El atacante puede interceptar y modificar fácilmente todos los mensajes intercambiados por cualquier nodo de la red, en cualquier lugar de ella.

- Por último, el protocolo ha sido diseñado para escenarios estáticos o de movidad reducida, en el que las tasas de autenticación de nuevos nodos no sea excesivamente alta. Este es el caso para las redes de sensores habituales, y así es asumido por otras importantes contribuciones en el campo [7, 8, 9].

Como puede observarse, se trata de un escenario de trabajo muy restrictivo, con condiciones muy severas. Sin embargo, esto permitirá que la solución aportada sea viable, práctica y fácilmente aplicable en entornos reales.

El resto de este capítulo se estructura de la siguiente manera. En la sección siguiente se introduce la notación utilizada. En la sección V.2 dos importantes esquemas de autenticación, ampliamente analizados y discutidos en la literatura, son descritos y comparados con nuestra propuesta, que se introduce en la sección V.3. Por otro lado, en las secciones V.4 y V.5 se analizan la resistencia del esquema a diversos ataques comunes en este tipo de redes y la evaluación de su rendimiento y consumo energético, respectivamente. En la sección IV.4.1.1 se describe el proceso de implementación del protocolo, junto con la plataforma hardware utilizada y una librería desarrollada como parte de dicha implementación. Por último, en la sección V.6 se presentan las conclusiones obtenidas.

V.1.2. Notación

A continuación se resume la notación que se utilizará en el resto del capítulo:

k_M	Clave maestra de red.
$\{M\}_k$	Cifrado del mensaje M con la clave k.
$[M]$	Resumen (hash) del mensaje M.
$[M]_k$	Resumen del mensaje M con la clave k (HMAC).
$[M]^i$	El mensaje M es resumido i veces sin clave.
$[M]^i_k$	El mensaje M es resumido i veces con la clave k.
k^A_{enc}	Clave de cifrado del nodo A.
∇^i_j	j-ésima tupla del autenticador en ciclo i-ésimo.
k^j_{auth}	Clave de autenticación del ciclo de autenticación j-ésimo, es decir, $k^j_{auth} = [k_M]^j$

V.2. ESTADO DEL ARTE Y TRABAJO PREVIO

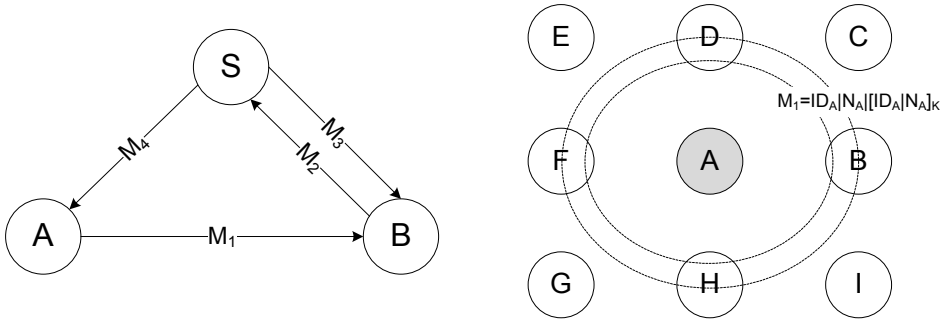
En esta sección se analizarán dos conocidos protocolos de gestión de claves basados en el esquema de clave maestra: SPINS [10] y BROSK [11], elegidos debido a que están consideradas contribuciones importantes en el campo y han sido ampliamente discutidos en la literatura [12, 13, 14].

V.2.1. SPINS

SPINS está formado, a su vez, por dos protocolos, SNEP y μTESLA, que proporcionan confidencialidad y autenticación en la difusión (*broadcasting*) de mensajes, respectivamente.

Como se muestra en la Figura V.1, SPINS asume que el nodo A quiere establecer una clave de sesión k_{AB} con el nodo B a través de una tercera parte confiable S, denominadado *centro de distribución de claves* (***K**ey **D**istribution **C**enter*). Este nodo S es un servidor, habitualmente con mayor capacidad de cómputo que el resto, que lleva a cabo las operaciones criptográficas necesarias para la autenticación y la distribución de claves.

Por otro lado, los nodos de SPINS tienen claves individuales, de forma que cada nodo y el servidor S comparten una clave simétrica única, que se utiliza para autenticar al propio nodo frente a S.



$M1 : N_A|ID_A$
$M2 : ID_B|N_A|N_B|ID_AK_{BS}$
$M3 : \{K_{AB}|N_A|ID_B\}K_{AS}$
$M4 : \{K_{AB}|N_B\}K_{BS}$

$M1 : ID_A|N_A|[ID_A|N_A]_K$
$M2 : ID_B|N_B|[ID_B|N_B]_K$
$K_{AB} : [N_A|N_B]_K$

Figura V.1: Protocolos de distribución de claves SPINS (izquierda) y BROSK (derecha)

Sin embargo, SPINS adolece de una serie de importantes inconvenientes, que se resumen a continuación:

- A pesar de que SPINS no puede ser considerado estrictamente un protocolo basado en clave maestra, posee una vulnerabilidad equivalente. En efecto, una vez que la clave que un nodo y el servidor S comparten ha sido comprometida, el adversario puede negociar claves de sesión válidas con otros nodos a través del servidor.
- Obviando el hecho de que la propia existencia de un nodo servidor rompe una de las premisas de la definición de una red de sensores ("...todos los nodos son iguales..."), las altas cargas de computación y consumo de ancho de banda puede

convertir al servidor S en un cuello de botella del sistema. Esto provoca que SPINS no se adapte de forma adecuada a un número alto de nodos en la red.

V.2.2. **BROSK**

Comparado con SPINS, BROSK puede considerarse una propuesta más reciente y mejorada de un protocolo de establecimiento de claves [11]. En este esquema no existe ningún servidor ni tercera parte confiable, y cada nodo negocia su clave de sesión con sus vecinos difundiendo un *mensaje de negociación de clave*.

Una vez que un nodo recibe este mensaje, deriva la clave de sesión compartida generando el MAC de dos valores aleatorios. Por ejemplo, en la Figura V.1 todos los nodos desde B hasta I recibirán el mensaje de negociación de clave de A. A su vez, A también recibirá el del resto de nodos ($M2$ en la figura). En ese momento, pueden utilizar K_{AB} como su clave compartida.

Por otro lado, los autores de BROSK no especifican qué ocurre con la clave maestra de red una vez que el proceso de difusión inicial ha finalizado, por lo que parece lógico asumir que esta clave no es borrada ni procesada de alguna forma. Si esta suposición es cierta, significa que BROSK sería vulnerable a una intrusión física, en la misma manera que SPINS.

Con el fin de evitar esta vulnerabilidad, nuestro protocolo *"olvida"* la clave maestra, pero, aún después, sigue siendo capaz de autenticar nuevos nodos, utilizando una estructura que hemos denomidado *operador de autenticación* o *autenticador*, que será analizado con profundidad en V.3.3.

V.3. PROTOCOLO DE AUTENTICACIÓN Y ESTABLECIMIENTO DE CLAVES

En esta sección se presentan las características y propiedades básicas de nuestra propuesta. El protocolo está compuesto, a su vez, de dos subprotocolos o procedimientos: un *esquema de establecimiento de claves*, que se lleva a cabo durante el despliegue inicial de la red, y un *protocolo de autenticación*, que se utiliza cuando, posteriormente, un nuevo nodo desea ingresar en la red, una vez que la fase anterior ha finalizado.

El criterio de diseño esencial de la propuesta ha sido optimizar al máximo su eficiencia energética. De esta forma, se trata de una propuesta computacional y criptográficamente muy "ligera": sólo utiliza funciones hash y no necesita de pesadas operaciones de clave pública, lo que la hace órdenes de magnitud más eficientes que los esquemas asimétricos. Por otra parte, la propuesta no impone ningún tipo de restricciones a la red, tales como algoritmos de enrutamientos, topologías o algoritmos de cifrados específicos.

La solución propuesta en esta Tesis implica tres fases, que deben llevarse a cabo en el siguiente orden:

- *Fase de pre-distribución de claves (F_0)*: se realiza antes del despliegue de la red, habitualmente durante el propio proceso de fabricación de los nodos.
- *Fase de inicialización de la red (F_1)*: encargada de establecer los primeros parámetros de seguridad de la red. Se lleva a cabo durante el despliegue inicial de la red sobre su ubicación física.
- *Fase de autenticación (F_2)*: se realiza cada vez que las fases anteriores han finalizado y un nuevo nodo solicita ingresar en la red.

A continuación se analizará cada una de estas fases en detalle.

V.3.1. F_0: Fase de pre-distribución de claves

Durante esta fase, que puede ser llevada a cabo directamente por el fabricante o el gestor de la red, debe generarse y almacenarse de forma segura una clave maestra de red, k_M. Esta clave debería tener una longitud suficiente para evitar ataques de fuerza bruta, por lo que se recomienda un mínimo de 128 bits. En cualquier caso para obtener una fortaleza criptográfica equivalente a una clave RSA de 1024 bits bastaría con una clave simétrica de sólo 80 bits, de acuerdo a las recomendaciones del organismo NIST [15].

Tras esta generación, se carga en cada nodo una estructura denominada *autenticador inicial*. Un *autenticador en ciclo i-ésimo* ∇^i es un operador utilizado para la autenticación de los nodos. Está compuesto de una serie de n tuplas de números aleatorios (*nonces*) y el resultado de aplicar una función HMAC con la clave de autenticación actual sobre dichos números aleatorios.

Durante el primer ciclo de autenticación, tras la fabricación de los nodos pero antes de su despliegue inicial, la clave de autenticación es igual a la clave maestra, $k^0_{auth} = k_M$, de forma que:

$$\nabla^0 = \{(r_i, [r_i]_{k_M})\},\ i = 0, ..., n-1.$$

En general, la clave de autenticación en el ciclo j-ésimo del autenticador será $k^j_{auth} = [k_M]^j$. Por otro lado, el contenido del autenticador será:

$$\nabla^j = \{(r_i, [r_i]_{k^j_{auth}})\},\ i = 0, ..., n-1.$$

Elemento	Notación	Tamaño
Clave de cifrado del nodo A	k^A_{enc}	16 bytes
Clave de cifrado de los nodos vecinos B...Z	k^B_{enc}	16 bytes
	$\vdots$	16 bytes
	k^Z_{enc}	16 bytes
Clave de autenticación del primer ciclo	k^1_{auth}	16 bytes
Autenticador de primer ciclo	∇^0	$n \times 24$ bytes

Cuadro V.1: Organización de la memoria del nodo al término de la fase F_1

El autenticador cambia de ciclo, transitando al siguiente, cuando éste está cerca de su agotamiento. Este proceso se analiza en detalle en la sección V.3.3.1.

V.3.2. F_1: Fase de inicialización de la red

Esta fase comienza una vez que la etapa anterior F_0 ha finalizado. Esta fase se realiza durante el despliegue de la red, en el mismo entorno físico y operativo que tendrá durante su normal funcionamiento. En este proceso, cada nodo "descubre" sus nodos vecinos que están en rango de comunicación y realiza los siguientes pasos:

1. Cada nodo i genera una clave única y simétrica, k^i_{enc}, denominada *clave de cifrado del nodo i*. Esta clave se obtiene generando un número aleatorio, r_i, y calculando $k^i_{enc} = [k_M, r_i]$. De esta forma, por ejemplo, la clave de cifrado de A se calcularía con la siguiente operación: $k^A_{enc} = [k_M, r_A]$.
2. Cada nodo difunde su valor aleatorio, r_i, durante un breve periodo de tiempo, que puede ser tan sólo de unos pocos segundos [16]. Una valor razonable podría ser alrededor de un minuto. De esta forma, si un atacante estuviera ya presente en esta fase, sólo obtendría una conjunto de números aleatorios.
3. Cada nodo recibe los números aleatorios de sus nodos vecinos y calcula sus claves de cifrado correspondientes utilizando la clave maestra común a todos ellos. En este punto, cada nodo almacena una lista con las claves de cifrado que le permiten comunicarse con sus vecinos.
4. Cada nodo calcula el hash de la clave maestra y almacena el resultado como la primera de las claves de autenticación, $k^1_{auth} = [k_M]$. El objetivo de esta operación es que, claramente, almacenar la clave maestra en claro en la memoria del nodo comprometería la seguridad de toda la red si el nodo es capturado. Esta es, por otra parte, la razón principal de la existencia del propio autenticador, que proporciona una forma de autenticar otros nodos, verificando su conocimiento de la clave maestra común, sin almacenar la propia clave.
5. Al final de esta fase, cada nodo almacena su propia clave de cifrado, k^i_{enc}, el conjunto de claves de cifrado de sus nodos vecinos, la clave del próximo ciclo de autenticación, k^1_{auth}, y el autenticador del ciclo actual, ∇^0, que está compuesto de un conjunto de n tuplas. La estructura interna de un autenticador se discutirá con detalle en la sección V.3.3.

Alcanzado este punto, los nodos de la red pueden empezar a comunicarse entre sí utilizando las claves de cifrado obtenidas en los pasos previos.

V.3.2.1. Requisitos de memoria

Una vez finalizada la fase anterior, cada nodo almacena un conjunto de claves de cifrado y un autenticador, que se utilizará en la siguiente fase para la autenticación de nuevos nodos que deseen ingresar en la red. La organización de la memoria del nodo, al final de la ejecución de esta fase, queda tal y como puede encontrarse en el Cuadro V.1.

Teniendo en cuenta el número habitual de nodos vecinos que suele darse en este tipo de redes [16], y la longitud de clave que estamos considerando (128 bits), los requisitos de memoria necesitada por el protocolo resultan adecuados. Por ejemplo, considerando una alta densidad de nodos vecinos, 5, una longitud de clave de 128 bits, funciones HMAC con una salida de 160 bits y un autenticador con 10 tuplas, la memoria necesaria para nuestra propuesta es de sólo 360 bytes, que se distribuyen de la siguente manera:

$$\begin{aligned}\text{Requisitos memoria} = &\ \text{Clave cifrado (128 bits)}+\\ &\ \text{Clave nodo vecino} \times 5 \text{ vecinos} \times 128 \text{ bits/clave}+\\ &\ 192 \text{ bits/tupla} \times 10 \text{ tuplas}\\ = &\ 2\,880 \text{ bits}\\ = &\ 360 \text{ bytes}\end{aligned}$$

En comparación, otros protocolos que pueden encontrarse en la literatura cuentan con unos requisitios mucho mayores. Puede citarse el caso del conocido protocolo de Eschenauer y Giglor [8] que necesita alrededor de 750 bytes para el almacenamiento de claves para una red de 10 000 nodos. Por otro lado, el protocolo BROSK requiere entre 1 100 y 1 300 bits aproximadamente. Finalmente, nuestra propuesta, que necesita aproximadamente 2 800 bits, se sitúa entre los valores previos.

V.3.3. **Operador de autenticación**

El *operador de autenticación o autenticador* es utilizado por los nodos de la red para autenticarse mutuamente. El objetivo de esta estructura es dotar a los nodos de la capacidad de autenticar a nuevos nodos, una vez que la fase de inicialización F_1 ha finalizado y la topología de la red es estable.

Con se ha comentado en la sección V.2.2, retener la clave maestra en la memoria de los nodos supone un grave riesgo de seguridad, que podría llevar a un fácil compromiso de toda la red. La idea detrás del autenticador, y su verdadera razón de ser, es, por tanto, sencilla: el material necesario para la autenticación, las tuplas de retos/respuestas, son precalculados con una clave que es posteriormente borrada. De esta forma, se mantiene la capacidad de verificar el conocimiento de dicha clave, sin necesidad de almacenar la propia clave en memoria.

Como se acaba de mencionar, el operador de autenticación se basa en el uso de dos primitivos criptográficas conocidas, como los *esquemas de reto/respuesta* [17] y las *cadenas de hashes o claves* [18]. Nuestro operador tienes ciertas similitudes con otros esquemas, como μTESLA [**?**] (parte del protocolo BROSK, anteriormente estudiado), pero proporciona una serie de importantes ventajas:

- No hace uso de un *esquema de publicación de la clave basado en tiempo* (*Time-based key disclosure mechanism*), de forma que no necesita una estación base, ni un mecanismo de sincronización de tiempo.

- Utiliza una cadena de claves para la autenticación de nuevos nodos, que resulta un mecanismo más sencillo y eficiente que una tercera parte confiable S y el consecuente intercambio de cuatro mensajes, necesarios en SPINS (sección V.2.1). Por contra, nuestra propuesta alcanza el mismo objetivo haciendo uso de una clave maestra y la transmisión de un único mensaje.

V.3.3.1. Generación del operador de autenticación

El autenticador ∇ en un ciclo arbitrario j-ésimo se construye utilizando el material de claves del ciclo anterior $j-1$. De esta forma, el nodo es capaz de verificar el conocimiento de la clave maestra, ya que la clave del ciclo j-ésimo sólo puede derivarse de la del ciclo $j-1$-ésimo, sin almacenar la propia clave maestra.

En esta situación, si un nodo es comprometido y es robado el material de claves de su memoria, el atacante sólo obtiene las claves del actual ciclo de autenticación y, por tanto, no es capaz de comprometer la autenticación y los intercambios de claves realizados en anteriores ciclos del mismo autenticador.

Cuando un nodo está próximo a agotar las tuplas del ciclo de su autenticador, simplemente genera un nuevo conjunto, es decir, comienza un nuevo ciclo. El proceso para transitar del ciclo j-ésimo al siguiente se compone de estos pasos:

1. Genera un nuevo conjunto de n números aleatorios y les aplica la actual clave de autenticación, k^j_{auth}, para obtener el conjunto de tuplas:

$$\nabla^{j+1} = \left\{ (r_i, [r_i]_{k^j_{auth}}) \right\},\ i = 0, ..., n-1$$

2. Actualiza la clave de autenticación actual, aplicándole una función *hash* y obteniendo:

$$k^{j+1}_{auth} = [k^j_{auth}]$$

3. En resumen, el contenido del nuevo ciclo del autenticador es el siguiente:

$$\left[k^{j+1}_{auth}, \nabla^{j+1} = \left\{ (r_i, [r_i]_{k^j_{auth}}) \right\} \right],\ i = 0, ..., n-1$$

V.3.3.2. Cuestiones de implementación

A la hora de implementar nuestra propuesta, cada una de las tuplas de un autenticador necesita disponer de una *etiqueta de estado* asociada, que describe el estado actual de la tupla en el proceso de autenticación. Los posibles valores para este estado es:

- *LIBRE* - la tupla no ha sido utilizada hasta el momento.
- *ASIGNADA* - la tupla ha sido temporalmente asignada a un nodo en un proceso de autenticación en marcha. Si el proceso falla o no se completa, la etiqueta vuelve al estado LIBRE y la tupla vuelve a estar libre para un nuevo uso.
- *USADA* - la tupla ha sido ya utilizada en un proceso de autenticación que ha finalizado correctamente y, por tanto, no está disponible para otros procesos. De esta forma, se hace resiste el protocolo a posibles ataques de repetición.

