UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERO EN INFORMÁTICA

ANÁLISIS TEÓRICO Y EXPERIMENTAL SOBRE SEGURIDAD EN REDES WI-FI

Realizado por:

Fernando Pablo Romero Navarro

Dirigido por:

Antonio Jesús Nebro Urbaneja

José Francisco Chicano García

Departamento:

Lenguajes y Ciencias de la Computación

Málaga, Julio 2013

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERO EN INFORMÁTICA

Reunido el tribunal examinador en el día de la fecha, constituido por:

Presidente/a Dº/Dª._____

Secretario/a Dº/Dª._____

Vocal Dº/Dª._____

para juzgar el proyecto fin de carrera titulado:

Análisis teórico y experimental sobre seguridad en redes Wi-Fi

realizado por Dº/Dª.

Fernando Pablo Romero Navarro

tutorizado por Dº/Dª.

Antonio Jesús Nebro Urbaneja

José Francisco Chicano García

y, en su caso dirigido académicamente por:

Dº/Dª._____

Acordó por _____ otorgar la calificación

de _____

Y para que conste se extiende firmada por los comparecientes del tribunal, la presente diligencia:

Málaga a ____ de _____ del 2013

Índice de contenido

Índice de ilustraciones y tablas

Análisis teórico y experimental sobre seguridad en redes Wi-Fi

Fernando Pablo Romero Navarro

Dedicado a mi madre, que ha conseguido un logro mayor que el autor de este trabajo, como es criar y educar a dos hijos que han obtenido sendas titulaciones de ingeniería, para lo cual no ha dudado en trabajar en el servicio doméstico o en labores agrícolas.

Este documento ha sido editado íntegramente con las siguientes aplicaciones de fuentes abiertas o basadas en software libre (FOSS): OpenOffice.org 2.4, GIMP 2.6.1, Dia 0.96.1, BOUML 3.3.4, GNUPlot 4.0

Capítulo 1

1. Introducción

El avance y la expansión de las comunicaciones inalámbricas es un hecho que no ha pasado inadvertido para casi ningún individuo de un país desarrollado. Desde que a finales del siglo XIX se llevasen a cabo las primeras transmisiones mediante señales de radiofrecuencia, el progreso de estas tecnologías ha sido constante, así como la incorporación a nuestra vida cotidiana de nuevos servicios o facilidades proporcionados por las mismas. La radiodifusión, la televisión analógica o los sistemas de navegación por satélite son algunos ejemplos de estos servicios, que están disponibles en la mayoría de países e incluso en zonas no habitadas del planeta desde hace tiempo. Sin embargo, no fue hasta aproximadamente dos décadas atrás cuando comenzó la expansión comercial de las comunicaciones inalámbricas digitales, gracias a las cuales disponemos de nuevos servicios, o bien de servicios que han reemplazado a sus predecesores analógicos para mejorar su calidad o incluir nuevas características. La telefonía móvil posterior a la primera generación, los sistemas de posicionamiento mediante GPS, la radio digital y la televisión digital terrestre, o bien por satélite, pertenecen a este tipo de comunicaciones.

También se incluyen en este grupo las redes locales inalámbricas (a veces referidas con la abreviatura WLAN) cuyo principal representante actualmente son las redes Wi-Fi. Tanto es así, que hoy en día puede considerarse a la tecnología Wi-Fi como el estándar “de facto” para esta clase de redes. Algunos factores que han contribuido a la consolidación de esta tecnología son: su estandarización por parte del grupo de trabajo 802.11 del IEEE, el apoyo recibido por la asociación de empresas que constituyen la *Wi-Fi Alliance* y la labor desempeñada por estas empresas para fomentar la interoperabilidad de los dispositivos inalámbricos, sin restar mérito alguno al esfuerzo realizado por los fabricantes de tales dispositivos para que éstos reciban las certificaciones correspondientes de la *Wi-Fi Alliance* y cumplan de forma suficientemente estricta con las restantes especificaciones del estándar 802.11.

Tan intensa ha sido la actividad de certificación de la *Wi-Fi Alliance* y tantas marcas comerciales han participado en este proceso, que el término *Wi-Fi* se usa frecuentemente para designar a cualquier dispositivo inalámbrico que se ajuste a las especificaciones del estándar IEEE 802.11. No obstante, a pesar del fuerte crecimiento experimentado por las tecnologías inalámbricas en las últimas décadas, presentan también bastantes inconvenientes con respecto a la transmisión por medios guiados, sobre todo en lo que respecta a la atenuación de la señal, las interferencias con otras señales, el nivel de ruido presente en los canales de transmisión, la propagación multitrayectoria de la señal, la seguridad, etc. En cuanto a la seguridad de las redes inalámbricas, este asunto se aborda en ocasiones en el nivel de enlace del modelo OSI, ya sea por motivos de mayor eficiencia o de mayor protección de los protocolos con respecto a las soluciones proporcionadas por niveles superiores, o bien porque no resultan factibles o prácticas implementar medidas a nivel físico.

1.1 Objetivos

Uno de los objetivos de este proyecto ha sido proporcionar una visión general de los procedimientos de seguridad utilizados, en el nivel de enlace lógico de datos del modelo OSI, por las redes Wi-Fi. Estos procedimientos incluyen, aunque no están limitados, a los que se especifican en el estándar IEEE 802.11. Además, esta visión general se completará con una breve descripción de las debilidades más importantes que se conocen para los procedimientos de seguridad principales estudiados en los próximos capítulos, incluyendo también los ataques más destacados que aprovechan estas debilidades.

El otro objetivo fundamental ha sido el desarrollo de una aplicación que implemente una prueba de concepto de una debilidad descubierta en algún mecanismo de protección asociado a uno de los protocolos de seguridad estudiados. Para tal fin, se ha escogido el protocolo WEP y, en concreto, el algoritmo de cifrado en flujo empleado por este protocolo, debido a que está basado en una variante de cifrado aditivo denominada cifrado Vernam. De hecho, el diseño de este algoritmo es extremadamente sencillo y se utiliza, desde hace algunos años, en diferentes criptosistemas. Esto ha propiciado que se realicen numerosos estudios sobre la seguridad de dicho algoritmo y que se hayan detectado bastantes debilidades en el mismo.

En concreto, esta aplicación explota una debilidad que reside principalmente en el cifrado *Vernam,* la cual es susceptible de ser explotada siempre que sea posible obtener un prefijo de longitud suficiente del texto en claro. No obstante, es la falta de un mecanismo que asegure la autenticidad de los mensajes lo que posibilita el ataque implementado. Este ataque fue ideado por *William A. Arbaugh* y podría describirse como la combinación de dos ataques diferentes: uno de reutilización de la secuencia de cifrado y otro de fuerza bruta sobre el último byte de la secuencia de cifrado obtenida.

Para finalizar, también hay que destacar que la aplicación desarrollada no solo implementa el ataque ya mencionado, sino también una colección de rutinas mediante las cuales el usuario puede incrementar el tráfico de datos transmitido a través de una red Wi-Fi protegida con el protocolo WEP. No obstante, para esto es necesaria la presencia de alguna estación en la red Wi-Fi que transmita tramas de datos, al menos de forma eventual, y con la cual el atacante sea capaz de intercambiar tramas de esta clase sin que ocurran demasiados errores de transmisión. El propósito de incrementar el tráfico de datos en esa red Wi-Fi es aumentar las posibilidades de que un ataque estadístico contra la clave WEP, que puede ser llevado a cabo mediante otras aplicaciones, consiga descubrir dicha clave. Para realizar estas funciones, se han propuesto como objetivos de diseño que la interacción con el usuario se efectúe a través de una interfaz gráfica, de forma sencilla e intuitiva, y que la aplicación sea robusta, de modo que sea capaz de recuperarse de errores de transmisión cuando el usuario lo autorice.