Además de estas etiquetas, la estructura de un autenticador necesita de un campo adicional, denominado *índice de tupla actual*, δ, que almacena la primera tupla con estado LIBRE y que se incrementa en uno cada vez que una tupla cambia su estado de LIBRE a ASIGNADA.

Autenticador ∇^2	
δ=3	
$(r_o, [r_o]_{k^2_{auth}})$	USADA
$(r_1, [r_1]_{k^2_{auth}})$	USADA
$(r_2, [r_2]_{k^2_{auth}})$	ASIGNADA
$(r_3, [r_3]_{k^2_{auth}})$	LIBRE
...	...

Cuadro V.2: Ejemplo de autenticador con estado $\delta = 3$

Tamaño n del autenticador Una cuestión importante que debe ser analizada es las implicaciones del tamaño o número de tuplas n del autenticador. Como un parámetro del sistema, debe ser ajustado de acuerdo al número de nodos y, sobre todo, a la tasa de nuevos nodos esperada.

Por supuesto, el valor de n puede ser ajustado de forma dinámica, conforme las características de la red cambian. Un valor típico para n estará alrededor de 10, que se considera un valor razonable teniendo en cuenta la tasa esperada de nuevas autenticaciones y el número típico de vecinos en esta clase de redes [16].

A pesar de que no tiene consecuencias directas en la seguridad del sistema, un valor excesivamente pequeño de n podría llevar a problemas de rendimiento si la red tuviera una alta tasa de autenticaciones. Para evitar este inconveniente, un nuevo ciclo del autenticador actual puede ser calculado "lentamente", en segundo plano. En función de la tasa de autenticaciones, este cálculo podría ser tan lento como generar una nueva tupla cada varias horas o incluso en días.

Un ejemplo de autenticador puede encontrarse en el Cuadro V.2, que contiene los valores de un autenticador de segundo ciclo. En este caso, $k^2_{auth} = [[k_M]] = [k_M]^2$. Según el valor del índice de tupla actual, δ, puede inferirse que este ciclo ha realizado ya dos autenticaciones con éxito.

V.3.4. F_2: Protocolo de autenticación

La fase F_2 comienza cuando las anteriores han acabado, han sido descubiertos los nodos que formaban parte de la red en su inicialiación, pero aparecen nuevos nodos que desean ingresar en la red. En ese momento comienza un protocolo de autenticación mutua que hace uso de sus autenticadores.

Sea A un nuevo nodo, que no pertenece a la red, y que solicita el ingreso en la misma. Puesto que no ha llevado hasta el momento ningún proceso de autenticación, su autenticador permanecerá aún en su primer ciclo, ∇^1. Por otro lado, sea B el nodo que ya pertenece a la red y que, por tanto, puede estar en cualquier ciclo arbitario j. El protocolo para la autenticación mutua de A y B debe ejecutarse, pues, como sigue:

1. A produce un reto para B generando un número aleatorio, r_A, que envia a B en el interior de un mensaje M_1 con el siguiente formato:

$$M_1 = r_A$$

2. B recibe M_1 de A y realiza las siguientes operaciones:
   a) Abre la primera tupla libre de su autenticador, marcada con el índice de tupla actual, δ, y extrae el reto correspondiente, r_B, y su respuesta correspondiente

$[r_B]_{k_{auth}^{j-1}}$. A continuación, cambia la etiqueta de estado de dicha tupla de *LIBRE* a *ASIGNADA*.

b) Genera la respuesta al reto recibido de A utilizando la función HMAC con la clave de autenticación actual, k_{auth}^{j-1}, sobre el reto r_A, para obtener $[r_A]_{k_{auth}^{j-1}}$.

c) Recupera su clave de cifrado, k_{enc}^{B}, y la cifra con la clave de autenticación actual, obteniendo $\{k_{enc}^{B}\}_{k_{auth}^{j}}$.

d) Finalmente incluye el ciclo actual de su autenticador, j, para que A pueda sincronizarse en los pasos siguientes. Por último, manda un mensaje M_2 a A con el siguiente formato:

$$M_2 = r_B, [r_A]_{k_{auth}^{j-1}}, \{k_{enc}^{B}\}_{k_{auth}^{j}}, j$$

3. A recibe M_2 de B y lleva a cabo las siguientes operaciones:

a) Recupera el ciclo actual de B, j, del mensaje y calcula la diferencia con su propio ciclo. Dado que A es un nodo "fresco", este ciclo será 1. Por tanto, A debe realizar $j-1$ hashes sobre la clave maestra k_M hasta sincronizarse con B y obtener $k_{auth}^{j} = [k_M]^{j-1}$. Puede encontrase más información sobre este paso en las notas de esta sección.

b) Comprueba que la respuesta de B es correcta, comparándola con su propio cálculo sobre el valor recibido. En este punto, B ha demostrado conocimiento de la clave maestra original y ha sido autenticado satisfactoriamente.

c) Calcula la respuesta al reto recibido r_B, obteniendo $[r_B]_{k_{auth}^{j-1}}$.

d) Genera su propia clave de cifrado, k_{enc}^{A}, y la cifra con su clave de autenticación actual, obteniendo $\{k_{enc}^{A}\}_{k_{auth}^{j}}$.

e) Por último, envia un mensaje M_3 conteniendo los dos valores anteriores a B, con el siguiente formato:

$$M_3 = [r_B]_{k_{auth}^{j}}, \{k_{enc}^{A}\}_{k_{auth}^{j}}$$

4. Por último, B recibe el mensaje M_3 de A y ejecuta las siguientes acciones:

a) Compara la respuesta a su reto utilizando la tupla correspondiente de su autenticador, ∇_k^j:

1) Si son iguales, el nuevo nodo ha sido autenticado con éxito y, por tanto, se puede permitir su acceso a la red. Por último, cambia la etiqueta de estado de ∇_k^j de *ASIGNADA* a *USADA*.
2) Si no son iguales, el nuevo nodo no se ha autenticado correctamente. La etiqueta de estado vuelve al valor *LIBRE* de nuevo.

El proceso completo puede resumirse en los siguientes pasos:

$$\begin{aligned} A \rightarrow B &: r_A \\ A \leftarrow B &: r_B, [r_A]_{k_{auth}^{j-1}}, \{k_{enc}^{B}\}_{k_{auth}^{j}} \\ A \rightarrow B &: [r_B]_{k_{auth}^{j}}, \{k_{enc}^{A}\}_{k_{auth}^{j}} \end{aligned}$$

Como puede observarse en los mensajes M_2 y M_3, nuestro protocolo constituye también un *procedimiento de establecimiento de claves*, puesto que permite transferir las correspondientes claves de cifrado de A y B a su contraparte correspondiente.

Notas

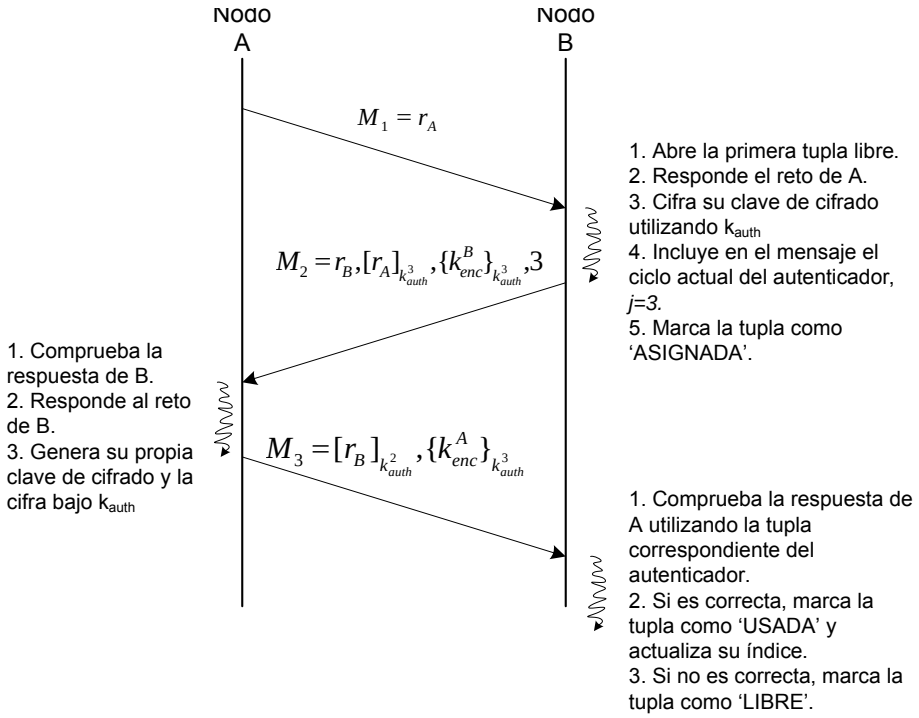


Figura V.2: Ejemplo de ejecución completa del protocolo de autenticación

- Otro tipo de retos son también posibles, como números de secuencia, en lugar de los números aleatorios utilizados. Los sellos de tiempo, aunque podrían reducir el número de mensajes intercambiados en uno, no resultarían prácticos en este escenario, pues implicarían la necesidad de disponer de relojes sincronizados en los nodos de la red.

- Este protocolo ha sido diseñado para redes de sensores, que son escenarios estáticos o cuasi-estáticos, con una tasa esperada de nuevos nodos baja. Por esta razón, los casos en los que el número de ciclos del autenticador de un nodo supere unas pocas decenas serán raros y, por tanto, el coste computacional asociado al proceso de sicronización realizado en el paso 2 por A será también bajo.

- El protocolo permite realizar varios procesos de autenticación simultáneos sobre el mismo autenticador, para lo que es necesario, simplemente, utilizar diferentes índices δ.

La Figura V.2 muestra un ejemplo de una ejecución completa del esquema de autenticación propuesto entre dos partes A y B. Los parámetros del autenticador de B son ∇^3 y $\delta = 2$, lo que significa que el operador está en su tercer ciclo y, hasta el momento, ha autenticado con éxito a dos nodos y ha utilizado el mismo número de tuplas. A es un nuevo nodo que desea ingresar en la red y cuya autenticador no ha sido todavía utilizado.

V.4. ANÁLISIS DE SEGURIDAD

En esta sección se analizará el grado de seguridad proporcionado por el protocolo propuesto frente a los ataques más comunes y peligrosos de este tipo de redes. Cuando el ataque sea inevitable, debido a las características inherentes de las redes, se aportan ciertas contramedidas que tratan de mitigar en lo posible dicho ataque.

V.4.1. Ataque físico a los nodos

El esquema presenta un grado de resistencia excelente frente a la que se considera la principal amenaza de estas redes, que es el *compromiso físico* de los nodos. Con el fin de evaluar cuantitativamente esta resistencia, haremos uso de la métrica definida en [5], aceptada en el ámbito como adecuada y ampliamente utilizada en otros trabajos.

Esta métrica define la resistencia de un nodo frente al compromiso físico como la fracción de enlaces, *en los que el nodo atacado no participe directamente*, respecto al total de la red cuya seguridad se pierde como consecuencia del compromiso. Por ejemplo,

De acuerdo a esta métrica, nuestro protocolo obtiene una puntuación perfecta, puesto que el compromiso de un nodo no revela información sobre otros enlaces en los que no está directamente involucrado.

V.4.2. Denegación de servicio

Los ataques de denegación de servicio tienen como objetivo agotar los recursos de una red, como su ancho de banda, a su capacidad de proceso, con el fin de que ésta no pueda atender las peticiones de clientes legítimos [19].

Cuanto el ataque se aplicar a nuestro protocolo, un atacante podría tratar de extenuar el conjunto de autenticadores iniciando continuamente un proceso de autenticación contra un nodo y no respondiendo a los retos que éste proporciona o proporcionando respuestas falsas (aleatorias, por ejemplo). De esta forma, el nodo atacado podría agotar rápidamente su conjunto de autenticadores, y no disponer de tuplas libres para atender a nodos legítimos.

Para resultar inmune a este ataque, o al menos mitigar lo suficiente sus efectos, la implementación de nuestro protocolo dispone de un mecanismo de protección. Este mecanismo se basa en el uso de un intervalo temporal, α, en el que el nodo que inicia la autenticación debe responder correctamente al reto proporcionado. Si finalizado el periodo α no se ha recibido una respuesta correcta, el reto se etiqueta como 'LIBRE' y pasa a estar disponible para su uso de nuevo.

Este parámetro puede ser ajustado dinámicamente si se detecta que un nodo se encuentra bajo ataque, decrementando su valor en función de la severidad del ataque.

V.4.3. Atacante presente en el despliegue de la red

Algunas de las propuestas que pueden encontrarse en la literatura son vulnerables cuando el atacante está presente ya en el momento del despliegue de la red, como la conocida propuesta de la infección de claves [16], que difunde sus claves en claro durante un breve periodo de tiempo.

Nuestra propuesta es segura incluso en este escenario, sin importar el número de atacantes presentes o su localización física.

V.4.4. **Borrado seguro de claves**

Debido al fenómeno denominado *remanencia magnética* [20], el proceso de eliminación de información sensible, como claves criptográficas, de las memorias de almacenamiento no es tan sencillo como pudiera parecer.

Este fenómeno consiste en cierta tendencia de los materiales con propiedas magnéticas, con los que están fabricados los discos duros y memorias dinámicas de cada computador o nodo de la red, en mantener trazas de la información que almacenaban cuando ésta cambia. Por ejemplo, se ha demostrado experimentalmente que las memorias RAM actuales son capaces de mantener sus contenidos originales durante horas, en ciertas condiciones, aún cuando su alimentación ha sido interrumpida [21].

Debido a estas razones, la obtención de una nueva clave de autenticación k^i_{auth}, en el ciclo i-ésimo, puede dejar trazos de la clave del ciclo anterior si este paso no es realizado con cuidado.

El peligro radica en el hecho de que sobreescribir de forma reiterada la información almacenada en una memoria RAM dinámica (DRAM) con una secuencia fija (o aleatoria) no es tan efectiva como en los medios magnéticos, como los discos duros. Por tanto, en este tipo de memoria, que es el elemento constituyente principal de un nodo típico de una red de sensores, el objetivo para realizar un borrado seguro de la información será almacenar la nueva información por el mayor periodo de tiempo posible, en lugar de alterar su contenido tan rápidamente como sea posible.

De acuerdo a los experimentos realizados en células de memoria DRAM [22], un almacenamiento por un periodo de un segundo provoca que los contenidos anteriores puedan ser todavia recuperados con cierta facilidad. Un minuto después, la probabilidad se reduce, pero sigue siendo posible con cierto equipamiento. Diez minutos más tarde resulta prácticamente irrecuperable.

Por tanto, la solución más práctica y eficiente al problema de la recuperación de la clave de autenticación previa es tan simple como almacenar la nueva clave en la misma posición de memoria de la anterior. De esta forma, en aproximadamente 10 minutos, el nuevo contenido sobreescribirá completamente al antiguo, haciendo su recuperación inviable.

V.5. EVALUACIÓN DEL RENDIMIENTO Y CONSUMO ENERGÉTICO

Por otro lado, se estudiará uno de los aspectos más importantes y uno de los criterios básicos de diseño: la eficiencia energética de las soluciones.

Como hemos mencionado, la evaluación del rendimiento y la eficiencia energética resultan especialmente importantes en entornos como las redes de sensores, cuyos recursos computacionales y energéticos son muy limitados.

Para evaluar el coste energético de los protocolos de seguridad se utiliza un método indirecto, en lugar de medidas directas de consumo. Este método consiste en calcular el número total de bytes transmitidos de forma inalámbrica, a través del número y tamaño de cada uno de los mensajes intercambiados por el protocolo. Este método es el más utilizado por dos razones.

Otras propuestas La primera es muchas de las grandes propuestas existentes en la literatura, incluyendo las dos seleccionadas en esta Tesis, SPINS y BROSK, utilizan este método y, por tanto, no incluyen medidas directas de consumo que puedan ser utilizadas en una comparación. Por otra parte, las medidas directas adolecen de un gran inconveniente y es que existen multitud de diferentes parámetros, como el hardware específico utilizado, los módulos de radio o las baterías utilizadas, que pueden afectar significativamente a los resultados, haciendo finalmente que éstos no sean comparables [23].

Costes energéticos La segunda de las razones radica en el hecho de que, examinando los costes energéticos de cada uno de los componentes de los protocolos de seguridad, se observa que la mayor parte de ellos se deben a la transmisión de radio [10]. Por ejemplo, en un protocolo de clave maestra como SPINS, la energía utilizada en la transmisión alcanza el 97 % del coste energético total, mientras que el utilizado por los procesos de cifrado apenas alcanza el 3 %.

Algunos estudios ahondan en este sentido [24] y muestran cómo en este tipo de redes la relación energía/bit transmitido por radio puede alrededor de tres órdenes de magnitud mayores que la necesario en las operacoines de cifrado simétrico. Otros estudios más recientes [25, 26] parecen confirmar este hecho. Resulta obvio, por tanto, que *minimizar el número de los mensajes transmitidos y el tamaño de éstos* es uno de los objetivos fundamentales en el diseño de cualquier protocolo de seguridad.

Por otro lado, debido a las interferencias inherentes e inevitables de la transmisión inalámbrica, existen muchos factores que pueden influir en el número final de transmisiones, como las interferencias de frecuencias cercanas, la atenuación propia de la señal (Figura V.3) o las colisiones con los nodos vecinos. En cualquier caso, estos factores afectan por igual a todos los protocolos y, por simplicidad, asumiremos en lo sucesivo que todos los datos se transmiten correctamente.

Sin embargo, ampliaremos la metodología utilizada en otros trabajos [10, 11], y no sólo compararemos el número de transmisiones, sino el tamaño de cada uno de los mensajes intercambiados como parte de la negociación del protocolo. Por las razones anteriormente expuestas, ésta se considera una buena aproximación al consumo energético real de la propuesta.

V.5.1. Consumo energético

Para llevar a cabo la evaluación del consumo energético utilizaremos el escenario de caso peor. Para ello, se utilizará una topología de red en forma de cuadrícula de $N \times N$

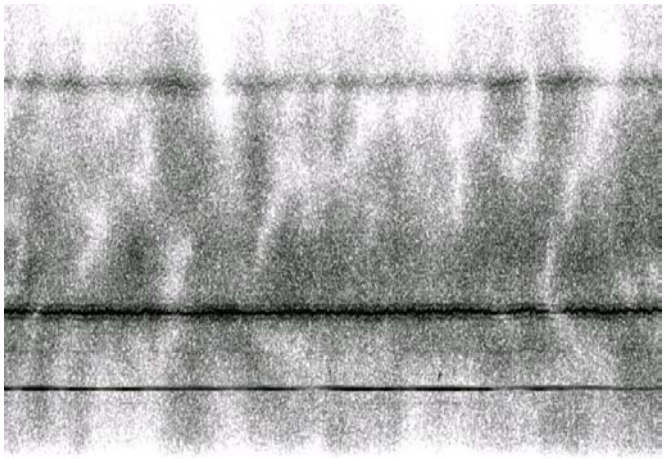


Figura V.3: El tiempo se muestra en el eje horizontal, la frecuencia en el vertical y la intensidad de la señal como una escala de grises.

y asumiremos que cada nodo puede recibir los datos transmitidos por sus vecinos más inmediatos (cada nodo tiene, por tanto, 8 nodos vecinos)

En este escenario, el número medio de transmisiones en el protocolo SPINS es de aproximadamente 8, mientras que BROSK y nuestra propuesta únicamente requieren una. En cualquier caso, cada una de estos mensajes transmitidos tiene una longitud diferente, que puede afectar significativamente al resultado final.

Para obtener una evaluación más exacta, calculamos el número total de bits realmente transmitidos, asumiendo valores de nonce de 64 bits, valores identificadores de 14 bits, claves simétricas de 128 bits y resúmenes de HMAC de 160 bits. Los resultados finales son que las longitudes de los mensajes son de 632 bits para SPINS (que necesita 8 de estos mensajes, haciendo un total de 8×632 =5.056 bits transmitidos), 238 bits para BROSK y 78 bits para nuestra propuesta.

Por tanto, puede concluirse que nuestra propuesta permite disminuir los costes energéticos en un 98 % y 67 % respecto a SPINS y BROSK respectivamente. El número total de bits transmitidos, directamente proporcional al consumo energético final, puede encontrarse en la Figura V.4.

V.5.2. Escalabilidad

Además del consumo energético, la escalabilidad es una característica extremadamente importante para un protocolo de seguridad en redes de sensores. Por ejemplo, con el fin de aliviar el número de mensajes que debe transmitir cada nodo, SPINS transfiere parte de esa carga a un nodo servidor.

Esto provoca, sin embargo, que el protocolo no resulte escalable, pues ahora es el servidor el que puede convertirse en un cuello de botella para toda la red. Por ejemplo, el número de transmisiones que debe realizar en una red de sólo 4.096 es superior a 10^7, lo que produciría múltiples colisiones y degradaría el rendimiento globar de toda la red.

Sin embargo, tanto BROSK como nuestra propuesta sí se adaptan adecuadamente al tamaño de la red, ya que ambos protocolos requieren intercambiar un único mensaje, independientemente del número de nodos de la red, que puede ser arbitrariamente

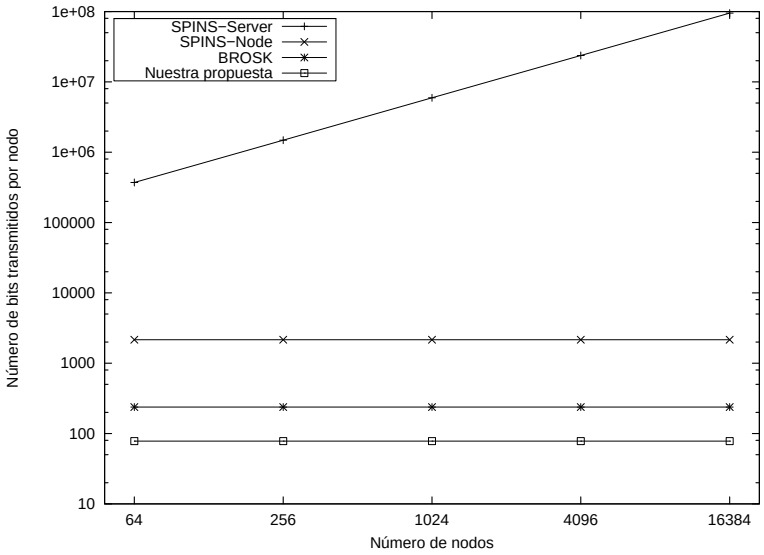


Figura V.4: Consumo energético (bits transmitidos por nodo). El eje de ordenadas utiliza una escala logarítmica.

grande.

V.6. CONCLUSIONES

En este capítulo se ha presentado un nuevo protocolo de autenticación y gestión de claves muy ligero, especialmente diseñado para redes de sensores de gama baja, con poca capacidad computacional y de memoria. El esquema se basa en el uso de primitivas criptográficas sencillas, como funciones hash y HMAC, y utiliza una clave maestra para derivar de ella una serie de claves simétricas. Este enfoque trata de aprovechar los beneficios del esquema de claves simétricas, es decir, una mayor resistencia al compromiso, sin la necesidad de una pre-distribución o almacenamiento de claves masivos.

Como consecuencia, nuestro esquema presenta una *buena escalabilidad*, y podría aplicarse a redes arbitrariamente grandes sin ninguna modificación o pérdida de rendimiento, puesto que sólo requiere intercambiar un mensaje para el proceso de autenticación y establecimiento de clave inicial. Esta escalabilidad se aplica también a los requisitos de memoria, que se mantienen pequeños y constantes, dependiendo únicamente del número de nodos vecinos, y no del total de nodos de la red.

Por otro lado, el protocolo posee también la *capacidad de transportar claves de forma segura*, lo que permite alcanzar ambos objetivos, autenticación y gestión de claves, con un número mínimo de mensajes intercambiados. Proporciona, también, autenticación *nodo a nodo*, de forma que los nodos son capaces de verificar las identidades de las partes con las que se comunican. De esta forma, un atacante no es capaz de suplantar la identidad ni los datos provenientes de un nodo legítimo de la red, a no ser que éste haya sido capturado físicamente.

El protocolo cuenta con otra serie de buenas propiedades, que lo hacen adecuado para este tipo de redes. En primer lugar, proporciona una *muy buena resistencia a las consecuencias del compromiso físico de los nodos*. De acuerdo a la métrica definida en [5], la captura de uno nodo no revela información sobre enlaces en los que dicho nodo no está directamente involucrado.

Otro de los grandes objetivos de este esquema de autenticación es el que tratar de conseguir su máxima *eficiencia energética*. Como se ha comentado con anterioridad, las restricciones energéticas en este tipo de redes son muy importantes, y cualquier esquema o protocolo de seguridad debe tener en cuenta esta característica si pretende ser viable en la práctica. En este sentido, el esquema presenta mejoras sobre las propuestas existentes hasta el momento en la literatura. Debido al menor tamaño y número de los mensajes intercambiados por el protocolo, éste resulta un 98 % y un 67 % más eficiente que SPINS y BROSK, respectivamente. Por último, este bajo consumo energético puede ayudar a paliar en lo posible los efectos derivados de los ataques de consumo.

Por otro lado, con el objetivo de demostrar la viabilidad del protocolo, y obtener medidas precisas y exactas del rendimiento del mismo, se ha llevado a cabo su *implementación en dispositivos hardware reales*. Para ello ha sido necesario desarrollar, previamente, la primera librería criptográfica existente para esta plataforma hasta la fecha, que hemos denominado *Cryptonite*. La librería ha sido licenciada bajo GPLv3 y puesta a disposición de la comunidad de software libre y de cualquier desarrollador que desee hacer uso de ella. La descripción del sistema y el análisis detallado de los resultados obtenidos pueden encontrarse en la sección VI.3 del Capítulo V.

En lo concerniente a su *rendimiento*, el protocolo muestra unas prestaciones excelentes. Los datos obtenidos muestran que una ejecución completa del protocolo, incluyendo la autenticación mutua del nodo iniciador y del anfitrión, en menos de 40 ms.

Finalmente, se resumen a continuación otras de las principales contribuciones de nuestro protocolo, que suponen importantes diferencias con los esquemas existentes:

- Durante la fase de despliegue, nuestra propuesta presenta también algunas ventajas sobre otros protocolos presentes en la literatura, como el conocido esquema de *infección de claves* [16]. que difunde las claves de cifrado en claro durante un breve periodo de tiempo. En este escenario, nuestro esquema es seguro frente a un atacante que esté presente ya en el mismo momento del despliegue de la red, sin importar el número de atacantes ni su localización física.

- Tras la fase de despliegue, el sistema "olvida" la clave maestra, de forma que resulta mucho menos vulnerable a la intrusión física [8]. Con el fin de poder cuantificar la resistencia del sistema a dicho ataque, se ha definido una métrica en [5] que ha sido ampliamente utilizada en otras propuestas. De acuerdo a dicha métrica, nuetro protocolo obtiene en ella una puntuación perfecta, que significa que, en el caso del compromiso de un nodo, éste no revela información sobre enlaces en los que no está directamente implicado.

- Por otro lado, el esquema ha sido diseñado con el fin de resultar resistente a ciertos ataques de denegación de servicio, potencialmente muy peligrosos en este tipo de redes. El protocolo cuenta con un mecanismo de protección, a través de un valor temporal α frente a un intento deliberado de agotar su conjunto de autenticadores. Este parámetro puede ajustarse dinámicamente cuando se detecta que un nodo está sufriendo un ataque.

CAPÍTULO VI

IMPLEMENTACIÓN Y APLICACIÓN A ESCENARIOS DE TRANSMISIÓN MULTIMEDIA

VI.1. INTRODUCCIÓN

Presentadas a lo largo de esta tesis las diversas soluciones aportadas en el marco de la seguridad de redes ad-hoc inalámbricas, este capítulo pretende describir y analizar las implementaciones realizadas de dichas soluciones. Algunas de ellas, como la correspondiente a los mecanismos de cifrado, se han aplicado también en entornos reales, que forman parte de diversos proyectos de investigación con financiación gubernamental.

El principal objetivo de este sección consiste, por tanto, en demostrar la viabilidad y utilidad de las soluciones a través de sus implementaciones o aplicación en entornos reales.

A pesar de que este enfoque resulta claramente más costoso que las habituales simulaciones, los autores creen que posee una serie de ventajas importantes. La primera, por supuesto, es que constituye una demostración innegable de la viabilidad de la propuesta. Proporciona, por otro lado, datos fiables y exactos relativos al tiempo de ejecución, consumo energético y otros aspectos de la misma, que no son siempre adecuadamente recogidos a través de simulaciones.

Ésta supone, de hecho, la principal de las razones por las que los autores se han decantado por una implementación real de las propuestas es que, a pesar de que la simulación es la herramienta por excelencia para la evaluación de soluciones en este tipo de redes, su utilización puede no resultar adecuada. Esta opinión es compartida, además de por los autores de esta tesis, por otros investigadores, como [1, 2], que plantean serias dudas sobre la credibilidad de los resultados obtenidos en base a simulaciones.

De acuerdo a estos principios, los mecanismos de cifrado propuestos en los Capítulos III y IV han sido implementados y evaluados en entornos MANet, donde han sido utilizados para la protección de la transmisión de contenido multimedia. Los resultados obtenidos se describen en la sección VI.2.

Por otro lado, el protocolo de autenticación presentado en el Capítulo V se analiza en la sección VI.3. En ella se describe la implementación realizada para redes de sensores, y se presenta una nueva librería, Cryptonite, desarrollada para la plataforma libre Arduino, debido a la inexistencia de librerías criptográficas en ella.

Por último, en la sección VI.4 se analiza otro de los desarrollos llevados a cabo en el marco de esta tesis, que consta de dos módulos denominados Apolo y Dafne, diseñados para representar gráficamente la topología de una red ad-hoc.

VI.2. IMPLEMENTACIÓN DE LOS MECANISMOS DE CIFRADO

Al margen de los desarrollos realizados con el fin de evaluar el rendimiento y otros aspectos del LFSRe, éste ha sido implementando dentro de un marco más general, que le permite proteger todas las comunicaciones de una máquina.

Más concretamente, este software ha sido desarrollado y probado en el marco del proyecto Hesperia[1], donde ha sido validado con la presentación definitiva del mismo al CDTI. Actualmente está integrado en el *middleware* del proyecto, donde se encarga del cifrado de todas las comunicaciones del mismo.

VI.2.1. Selección de los entornos y herramientas de implementación

El objetivo de esta implementación ha sido, por tanto, dotar al algoritmo LFSRe de un sistema completo capaz de cifrar todas las comunicaciones de una máquina. Para ello, es necesario que este sistema interactúe con el sistema operativo, más concretamente con la pila de comunicaciones (habitualmente, TCP/IP).

El sistema de cifrado ha sido finalmente diseñado y desarrollado en lenguaje C para entornos Linux, haciendo uso de la librería IP_QUEUE, formalmente *libnetfilter_queue*, que forma parte, a su vez, del proyecto Netfilter[2].

Esta librería proporciona acceso completo al subsistema de red y a las colas que almacenan los paquetes en tránsito por el filtro de paquetes del núcleo del sistema operativo. De esta forma, en espacio de usuario, es posible descartar, inspeccionar o modificar estos paquetes antes de que continúen su camino por la pila TCP/IP de la máquina. En nuestro caso, la modificación de los paquetes consistirá en su cifrado o descifrado, según estén siendo enviados o recibidos, respectivamente.

Obviamente, no todo el contenido de un paquete puede ser cifrado pues éste incluye, además de los propios datos, cabeceras de control y otras estructuras, utilizadas por el protocolo de red. Por esta razón, un paquete, o trama, no puede ser cifrado por debajo de la capa de red, incluyendo a ésta.

Pila de comunicaciones y seguridad La siguiente cuestión que fue necesario resolver consiste en decidir en qué lugar de la pila de comunicaciones situar nuestra solución de seguridad. En general, los protocolos de seguridad tienden a situarse en niveles altos de la pila OSI. Pueden citarse literalmente decenas de ejemplos, pero algunos de los más significativos podrían ser los protocolos SSH, S-HTTP o SFTP, todos ellos de nivel de aplicación.

El inconveniente obvio de este enfoque es que cada aplicación necesita de una solución propia, que no sirve para proteger a ninguna otra aplicación. Esto supone una

[1] El proyecto Hesperia ha tenido por objeto el desarrollo de tecnologías que permitan la creación de sistemas punteros de seguridad, vídeo vigilancia y control de operaciones de infraestructuras críticas y espacios públicos.

El CDTI, organismo adscrito al Ministerio de Industria Turismo y Comercio, creó en 2005 un programa de Consorcios Estratégicos Nacionales en Investigación Técnica (CENIT), cuyo principal objetivo es fomentar la cooperación público privada en I+D+i mediante la financiación de proyectos conjuntos de investigación industrial.

El consorcio está integrado por un conjunto de empresas y universidades españolas. Entre estas últimas se encuentran la Universidad de Castilla-La Mancha, la Universidad de Granada, la Universidad de Extremadura, la Universidad Politécnica de Madrid, la Universidad de Las Palmas de Gran Canaria, la Universidad Politécnica de Valencia y la Universidad Politécnica de Cataluña. La lista se completa con la colaboración del Consejo Superior de Investigaciones Científicas (CSIC) y el Centro Tecnológico del País Vasco (Ikerlan).

[2] La librería, junto con la documentación pertinente, es accesible en la siguiente URL: http://www.netfilter.org/projects/libnetfilter_queue/index.html

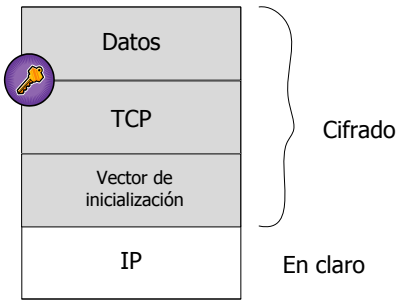


Figura VI.1: Trama de datos de nuestra implementación, donde puede observarse la parte cifrada y la que permanece en claro. El ejemplo supone una pila de comunicaciones TCP/IP.

duplicidad innecesaria de esfuerzos y riesgos. Además, cualquier cambio en el protocolo de comunicaciones de cada aplicación puede implicar el rediseño de la solución de seguridad.

Por tanto, siempre que sea posible, resulta recomendable situar las protecciones en el nivel más bajo posible de la pila de comunicaciones. En este sentido, protocolos como SSL o IPsec resultan más eficientes pero, desgraciadamente, por su modo intrínseco de funcionamiento, no resultan adecuados en redes de naturaleza distribuida como una red ad-hoc.

Por estas razones, nuestra implementación se sitúa a nivel de red, protegiendo todas las capas superiores. La principal ventaja de este enfoque es que resulta completamente transparente al usuario y a las aplicaciones, que no deben ser modificadas ni reescritas en absoluto. Permite además, proteger cualquier aplicación de forma automática, sin necesidad de escribir protocolos de protección específicos para ella.

Estructura de la trama cifrada Como se ha explicado en los capítulos correspondientes, todo cifrador en flujo necesita de un vector de inicialización, IV, que diversifique el contenido de los datos de entrada. Éste valor debe ser único para cada paquete, y enviado junto a éste, para que el receptor pueda utilizarlo en el proceso de descifrado.

Una trama de datos de nuestro cifrador tendrá, por tanto, una estructura característica, que se muestra en la Figura VI.1. En ella se observa cómo las únicas porciones de los datos que permanecen sin cifrar son el propio IV y la cabecera del nivel de red (habitualmente IP), con el fin de que el paquete pueda ser correctamente encaminado en su viaje entre emisor y receptor.

VI.2.2. Funcionamiento

La implementación hace uso de una serie de *hooks*, proporcionados por la librería de Netfilter. Estos *hooks* o funciones son llamados por el sistema operativo cuando se cumple un criterio predefinido. Descrito en lenguaje natural, uno de estos criterios podría ser, por ejemplo:

> "...captura todos los paquetes de entrada cuyo puerto TCP de destino sea 23..."

De esta forma, todos los paquetes que cumplan esta regla serán entregados a nuestro código, que lleva a cabo, entonces, los siguientes pasos:

1. Se realizan una serie de comprobaciones rutinarias sobre el paquete, como que posee una longitud mínima adecuada.
2. Si el paquete debe ser cifrado, se genera un nuevo *IV* y se añade tras la cabecera de nivel de red (IP).
3. Si el paquete debe ser descifrado, se recupera el *IV* de la trama.
4. Se llama a función de cifrado (que es la misma que para el descifrado), que toma como parámetros el *IV* y los datos a cifrar (la porción del paquete de nivel de transporte y capas superiores).
5. Se deja continuar a la trama su viaje por la pila de comunicaciones. El sistema operativo decidirá si es un paquete de salida, por lo que debe ser enviado a un receptor remoto, o es de entrada, por lo que debe ser entregado a la aplicación correspondiente.

VI.2.3. Rendimiento y pruebas

Como se ha comentado con anterioridad, este sistema forma parte actualmente del activo experimental definitivo del proyecto Hesperia. Durante la pruebas realizadas durante su desarrollo, consistente en la protección de diferentes flujos de audio y vídeo, el sistema funcionó correctamente, sin ninguna degradación medible en el rendimiento.

La razón puede encontrarse en que las tasas de transmisión reales fueron muy inferiores a las teóricas, debido a multitud de factores, como interferencias, pérdidas y atenuación de señal o las colisiones inherentes del medio inalámbrico.

VI.3. IMPLEMENTACIÓN DE LA AUTENTICACIÓN

El objetivo de este sección es demostrar, una vez presentado el diseño del protocolo de autenticación, la viabilidad del mismo en un entorno completamente real. Para ello se ha llevado a cabo su implementación práctica en un entorno de laboratorio.

A pesar de que este enfoque resulta claramente más costoso que las habituales simulaciones, los autores creen que posee una serie de ventajas importantes. La primera, por supuesto, es que supone una demostración innegable de la viabilidad de la propuesta. Proporciona, por otro lado, datos fiables y exactos en cuanto al tiempo de ejecución, consumo energético y otros aspectos, que no siempre son adecuadamente tratados en las simulaciones.