1.2 Fases del proyecto y materiales utilizados

En esta sección se describen las principales fases en las que se ha acometido la realización de este proyecto. Esta sucesión de etapas coincide, en líneas generales, con las previstas en la planificación inicial, sin embargo, ahora se enumeran y se describen en concordancia con el tiempo y con el esfuerzo invertidos finalmente en las diferentes fases del proyecto:

- Recopilación de información, organización de la documentación recopilada y síntesis de los contenidos relevantes para este trabajo. Estas acciones se realizaron de forma reiterada hasta adquirir las nociones básicas sobre redes Wi-Fi y seguridad necesarias para afrontar la parte práctica del trabajo.

- Estudio de herramientas que permitan el desarrollo de la aplicación práctica propuesta, incluyendo lenguajes de programación, entornos de desarrollo para estos lenguajes, librerías para desarrollar aplicaciones de redes a bajo nivel y también las primitivas del sistema operativo utilizadas por tales librerías. Después de seleccionar las alternativas que ofrecían mejores facilidades, se implementaron algunos prototipos con el objetivo de poner a prueba las primitivas fundamentales, proporcionadas por estas alternativas (lenguajes, librerías, llamadas al sistema del S.O.), que resultaban imprescindibles para realizar el desarrollo. Estas pruebas se efectuaron con diferentes configuraciones de software del sistema y con diferentes dispositivos inalámbricos, para comprobar la independencia de los prototipos desarrollados con respecto al hardware y a S.S.O.O. particulares, y se evaluaron utilizando un analizador de protocolos.
- Después de que las pruebas apoyasen la viabilidad del desarrollo con las herramientas escogidas, se indagaron algunas debilidades del protocolo de seguridad WEP, así como algunos ataques que las explotaban. A la hora de seleccionar la debilidad de la cual se aprovecharía la aplicación, se valoró preferentemente la utilidad práctica de explotar dicha debilidad, así como el conocimiento de ataques cuya implementación entrañase la menor dificultad y cuya ejecución acarrease la menor complejidad temporal o bien la menor cantidad de recursos que fuese posible. Una vez seleccionada la debilidad, se implementó un prototipo del ataque y se hicieron varias pruebas hasta conseguir su correcto funcionamiento. A continuación, se elaboró un diseño inicial de la aplicación, que se fue refinando a medida que se implementaban nuevos módulos o subsistemas, como las funciones del servidor, las funciones para el procesamiento y el intercambio de mensajes entre el cliente y el servidor, la interfaz gráfica, etc.
- Tras completar el desarrollo de la aplicación, se retomó la parte teórica concerniente a los mecanismos de seguridad introducidos por los protocolos WEP, WPA y WPA2, para ampliar y detallar más exhaustivamente la información recopilada. Seguidamente, se establecieron los contenidos iniciales abarcados por la parte teórica y se comenzó a redactar la memoria. Al mismo tiempo que se iba redactando la memoria, se revisaron y se analizaron con una mayor profundidad los temas abordados, hasta ese momento, en la memoria.

Para la elaboración de la parte teórica del proyecto se ha recurrido a fuentes muy diversas, como a bibliografía genérica sobre redes de computadores, para adquirir las nociones básicas sobre tecnologías de comunicación inalámbrica, redes Wi-Fi y seguridad, gracias a las cuales se redactaron algunas secciones de la introducción de este trabajo. También se han consultado libros y artículos específicos sobre seguridad en redes inalámbricas, para realizar una primera aproximación a los capítulos que abordan los protocolos de seguridad WEP y WPA/WPA2. Para conocer mejor las medidas de seguridad que emplean estos protocolos se han recopilado monografías, artículos de revistas, informes técnicos, resúmenes ejecutivos, etc, e incluso se han examinado las especificaciones de los correspondientes estándares o RFCs para resolver las cuestiones más técnicas. En cuanto al estudio de las debilidades de estos mecanismos de seguridad, las principales fuentes de documentación han sido artículos científicos y trabajos monográficos sobre materias relacionadas con esta temática, entre los que se incluyen trabajos universitarios como proyectos y tesis de maestría

Para el desarrollo de la aplicación se han utilizado los lenguajes de programación *Java*, para implementar el cliente, y *Python,* junto con las librerías *Scapy* y *Python-Wifi* escritas en este mismo lenguaje, para implementar el servidor. Además, se han utilizado herramientas como el entorno *NetBeans*, con un plugin para programación en Python, el depurador *Winpdb*, también para Python, y el analizador de protocolos *Wireshark*, durante las pruebas de software correspondientes al desarrollo de la aplicación. No obstante, no puede obviarse la restricción de que la implementación de la aplicación desarrollada fue destinada a sistemas *GNU/Linux*. En concreto, ha sido probada con el sistema operativo *Ubuntu,* versión 8.10, con un núcleo compilado a medida que incluía tanto el driver correspondiente a las interfaces de red Wi-Fi con chipset *Zydas*, parcheado para inyección, como el driver original para interfaces Wi-Fi con chipset *Ralink rt73*. Por lo tanto, se han realizado pruebas con adaptadores inalámbricos de diferentes fabricantes y, a continuación, se ha procedido de la misma forma con los puntos de acceso.

1.3 Contenido de la memoria

En esta sección se describe brevemente el contenido de los capítulos posteriores a éste y de los apéndices que se incluyen en esta memoria:

- **Capítulo 2**, realiza un breve repaso sobre algunas de las tecnologías de comunicación inalámbricas más populares en la actualidad y, a continuación, amplía la presentación previa de la tecnología Wi-Fi con una descripción más detallada de las características fundamentales de esta clase de redes, basándose en las especificaciones de la versión inicial del estándar IEEE 802.11.
- **Capítulo 3**, profundiza en los aspectos de seguridad abarcados por la primera versión del estándar 802.11, que conciernen casi exclusivamente al protocolo WEP, y también describe someramente las principales debilidades de este protocolo de seguridad y los ataques conocidos más importantes contra tales debilidades.
- **Capítulo 4**, se centra en las técnicas introducidas en la enmienda 802.11i para mejorar la seguridad provista por las especificaciones originales del estándar 802.11, como son la autenticación mediante PSK o mediante el estándar IEEE 802.1X y los protocolos de seguridad TKIP y CCMP, aunque aborda las mismas cuestiones de seguridad que el capítulo anterior
- **Capítulo 5**, adopta un enfoque práctico para revisar los principales procesos y otras cuestiones relevantes, incluyendo el análisis, el diseño y las dificultades asociadas con la programación, afrontadas durante el desarrollo de una sencilla aplicación destinada a vulnerar la seguridad de una red Wi-Fi mediante la implementación de una prueba de concepto de una debilidad del protocolo WEP.
- **Capítulo 6**, extrae algunas conclusiones generales, aunque de carácter práctico, sobre los diferentes protocolos o técnicas de seguridad, incluidos en la versión más reciente del estándar 802.11, que facilitan la autenticación de las estaciones o que protegen la confidencialidad, la integridad y la autenticidad de los datos intercambiados. Además, propone algunas mejoras para la aplicación desarrollada, así como determinadas líneas de investigación sobre estos protocolos que pueden ser iniciadas o bien ampliadas.
- **Apéndice A**, describe el contenido de los CD-ROMs que se entregan conjuntamente con esta memoria.
- **Apéndice B**, contiene un glosario que explica brevemente el significado de la mayoría de los acrónimos utilizados que guardan relación directa con la materia de este trabajo.
- **Apéndice C**, cita la bibliografía más relevante usada para la elaboración del trabajo.