De hecho, la principal de las razones que han hecho que los autores se decanten por una implementación real de las propuestas es que, a pesar de que la simulación es la herramienta por excelencia utilizada para la evaluación de las soluciones propuestas en este tipo de redes, su aplicación puede no resultar siempre adecuada. Esta opinión es, por otra parte, compartida por los propios autores y otros investigadores, como [1, 2], que plantean serias dudas sobre la credibilidad de los resultados obtenidos en base a simulaciones.

Selección de los entornos y herramientas de implementación Una vez tomada la decisión de implementar en entorno real, restaba la elección de la plataforma concreta donde trabajar y qué herramientas de desarrollo utilizar.

Tras evaluar diversas alternativas, como la conocida arquitectura del fabricante Microchip, finalmente se decidió hacer uso de la plataforma libre Arduino [3, 4], basada en microcontroladores Atmel, por el potencial y multitud de ventajas que ésta presenta.

Las más significativa de ellas es que se trata de una plataforma completamente flexible, de código abierto, por lo que tanto el hardware como el software pueden ser fácilmente modificados, sin ningún de restricciones, para adaptarlos a cada necesidad. Por otro lado, Arduino también se ajusta a otro de los grandes objetivos de esta tesis, que consiste en proporcionar soluciones *realistas y económicas*, especialmente orientadas a dispositivos de gama baja. El bajo precio de la plataforma hace que sus características técnicas encajen en esta categoría, resultando, incluso, como veremos en la siguiente sección, algo limitadas.

VI.3.1. La plataforma Arduino

El proyecto Arduino es una de las primeras plataformas hardware libres del mundo. Orientada al desarrollo electrónico, está basada en software y hardware flexible y fácil de utilizar. De hecho, uno de los principales objetivos es servir de plataforma educativa para estudiantes.

Todos los elementos del proyecto son abiertos y libres: la placa física puede montarse a mano o comprarse ya montada, y el software de desarrollo es gratuito. El diseño hardware de la placa es, también, libre y los ficheros CAD están disponible bajo una licencia de código abierto, de forma que pueden modificarse para adaptar el diseño a las necesidades de cada proyecto.

En esta tesis, se ha trabajado con el modelo Diecimila[3] de esta plataforma. Puede encontrarse una fotografía del diseño en la Figura VI.2 y un resumen de sus características técnicas más importantes en el Cuadro VI.1.

[3]La palabra *Diecimila* significa *diez mil* en italiano y hace referencia al hecho de que las primeras diez mil unidades de la plataforma habían sido ya fabricadas.

Microcontrolador	ATmega168
Voltaje de operación	5 V
Voltaje de entrada (máximo)	20 V
Pines de E/S digitales	14 (6 de los cuales pueden producir salidas PWM)
Pines de entrada analógicos	6
Corriente disponible en pines E/S	40 mA
Memoria FLASH	16 Kb (2 de los cuales son utilizados por el cargador)
SRAM	1 Kb
EEPROM	512 bytes
Velocidad de reloj	16 MHz

Cuadro VI.1: Resumen de características técnicas del Arduino Diecimila

Comunicaciones El Diecimila cuenta con una serie de interfaces de comunicaciones, que le permiten comunicarse con un ordenador, otro Arduino u otros microcontroladores. El corazón de la placa, un microcontrolador ATmega168, dispone de una UART[4] accesible en los pines 0 (RX) y 1 (TX). Un chip FTDI FT232RL permite simular un puerto serie sobre un puerto USB estándar, de forma que la comunicación con el ordenador para la programación o alimentación de la placa es muy sencilla. Por último, la plataforma también soporta los protocolos I2C y SPI[5], que permiten la comunicación con otros dispositivos, como memorias externas.

VI.3.2. Cryptonite: librería criptográfica ligera

A pesar de la popularidad de la plataforma, y de las múltiples librerías y utilidades disponibles para la misma[6], uno de los primeros problemas que se presentó en el desarrollo fue la inexistencia de una librería con funciones criptográficas.

Se decidió, por tanto, desarrollar una nueva librería criptográfica, que hemos denominado Cryptonite, para poder llevar a cabo la implementación del protocolo de autenticación y, adicionalmente, llenar este vacío, aportando la librería a la comunidad de software libre[7].

En las siguientes secciones se describirán algunos conceptos criptográficos básicos y las principales características de la librería desarrollada.

VI.3.2.1. Diseño y consideraciones iniciales

La principal dificultad a superar en el desarrollo de Cryptonite fue las fuertes limitaciones del microcontrolador de Arduino. Su escaso 1 Kb de memoria RAM hacen imposible el uso de las primitivas criptográficas tradicionales, como SHA-1. En este caso concreto, sólo el tamaño del código correspondiente a esta función hash superaba ampliamente el total de memoria disponible.

[4]Una unidad UART (*Universal Asynchronous Receiver-Transmitter*) se encarga de traducir los datos enviados en formato paralelo a formato serie y viceversa. Normalmente se utiliza para comunicar distintos periféricos, que suelen tratar los datos interiormente de forma paralela, a través de un enlace, que suele empaquetar los datos en serie

[5]Tanto I2C como SPI son dos protocolos de comunicaciones, desarrollados por Philips y Motorola, respectivamente, diseñados para comunicar dispositivos de bajas velocidades de transmisión

[6]Accesibles en la siguiente URL: http://arduino.cc/en/Reference/Libraries

[7]La librería se distribuye bajo una licencia libre GPLv3, y puede encontrarse en la siguiente URL: http://code.google.com/p/crypto-arduino-library/

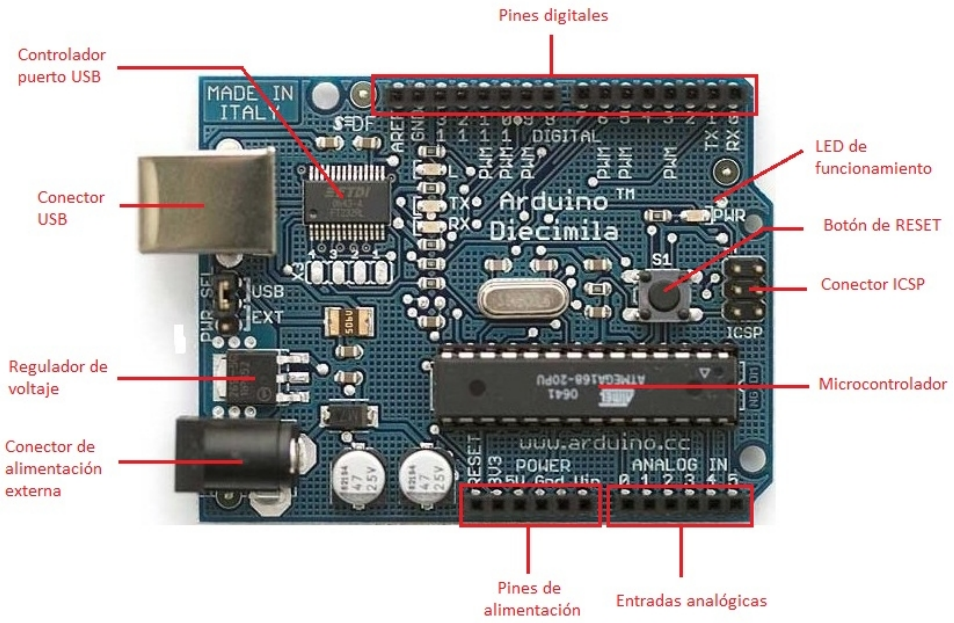


Figura VI.2: Arduino Diecimila

TEA Dada la situación, se hizo imprescindible, por tanto, considerar otras primitivas criptográficas cuyo código y huella de memoria fueran mínimas. En este sentido, destaca el algoritmo TEA (*Tiny Encryption Algorithm*), presentado en 1995 por David Wheeler y Roger Needham, de la Universidad de Cambridge [5, 6]. La cualidad más característica de TEA es, sin duda, su mínimo tamaño, pues puede ser implementado en unas pocas líneas de código C. En la Figura VI.3 puede encontrarse una implementación de referencia.

TEA opera en bloques de 64 bits y utiliza una clave de 128 bits. El proceso de cifrado consiste esencialmente en una estructura de Feistel, con 64 rondas recomendadas. Cuenta con un mecanismo de mezcla de clave muy sencillo y efectivo, puesto que funciona de igual manera en todos los ciclos. Por último, hace uso de las denominadas constantes mágicas, con valor $2^{32}/\phi$, siendo ϕ el número áureo, con el fin de evitar ataques sencillos basados en la simetría de las rondas.

TEA sufre, sin embargo, de ciertas vulnerabilidades. La más importante consiste en que cuenta con una serie de claves equivalentes, de forma que cada clave corresponde a otras tres. Esto significa que, en la práctica, el espacio de claves se reduce de 2^{128} a $2^{128}/2^2 = 2^{126}$. Debido a éste y otros inconvenientes [7, 8], los autores han desarrollado nuevas versiones, que se describen brevemente a continuación, para solucionar estos problemas.

Revisiones de TEA La primera de las revisiones se denominó Block TEA [9], aunque el nombre más extendido es XTEA. Al margen de solucionar los problemas mencionados, cuenta con la capacidad de operar sobre bloques de longitud arbitraria, en lugar de los 64 bits de la propuesta original. Finalmente, en 1998, se publicó la que hasta la fecha es la última revisión, denominada XXTEA, que será la utilizada en este trabajo, y que incluye algunas mejoras adicionales.

Algoritmo 17 Implementación de Block TEA corregido (XXTEA)

```
#define DELTA 0x9e3779b9
#define MX ((z>>5^y<<2) + (y>>3^z<<4)) ^
  ((sum^y) + (k[(p&3)^e] ^ z));

void XXTEA(uint32_t *v, int n, uint32_t *k)
 uint32_t y, z, sum;
 unsigned p, rounds, e;

 if (n > 1) {          // Cifrado
  rounds = 6 + 52/n;
  sum = 0;
  z = v[n-1];
  do {
    sum += DELTA;
    e = (sum >> 2) & 3;
    for (p=0; p<n-1; p++)
      y = v[p+1], z = v[p] += MX;
    y = v[0];
    z = v[n-1] += MX;
  } while (--rounds);
 } else if (n < -1) {  // Descifrado
  n = -n;
  rounds = 6 + 52/n;
  sum = rounds*DELTA;
  y = v[0];
  do {
    e = (sum >> 2) & 3;
    for (p=n-1; p>0; p--)
      z = v[p-1], y = v[p] -= MX;
    z = v[n-1];
    y = v[0] -= MX;
  } while ((sum -= DELTA) != 0);
 }
}
```

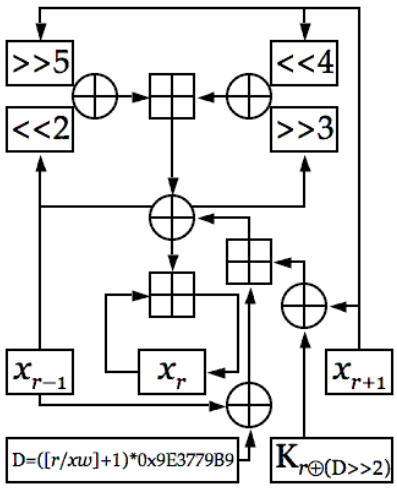


Figura VI.3: A la izquierda, implementación en lenguaje C de Block TEA corregido (XXTEA). El algoritmo cifra n palabras de 32 bits como un bloque, donde v es el texo en claro (en bloques de n palabras), k es la clave de cifrado (de 4 palabras de longitud) y n es un parámetro que toma valores positivos para cifrar y negativos para descifrar. A la derecha, un esquema de una ronda de XXTEA.

Una vez seleccionado y analizado XXTEA como bloque básico con el que construiremos nuestra librearía criptográfica Cryptonite, resta definir cada una de las primitivas criptográficas básicas, *cifrado, hash y HMAC*, a partir de TEA. Este proceso se analizará en las siguientes secciones.

VI.3.2.2. Primitivas de cifrado

Puesto que XXTEA está definido como un algoritmo de cifrado, definir dicho proceso resulta una tarea trivial. Hay que realizar, sin embargo, algunas consideraciones, sobre todo en lo concerniente al tamaño de los bloques de texto en claro.

Como se ha comentado anteriormente, XXTEA permite trabajar con bloques de en-

Algoritmo 18 Implementación de Block TEA corregido (XXTEA)

```
/* FUNCION: encrypt
 * ARGS:
 *      - inputText : puntero al texto en claro
 *      - inputTextLength : longitud del texto en claro (en bytes)
 *      - k[4] : clave de cifrado, en forma de array de cuatro elementos de
                                                    32 bits = clave de 128 bits.
 * SALIDA: numero de bloques cifrados, -1 en caso de error
 */
int Cripto::encrypt(unsigned char *inputText,
                    unsigned int inputTextLength,
                    uint32_t *k)
{
  unsigned int numBlocks = (inputTextLength <= BLOCK_SIZE ? 1 :
                                          inputTextLength/BLOCK_SIZE);
  unsigned int offset, i;

  // Rellenar (con ceros) si es necesario hasta el tamaño de un bloque
  if ((offset = inputTextLength % BLOCK_SIZE) != 0)
          memset(inputText+inputTextLength, 0x00, BLOCK_SIZE - offset);

  for (i=0; i<numBlocks;i++){
     btea((uint32_t *)inputText, BLOCK_SIZE/4,k);
     inputText+=BLOCK_SIZE;
  }

  return numBlocks;
}
```

trada de longitud variable. Sin embargo, como suele ser habitual en criptografía, es necesario encontrar un equilibrio para el tamaño de bloque entre rendimiento y seguridad.

En cuanto al rendimiento, se pretendía que para cifrar un bloque de entrada dado, el cifrador sólo necesitara llevar a cabo una ronda en la mayoría de los casos. Obviamente, cuando mayor sea el tamaño de bloque interno del cifrador, mayor es su rendimiento. Por ejemplo, para un texto de entrada de 512 bits y un tamaño de bloque de $B = 128$ bits, el cifrador necesitará realizar $512/128 = 4$ rondas. Por otro lado, con $B = 32$ bits, el número de rondas se incrementa hasta 16.

Sin embargo, un tamaño de bloque B grande, sobre todo en la plataforma en la que trabajamos, tiene la contrapartida de que en el caso de que sea necesario cifrar un bloque de datos menor a B, es necesario realizar un relleno del mismo hasta completar la longitud de B, con el consiguiente gasto de tiempo y memoria.

Teniendo en cuenta todas las consideraciones anteriores, se decidió establecer el tamaño de bloque B en 128 bits.

VI.3.2.3. Funciones hash

En general, existen tres grandes categorías de funciones hash, que se clasifican según su proceso de construcción:

- Basadas en cifradores en bloque: aprovechan esta primitiva criptográfica, que se supone ya disponible.
- Funciones ad-hoc: las funciones hash tradicionales, específicamente diseñadas para su propósito.

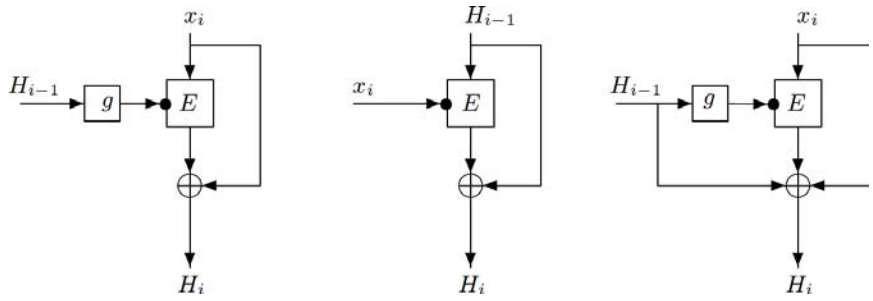


Figura VI.4: Esquemas existentes para la construcción de funciones hash a partir de un cifrador en bloque: Matyas-Meyer-Oseas (izquierda), Davies-Meyer (centro) y Miyaguchi-Preneel (derecha).

- Basadas en aritmética modular.

Como se ha comentado al inicio de esta sección, las funciones hash tradicionales, como MD5 o toda la familia SHA, resultan demasiado "pesadas" para ser implementadas en una plataforma de gama baja como Arduino. Por esta razón, se decidió crear una función hash a partir de una cifrador en bloque. A continuación se detallan los pormenores del proceso.

Construcción de una función hash a partir de un cifrador La obtención de funciones hash basadas en cifradores en bloque es una técnica bien conocida y estudiada en la literatura [10]. La razón principal para utilizar este enfoque radica en aprovechar la implementación eficiente existente de un cifrador en bloque, XXTEA en nuestro caso, y utilizar éste como elemento central en el diseño de la función hash.

A continuación se presentan unos breves conceptos necesarios para la argumentación posterior.

VI.1 Definición. Un *cifrador en bloque*, que denotaremos $E_k(x)$, es una función invertible que transforma textos en claro x de n bits en textos cifrados de n bits utilizando un clave k de r bits de longitud.

A la luz de la definición anterior, el proceso de construcción de la función hash se divide, básicamente, en aquellos que producen hashes de *longitud simple* (n bits) o de *doble longitud* ($2n$ bits), siendo n, como acabamos de ver, la longitud del bloque de salida del cifrador subyacente. Además de E_k, son necesarios otros elementos para completar el proceso:

1. Una función g que transforma los bloques de entrada de texto en claro de n bits a claves k adecuadas para ser utilizadas en E. Obviamente, si las claves de E son también de longitud n, g consiste simplemente en la función identidad.
2. Un vector de inicialización IV, habitualmente de n bits de longitud.

Con estos elementos, existen tres grandes enfoques para la definición de la función hash H a partir de un cifrador en bloque E, denominados de acuerdo a los nombres de los autores que los presentaron, Matyas-Meyer-Oseas, Davies-Meyer y Miyaguchi-Preneel, respectivamente, que pueden encontrarse esquematizados en la Figura VI.4.

VI.2 Definición. Sea H una función hash construida a partir de un cifrador en bloque E, que necesita realizar s cifrados para procesar cada bloque de texto de entrada. Entonces el *ratio* de H es $1/s$.

Función hash	(n, k, m)	Ratio
Matyas-Meyer-Oseas	(n, k, n)	1
Davies-Meyer	(n, k, n)	k/n
Miyaguchi-Preneel	(n, k, n)	1
MDC-2 (con DES)	$(65, 56, 128)$	$1/2$
MDC-4 (con DES)	$(65, 56, 128)$	$1/4$

Cuadro VI.2: Resumen de algunas funciones hash basada en cifradores de bloque de n bits, done k es el tamaño de clave de los mismos y m la longitud de la salida producida por las funciones hash. MDC-2 y MDC-4 son funciones de *doble longitud*, basadas en el uso de DES como cifrador subyacente y que requieren 2 y 4 operaciones de cirado por cada bloque de texto de entrada, respectivamente.

En el Cuadro VI.2 se resumen los ratios de cada uno de los enfoques existentes. Por ejemplo, el enfoque Matyas-Meyer-Oseas y el algoritmo MDC-2 son la base de las dos funciones hash definidas en el estándar ISO 10118-2 ("*Information technology – Security techniques – Hash-functions*"), pudiendo hacer uso de cualquier cifrador E de n bits y proporcionando salidas de longitud $m \leq n$ y $m \leq n$, respectivamente.

Implementación Teniendo en cuenta los argumentos presentados, finalmente se decidió utilizar el enfoque Davies-Meyer para la construcción de la función hash, por resultar eficiente, siempre que $k = n = m$, ya que no implica el uso de una función intermedia g, y ligeramente más sencilla de implementar que el resto de opciones. En una plataforma tan limitada como Arduino, ésta características se convierte es una ventaja importante que debe tenerse en cuenta.

En la implementación realizada, se decidió utilizar un tamaño de bloque interno B y una salida m de 128 bits, obteniendo así un ratio de 1. El código correspondiente puede encontrase en el Algoritmo 19.

VI.3.2.4. Primitiva de HMAC

La tercera y última primitiva criptográfica implementada en Cryptonite corresponde a una *función hash con clave* o HMAC. Por la mismas razones expuestas en las sección anterior, en este caso también se decidió aprovechar la implementación de XXTEA para utilizarla como base, en lugar de implementar una función ad-hoc.

De acuerdo a procedimientos conocidos [10], la construcción del HMAC puede realizarse haciendo usode una clave k y una función hash h. De esta forma, para un texto de entrada x, la función queda de la siguiente forma:

$$HMAC(x) = h(k||p_1||h(k||p_2||x))$$

donde p_1, p_2 son dos cadenas arbitrarias de longitud suficiente para completar la longitud de k hasta el tamaño de un bloque completo de la función hash. A pesar de la doble iteración de h, la función final resulta eficiente, debido a que la iteración externa solo procesa una entrada de dos bloques de longitud, independientemente de la longitud de x.

La implementación final realizada para la plataforma Arduino puede encontrarse en el Algoritmo 20.

Algoritmo 19 Implementación de una función hash basada en XXTEA, que hace uso del enfoque Davies-Meyer

```
/* NOMBRE: calculateHash
 * ARGUMENTOS ENTRADA:
 *      - inputText : puntero al texto de entrada
 *      - inputTextLength : longitud del texto de entrada (en bytes)
 */
void Cripto::calculateHash(unsigned char *inputText,
                           unsigned int inputTextLength,
                           unsigned char *hash)
{
  unsigned int numBlocks =
      (inputTextLength <= HASH_LENGTH ? 1 : inputTextLength/HASH_LENGTH), i,j;
  unsigned char H[HASH_LENGTH]={HASH_IV};
  unsigned char tempBuffer[BLOCK_SIZE];

  for (i=0; i<numBlocks;i++){
     memcpy(&tempBuffer,inputText,BLOCK_SIZE);
     XXTEA((unsigned char *)&tempBuffer,KEY_SIZE,(uint32_t *)&H);

     // Se actualiza el puntero de la entrada, x_i
     for (j=0; j<HASH_LENGTH;j++)
         H[j]=tempBuffer[j]^(*inputText++);
  }

  memcpy(hash,&H,HASH_LENGTH);

}
```

VI.3.3. Detalles de implementación y resultados

En esta sección describiremos los mayores problemas encontrados durante la implementación de Cryptonite, cómo éstos fueron solucionados, y algunas de las lecciones aprendidas. Por último, se analizará el rendimiento de la librería y cómo se utiliza ésta para la implementación del protocolo de autenticación presentado en este capítulo.

Problemas de desarrollo El desarrollo de la librería resultó, sin duda, algo problemática debido a las importantes restricciones del hardware de la plataforma Arduino. Por ejemplo, el reducido tamaño de su memoria RAM producía frecuentes agotamientos de la misma. Debido a las limitaciones inherentes de la arquitectura, esta circunstancia no se notifica al desarrollador, a diferencia de la plataforma Intel PC, produciéndose mal funcionamientos erráticos y difíciles de reproducir.

Por otro lado, las herramientas de depuración para la arquitectura AVR, a la que pertenece el microcontrolador de Arduino, resultan todavía algo deficientes y permanecen en un estado de desarrollo poco maduro. Por estas razones, la depuración de los programas en esta plataforma se convierte habitualmente en una tarea lenta y tediosa.

VI.3.3.1. Resultados

En una arquitectura tan limitada, un parámetro crítico a la hora de evaluar la validez de un código es la huella de memoria del mismo, dada la escasez de dicho recurso. En este sentido Cryptonite obtiene buenos resultados, pues ocupa 2108 bytes, que supone poco más del 13 % de los 16Kb totales disponibles.

Algoritmo 20 Implementación de una función HMAC basada en XXTEA

```
/* NOMBRE: calculateHMAC
 * ARGUMENTOS ENTRADA:
 *      - inputText : puntero al texto de entrada
 *      - inputTextLength : longitud del texto de entrada (en bytes)
 *      - k : clave (128 bits)
 */
void Cripto::calculateHMAC(unsigned char *inputText,
                           unsigned int inputTextLength,
                           uint32_t *k,
                           unsigned char *hash)
{
  unsigned char tempHash[HASH_LENGTH];
  unsigned char tempBuffer[inputTextLength + KEY_SIZE];

  memcpy((void *)&tempBuffer,(void *)k,KEY_SIZE);
  memcpy((unsigned char *)&tempBuffer+KEY_SIZE, inputText,inputTextLength);
  calculateHash((unsigned char *) &tempBuffer, inputTextLength + KEY_SIZE,
                                               (unsigned char *) &tempHash);

  memcpy(&tempBuffer,k,KEY_SIZE);
  memcpy((unsigned char *)&tempBuffer + KEY_SIZE, &tempHash, HASH_LENGTH);
  calculateHash((unsigned char *)&tempBuffer, HASH_LENGTH + KEY_SIZE,
                                               (unsigned char *)hash);
}
```

Cryptonite En comparación, otras librerías disponibles para Arduino, como Ethernet o LiquidCrystal[8], poseen tamaños similares, concretamente 1777 y 2172 bytes, respectivamente.

En el Cuadro VI.3 pueden encontrarse más detalles, con el desglose del tamaño que ocupa cada cada una de las funciones de la librería.

Por otro lado, en lo concerniente al rendimiento de la librería, los datos muestran que éste puede considerarse también muy bueno. De acuerdo a los tiempos recogidos en la tabla anterior, una operación de cifrado o de cálculo de hash sobre un bloque de datos de 128 bytes consume unos 9.5 milisegundos. En el caso del cálculo del hash con clave, debido a la forma en el que éste se lleva a cabo (veáse la sección VI.3.2.4), este tiempo se eleva hasta los 13.5 ms.

Protocolo de autenticación Una vez presentado el análisis de Cryptonite, herramienta básica sobre la que está construido el protocolo de autenticación, podemos pasar a evaluar el rendimiento del mismo.

Comenzaremos por recordar brevemente los mensajes intercambiados durante la ejecución del mismo (expuestos con mayor detalle en la sección V.3):

$$\begin{aligned} A \rightarrow B &: r_A \\ A \leftarrow B &: r_B, [r_A]_{k^{j-1}_{auth}}, \{k^B_{enc}\}_{k^j_{auth}} \\ A \rightarrow B &: [r_B]_{k^j_{auth}}, \{k^A_{enc}\}_{k^j_{auth}} \end{aligned}$$

[8]La librería Ethernet permite conectar una placa Arduino a una red de este tipo y, por tanto a Internet. LiquidCrystal, por su parte, sirve para utilizar pantallas de cristal líquido basadas en el chipset Hitachi HD44780, o compatibles, utilizado por la mayoría de LCDs de texto del mercado. Ambas librerías están disponibles en http://arduino.cc/en/Reference/Libraries.

Función	Tamaño (bytes)	Rendimiento (ciclos/byte)	Tiempo operación (μs)
XXTEA	1330	—	—
encrypt	132	1183	9464
decrypt	96	1183	9780
calculateHash	276	1222	9756
calculateHMAC	274	1689	13516
Total	2108		

Cuadro VI.3: Huella de memoria y rendimiento de la librería Cryptonite. Tanto el rendimiento como el tiempo de operación, en microsegundos, hacen referencia a los ciclos por byte y el tiempo empleado en cifrar, descifrar, calcula el hash y el HMAC, respectivamente, de un texto de entrada de 128 bytes.

Función	Tiempo operación (μs)	Rendimiento (ciclos/byte)
Lectura	4	128
Escritura	8	64

Cuadro VI.4: Tiempos de acceso a la memoria EEPROM de la plataforma Arduino. Los tiempos mostrados correspoden a la lectura o escritura de un byte en posiciones aleatorias de la memoria.

Nótese que A es el nodo que desea ingresar en la red e inicia, por tanto, el proceso de autenticación y B el nodo que ya pertenece a la misma y hace las veces de "anfitrión". Como puede observarse en el esquema anterior, A y B intercambian un total de tres mensajes, que involucran el uso de unos retos precalculados.

En nuestra implementación, estos retos, que forman parte de los autenticadores de A y B, ∇^A y ∇^B, respectivamente, son almacenados en la memoria EEPROM de la placa Arduino. Por tanto, el primer paso de la evaluación final consistió en medir los tiempos de acceso a esta memoria. Los resultados pueden encontrarse en el Cuadro VI.4. Como puede apreciarse, estos tiempos, unos $4\mu s$ para la operación de lectura y $8\mu s$ para la escritura, son prácticamente despreciables frente a las operaciones criptográficas, mucho más costosas, de la librería Cryptonite (suponen apenas un 0.08 % de la operación criptográfica más rápida).

Con estos elementos, los resultados finales obtenidos para el protocolo de autenticación se detallan en el Cuadro VI.5. Según los datos mostrados, cada nodo necesita invertir algo menos de 20 ms en la generación y procesamiento de cada uno de sus mensajes, lo que supone unos 38 ms para la ejecución completa del protocolo. Teniendo en cuenta que el protocolo ha sido diseñado para ser utilizado principalmente en redes de sensores, donde la tasa de autenticaciones se estima baja, estos valores ofrecen un rendimiento muy superior al que podría considerarse mínimo.

Nodo	Mensaje	Operación			Tiempo	
		EEPROM	HMAC	Cifrado	Parcial	Total
A	M_1	1	0	0	4	19224
	M_3	0	1	1	19220	
B	M_2	1	1	1	19224	19224

Cuadro VI.5: Tiempos de ejecución del protocolo de autenticación presentado, desglosado por cada uno de los mensajes intercambiados por los nodos A y B. Se muestran asimismo el número de operaciones realizadas y los tiempos invertidos en cada mensaje.

VI.4. APOLO Y DAFNE

Finalmente, en esta sección se describe la última de las aportaciones en forma de software realizadas en esta tesis. Si bien no está directamente relacionada con la seguridad de las redes ad-hoc, ha sido de gran valor durante la realización de este trabajo y, a día de hoy, resuelve un área que carece de herramientas adecuadas en la comunidad que trabaja en el campo.

El área a la que nos referimos está relacionada con la representación gráfica o visualización de la topología de estas redes. Hasta el momento, esta tarea resulta complicada y exige conocimientos profundos de la plataforma, debido a la falta de herramientas que faciliten este trabajo.

Por esta razón, se decidió desarrollar una aplicación gráfica específica, que resolviera esta cuestión. Esta herramienta ha sido diseñada en forma de cliente/servidor, y está constituida por dos módulos que hemos denominado *Apolo* y *Dafne*, respectivamente.

VI.4.1. Apolo

El agente Apolo constituye el cliente del sistema, y ha sido desarrollado en forma de *plugin*, o extensión, para la plataforma OLSRd [11, 12]. Ésta es, probablemente, la implementación libre del protocolo OLSR (véase sección II.2.2.2, pág. 43) más utilizada y extendida del mundo.

Funcionamiento El funcionamiento de Apolo es muy sencillo. Esta extensión debe ser instalada en cada uno de los nodos de la red cuya topología se desea monitorizar. Cuando ésta se pone en marcha, los agentes comienzan a reunir y enviar información a Dafne sobre la topología de la porción de la red en la que se encuentra cada uno de ellos. Posteriormente, Dafne, como veremos en la siguiente sección, recolecta esta información, la normaliza y "reconstruye" la estructura completa de la red a partir de los datos parciales de cada agente Apolo. En la Figura VI.5 puede encontrarse una esquematización de una red de ejemplo y sus agentes.

Formato de los mensajes Para ahorrar los valiosos recursos energéticos y el ancho de banda disponibles, el agente Apolo no envía sus mensajes a Dafne a intervalos regulares, sino únicamente cuando se produce un *cambio en su vecindad* (se ha detectado un nuevo nodo, o alguno ha abandonado la red), o en la *topología* (ha cambiado el enlace entre dos nodos vecinos, por ejemplo).

De esta forma, cuando un agente Apolo dispone de nueva información, genera y envía un mensaje al nodo Dafne con el siguiente formato:

```
LINK#<<ID_nodo_origen>>#<<ID_nodo_destino>>#<<calidad_enlace>>#<<tipo_nodo>>
```

Los parámetros anteriores tienen el siguiente significado:

- **LINK**: cabecera del mensaje. Hace referencia a que éste mensaje informa sobre un enlace existente entre el nodo que envía la información y otros.
- **ID_nodo_origen** y **ID_nodo_destino**: constituyen los identificadores de los nodos que participan en el enlace.
- **calidad_enlace**: es una medida de la calidad del enlace entre ambos nodos, en una escala que varía de 0.0 (no existe enlace) a 1.0 (calidad máxima). Esta medida se obtiene a está basado en el RSSI de la señal (*Received Signal Strength Indicator*), que proporciona una medida de la potencia de la misma.

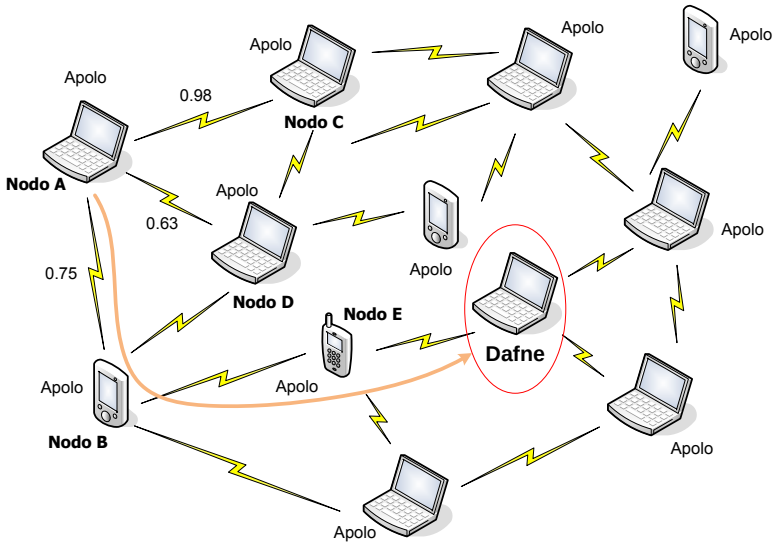


Figura VI.5: Estructura de una red ad-hoc inalámbrica con los agentes Apolo y Dafne instalados en sus nodos. El nodo que contiene a Dafne está señalado por un círculo rojo. Las etiquetas numéricas sobre los enlaces entre los nodos A, B, C y D indican la calidad del mismo, en una escala de 0.0 (no hay enlace) a 1.0 (calidad de enlace máxima).

- **tipo_nodo**: es un parámetro opcional, e indica, en el caso de utilizarse, el tipo de nodo en el que el agente Apolo está instalado. Debido a que prácticamente cualquier plataforma hardware móvil puede constituir el nodo de una red ad-hoc móvil, este parámetro proporciona una forma de identificarla fácilmente. Algunos de los tipos existentes hasta el momento incluyen los siguientes: {LINUX, FR (FreeRunner), N810 (Nokia N810), ACER, HP-DX5150}.

Por ejemplo, en la red representada en la Figura VI.5, el nodo A enviaría a Dafne el siguiente mensaje (que sería enrutado, probablemente, a través de los nodos B y E):

```
LINK#Nodo A#Nodo B#0.75#FR-
    Nodo A#Nodo C#0.98#LINUX-
    Nodo A#Nodo D#0.63#LINUX
```

Plataforma FreeRunner Con el fin de avanzar en la prueba de las propuestas de esta tesis en entornos y plataformas reales, se decidió trabajar en un nuevo hardware. Teniendo en cuenta el reciente y espectacular auge de los denominados *smartphones*, cuyas capacidades técnicas no están ya lejos de cualquier ordenador de sobremesa o portátil, se eligió una de estas plataformas.

Concretamente, se trata del *Neo FreeRunner*, que puede considerarse el primer *smartphone* libre del mundo, desarrollado por el proyecto Openmoko. Al margen de sus excelentes características (dispone, en otras, de GSM, Bluetooth, WiFi o GPS), lo más importante de esta plataforma, sin duda, es que tanto el hardware como el software de la misma son libres, de forma que todas las especificaciones técnicas son accesibles y modificables sin restricciones.

Como no podía ser de otra forma, el sistema operativo del teléfono consiste en una

distribución estándar de Linux ligeramente modificada. Por esta razón, el desarrollo de aplicaciones para esta plataforma resulta sencillo.

Con el fin de conseguir que el teléfono pudiera formar parte de las redes ad-hoc de nuestros experimentos, se inició el transporte del código del demonio OLSR a esta nueva plataforma. El proceso no resultó sencillo, puesto que hubo que compilar, una a una, cada dependencia, en forma de librerías, del ejecutable principal. Finalmente, se consiguió que el código pueda ejecutarse sin problemas en FreeRunner, incluido el agente Apolo. Tras la finalización de esta tesis, se pretende liberar éste código, para que pueda ser utilizado por la comunidad de software libre.

VI.4.2. Dafne

Como hemos comentado ya brevemente, Dafne constituye el servidor del sistema, y su objetivo es recoger, normalizar y representar gráficamente la información topológica que recibe de los agentes Apolo.

Esta representación cobra especial importancia debido a que cada nodo sólo posee información local sobre sus vecinos más inmediatos y los enlaces que le unen con ellos. Haciendo uso de nuestra herramienta es posible conocer, en todo momento, el estado global de la red, con información sobre el número de nodos y la calidad de los enlaces entre ellos, en otras.

Es importante señalar, sin embargo, que la topología representada en el interfaz gráfico sólo muestra la estructura "lógica", y no la disposición física de los nodos en el espacio. Para poder hacerlo de esta última forma, los nodos deberían disponer de hardware específico, como un módulo GPS, que encarecería muy significativamente el coste total de la red. En cualquier caso, ésta constituye una línea de investigación futura (más información sobre ella en la sección VII.2 del Capítulo VII).

Funcionamiento De acuerdo a una de las premisas básicas de las redes ad-hoc ("Todos los nodos deben ser iguales"), el nodo que acoge al servidor Dafne es un nodo más de la red, que no debe cumplir ningún requisito ni poseer ninguna característica especial.

Dafne ha sido desarrollado en lenguaje Java, lo que permite el uso de la herramienta en diferentes sistemas operativos y facilita el desarrollo de la interfaz gráfica.

Una vez instalado, Dafne abre un socket UDP, en el puerto 4447, y permanece a la escucha. Cuando recibe un paquete de datos, con el formato correcto de acuerdo al descrito en la sección anterior, la herramienta comienza la representación gráfica.