Capítulo 2

2. Introducción a las redes Wi-Fi

Desde que en Junio de 1997 fuese ratificada la primera versión del estándar 802.11, hasta que en Octubre del año 2009 se hiciese lo mismo con la enmienda 802.11n, han sido muchos los cambios y las mejoras introducidas en las redes Wi-Fi, en parte, gracias a la labor de estandarización del grupo de trabajo 802.11 del IEEE. Este trabajo de estandarización se centra en los niveles físico y de enlace lógico de datos del modelo OSI y, por lo tanto, se ocupa de cuestiones como los niveles de potencia permitidos de la señal, las bandas y los canales habilitados para la transmisión, las técnicas empleadas en la modulación de la señal, los mecanismos de control de acceso al medio o de control de errores, el formato de las tramas y los intercambios válidos de tramas, entre los dispositivos que integran una red inalámbrica, para cada una de las operaciones definidas en el estándar.

En este capítulo se pretende realizar una descripción general de los aspectos más importantes del mencionado estándar que son abordados también por alguna certificación de la Wi-Fi Alliance. Sin embargo, esta descripción estará restringida a la revisión del estándar de 1999, y no hará referencia a ninguna característica incluida en enmiendas posteriores, salvo en lo que respecta a las novedades introducidas, concernientes al nivel físico, por las enmiendas 802.11g y 802.11n. Tampoco se abordarán aquellas características calificadas como opcionales en el estándar, ni las relacionadas con la seguridad, ya que estas últimas serán tratadas en posteriores capítulos.

No obstante, antes de adentrarnos en el insondable mundo de las redes Wi-Fi, se precederá dicha descripción con una breve y genérica exposición sobre diferentes tecnologías de comunicación inalámbricas. En esta exposición se intentarán abstraer, para cada una de las tecnologías expuestas, algunos valores típicos para determinadas características generales asociadas a esta clase de redes. De esta manera, se aportarán algunas nociones sobre estas tecnologías, que nos permitirán realizar comparaciones y dilucidar algunos aspectos que hayan favorecido su consolidación.

Al final de la primera sección se muestra una gráfica (**Figura 1**) donde se representan varios rangos de velocidades de transmisión y de radios de alcance soportados por cada una de estas tecnologías, si bien hay que tener en cuenta que, en general, la velocidad de transmisión no es constante en todo el área cubierta por la red inalámbrica, sino que suele reducirse conforme nos acercamos a los límites de dicha área. Además, es posible que cualquiera de estos dos parámetros tomen valores fuera del rango representado en la gráfica, puesto que dependen de numerosos y, en ocasiones, muy variables factores, como la potencia de transmisión, el ancho de banda disponible, la clase de antenas empleadas, la movilidad de los terminales, etc.

2.1 Algunas tecnologías inalámbricas

El rápido crecimiento experimentado por las comunicaciones inalámbricas en las últimas décadas ha sido acompañado por una proliferación, no menos rápida, de las tecnologías de comunicación de este tipo, que puede ejemplificarse en la plétora de tecnologías que se incluyen dentro de la telefonía celular de segunda o tercera generación. El principal factor que ha contribuido al desarrollo de esta amplia gama de tecnologías inalámbricas ha sido la demanda, por parte de las aplicaciones que son provistas mediante este tipo de comunicación, de diferentes requisitos en lo que respecta a la comunicación de datos o los dispositivos utilizados para la transmisión. Algunos de estos requisitos afectan a la velocidad o la fiabilidad de las transmisiones, al ancho de banda, la potencia o las propiedades espectrales de la señal, o incluso al consumo de energía o el coste de los dispositivos de transmisión.

Tampoco deben menospreciarse otros factores, como la competencia entre las empresas del sector, pues a menudo estas tecnologías originan patentes o marcas comerciales, o el empeño de las organizaciones de estandarización en producir estándares abiertos y consensuados que respalden a las tecnologías en desarrollo, de manera que estos estándares se adapten a la demanda del mercado, los intereses de la empresas o el marco regulatorio de las telecomunicaciones en los países representados en tales organizaciones. Seguidamente, se describen algunas características generales de diversas tecnologías de comunicación inalámbrica que han sido seleccionadas por su extensa implantación o bien porque destacan en algún nicho del mercado.

2.1.1 Bluetooth

Esta tecnología, desarrollada inicialmente por la compañía *Ericsson*, en la actualidad es gestionada por una alianza tecnológica en la que participan más de trece mil empresas, denominada *Bluetooth SIG* (*Special Interest Group*). Uno de los propósitos para los que fue creada era la conexión de teléfonos móviles y accesorios que podían utilizarse con estos teléfonos. Aunque también soporta muchas otras aplicaciones que, además, pueden beneficiarse de ciertas facilidades detalladas en las especificaciones de esta tecnología. Estas facilidades están disponibles para estas aplicaciones mediante determinados servicios, los cuales están vinculados típicamente a capas de nivel superior a la de enlace de datos. Sin embargo, no todas estas facilidades han sido incluidas en especificaciones alternativas a la del *Bluetooth SIG*. Tal es el caso del estándar 802.15.1 del IEEE, gracias al cual la anterior organización adopta esta tecnología basándose en la versión 1.1 de las especificaciones del *Bluetooth SIG*, aunque está restringido, como es habitual en los estándares del grupo de trabajo 802.11, a los dos niveles inferiores del modelo OSI.

En aquellas aplicaciones que demandan transmisiones inalámbricas de corto alcance y a baja velocidad, y que requieren dispositivos de bajo consumo y bajo coste, *Bluetooh* representa una opción tecnológica adecuada. Algunas aplicaciones en las que se ha utilizado esta tecnología son: la transferencia de archivos, tarjetas de presentación o incluso citas de agenda entre dispositivos, la sincronización entre ordenadores y dispositivos móviles y la conexión de ordenadores o de otros dispositivos con periféricos o accesorios específicos de estos aparatos, etc.

En cuanto a las características técnicas, hay que destacar que se escogió una banda situada en torno a los 2,4 GHz del espectro electromagnético para la transmisión de los dispositivos *Bluetooth*. En muchos países, esta banda está reservada para aplicaciones industriales, científicas o médicas (ICM) y no requiere licencia para transmitir. A consecuencia de esto, es utilizada también por otras tecnologías y en muchos lugares su uso está saturado. De manera que, para reducir los efectos de las frecuentes interferencias, se emplea para la transmisión una técnica de espectro expandido mediante salto en frecuencias, en la que los datos se transmiten sucesivamente a través de una secuencia de 79 canales (o 23 en algunos países), conmutando de canal a una tasa de 1600 veces por segundo. Con el uso de esta técnica la velocidad de transmisión no supera 1 Mbps, aunque en versiones más recientes de las especificaciones se introducen modulaciones que permiten la velocidad de 3 Mbps. Mientras que el alcance típico de la señal es de 10 metros, existe una clase de dispositivos *Bluetooh* que alcanza hasta 100 metros, a costa de incrementar el consumo de energía.

2.1.2 Wi-Fi

Aunque ya en 1979, los experimentos realizados por ingenieros de IBM en Suiza demostraron que podía desplegarse una red de área local mediante enlaces de infrarrojos, no fue hasta 1985, fecha en la que la FCC (acrónimo de *Federal Communications Commision*) asigna en EEUU las bandas comprendidas entre los 902 - 928 MHz, los 2400 - 2483 MHz y los 5725 - 5850 MHz para el uso de aplicaciones ICM sin necesidad de licencia, cuando la investigación sobre redes locales inalámbricas recibió el impulso suficiente para posibilitar el desarrollo de productos comerciales, que saldrían al mercado años más tarde.

De esta forma, a principios de los noventa, existen varios productos de esta clase en el mercado, entre ellos los comercializados bajo la marca *Wavelan*, que se consideran precursores de los que actualmente reciben la denominación Wi-Fi. Sin embargo, no tuvieron mucho éxito debido a que la mayoría eran incompatibles. Por esas fechas, se constituye el grupo de trabajo 802.11 con la intención de elaborar un estándar que solucionase el problema anterior. En 1994 se publica el primer borrador, que será ratificado tres años después. No obstante, las velocidades de transmisión establecidas no superan los 2 Mbps y los productos que se adhieren al estándar tampoco tienen demasiada aceptación.

Cuando llega el año 1999, sale a la luz la enmienda 802.11b, en la que se especifican nuevas técnicas mediante las que se alcanzan los 11 Mbps y en agosto de ese año varias empresas del sector fundan la WECA (*Wireless Ethernet Compatibility Alliance*), que después pasaría a denominarse *Wi-Fi Alliance*, con el objetivo de promocionar esta tecnología y de fomentar la interoperabilidad de los productos de este tipo. Desde entonces, la penetración en el mercado de esta tecnología ha continuado su avance hasta adquirir un dominio casi absoluto frente a otras alternativas que compiten por el mismo segmento del mercado, incluso cuando ofrecen un rendimiento mayor, como es el caso de *HiperLAN*. Como ya se ha mencionado, la principal aplicación de la tecnología discutida en este apartado son las redes locales inalámbricas, pero también la comunicación directa e instantánea entre computadores o dispositivos móviles mediante redes provisionales creadas para tal fin. Además, con el hardware apropiado, es posible establecer enlaces punto a punto de larga distancia, que pueden extenderse varias decenas de kilómetros.

Otra aplicación muy común es el despliegue de redes de acceso a intranets, a Internet o a cualquier otra red de un proveedor de contenidos. Para dar soporte a todas estas aplicaciones, se han habilitado en muchos países una banda en la región de los 2,4 GHz y varias en la región de los 5 GHz (bandas ICM que no requieren licencia) en las que se permite la transmisión con restricciones en la potencia de la señal emitida. En la actualidad, la mayoría de las redes Wi-Fi permiten transmitir a 54 Mbps, pero usando las nuevas características incluidas en la enmienda 802.11n, que ha sido ratificada hace pocos años, es posible alcanzar los 600 Mbps. La distancia máxima a la que puede ser decodificada la señal depende de muchos factores pero, en general, supera los 30 metros en interiores y los 100 metros en exteriores, aunque las estaciones que implementan la capa física basada en la enmienda 802.11n pueden aumentar en más del doble esa distancia en determinados entornos.

2.1.3 WiMAX

A finales de los noventa algunas compañías telefónicas desarrollaron y desplegaron varias tecnologías inalámbricas propietarias que proporcionaban conexiones de banda ancha, en sustitución del cable o de líneas ADSL. Algunas de estas tecnologías recibieron el nombre genérico de MMDS o LMDS, y normalmente realizaban transmisiones mediante microondas en bandas con licencia habilitadas para ese fin. Los sistemas MMDS podían abarcar un área de unas pocas decenas de kilómetros, ofreciendo velocidades de transmisión entre 0.5 y 30 Mbps, en función de la distancia del cliente a la estación base, entre otros factores.

Por otro lado, los sistemas LMDS limitaban su cobertura a unos cuantos kilómetros, pero permitían velocidades agregadas de transmisión superiores a 30 Mbps. No obstante, funcionaban a frecuencias más altas, generalmente superiores a 20 GHz, lo que dificultaba la comunicación cuando no existía línea de visión directa entre el emisor y el receptor. Al no estar estandarizadas estas tecnologías, casi siempre presentaban inconvenientes relacionados con la interoperabilidad, el coste de los equipos, la disponibilidad de proveedores, etc. Así que éstas fueron algunas de las causas que propiciaron que estas tecnologías no consiguieran consolidarse y también de que se estableciese el grupo de trabajo 802.16 del IEEE en 1999, para elaborar un estándar que diese respuesta a estas necesidades.

En el mismo año en el que se aprueba el primer estándar del grupo de trabajo 802.16, se funda el *WiMAX Forum*, con el apoyo de varias empresas y con objetivos idénticos a los de la *Wi-Fi Alliance*, pero dirigidos a la tecnología *WiMAX*, que está basada en los estándares producidos por dicho grupo de trabajo. Esta versión inicial del estándar se centraba en la comunicación punto a multipunto con dispositivos fijos ubicados en la línea de visión de la estación base, mediante la utilización de microondas en alguna banda comprendida entre los 10 GHz y los 66 GHz, razón por la que se requería licencia para la transmisión en muchos países. En posteriores modificaciones del estándar 802.16 se habilitaron bandas de menor frecuencia para transmitir prescindiendo de la línea de visión, incluso bandas sin licencia, y se posibilitó la comunicación con estaciones móviles que se desplazaran a menos de 60 Km/hora.

El área cubierta por una estación base WiMAX puede alcanzar los 50 km de radio y su velocidad agregada de transmisión los 130 Mbps, aunque cuando no hay línea de visión con los equipos de los clientes, la velocidad máxima se encuentra en torno a 75 Mbps y cuando éstos se desplazan no excede de los 30 Mbps. También la distancia de los clientes a la estación base, los obstáculos presentes en la trayectoria de la señal y las condiciones atmosféricas, entre otros factores, pueden afectar negativamente a la velocidad de transmisión. En la práctica, para ofrecer conexiones de suficiente calidad a todos los usuarios y por razones de escalabilidad, se limita el área de una celda a 16 Km para los clientes con línea de visión y a 8 Km para los que no tienen línea de visión, y la velocidad de transmisión agregada no suele exceder los 40 Mbps.

El objetivo principal de esta tecnología fue impulsar el bucle local inalámbrico y ofrecer una alternativa factible al par trenzado y al cable coaxial para conectar a los abonados con la red del operador de telecomunicaciones. También se ha utilizado para la implementación de intranets de área metropolitana, ya que mediante las infraestructuras apropiadas posibilita, por ejemplo, interconectar las facultades o varios campus de una universidad, las oficinas o sucursales de una empresa con la central ubicada en la misma ciudad, etc. Otra aplicación, cada vez más frecuente de esta tecnología, consiste en el despliegue de redes troncales o de interconexión de otras redes implementadas mediante otras tecnologías inalámbricas, como pueden ser diferentes redes Wi-Fi o incluso redes celulares de telefonía móvil de tercera generación, proporcionando un enlace entre *hot-spots* o estaciones base y la red troncal del proveedor de servicios.

2.1.4 Telefonía celular

Aunque los inicios de la radio-telefonía se remontan a principios del siglo XX, el primer sistema de telefonía móvil completamente automatizado se considera que fue implantado por la compañía *Ericsson* a mediados del citado siglo. Por entonces, estos servicios eran bastante caros, estaban disponibles en sólo unas pocas ciudades y para un número muy reducido de usuarios. Además, los terminales móviles eran grandes, pesados y tenían un coste muy elevado. Con los avances de la tecnología electrónica y la invención de los sistemas celulares se superan muchos de estos inconvenientes. De modo que, en la década de los 80, se produce una gran expansión de la telefonía móvil, que se pone al alcance, tanto económicamente como geográficamente, de millones de personas. Esta es la época de la primera generación de telefonía móvil celular, basada en las transmisiones analógicas.