Representación Esta representación se hace en forma de grafo no dirigido. Los vértices del mismo representan los nodos de la red, y las aristas, los enlaces entre ellos. Por último, el peso de cada arista hace referencia a la calidad del enlace, y se representa con una etiqueta numérica sobre la misma.

Conforme recibe los mensajes de topología, Dafne recalcula el grafo en tiempo real. En la Figura VI.6 puede observarse la representación gráfica que proporciona nuestra herramienta de la red esquematizada en la Figura VI.5.

Como puede observarse en la figura anterior, Dafne utiliza diferentes imágenes para cada uno de los tipos de nodos especificados en la sección anterior. Por supuesto, pueden definirse nuevos tipos de nodos fácilmente, a través de un fichero de configuración.

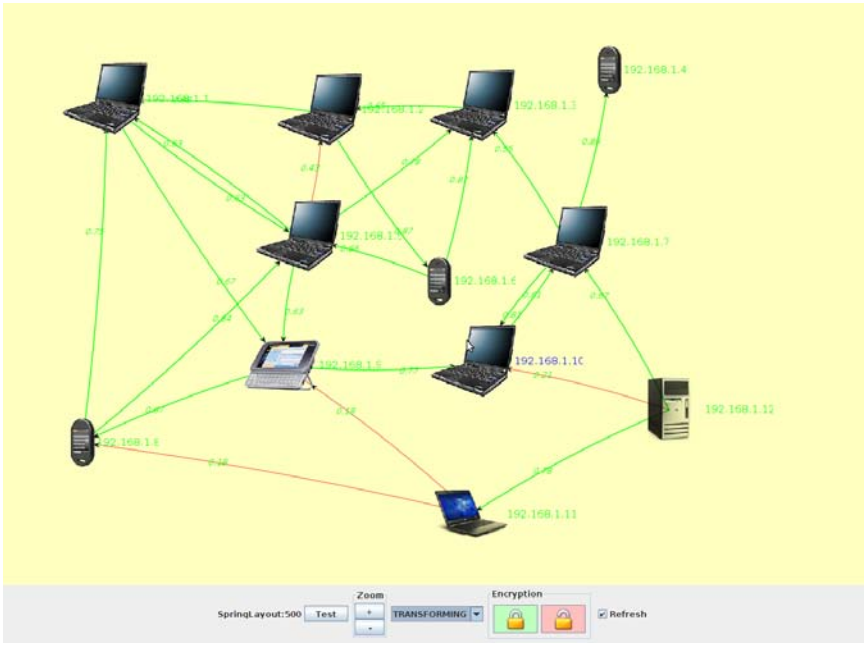


Figura VI.6: Herramienta Dafne en funcionamiento. Las aristas en color rojo indican una calidad de enlace por debajo de 0.50.

VI.5. CONCLUSIONES

Durante esta fase del trabajo hemos presentado los resultados obtenidos a partir la implementación de las diferentes propuestas realizadas a lo largo de la tesis en entornos reales.

El algoritmo de cifrado en flujo LFSRe ha sido dotado de toda la infraestructura adicional necesaria para poder proteger todas las comunicaciones entrantes y salientes de una máquina. Este desarrollo se ha llevado a cabo inicialmente para entornos Linux, donde, gracias a la disponibilidad de su código fuente, es posible profundizar con mayor facilidad en la estructura interna de la pila de comunicaciones.

Este software de cifrado de tráfico con LFSRe ha sido probado con éxito en el contexto del proyecto Hesperia, donde actualmente forma parte de su núcleo de comunicaciones y se encuentra en uso. Como parte de sus tareas habituales, se encarga del cifrado de flujos de datos multimedia provenientes de los diferentes componentes de la red, entre las que se incluyen cámaras de vigilancia, micrófonos o PDA's portadas por vigilantes de seguridad.

Por otro lado, el protocolo de autenticación propuesto en el Capítulo V ha sido también implementado en un entorno real. Dadas las inherentes restricciones computacionales y energéticas, la codificación de las primitivas criptográficas necesarias para poder llevar a cabo el proceso de autenticación fue una tarea complicada, aunque al mismo tiempo estimulante.

Tras barajar diversas alternativas, se optó por derivar todas las primitivas (cifrado, funciones hash y HMAC) de una única implementación de TEA, que se caracteriza por su pequeña huella de memoria. Como resultado, se ha desarrollado la librería Cryptonite, que constituye hasta la fecha, la primera librería criptográfica libre para la plataforma Arduino. Sin embargo, una de las líneas futuras de trabajo proyectadas incluye el desarrollo de Cryptonite en otras plataformas típicas de redes de sensores. Dado que esta librería ha sido desarrollada completamente en ANSI C y es autocontenida, este traslado no debería suponer problemas importantes.

Por último, como consecuencia de todo el trabajo y experiencia acumulada durante los desarrollos anteriores, se detectaron ciertas carencias en diversas áreas que han sido también suplidas. La primera de ellas está relacionada con la dificultad existente para visualizar, de forma fácil y global, la topología de una red de sensores ya desplegada y en funcionamiento. Para subsanar este problema, hemos desarrollado una solución denominada Apolo y Dafne, que representa de forma gráfica la estructura de toda la red a través de unos pequeños agentes instalados en cada nodo.

Otra de las carencias detectadas, que surgió de la necesidad de incluir el máximo número posible de plataformas diferentes en nuestras redes MANet y de sensores, fue la imposibilidad de ejecutar el software de enrutamiento OLSR, uno de los más conocidos y utilizados, en la plataforma FreeRunner. Tras llevar a cabo el oportuno proceso de compilación, este teléfono libre puede ser utilizado actualmente sin problemas, como un nodo más, en este tipo de redes. Una vez finalizada esta tesis, éste código será liberado con una licencia libre que permita su uso por otras universidades e instituciones.

Capítulo VII

CONCLUSIONES Y TRABAJO FUTURO

VII.1. Conclusiones

Basta con detenernos unos instantes y echar un vistazo a nuestro alrededor, para tomar conciencia de la espectacular revolución que la informática y la electrónica han provocado en nuestras vidas en los últimos años. Internet ha irrumpido en ellas sin ningún disimulo, prometiendo hacerlas más fáciles y divertidas, y, al margen de la personal percepción de que lo haya conseguido o no, es innegable que resulta imposible evitar su influencia en casi todos los aspectos de nuestra vida cotidiana.

Internet de las cosas En un momento como éste, obviamente, la seguridad de la información y las comunicaciones cobra una importancia capital. Resulta inconcebible, por ejemplo, el despegue definitivo de la denominada *Internet de las cosas* sin que se resuelvan primero los actuales problemas y carencias existentes en sus esquemas de seguridad.

Problemas que, sin embargo, no resultan fáciles de resolver, habida cuenta de las importantes restricciones del hardware típico de este tipo de redes. Entre ellas se cuentan unas reducidas capacidades de proceso, de memoria y energéticas de las que disponen sus nodos. A esto se añade que, habitualmente, no es posible disponer de hardware anti-manipulación en los mismos, que elevaría el coste total de la red de forma prohibitiva. Con estos mimbres, resulta obvio que el diseño de nuevos algoritmos para la protección de la información se convierte, desde luego, en una tarea desafiante.

Sin embargo, esta preocupación por la seguridad en los dispositivos no abarca únicamente a la denominada Internet de la cosas, sino que alcanza a infraestructuras críticas y ámbitos gubernamentales. En efecto, al margen de los aspectos más visibles o comerciales, existe una inagotable fuente de otros aspectos de nuestras vidas que han sido también informatizados. La gestión del tráfico aéreo, mirtino, ferroviario o vial, las centrales energéticas tradicionales y nucleares o gran parte de los aparatos de diagnóstico y tratamiento médicos constituyen sólo una pequeña muestra.

Seguridad de las infraestructuras críticas Nótese, además, que todas las infraestructuras anteriores son críticas para el funcionamiento normal de un país, y que el fallo o colapso de sólo una de ellas puede suponer una gran caos con repercusiones impredecibles. Y, curiosamente, no resulta difícil encontrar noticias sobre recientes ataques a otras infraestructuras críticas, como a la señalización ferroviaria en Alemania, o a los paneles informativos de las carreteras en nuestro propio país [1].

Por otro lado, aunque resulte difícil de creer, los expertos llevan años advirtiendo de que este tipo de infraestructuras críticas no están, a día de hoy, adecuadamente protegidas a pesar de su importancia estratégica y elevadísimo riesgo.

Afortunadamente, parte de la percepción de este riesgo parece haber llegado, por fin, al estrato gubernamental europeo y, en este sentido, la reciente creación del *Centro Nacional de Protección de Infraestructuras Críticas* (CNPIC), parece dar fe de ello. Baste como ejemplo las declaraciones del secretario de Estado de Seguridad, que, durante la apertura del I Encuentro Internacional sobre el tema, celebrado en Madrid en Febrero de 2010 [1], señaló que:

> "...no basta con proteger físicamente las infraestructuras sensibles sino que es necesario hacer seguras e invulnerables sus redes de comunicaciones, imprescindibles para su funcionamiento. El peligro del ciberterrorismo es real y puede acarrear consecuencias dramáticas en servicios básicos como la energía o el transporte."

Otra iniciativa que ahonda en la creciente concienciación sobre estas necesidades la constituye la misma existencia del proyecto Hesperia, bajo cuyo amparo se ha llevado a cabo buena parte del trabajo de esta tesis, y cuyo objetivo ha sido el desarrollo de nuevas tecnologías específicamente orientadas a la protección de infraestructuras críticas.

Con este escenario y perspectivas en el horizonte, esta tesis ha elegido uno de los múltiples aspectos del escenario presentado, las denominadas *redes ad-hoc*, para tratar de aportar soluciones de seguridad noveles, eficientes y prácticas, que ayuden a mitigar en lo posible algunos de los riesgos anteriores.

Objetivos y antecedentes de la investigación En este sentido, los grandes objetivos de nuestro trabajo han sido tratar aspectos relacionados con la privacidad y autenticación de la información en este tipo de redes.

Pero toda investigación debe comenzar con un proceso previo de estudio, que permita contextualizar las soluciones que se aporten con posterioridad. En nuestro caso comenzamos por el diseño de una taxonomía que clasifica los tipos de redes ad-hoc existentes en base a los dos criterios que, a nuestro juicio, resultan más diferenciadores: la *movilidad* y *capacidad de proceso* de los nodos que las conforman.

Posteriormente, se procedió a estudiar las principales características de cada una de la redes que surgieron de la clasificación anterior, haciendo especial hincapié en aquellos aspectos que influyen directa o indirectamente en las técnicas de seguridad que le son aplicables. Por ejemplo, la enorme disparidad en las capacidades de cómputo de cada categoría, hacen inviable obtener una única solución, que se ajuste a todas ellas, y resuelva de forma adecuada los problemas que se le platean.

Con los resultados obtenidos en esta etapa, fue posible iniciar el trabajo en la primera de las grandes áreas afrontadas en esta tesis: la privacidad de la información.

VII.1.1. Confidencialidad y registros de desplazamiento extendidos

Dadas las inherentes restricciones de estas redes, ¿resultarán adecuados para ellas los esquemas de cifrado en flujo, sencillos y rápidos? Y, particularmente, ¿cómo se comportarían los registros de desplazamiento (LFSRs) situados en sus nodos?. Es más, ¿cómo podrían ser implementados más eficientemente y cuál es el rendimiento máximo que podría obtenerse de ellos?. Éstas fueron las preguntas que motivaron las primeras etapas de la investigación en este área.

Nuevo enfoque Las respuestas a estas preguntas mostraron que el enfoque tradicional, según el cual los LFSRs sobre el cuerpo algebraico binario $GF(2)$, puede ser adecuado en las implementaciones hardware, pero no en software. Dado que el tamaño de

los registros de los ordenadores actuales es de, como mínimo, 8 bits en las máquinas menos potentes y de hasta 64 bits en las de mayor potencia, realizar implementaciones orientadas a bit resulta claramente ineficiente.

En este sentido, nos plateamos utilizar cuerpos finitos más apropiados para la arquitectura de los actuales microprocesadores y manipular registros de desplazamiento definidos sobre cuerpos algebraicos compuestos, en contraposición a los tradicionales $GF(2)$.

Por supuesto, se llevó a cabo un estudio del rendimiento obtenido por estos registros, que denominaremos *LFSR extendidos*, en comparación con los tradicionales. Los experimentos mostraron los claros beneficios de este enfoque, proporcionando datos empíricos que avalan los resultados y demostrando que éste método resulta mucho más eficiente que los LFSR tradicionales.

Asimismo, con el fin de optimizar al máximo su rendimiento, y hacerlos eficientes también en plataformas como las redes de sensores, se ha realizado un análisis exhaustivo de los diferentes métodos para la implementación software de estos registros, comparándolos entre sí, enumerando las ventajas e inconvenientes de cada uno y proporcionando un veredicto final en base a los datos empíricos obtenidos.

Resultados Los resultados de los experimentos realizados fueron concluyentes. Los datos muestran que técnicas propuestas permiten obtener incrementos en el rendimiento final realmente significativos, con factores de mejora de hasta 10.15, que supone un aumento en la eficiencia de más de un 900 %.

Los estudios han mostrado también otros hechos relevantes y llamativos. El primero es que, tal y como se esperaba, el rendimiento obtenido por un LFSR extendido aumenta al mismo tiempo que lo hace el tamaño del cuerpo base. Curiosamente, sin embargo, y debido a diversos factores, esta mejora no es completamente lineal, sino que decrece ligeramente en $GF(2^{32})$ respecto a $GF(2^{16})$.

Las razones que explican este fenómeno están relacionadas con el equilibrio entre el beneficio proporcionado por la mayor salida por ciclo del cuerpo base y el coste computacional de sus operaciones, y nos llevan directamente a una de las conclusiones más importantes de esta parte de la tesis.

En pocas palabras, ésta consiste en que el coste de las operaciones internas del cuerpo base llegan a hacerse tan costosas conforme crece su dimensión, que la ventaja de obtener en cada ciclo de funcionamiento del LFSR una salida de mayor tamaño, de 8, 16 y 32 bits respectivamente, termina perdiéndose de forma gradual. Según los datos obtenidos, *el límite parece situarse en* $GF(2^{16})$, *que constituye el cuerpo que obtuvo mejor rendimiento en los experimentos realizados*.

Por otro lado, se ha mostrado cómo otras técnicas de mejora, como búfers circulares o desdoblamiento de bucles se adaptan perfectamente a la implementación de LFSRs, mejorando muy significativamente la tasa de generación de salida de los mismos. Los resultados concluyen, también, que estas técnicas resultan más eficaces en cuerpos de pequeño tamaño, como $GF(2^8)$, $GF(2^{16})$ e, incluso, $GF(2)$, y que van perdiendo eficacia para cuerpos de mayor tamaño.

En resumen, los experimentos realizados concluyen, por tanto, que *la forma más eficiente de implementar registros de desplazamiento en software es trabajar en el cuerpo extendido* $GF(2^{16})$, *combinando esta técnica con la de desdoblamiento de bucles*.

VII.1.2. Cifrado en flujo: LFSRe

Como continuación natural del trabajo anterior, se pensó en la posibilidad de aplicar lo aprendido hasta el momento sobre registros extendidos como base de un cifrador

en flujo. De nuevo guiados por las limitaciones computacionales, se decidió utilizar los principios del generador shrinking, que se caracteriza por su sencillez y excelente resistencia al criptoanálisis, en el diseño de un nuevo cifrador que también hiciera uso de los registros anteriores.

Así surgió la idea del LFSRe, especialmente orientado a su uso en dispositivos con poca capacidad de cómputo, como los que constituyen las redes de sensores y redes ad-hoc en general.

La primera de las etapas de su diseño consistió en la definición de nuevos *criterios de decimación*, evaluando sus respectivos rendimientos y los pros y contras de cada uno de ellos. En este sentido, se ha profundizado asimismo en diversos aspectos asociados a la característica salida irregular de éste y otros generadores, que no han sido convenientemente estudiados hasta la fecha.

Uno de los resultados más importantes es que hemos demostrado, haciendo uso de cadenas de Markov para la modelización del nivel de ocupación del búfer interno, que atendiendo al planteamiento original del generador shrinking, no es posible evitar simultáneamente el agotamiento y el desbordamiento de dicho búfer. Sin embargo, sí se han aportado dos formas nuevas de gestionar esta salida irregular, que hemos denominado *pausa* y *gestión dinámica del ratio*, que evitan completamente los problemas anteriores.

Por supuesto, se ha realizado asimismo un análisis del rendimiento que es capaz de proporcionar LFSRe en diferentes escenarios, desde flujos continuos de datos a paquetes de diversa longitud, obteniendo excelentes resultados, comparables en todos los casos a los de conocidas referencias en el campo, como RC4 o AES-CTR. Para ello se ha hecho uso de la infraestructura proporcionada por el proyecto eSTREAM, de donde se han elegido también diversos cifradores en flujo de vanguardia, para ser usados en la comparativa.

VII.1.3. Autenticación y *Criptonite*

La segunda de las grandes áreas de investigación de esta tesis la constituye el aseguramiento de la autenticidad de la información intercambiada en este tipo de redes.

Esquemas existentes El estado del arte en el área en el momento de iniciar esta etapa era variado. Por un lado, se encuentran aquellos esquemas diseñados para redes de uso personal (PAN), que, en muchas casos, no contemplan en absoluto opciones de autenticación o de privacidad, y que basan su seguridad en el extremadamente corto rango de comunicaciones y la necesidad de línea de visión directa puede proporcionar una suerte de autenticación "física".

Otra gran categoría incluye los esquemas simétricos, donde se enmarca también la solución que aportamos en esta tesis. En este caso, las propuestas existentes en la literatura adolecen de ciertos inconvenientes. Uno de los más comunes que muchos de los esquemas hacen uso de un canal seguro pre-existente para la transmisión de dicho secreto, lo que en ciertos escenarios, como el que nos ocupa, no resulta posible. Es el caso del estándar IEEE 802.11, utilizado en muchas de las redes inalámbrica de ámbito local (WLANs).

Por otro lado, esquemas como el de Bluetooth, dejando de lado que la seguridad de su algoritmo de cifrado, E_0, se considera insuficiente [2], presentan el inconveniente que no se adaptan convenientemente a un alto número de nodos, debido a la necesidad de que el usuario teclee manualmente el secreto en cada dispositivo participante en la comunicación. El modelo del "patito resucitador", esquema serio de cómico nombre propuesto por Stajano y Anderson [3], resuelve de forma imaginativa el problema del

intercambio de claves, pero adolece de los mismos problemas que Bluetooth, pues aún necesita de un contacto físico entre dispositivos. Este requisito invalida ambas soluciones para uso en redes de sensores.

Finalmente, como ejemplo paradigmático de la última gran categoría de esquemas de autenticación podemos citar a los *modelos de pre-distribución de claves*. Estos esquemas proponen diferentes formas de realizar una distribución previa de las claves simétricas, con el fin de no utilizar una única clave en toda la red. Este hecho es especialmente importante en las redes de sensores, que disponen de una seguridad física débil, y cuyo material de claves puede ser fácilmente recuperado. Sin embargo, el inconveniente de estos esquemas es que suelen requerir una gran cantidad de memoria en cada nodo que, como hemos visto, constituye un recurso realmente preciado y escaso.