En la siguiente década, con la llegada de la segunda generación, se produce un fuerte crecimiento del mercado de la telefonía móvil, a la vez que se impone la transmisión digital. Además de mejorar la calidad de las comunicaciones de voz, se introducen servicios de datos como los mensajes de texto, el acceso a contenido multimedia o incluso a Internet. Con la entrada del nuevo siglo comienza el despliegue de redes de la tercera generación. Las tecnologías de esta generación emplean conmutación de paquetes para la transmisión de datos y ofrecen un ancho de banda que posibilita una mayor gama de servicios multimedia, como algunas formas de streaming. Desde hace algunos años, algunas compañías anuncian la puesta en servicio de redes de cuarta generación, si bien existen importantes discrepancias entre diferentes organizaciones y empresas sobre qué características deben reunir estas redes y cómo diferenciarlas de las de tercera generación, aunque parece que el acceso móvil de banda ancha y la provisión de servicios mediante redes IP marcarán esta generación.

Debido a la gran variedad de tecnologías que se agrupan bajo las distintas generaciones de telefonía celular, es difícil generalizar sobre características que presenten todas estas tecnologías. En lo que respecta a la tasa de datos, las tecnologías de segunda generación, por lo general, no superan los 200 Kbps, mientras que las de tercera generación sí que lo hacen. En cambio, estas últimas típicamente estaban limitadas a unos pocos megabits por segundo. Para la cuarta generación, la ITU fijo como objetivo alcanzar los 100 Mbps para terminales de alta movilidad y 1 Gbps para aquellos de baja movilidad.

Actualmente, los operadores ofrecen tasas de datos más modestas, de modo que cada terminal puede disponer de una tasa comprendida entre 2 a 6 Mbps, en la mayoría de los casos. En cuanto a las bandas utilizadas para la transmisión, normalmente son bandas que requieren licencia y que han sido asignadas por cada país para dicho uso. No obstante, existen varias bandas en torno a los 800, 1800 y 2100 MHz que se utilizan en muchos países para tal fin. Finalmente, la cobertura de una red de telefonía celular depende fundamentalmente de las estaciones base que forman parte de esa red, ya que, por ejemplo, las que se despliegan en ciudad no suelen exceder de 1 o 2 Km de alcance, mientras que las ubicadas en el exterior puede alcanzar los 30 km o incluso distancias mayores si se utiliza transmisión analógica o frecuencias bajas en la banda UHF.

2.1.5 Comunicaciones por satélite

El primer satélite artificial puesto en órbita fue el *Sputnik* I, lanzado por la Unión Soviética en 1957. Aunque sus funciones eran muy limitadas y su vida útil fue muy corta, despertó el temor de EEUU a que los soviéticos alcanzasen una notable ventaja tecnológica frente a ellos, por lo que tomaron medidas al respecto, iniciándose así la carrera espacial. A consecuencia de esto, en enero de 1958 los americanos lanzan el *Explorer* I, que se convertiría en el primer satélite en órbita de esta nación. Pero no es hasta 1962, cuando la NASA, en cooperación con otros organismos y empresas, pone en órbita el precursor de los satélites de comunicación modernos, el *Telstar* I. Este satélite permitió la retransmisión de señales de televisión, llamadas telefónicas, faxes y datos, entre varias estaciones terrestres y fue el primero que llevó a cabo retransmisiones transatlánticas de televisión en directo. Transcurridos cinco meses después de su lanzamiento, sus circuitos se sobrecargaron y comenzó a fallar, aunque de todas formas ya había superado el tiempo de servicio de muchos de los satélites predecesores.

Algunos años después, otros países comenzaron a desplegar sus propios satélites, de modo que en la actualidad se han efectuado miles de lanzamientos de satélites y se estima que hay más de 900 satélites activos en órbita, junto a más de 18.000 objetos inservibles de tamaño apreciable en órbitas terrestres próximas, incluyendo satélites sin servicio, etapas de cohetes de lanzamiento y fragmentos originados en colisiones. Aparte de sus numerosas aplicaciones científicas y militares, los satélites se han utilizado en diferentes formas de comunicación, como la difusión de radio y televisión en regiones extensas, la transmisión de voz o datos hasta lugares deshabitados o sin infraestructuras, para enlazar redes o estaciones terrestres situadas a gran distancia, etc.

Al igual que sucede con la telefonía móvil celular, muchos países reservan algunas bandas para la transmisión por satélite de aquellos operadores a los que conceden las licencias. Generalmente, estas bandas coinciden con algunas de las establecidas por la ITU para este propósito, como son las bandas L (1,5/1,6 GHz), S (1,9/2,2 GHz), C (4,0/6,0 GHz), Ku (11/14 GHz) y Ka (20/30 GHz). Cada banda de las anteriores, a su vez, se divide en al menos en dos bandas: la de menor frecuencia se utiliza en la transmisión en sentido descendente (del satélite a tierra) y la de mayor frecuencia en sentido ascendente (desde tierra hacia el satélite). No obstante, el ancho de banda de estas bandas unidireccionales no es el mismo en todos los casos, sino que depende de la banda principal a la que pertenecen y suele oscilar entre decenas de megahercios y varios gigahercios. Tampoco coinciden en muchos casos las asignaciones de la anchura o la frecuencia central de las bandas principales o secundarias, realizadas por diferentes organismos, como el IEEE, la OTAN, etc.

La tasa de datos a la que puede transmitir un satélite depende del número y del tipo de transpondedores de los que dispone. Un transpondedor típico de 36 MHz de ancho de banda puede retransmitir varios cientos o miles de llamadas telefónicas, dependiendo de la compresión utilizada, o un canal de televisión analógica, o entre 2 a 16 de televisión digital, dependiendo de la calidad, compresión, definición y otros parámetros de cada canal. Los satélites modernos pueden llevan varias decenas de transpondedores, por lo que algunos pueden superar la velocidad de transmisión agregada de 1 Gbps. Esto permite que, por medio de ciertos satélites, se ofrezca conexión a Internet de banda ancha, con tasas de datos agregadas que pueden rebasar los 30 Mbps, de los cuales cada usuario puede disfrutar, en general, de tasas de descarga entre 768 Kbps y 5 Mbps.

Por último, el área de la superficie terrestre donde puede recibirse la señal de un satélite con la suficiente potencia para permitir su decodificación, cuando alcanza una posición determinada de su órbita, se denomina huella del satélite y depende no solo del tipo de órbita que recorre el satélite, sino también del número y del tipo de antenas que incorpora dicho satélite. Normalmente, la huella tiene un radio mayor de 100 km (por ejemplo, empleando antenas de pequeña abertura es posible reducir la huella hasta muy cerca de este límite), pero también puede abarcar más de un tercio de la superficie terrestre.