Con el objetivo de superar gran parte de los problemas anteriores y aportar nuevas ventajas, el análisis realizado en la etapa anterior concluye finalmente con la contribución de un nuevo protocolo de autenticación y gestión de claves, que destaca por sus escasos requisitos computacionales, pues únicamente hace uso de primitivas criptográficas sencillas y simétricas.

Otro de sus puntos fuertes se encuentra en que utiliza una clave maestra, que es posteriormente eliminada de la memoria del nodo, para derivar de ella una serie de claves simétricas. Este enfoque trata de aprovechar los beneficios del esquema de claves simétricas, es decir, una mayor resistencia al compromiso, sin la necesidad de una predistribución o almacenamiento de claves masivos.

Como consecuencia, nuestro esquema presenta una *buena escalabilidad*, una *muy buena resistencia a las consecuencias del compromiso físico de los nodos* y una gran *eficiencia energética*.

Como se ha comentado a lo largo de toda la tesis, las restricciones energéticas son uno de los grandes retos a superar en este tipo de redes, y cualquier esquema o protocolo de seguridad debe destacar en este apartado si pretende ser viable en la práctica. En este sentido, nuestro esquema presenta mejoras importantes, de alrededor del 98 % y 67 %, respectivamente, sobre dos conocidas propuestas existentes en la literatura como SPINS y BROSK.

Este protocolo, al igual que el algoritmo LFSRe, ha sido también *implementado en dispositivos hardware reales*, concretamente de la plataforma Arduino. Como consecuencia, se ha desarrollado asimismo la primera librería criptográfica existente para esta plataforma hasta la fecha, que hemos denominado *Cryptonite*.

En lo relativo al rendimiento del protocolo, los resultados muestran que éste presenta unas prestaciones excelentes. Por ejemplo, una ejecución completa del protocolo, incluyendo la autenticación mutua del nodo iniciador y del anfitrión, puede llevarse a cabo en menos de 40 ms.

VII.2. TRABAJO FUTURO

A lo largo del desarrollo del trabajo de esta tesis se han identificado una serie de líneas investigación relacionadas con los aspectos que se han tratado. Algunas de ellas se han incorporado al cuerpo principal de la tesis, ampliando los objetivos iniciales, pero otras han quedado finalmente al margen de los mismos. La continuación del trabajo en estas áreas estimula futuras tareas que complementarían las contribuciones realizadas en esta tesis.

VII.2.1. Localización

La localización de los nodos hace referencia al proceso según el cual éstos pueden estimar su posición física en el espacio por diversos métodos. Ésta es una cualidad muy interesante, pues abre todo un abanico de nuevas posibilidades.

En los últimos años, ésta se ha convertido en un área de intensa investigación en el ámbito académico e industrial. La solución inmediata y obvia consiste en dotar a cada sensor de un receptor GPS, de forma que éstos puedan calcular su posición con precisión. Ésta no es, sin embargo, una opción factible desde el punto de vista económico, pues a menudo las redes de sensores están formados por un gran número de nodos y el coste total sería prohibitivo. De hecho, es habitual que, a día de hoy, un receptor GPS para una mota tenga mayor coste que la propia mota [4].

Por estas razones, el proceso de localización en redes de sensores y ad-hoc se convierte en un desafío. Cuando a esto se añade la necesidad de que el procedimiento sea seguro, de forma que un nodo atacante no pueda inyectar información de localización falsa en la red, encontrar una solución se complica aún más.

Éste sería, pues, el doble objetivo de esta línea de investigación. Por un lado, estudiar los mecanismos de localización segura existentes, analizándolos y apuntando sus puntos fuertes y débiles. Y, por otro lado, tratar de definir un nuevo método, que resulte novedoso y trate de superar los problemas identificados en el paso anterior. Especialmente interesante sería el hecho de que el proceso pudiera funcionar, con cierta precisión, en interiores, sin necesidad de hardware adicional.

Por último, este trabajo permitiría dotar a Apolo y Dafne de la capacidad de mostrar la posición real de los nodos en el espacio, lo que convertiría a esta herramienta en el primer *panel de control* existente, de uso y licencia libre, de redes de sensores.

VII.2.2. Criptonite en otras plataformas

Otro de los aspectos en los que se pretende trabajar en el futuro es en la extensión de la librería Cryptonite a otras plataformas, tanto de 8 bits, como otras más potentes de 16. Una de las principales motivaciones para este desarrollo es la escasez de librerías criptográficas libremente disponibles para estos entornos.

Por otro lado, el proceso no debería resultar excesivamente complicado, puesto que toda la librería ha sido codificada exclusivamente en ANSI C, sin dependencias externas.

Sin duda, una de las primeras plataformas candidatas es MICAz, con su sistema operativo TinyOS, uno de los más utilizados en el ámbito académico.

VII.2.3. Ataques a la implementación en microcontroladores

Los ataques a la implementación, más conocidos como *side-channel attacks*, constituyen una técnica reciente, a menudo no del todo conocida por criptógrafos e inge-

nieros, y realmente peligrosa. Este tipo de ataques se basan en la explotación de ciertas características del hardware que ejecuta los algoritmos criptográficos o de seguridad.

En función del aspecto concreto que analice, estos ataques pueden dividirse en ataques de monitorización, ataques de análisis de potencia, de emanación electromagnética o de análisis de errores. Al margen quedan otro tipo de técnicas, como la provocación de fallos, que implican ataques activos y, a menudo, la manipulación física de los dispositivos.

En este contexto, la línea de investigación detectada incluye estudiar este tipo de ataques y técnicas en las plataformas estudiadas en este tesis. Aunque la plataforma Arduino no ha sido diseñada, desde luego, para evitar estos ataques, lo cierto es que otros muchos microcontroladores de otros fabricantes, como Microchip, están empezando a ser utilizados en aplicaciones críticas, que manejan información confidencial o sensible, conforme la popularidad de este tipo de redes crece.

En este sentido, sería muy interesante conocer la resistencia de estos microcontroladores comunes a este tipo de ataques y proponer contramedidas que ayuden a mitigarlos en la medida de lo posible.

VII.2.4. Implementación hardware de LFSRe

Con las lecciones aprendidas en el estudio anterior, otra de las propuestas de trabajo futuro consiste en la implementación de LFSRe en un hardware específico de alto rendimiento, como una FPGA.

En este desarrollo, por supuesto, al margen de tratar de maximizar el rendimiento del mismo, deberían probarse todos los ataques perfeccionados en la etapa anterior y, si resulta necesario, aplicar las contramedidas que hayan sido reconocidas como más efectivas.

Otros de los objetivos que se pretende alcanzar sería comparar esta implementación con la realizada en esta tesis en un microcontrolador barato y fácilmente disponible. Se pretende, de esta forma, obtener el ratio coste/rendimiento de ambas plataformas hardware.

Por un lado, una FPGA proporciona un mayor rendimiento, a costa de un proceso de diseño y desarrollo más laboriosos y costosos económicamente, al tratarse de un hardware dedicado. Por otro lado, un microcontrolador permite un desarrollo mucho más rápido y sencillo, pero cuenta con evidentes limitaciones de velocidad de proceso, memoria y otras implicaciones, como los anteriores *side-channel attacks*, para los que no cuenta, en principio, con ningún tipo de protección.

El objetivo es, pues, dilucidar para qué aplicaciones será necesario el uso de FPGAs, a pesar de su mayor coste, y en cuáles es posible utilizar implementaciones hardware mucho más baratas y fáciles de desarrollar.

BIBLIOGRAFÍA DEL CAPÍTULO I

[1] H. Chaouchi, ed., *The Internet of Things: Connecting Objects*. Hoboken, NJ, London: Wiley-ISTE, 2010.

[2] L. S. Barney, *Developing hybrid applications for the iPhone: using HTML, CSS, and JavaScript to build dynamic apps for the iPhone*. Developer's library, Boston, MA: Pearson Education, 2008.

[3] Ji-Sun Kim, Denis Gracanin, Kresimir Matkovic, and Francis K. H. Quek, "iPhone/iPod Touch as Input Devices for Navigation in Immersive Virtual Environments.," in *VR*, pp. 261–262, IEEE, 2009.

[4] M. Rahnema, "Overview of the GSM system and protocol architecture," *Communications Magazine, IEEE*, vol. 31, no. 4, pp. 92–100, 1993.

[5] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "Wi-fi in ad hoc mode: a measurement study," *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pp. 145–154, 2004.

[6] Dong Zhou, Lifei Huang, and Ten H. Lai, "On the scalability of IEEE 802.11 ad-hoc-mode timing synchronization function," *Wirel. Netw.*, vol. 14, no. 4, pp. 479–499, 2008.

[7] Wen Hu, Peter Corke, Wen Chan Shih, and Leslie Overs, "secFleck: A Public Key Technology Platform for Wireless Sensor Networks," in *EWSN*, pp. 296–311, 2009.

[8] Md. Mokammel Haque, Al-Sakib Khan Pathan, and Choong Seon Hong, "Securing U-Healthcare Sensor Networks using Public Key Based Scheme," *CoRR*, vol. abs/0804.1184, 2008.

[9] C. Krauí, M. Schneider, and C. Eckert, "On handling insider attacks in wireless sensor networks," *Inf. Secur. Tech. Rep.*, vol. 13, no. 3, pp. 165–172, 2008.

[10] W. C. Shih, W. Hu, P. Corke, and L. Overs, "A public key technology platform for wireless sensor networks," in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, (New York, NY, USA), pp. 447–448, ACM, 2008.

[11] K. Hoeper and G. Gong, *Models of Authentications in Ad Hoc Networks and Their Related Network Properties*, ch. 6. Springer, 2006.

[12] W. Peterson and E. Weldon, *Error Correcting Codes*. MIT press, 1972.

[13] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *Proceedings of Symposium on Security and Privacy, 2003*, pp. 197–213, 2003.

Bibliografía del Capítulo II

[1] W. Peterson and E. Weldon, *Error Correcting Codes*. MIT press, 1972.

[2] Rudolf Lidl and Harald Niederreiter, *Finite fields*, vol. 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge: Cambridge University Press, 2nd ed., 1997.

[3] R. J. McEliece, *Finite field for scientists and engineers*. Norwell, MA, USA: Kluwer Academic Publishers, 1987.

[4] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," 1999.

[5] Daqing Wan, "Generators And Irreducible Polynomials Over Finite Fields," *Mathematics of Computation*, vol. 66, pp. 1195–1212, 1997.

[6] Keith Conrad, *Polynomials over a finite field*. Deparment of Mathematics, University of Connecticut, 2007.

[7] R. Lidl and H. Niederreiter, *Finite Fields - Encyclopedia of Mathematics and its Applications*, vol. 20. Addison-Wesley, 1983.

[8] Christof Paar, *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*. PhD thesis, Institute for Experimental Mathematics, University of Essen, Germany, 1994.

[9] J. Alanen and D. Knuth, "Tables of finite fields," *Sankhya, the Indian Journal of Statistics*, vol. Series A, 26(part 4):December, 1964, 1964.

[10] D. Knuth, *The Art of Computer Programming. Volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, Massachusetts, 1981.

[11] B. Ryu, T. Andersen, T. Elbatt, and Y. Zhang, "Multitier mobile ad hoc networks: architecture, protocols, and perfomance," vol. 2, pp. 1280–1285 Vol.2, 2003.

[12] Kwan-Wu Chin, John Judge, Aidan Williams, and Roger Kermode, "Implementation experience with MANET routing protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 5, pp. 49–59, 2002.

[13] Elgan Huang, Wenjun Hu, Jon Crowcroft, and Ian Wassell, "Towards commercial mobile ad hoc network applications: a radio dispatch system," in *In Proceedings of the sixth ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc*, pp. 355–365, ACM Press, 2005.

[14] M. Sulleman Memon, Manzoor Hashmani, and Niaz A. Memon, "A review for uniqueness and variations in throughput of MANET routing protocol due to performance metrics, characteristics and simulation environments," *WTOC*, vol. 7, no. 5, pp. 459–468, 2008.

[15] Frank Stajano, *Security for Ubiquitous Computing*. John Wiley and Sons, Feb. 2002.

[16] A. Juels, "Minimalist Cryptography for Low-Cost RFID Tags (Extended Abstract)," *Security in Communication Networks*, pp. 149–164, 2005.

[17] M. J. Robshaw, "An overview of RFID tags and new cryptographic developments," *Information Security Technical Report*, vol. 11, no. 2, pp. 82–88, 2006.

[18] J. Biskup and J. Lopez, eds., *Computer Security - ESORICS 2007, 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24-26, 2007, Proceedings*, vol. 4734 of *Lecture Notes in Computer Science*, Springer, 2007.

[19] A. Juels, R. L. Rivest, and M. Szydlo, "The blocker tag: selective blocking of RFID tags for consumer privacy," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 103–111, ACM, 2003.

[20] Raghu Das and Dr Peter Harrop, "Complete RFID Analysis and Forecasts 2008-2017," tech. rep., IDTechEx, 2009.

[21] A. Mahajan, N. Potnis, K. Gopalan, and A. Wang, "Urban Mobility Models for VANETs," in *Proc. of 2nd Workshop on Next Generation Wireless Networks*, 2006.

[22] M. Fiore, J. Harri, F. Filali, and C. Bonnet, "Vehicular Mobility Simulation for VANETs," in *Simulation Symposium, 2007. ANSS '07. 40th Annual*, pp. 301–309, 2007.

[23] Pandurang Kamat, Arati Baliga, and Wade Trappe, "An identity-based security framework For VANETs," in *VANET '06: Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, (New York, NY, USA), pp. 94–95, ACM, 2006.

[24] Charles E. Perkins, Elizabeth M. Belding Royer, and Samir R. Das, "Ad Hoc On-Demand Distance Vector Routing Protocol," internet-draft, IETF MANET Working Group, February 2003 2003. Expiration: August 17, 2003.

[25] Thomas Clausen and Philippe Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Internet Engineering Task Force, October 2003.

[26] A. A. Pirzada, R. Wishart, and M. Portmann, "Multi-Linked AODV Routing Protocol for Wireless Mesh Networks," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 4925–4930, 2007.

[27] T. Clausen, P. Jacquet, and L. Viennot, "Optimizing Route Length in Reactive Protocols for Ad Hoc Networks," in *Med-hoc-Net*, september 2002. Sardaigne.

[28] T. Clausen, P. Jacquet, and L. Viennot, "Comparative Study of Routing Protocols for Mobile Ad Hoc Networks," in *Med-hoc-Net*, september 2002. Sardaigne.

[29] C. Adjih, P. Jacquet, and L. Viennot, "Computing connected dominated sets with multipoint relays," *Ad Hoc and Sensor Wireless Networks*, vol. 1, no. 1-2, pp. 27–39, 2005.

[30] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot, "Performance Analysis of OLSR Multipoint Relay Flooding in Two Ad Hoc Wireless Network Models," in *The second IFIP-TC6 NETWORKING Conference*, may 2002. Pise.

[31] T. Plesse, C. Adjih, P. Minet, A. Laouiti, A. Plakoo, M. Badel, P. Muhlethaler, P. Jacquet, and J. Lecomte, "OLSR performance measurement in a military mobile ad hoc network," Tech. Rep. 5, September 2005.

[32] Carlos T. Calafate, M. P. Malumbres, and P. Manzoni, "Performance of H.264 Compressed Video Streams over 802.11b Based MANETs," *Distributed Computing Systems Workshops, International Conference on*, vol. 6, pp. 776–781, 2004.

[33] K. Hoeper and G. Gong, *Models of Authentications in Ad Hoc Networks and Their Related Network Properties*, ch. 6. Springer, 2006.

[34] M. M. McMahon, D. M. Needham, and J. B. Datko, "Performance of a Jini-based Ad Hoc Network Authentication Scheme," in *ISCA PDCS*, pp. 442–447, 2003.

[35] Sathishkumar Alampalayam and Anup Kumar, "An Adaptive and Predictive Security Model for Mobile Ad hoc Networks," *Wirel. Pers. Commun.*, vol. 29, no. 3-4, pp. 263–281, 2004.

[36] Lidong Zhou and Zygmunt J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, pp. 24–30, 1999.

[37] A. Fourati, K. A. Agha, and H. K. Ayed, "Secure and fair auctions over ad hoc networks," *International Journal of Electronic Business*, pp. 276–293, July 2007.

[38] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, pp. 521–534, September 2002.

[39] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 162–175, ACM, 2004.

[40] IEEE, "802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks," tech. rep., IEEE standard, 2006.

[41] Y. W. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 1, pp. 65–93, 2006.

[42] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, (Washington, DC, USA), pp. 245–256, IEEE Computer Society, 2008.

[43] Frazer Bennett, David Clarke, and Joseph B. Evans, "Piconet: Embedded mobile networking," *IEEE Personal Communications*, vol. 4, pp. 8–15, 1997.

[44] J. Lansford, "HomeRF. SWAP: a wireless voice and data system for the home," *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 6, pp. 3718–3721, 2000.

[45] I. 802.11, "Standard Specifications for Wireless Local Area Networks." http://standards.ieee.org/wireless.

[46] Scott R. Fluhrer and Stefan Lucks, "Analysis of the E0 Encryption System," in *SAC '01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, (London, UK), pp. 38–48, Springer-Verlag, 2001.

[47] Markus Jakobsson and Susanne Wetzel, "Security Weaknesses in Bluetooth," in *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, (London, UK), pp. 176–191, Springer-Verlag, 2001.

[48] R. Stajano, F.; Anderson, "The Resurrecting Duckling: security issues for ubiquitous computing," *Computer*, vol. 35, pp. 22–26, April 2002.

[49] Laurent Eschenauer and Virgil D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 41–47, ACM, 2002.

[50] D. Liu and P. Ning, "Location-Based Pairwise Key Establishments for Static Sensor Networks," 2003.

[51] Cungang Yang, Celia Li, and Jie Xiao, "Location-based design for secure and efficient wireless sensor networks," *Comput. Netw.*, vol. 52, no. 16, pp. 3119–3129, 2008.

[52] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*, (New York, NY, USA), pp. 47–53, Springer-Verlag New York, Inc., 1985.

[53] A. Khalili, J. Katz, and W. A. Arbaugh, "Toward Secure Key Distribution in Truly Ad-Hoc Networks," in *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, (Washington, DC, USA), p. 342, IEEE Computer Society, 2003.

[54] J.-P. Hubaux, L. Buttyán, and S. Capkun, "The quest for security in mobile ad hoc networks," in *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 146–155, ACM, 2001.

BIBLIOGRAFÍA DEL CAPÍTULO III

[1] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[2] Don Coppersmith, Hugo Krawczyk, and Yishay Mansour, "The Shrinking Generator," in *CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 22–39, Springer-Verlag, 1994.

[3] Solomon W. Golomb, *Shift Register Sequences*. Laguna Hills, CA, USA: Aegean Park Press, 1981.

[4] W. Peterson and E. Weldon, *Error Correcting Codes*. MIT press, 1972.

[5] Roberto Maria Avanzi and Nicolas Theriault, "Effects of Optimizations for Software Implementations of Small Binary Field Arithmetic," in *WAIFI*, pp. 69–84, 2007.