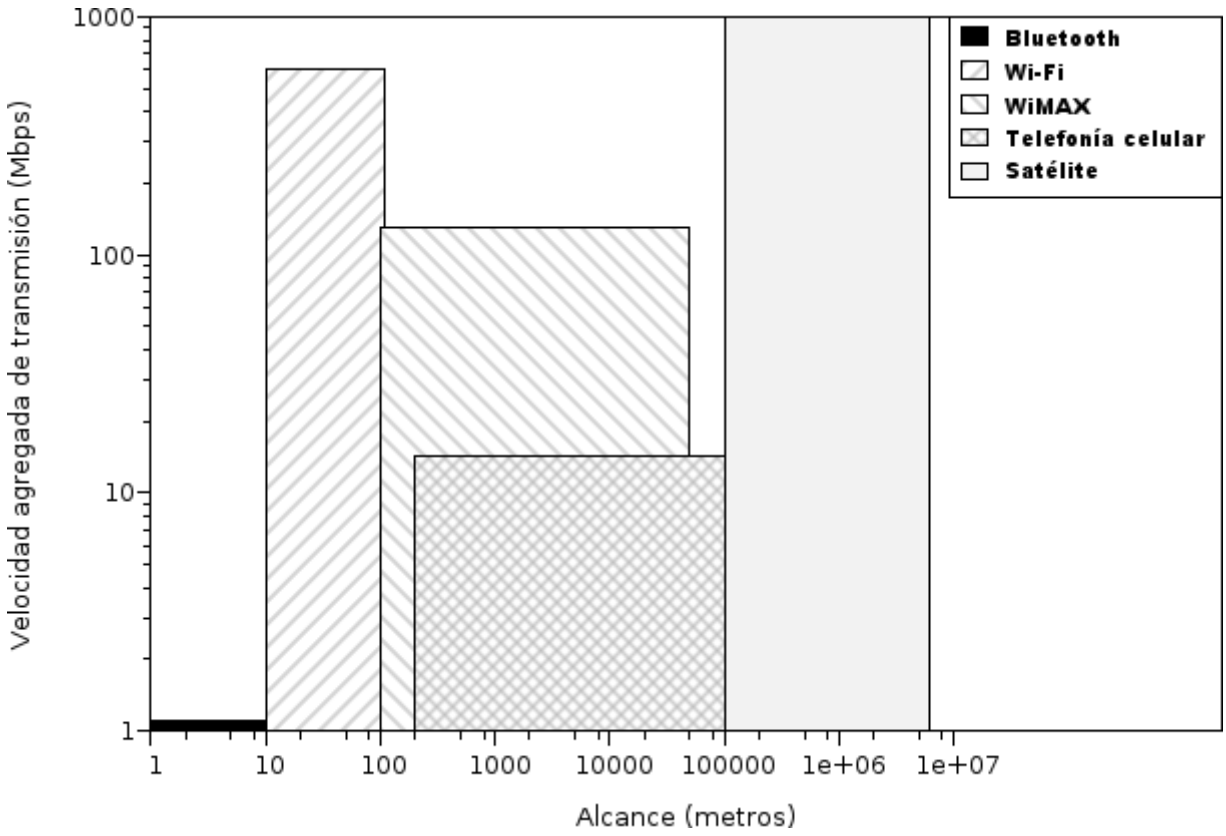

Figura 1: Velocidad de transmisión frente a alcance en algunos tipos de redes inalámbricas

2.2 Conceptos básicos sobre redes Wi-Fi

La transmisión a través del medio inalámbrico, esto es: el aire, es la causa de las diferencias fundamentales que existen entre las redes inalámbricas y las redes cableadas, algunas de las cuales se mencionan a continuación. En primer lugar, los límites de una red inalámbrica son imprecisos y no pueden ser fácilmente delimitados. Además, estas redes son inherentemente menos fiables y menos seguras que las desplegadas mediante cables, puesto que la trayectoria de propagación de la señal generalmente es variable, asimétrica y no está confinada a un área de acceso restringido. De igual manera, resulta muy complicado impedir su interferencia o interceptación por dispositivos externos. En segundo lugar, la topología de la red puede ser dinámica y la conectividad entre las estaciones puede no ser completa, como suele ocurrir cuando los clientes son móviles o cuando están dispersos por regiones extensas. Considerando estas características particulares, se diseñó una arquitectura genérica que pudiese dar soporte a las funciones y capacidades exigidas a estas redes. A continuación se describen brevemente los aspectos más importantes de esta arquitectura.

Los nodos que constituyen una red inalámbrica basada en el estándar 802.11 se denominan estaciones y, aparte de implementar las entidades de red del nivel físico y del nivel de enlace de datos conforme al estándar, se caracterizan por disponer de una dirección en la subcapa MAC. Esta dirección debería ser única en cada red Wi-Fi y debería permitir la comunicación entre las entidades del nivel de enlace. Un conjunto de estaciones pueden interconectarse de dos formas distintas, bien como una red en modo *Ad hoc*, en cuyo caso constituyen una LAN independiente denominada *IBSS* (*Independent Basic Service Set*), en la que cada estación se comunica de forma directa con las otras estaciones, o bien como una red en modo *Infraestructura*, en la cual existe una estación distinguida que recibe el nombre de punto de acceso y que, generalmente, participa en la comunicación entre un grupo de estaciones.

Las estaciones de una red en modo *Infraestructura* que están vinculadas al mismo punto de acceso compiten, o bien cooperan, para transmitir a través del medio inalámbrico y forman parte de lo que se llama un *BSS* (*Basic Service Set*), que se relaciona habitualmente con el concepto de celda en una red inalámbrica. A su vez, una red en modo *Infraestructura* puede estar compuesta por varios BSSs interconectados mediante un sistema de distribución, y en tal caso recibe el nombre de *ESS* (*Extended Service Set*). No obstante, las estaciones que pertenecen al mismo ESS, pero a distintos BSSs pueden direccionarse y comunicarse mutuamente, mediante las entidades correspondientes de la capa de enlace de datos. También es posible la conexión de LANs, cableadas o no basadas en el estándar IEEE 802.11, con un ESS. En este caso, la conexión se realiza a través del sistema de distribución, mediante unas entidades lógicas que se denominan *portales* y que efectúan las oportunas conversiones o adaptaciones entre los protocolos del nivel físico y de enlace de datos involucrados. A menudo, los puntos de acceso implementan estas entidades y frecuentemente actúan como portales entre redes 802.11 y redes Ethernet.

La arquitectura descrita anteriormente se diseñó con el objetivo de que los dispositivos que se ajustan a la misma pudiesen implementar una serie de servicios especificados en el estándar 802.11. Mediante la implementación de todos estos servicios se pretendía que una red 802.11 ofreciese una funcionalidad similar a la de una red cableada, incluyendo los aspectos de conectividad, seguridad e interoperabilidad, junto con algunas funciones adicionales, como la itinerancia. En función de los componentes de la arquitectura por los cuales son implementados los servicios, éstos se clasifican en dos categorías: servicios de estación, que están presentes en cualquier estación, y servicios del sistema de distribución, que como indica su nombre son provistos por el sistema de distribución, incluyendo los puntos de acceso.

A continuación se enumeran y se describen brevemente los servicios de estación:

- *Autenticación*, mediante este servicio se controla el acceso de las estaciones a una red inalámbrica, aunque no implica que se verifiquen sus identidades, del mismo modo que sucede, generalmente, cuando un equipo se conecta físicamente a una red cableada. Opcionalmente, mediante este servicio se puede restringir el acceso a estaciones que poseen una clave secreta predefinida.
- *Desautenticación*, es el servicio opuesto al anterior y permite finalizar una autenticación establecida entre dos estaciones, por ejemplo, un punto de acceso y un cliente, o bien dos estaciones cualesquiera de un IBSS. Puede ser invocado por diversas razones, como cuando un cliente desea desconectarse de una red o incluso lo puede invocar el punto de acceso periódicamente para mayor seguridad. En cualquier caso, una estación puede terminar una autenticación establecida con otra estación cualquiera de forma unilateral.
- *Privacidad*, este servicio proporciona confidencialidad a las comunicaciones de una red 802.11, por lo que si se prescinde del mismo, lo que supone una práctica desaconsejable pero posible (puesto que su implementación es opcional), cualquier dispositivo dentro del alcance de la red y con el transceptor adecuado puede recuperar los datos transmitidos. En cambio, cuando se activa este servicio se habilita el cifrado de los datos, de forma que tan solo las estaciones que comparten cierta clave privada deberían ser capaces de descifrar tales datos. Con frecuencia, las estaciones pertenecientes a un mismo BSS comparten las mismas claves secretas, aunque el estándar no obliga a ello.
- *Entrega de MSDUs*, es el servicio encargado del intercambio confiable, entre estaciones de una red 802.11, de los datos entregados a las entidades de la subcapa MAC de tales estaciones. Por lo tanto, se ocupa de tareas como el control de errores, la fragmentación y el reensamblado de los datos, la reordenación de tramas, etcétera. En una red en modo *Infraestructura*, es necesaria la participación de otros servicios para cumplir este cometido.

Las redes en modo *Infraestructura* también proporcionan los, así llamados, servicios del sistema de distribución, que se comentan seguidamente:

- *Asociación*, este servicio es invocado por una estación para vincularse a un punto de acceso, de tal forma que se permita a la estación utilizar el sistema de distribución y las tramas

enviadas por ésta sean dirigidas a su destino. No obstante, solo se permite que una estación se asocie con un único punto de acceso, ya que estableciendo esta restricción el sistema de distribución puede determinar de forma unívoca e inmediata el punto de acceso a través del cual puede enviar una trama a una estación.

- *Desasociación*, es el servicio recíproco del anterior y es invocado para extinguir una asociación existente, lo que provoca que una estación sea, de forma efectiva, expulsada de una red. Puede ser invocado por un punto de acceso, por ejemplo, cuando deja de estar operativo o se supera el número máximo de asociaciones que soporta, entre otras causas, aunque también puede ser invocado por una estación, por ejemplo, cuando un usuario se desconecta de la red. Como en el caso de la autenticación, se lleva a cabo de forma unilateral, sin que la otra estación pueda rechazar su ejecución.
- *Reasociación*, gracias a este servicio es posible la transición entre puntos de acceso de una estación, que es un proceso que se conoce comúnmente como itinerancia o "*roaming*". Este proceso es iniciado por una estación, por razones no especificadas en el estándar, pero en la práctica, casi siempre sucede debido a una mejor recepción de la señal de otro punto de acceso del mismo ESS. Adicionalmente, a través de este servicio una estación puede solicitar a un punto de acceso una modificación en los atributos de una asociación que mantiene con éste.
- *Distribución*, antes de enviar una trama de datos se debe invocar este servicio para determinar si el mensaje será retransmitido por el punto de acceso dentro del propio BSS, o bien si será enviado a otro BSS por medio del sistema de distribución. En el segundo caso, el sistema de distribución debe entregar el mensaje al punto de acceso correspondiente al BSS de la estación de destino. No obstante, en el estándar 802.11 se evita intencionadamente cualquier indicación sobre la estructura o la implementación del sistema de distribución.
- *Integración*, este servicio habilita el intercambio de datos entre una red 802.11 y cualquier otro tipo de LAN. Como ya se ha comentado, este servicio es provisto por unas entidades lógicas denominadas portales, que posibilitan la entrada al sistema de distribución de datos procedentes de LANs de otros tipos, así como la salida de los mensajes enviados por las estaciones 802.11 hacia otras LANs. Por esta causa, los portales deben realizar las oportunas adaptaciones, por ejemplo, en el formato de las trama, en los tipos de direccionamiento, en la información sobre prioridad o calidad de servicio, etc.

2.3 El nivel físico

La implementación de la capa del nivel físico, como es definido en el modelo OSI, determinará o influenciará en gran medida algunas características importantes de una red inalámbrica, como la velocidad de transmisión, el alcance, la tasa de errores, la tolerancia al ruido o a las interferencias o incluso el nivel de interferencia que provoca en otras señales externas. Después de la elaboración de la versión inicial del estándar 802.11 se han publicado varias enmiendas que especifican nuevas capas del nivel físico y que mejoran diversas prestaciones de las anteriores capas.

Una estación que se ajuste al mencionado estándar debe implementar al menos una de las capas físicas descritas en el mismo, aunque para formar parte de un BSS también se requiere que implemente al menos una de las capas soportadas por el punto de acceso. En el resto de esta sección se realizará un breve repaso de las diferentes capas físicas definidas actualmente en el estándar, indicándose en el epígrafe de cada subsección la correspondiente versión o enmienda del estándar en la que son introducidas. Por último, la sección concluye con una tabla en la que se recopilan algunas características fundamentales que se han mencionado sobre cada capa física.

2.3.1 802.11-1997

En la versión original del estándar 802.11, aprobada en 1997 y revisada dos años más tarde, se especificaban tres capas físicas distintas, una basada en infrarrojos y las otras dos en señales de radiofrecuencia que emplean distintas técnicas de codificación de canal mediante espectro expandido, una de ellas mediante salto en frecuencias (*FHSS*) y la otra mediante secuencia directa (*DSSS*). Las implementaciones de estas capas debían proporcionar, de forma obligatoria, una velocidad de transmisión de 1 Mbps y, opcionalmente de 2 Mbps, excepto para las implementaciones de la capa basada en DSSS, a las que se exige ambas velocidades.

La primera capa mencionada fue concebida para la transmisión difusa (no se requiere línea de visión directa entre emisor y receptor) mediante infrarrojos con longitudes de onda entre 850 y 950 nm. Aunque presentaba el inconveniente de un corto alcance (generalmente inferior a 10 m) y de no poder utilizarse en exteriores, también contaba con ciertas ventajas, como su inmunidad a las interferencias de radiofrecuencia y su mayor seguridad, al no propagarse la señal a través de la mayoría de obstáculos que encuentra en su camino, tales como paredes, techos, etc. A pesar de estas ventajas, no hay constancia de ninguna implementación comercial de este capa física.

La capa basada en FHSS fue diseñada para transmitir en la banda ICM en torno a los 2.4 GHz, así como para reducir las interferencias mutuas entre los dispositivos que implementan esta capa y otros dispositivos que utilizan esta banda. Por esta razón, dicha banda se dividió en 95 canales cuyas frecuencias centrales estaban separadas a 1 MHz, a efectos de uso por las implementaciones de esta capa. Aunque hay que advertir que no todos estos canales son utilizados en cada país, sino que depende del dominio regulador que corresponde a cada país y de su normativa particular sobre telecomunicaciones.

Para implementar la técnica de FHSS se transmitía la señal a través de una secuencia establecida de canales (con menos de 80 canales), siendo la tasa de cambio de canal configurable, sin embargo el tiempo máximo de permanencia en cada canal estaba limitado, siendo fijado este límite por el dominio regulador correspondiente. Aparte de estos cambios periódicos de canal, la frecuencia de la señal también oscilaba a causa de su modulación mediante desplazamiento en frecuencia gaussiano (*GFSK*), técnica que utiliza dos o cuatro frecuencias distintas y desplazadas respecto a la frecuencia central del canal (para las velocidades de transmisión de 1 o 2 Mbps, respectivamente). Tampoco en este caso los dispositivos que implementaron esta capa tuvieron demasiada repercusión en el mercado.