[6] S. Chowdhury and S. Maitra, "Efficient Software Implementation of LFSR and Boolean Function and Its Application in Nonlinear Combiner Model," *Applied Cryptography and Network Security - ACNS 2003*, vol. Lecture Notes in Computer Science 2846, pp. 387–402, 2003.

[7] S. Chowdhury and S. Maitra, "Efficient Software Implementation of Linear Feedback Shift Registers," *International Conference in Cryptology in India - INDOCRYPT '01*, vol. Lecture Notes in Computer Science 2247, pp. 397–407, 2001.

[8] Tore Herlestam, "On Functions of Linear Shift Register Sequences," *Advances in Cryptology - EUROCRYPT '85*, vol. Volume 219/1986, pp. 119 – 129, 1985.

[9] P. Deepthi, Deepa Sara John, and P. Sathidevi, "Design and analysis of a highly secure stream cipher based on linear feedback shift register," *Comput. Electr. Eng.*, vol. 35, no. 2, pp. 235 – 243, 2009.

[10] Jovan Dj. Golic, "On the Security of Nonlinear Filter Generators," in *Proceedings of the Third International Workshop on Fast Software Encryption* (L. N. I. C. Science, ed.), vol. 1039, 1993.

[11] Philip Hawkes and Gregory G. Rose, "Guess-and-Determine Attacks on SNOW," in *SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, (London, UK), pp. 37–46, Springer-Verlag, 2003.

[12] James S. Plank, *Fast Galois Field Arithmetic Library in C/C++*. Department of Computer Science, University of Tennessee, 2007.

[13] J. S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems," Tech. Rep. CS-96-332, University of Tennessee, July 1996.

[14] J. S. Plank, "Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Storage Applications," Tech. Rep. CS-05-569, University of Tennessee, December 2005.

[15] Tjerk Bijlsma, Marco Bekooij, Pierre Jansen, and Gerard Smit, "Communication between nested loop programs via circular buffers in an embedded multiprocessor system," in *SCOPES '08: Proceedings of the 11th international workshop on Software & compilers for embedded systems*, pp. 33–42, 2008.

[16] Yuichi Tsujita, "Effective Remote MPI-I/O on a Parallel Virtual File System using a Circular Buffer: A Case Study for Optimization," in *IASTED PDCS*, pp. 490–495, 2005.

[17] M. Robshaw, "The eSTREAM Project," pp. 1–6, 2008.

[18] Cetin Kaya Koc, "Analysis of sliding window techniques for exponentiation," *Computers and Mathematics with Applications*, vol. 30, pp. 17–24, 1995.

[19] Lavy Libman and Ariel Orda, "Optimal sliding-window strategies in networks with long round-trip delays," *Comput. Netw.*, vol. 46, no. 2, pp. 219–235, 2004.

[20] J. C. Huang and T. Leng, "Generalized Loop-Unrolling: A Method for Program Speedup," *Application-Specific Software Engineering and Technology, IEEE Workshop on*, vol. 0, p. 244, 1999.

[21] O. Dragomir, T. P. Stefanov, and K. Bertels, "Loop Unrolling and Shifting for Reconfigurable Architectures," in *Proceedings of the 18th International Conference on Field Programmable Logic and Applications (FPL08)*, September 2008.

[22] Afshin Ganjoo and Nian-Feng Tzeng, "Influence of High-Level Program Structures on Branch Prediction Accuracy," *EUROMICRO Conference*, vol. 1, p. 1316, 2000.

[23] J. W. Davidson and S. Jinturkar, "An Aggressive Approach to Loop Unrolling," tech. rep., Charlottesville, VA, USA, 2001.

[24] J. W. Davidson and S. Jinturkar, "Improving instruction-level parallelism by loop unrolling and dynamic memory disambiguation," in *MICRO 28: Proceedings of the 28th annual international symposium on Microarchitecture*, (Los Alamitos, CA, USA), pp. 125–132, IEEE Computer Society Press, 1995.

[25] S. Eranian, "What can performance counters do for memory subsystem analysis?," in *MSPC '08: Proceedings of the 2008 ACM SIGPLAN workshop on Memory systems performance and correctness*, (New York, NY, USA), pp. 26–30, ACM, 2008.

[26] Intel Corporation, "Intel Vtune Performance Analyzer." http://www.intel.com/software/products/vtune/239144.htm, 2009.

[27] S. Eranian, "The Perfmon2 project." http://perfmon2.sourceforge.net, 2009.

[28] A. Phansalkar, A. Joshi, L. Eeckhout, and L. K. John, "Measuring Program Similarity: Experiments with SPEC CPU Benchmark Suites," in *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*, pp. 10–20, 2005.

[29] Helger Lipmaa, Phillip Rogaway, and David Wagner, "Comments to NIST Concerning AES-modes of Operations: CTR-mode Encryption," in *Symmetric Key Block Cipher Modes of Operation Workshop*, (Baltimore, Maryland, US), 20 October 2000.

[30] W. Shi, H.-H. S. Lee, M. Ghosh, C. Lu, and A. Boldyreva, "High Efficiency Counter Mode Security Architecture via Prediction and Precomputation," *SIGARCH Comput. Archit. News*, vol. 33, no. 2, pp. 14–24, 2005.

[31] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," *Lecture Notes in Computer Science*, vol. 2259, pp. 1–??, 2001.

[32] Juan Soto, "Statistical Testing of Random Number Generators," in *In: Proceedings of the 22nd National Information Systems Security Conference*, 1999.

[33] Ueli M. Maurer, "A universal statistical test for random bit generators," *Journal of cryptology*, vol. 5, pp. 89–105, 1992.

[34] David Naccache, "An Accurate Evaluation of Maurer's Universal Test," in *Proceedings of SAC '98 (Lecture Notes in Computer Science)*, pp. 57–71, Springer-Verlag, 1998.

[35] J.-S. Coron and D. Naccache, "An Accurate Evaluation of Maurer's Universal Test," in *SAC '98: Proceedings of the Selected Areas in Cryptography*, (London, UK), pp. 57–71, Springer-Verlag, 1999.

[36] G. Marsaglia, "Diehard battery of tests of randomness." Florida State University, 1995.

[37] George Marsaglia and Wai Wan Tsang, "Some Difficult-to-pass Tests of Randomness," *Journal of Statistical Software*, vol. 7, no. i03, 2007.

[38] Robert G. Brown, "Dieharder: A Random Number Test Suite." Duke University Physics Department, http://www.phy.duke.edu/ rgb/General/dieharder.php, 2004.

[39] J. Andrew Rukhin, Juan Soto, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST, special publication 800-22, revision 1 ed., Agosto 2008.

BIBLIOGRAFÍA DEL CAPÍTULO IV

[1] T. Clausen, N. Larsen, T. Olesen, and L. Viennot, "Investigating Data Broadcast Performance in Mobile Ad Hoc Networks," in *The 5th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, october 2002. Honolulu.

[2] V. Frias, G. Delgado, and M. Igartua, "Multipath Routing for video-streaming services over IEEE 802.1le Ad hoc Networks," *International Conference on Software in Telecommunications and Computer Networks*, vol. 0, pp. 132–136, 2006.

[3] Helger Lipmaa, Phillip Rogaway, and David Wagner, "Comments to NIST Concerning AES-modes of Operations: CTR-mode Encryption," in *Symmetric Key Block Cipher Modes of Operation Workshop*, (Baltimore, Maryland, US), 20 October 2000.

[4] Niels Ferguson, Doug Whiting, Bruce Schneier, John Kelsey, Stefan Lucks, and Tadayoshi Kohno, "Helix - Fast Encryption and Authentication in a Single Cryptographic Primitive," in *Proc. Fast Software Encryption 2003, volume 2887 of LNCS*, pp. 330–346, Springer-Verlag, 2003.

[5] William E. Burr, "Selecting the Advanced Encryption Standard," *IEEE Security and Privacy*, vol. 1, no. 2, pp. 43–52, 2003.

[6] N. Sklavos and O. Koufopavlou, "Architectures and VLSI Implementations of the AES-Proposal Rijndael," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1454–1459, 2002.

[7] M. Robshaw, "The eSTREAM Project," pp. 1–6, 2008.

[8] C. Cannière, "eSTREAM Software Performance," pp. 119–139, 2008.

[9] I. Akyildiz, T. Melodia, and K. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, pp. 921–960, March 2007.

[10] Don Coppersmith, Hugo Krawczyk, and Yishay Mansour, "The Shrinking Generator," in *CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 22–39, Springer-Verlag, 1994.

[11] Solomon W. Golomb, *Shift Register Sequences*. Laguna Hills, CA, USA: Aegean Park Press, 1981.

[12] Pino Caballero Gil, Amparo Fúster Sabater, and Maria Eugenia Pazo Robles, "New Attack Strategy for the Shrinking Generator," in *WOSIS*, pp. 59–67, 2008.

[13] L. Simpson, J. D. Golic, and E. Dawson, "A Probabilistic Correlation Attack on the Shrinking Generator," in *ACISP '98: Proceedings of the Third Australasian Conference on Information Security and Privacy*, (London, UK), pp. 147–158, Springer-Verlag, 1998.

[14] Eli Biham and Orr Dunkelman, "Cryptanalysis of the A5/1 GSM Stream Cipher," in *INDOCRYPT*, pp. 43–51, 2000.

[15] T. Gendrullis, M. Novotný, and A. Rupp, "A Real-World Attack Breaking A5/1 within Hours," in *CHES '08: Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, (Berlin, Heidelberg), pp. 266–282, Springer-Verlag, 2008.

[16] Gregory G. Rose and Philip Hawkes, "Turing, a Fast Stream Cipher," in *In T. Johansson, Proceedings of Fast Software Encryption FSE2003, LNCS 2887, Springer-Verlag*, pp. 307–324, Springer-Verlag, 2003.

[17] Ilan Kessler and H. Krawczyk, "Minimum buffer length and clock rate for the shrinking generator cryptosystem," tech. rep., IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 1995.

[18] R. Beek, C. Vaucher, D. Leenaerts, E. Klumperink, and B. Nauta, "A Low-jitter 2.5-to-10 GHz Clock Multiplier Unit in CMOS," in *ProRISC 2003, 14th Workshop on Circuits, Systems and Signal Processing*, (Utrecht), pp. 173–176, STW, November 2003.

[19] S. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. New York, NY, USA: Cambridge University Press, 2009.

[20] C. M. Grinstead and J. L. Snell, *Introduction to Probability*. AMS Bookstore, second ed., 1997.

[21] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. New York, NY, USA: Cambridge University Press, 1986.

[22] Hongjun Wu and Bart Preneel, "Cryptanalysis of the Stream Cipher DECIM," in *FSE* (M. J. B. Robshaw, ed.), vol. 4047 of *Lecture Notes in Computer Science*, pp. 30–40, Springer, 2006.

[23] J. Bray and C. Sturman, *Bluetooth 1.1: Connect Without Cables, Second Edition*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[24] Markus Jakobsson and Susanne Wetzel, "Security Weaknesses in Bluetooth," in *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, (London, UK), pp. 176–191, Springer-Verlag, 2001.

[25] Hkan Englund, Martin Hell, and Thomas Johansson, "Two General Attacks on Pomaranch-Like Keystream Generators," in *FSE*, pp. 274–289, 2007.

[26] Hongjun Wu and Bart Preneel, "Resynchronization Attacks on WG and LEX," in *FSE*, pp. 422–432, 2006.

[27] Antoine Joux and Frédéric Muller, "A Chosen IV Attack Against Turing," in *Selected Areas in Cryptography*, pp. 194–207, 2003.

[28] Hongjun Wu and Bart Preneel, "Differential-Linear Attacks against the Stream Cipher Phelix," in *In Proc. FSE 2007, LNCS*, Springer, 2006.

[29] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," *Lecture Notes in Computer Science*, vol. 2259, pp. 1–??, 2001.

[30] E. Dawson and L. Nielsen, "Automated Cryptanalysis of XOR Plaintext Strings," *Cryptologia*, vol. 20, pp. 165–??, Apr. 1996.

[31] Phillip Rogaway, "Nonce-Based Symmetric Encryption," in *Proc. FSE 2004, volume 3017 of LNCS*, pp. 348–359, Springer, 2004.

[32] D. Bernstein, "Key recovery attacks on Phelix." http://www.ecrypt.eu.org/stream/phorum/read.php?1,883,921, 2006.

[33] Markku-juhani O. Saarinen, "Chosen-IV statistical attacks on estream stream ciphers," in *eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013*, pp. 5–19, 2006.

[34] E. Filiol, "A New Statistical Testing for Symmetric Ciphers and Hash Functions," in *ICICS '02: Proceedings of the 4th International Conference on Information and Communications Security*, (London, UK), pp. 342–353, Springer-Verlag, 2002.

Bibliografía del Capítulo V

[1] Frank Stajano, *Security for Ubiquitous Computing*. John Wiley and Sons, Feb. 2002.

[2] David D. Hwang, Bo-Cheng Charles Lai, and Ingrid Verbauwhede, "Energy-Memory-Security Tradeoffs in Distributed Sensor Networks," in *ADHOC-NOW 2004*, vol. 3158 of *LNCS*, 2004.

[3] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "Analyzing the energy consumption of security protocols," 2003.

[4] J. M. Sierra, J. C. H. Castro, N. Jayaram, and A. Ribagorda, "Low computational cost integrity for block ciphers," *Future Generation Comp. Syst*, vol. 20, no. 5, pp. 857–863, 2004.

[5] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *Proceedings of Symposium on Security and Privacy, 2003*, pp. 197–213, 2003.

[6] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 42–51, ACM, 2003.

[7] D. Liu and P. Ning, "Location-Based Pairwise Key Establishments for Static Sensor Networks," 2003.

[8] Laurent Eschenauer and Virgil D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 41–47, ACM, 2002.

[9] M. Bohge and W. Trappe, "An authentication framework for hierarchical ad hoc sensor networks," in *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security*, (New York, NY, USA), pp. 79–87, ACM, 2003.

[10] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, pp. 521–534, September 2002.

[11] Bo-Cheng Charles Lai, David D. Hwang, Sungha Pete Kim, and Ingrid Verbauwhede, "Reducing Radio Energy Consumption of Key Management Protocols for Wireless Sensor Networks," *Low Power Electronics and Design, International Symposium on*, vol. 0, pp. 351–356, 2004.

[12] Y. Wang, "Robust key establishment in sensor networks," *SIGMOD Rec.*, vol. 33, no. 1, pp. 14–19, 2004.

[13] B. Panja, S. Madria, and B. Bhargava, "Energy-Efficient Group Key Management Protocols for Hierarchical Sensor Networks," *Int. J. Distrib. Sen. Netw.*, vol. 3, no. 2, pp. 201–223, 2007.

[14] K. Kabri and D. Seret, "An Evaluation of the Cost and Energy Consumption of Security Protocols in WSNs," in *SENSORCOMM '09: Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications*, (Washington, DC, USA), pp. 49–54, IEEE Computer Society, 2009.

[15] NIST, *Recommendation for Key Management. Part 1: General Guideline. Special Publication 800-57*, p. 63. 2007.

[16] Ross Anderson, Haowen Chan, and Adrian Perrig, "Key Infection: Smart Trust for Smart Dust." Proc. of ICNP'04, October 2004.

[17] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[18] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, pp. 770–772, November 1981.

[19] Qijun Gu, Peng Liu, and Chao-Hsien Chu, "Analysis of area-congestion-based DDoS attacks in ad-hoc networks," *Ad Hoc Networks.*, vol. 5, no. 5, pp. 613–625, 2007.

[20] Peter Gutmann, "Data remanence in semiconductor devices," in *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, (Berkeley, CA, USA), pp. 4–4, USENIX Association, 2001.

[21] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, January 2001.

[22] Peter Gutmann, *Secure deletion of data from magnetic and solid-state memory*, pp. 8–8. Berkeley, CA, USA: USENIX Association, 1996.

[23] G. Bravos and A. G. Kanatas, "Energy efficiency comparison of MIMO-based and multihop sensor networks," *EURASIP J. Wirel. Commun. Netw.*, vol. 2008, pp. 1–13, 2008.

[24] D. W. Carman, P. S. Kruus, and B. J. Matt, "CONSTRAINTS AND APPROACHES FOR DISTRIBUTED SENSOR NETWORK SECURITY," Tech. Rep. 010, NAI Labs, The Security Research Division Network Associates, Inc., September 2000.

[25] M. Mallinson, S. Hussain, and J. H. Park, "Investigating Wireless Sensor Network Lifetime Using a Realistic Radio Communication Model," in *MUE '08: Proceedings of the 2008 International Conference on Multimedia and Ubiquitous Engineering*, (Washington, DC, USA), pp. 433–437, IEEE Computer Society, 2008.

[26] G. Anastasi, M. Conti, M. Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, pp. 537–568, May 2009.

BIBLIOGRAFÍA DEL CAPÍTULO VI

[1] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET simulation studies: The Incredibles," *ACM's Mobile Computing and Communications Review*, vol. 9, no. 4, pp. 50–61, 2005.

[2] D. Cavin, Y. Sasson, and A. Schiper, "On the Accuracy of MANET Simulators," in *Proceedings of the Workshop on Principles of Mobile Computing (POMC'02)*, pp. 38–43, 2002.

[3] M. Banzi, *Getting Started with Arduino*. Sebastopol, CA: Make Books - Imprint of: O'Reilly Media, 2008.

[4] J. D. Brock, R. F. Bruce, and S. L. Reiser, "Using Arduino for introductory programming courses," *J. Comput. Small Coll.*, vol. 25, no. 2, pp. 129–130, 2009.

[5] David J. Wheeler and Roger M. Needham, "TEA, a Tiny Encryption Algorithm," in *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, pp. 363–366, 1994.

[6] S. J. Shepherd, "The Tiny Encryption Algorithm," *Cryptologia*, vol. 31, no. 3, pp. 233–245, 2007.

[7] J. Kelsey, B. Schneier, and D. Wagner, "Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA," in *ICICS '97: Proceedings of the First International Conference on Information and Communication Security*, (London, UK), pp. 233–246, Springer-Verlag, 1997.

[8] B. Schneier, *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. New York, NY, USA: John Wiley & Sons, Inc., 1995.

[9] Roger M. Needham and David J. Wheeler, "TEA extensions," technical report, Computer Laboratory, University of Cambridge, 1997.

[10] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[11] Thomas Clausen and Philippe Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Internet Engineering Task Force, October 2003.

[12] A. Tonnesen, "Implementing and Extending the Optimized Link State Routing Protocol," Master's thesis, Department of Informatics, University of Oslo, 2004.

Bibliografía del Capítulo VII

[1] Periódico Público, "Los 'ciberpiratas' atacan el mundo real." http://www.publico.es/ciencias/296536/ciberpiratas/atacan/mundo/real, Febrero 2010.

[2] Markus Jakobsson and Susanne Wetzel, "Security Weaknesses in Bluetooth," in *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, (London, UK), pp. 176–191, Springer-Verlag, 2001.

[3] R. Stajano, F.; Anderson, "The Resurrecting Duckling: security issues for ubiquitous computing," *Computer*, vol. 35, pp. 22–26, April 2002.

[4] Libellium, 2009.