Finalmente, la capa basada en DSSS también se diseñó para operar en la misma banda ICM de 2.4 GHz, aunque con una distribución de canales diferente a la capa basada en FHSS. De modo que para la capa DSSS se definieron 14 canales, cuyas frecuencias centrales estaban separadas a 5 MHz (con la excepción del canal 14, para el cual la separación era de 12 MHz con respecto al canal precedente), aunque no siempre estaba permitida la transmisión en todos estos canales, dependiendo del dominio regulador aplicable y, en ocasiones, del país en cuestión (por ejemplo, EEUU y Canadá habilitaron los canales desde el 1 hasta el 11, mientras que en muchos países de Europa estaban disponibles los canales del 1 al 13). No obstante, debido a las propiedades espectrales de la señal que se establecieron para esta capa física, dos señales de esta clase podían provocar interferencias apreciables entre ellas si eran transmitidas por dos canales que estuviesen separados por menos de cuatro canales adyacentes (véase **Figura 2**).

La solución escogida para implementar la técnica de DSSS fue la transmisión de una palabra de 11 chips (bits o elementos de señal integrados en una secuencia de algún código de expansión) por cada bit de datos, aunque ésta podía transmitirse invertida en función del valor del bit. Además, esta palabra pertenecía a un código de *Barker*, mediante el cual se consigue distribuir la energía de

la señal por un ancho de banda mayor que con la codificación original (dado que la tasa de chips supera a la tasa de datos), de manera que para los receptores de banda estrecha, una señal codificada con DSSS se asemeja al ruido y no afectará a la señal que procesan, si se recibe con la suficiente relación señal/ruido. De forma recíproca, una señal de banda estrecha normalmente no influye en la decodificación de una señal DSSS, puesto que en este proceso la energía de la señal de banda estrecha se dispersa y su potencia se mantiene por debajo del nivel de potencia de la señal DSSS decodificada, a menos que la potencia de la señal de banda estrecha recibida supere cierto umbral.

Por último, la secuencia de chips se modulaba mediante desplazamiento en fase diferencial, con dos o cuatro símbolos distintos (correspondientes a las modulaciones DBPSK o DQPSK), para alcanzar las velocidades de transmisión de 1 o 2 Mbps, respectivamente. Estas velocidades estaban aún muy por debajo de las que ofrecían otras tecnologías de redes locales de bajo coste, así que ésta pudo ser una de las causas de la escasa aceptación en el mercado de los productos que implementaron esta capa física, especificada en la primera versión del estándar.

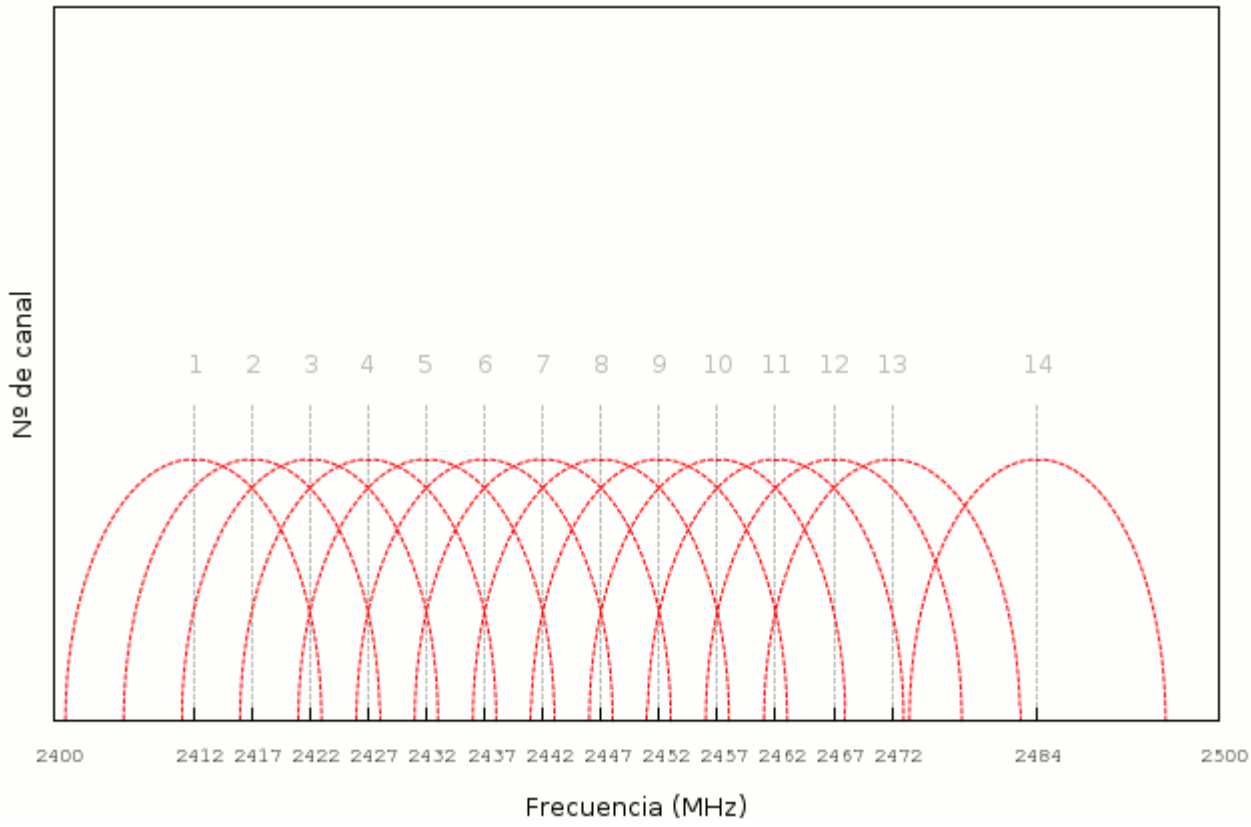

Figura 2: Canales para dispositivos 802.11 DSSS/b/g en la banda ICM de 2.4 GHz

2.3.2 802.11b

Pocos meses antes de terminar el año 1999, se aprobaron dos nuevas enmiendas para el estándar 802.11, que se denominaron *802.11a* y *802.11b*, respectivamente, de manera que cada una de éstas introducía una nueva capa física. Como se acordó que la capa detallada en la enmienda 802.11b debía ser compatible con la capa basada en DSSS, se describirá en primer lugar. Esta capa operaba en la misma banda y también contaba con los mismos canales y las mismas restricciones sobre los canales y sobre el espectro de la señal transmitida que la capa basada en DSSS. De hecho, el texto de esta enmienda incluye las especificaciones de la capa física basada en DSSS para la transmisión a 1 o 2 Mbps.

Pero además, esta enmienda introducía dos nuevas velocidades de conexión a 5.5 y 11 Mbps, soportadas mediante otra implementación de la técnica de espectro expandido, denominada *CCK* (*Complementary Code Keying*), aunque opcionalmente podía utilizarse otra técnica basada en una codificación convolucional y denominada *PBCC* (*Packet Binary Convolutional Coding*) para alcanzar las mismas velocidades. Debido al aumento de la velocidad de conexión que ofrecían estas técnicas, se distinguieron mediante el nombre de *HR-DSSS* (*High Rate Direct Sequence Spread Spectrum*) y aunque la primera técnica tiene alguna característica en común con la capa física basada en DSSS introducida en el estándar original, por ejemplo: la tasa de chips (11 millones por segundo), las diferencias entre ambas son sustanciales.

En primer lugar se distingue por el uso de diferentes secuencias de chips, pertenecientes a un código de *Walsh-Hadamard* de 8 chips de longitud. Por lo tanto, el propósito de tales secuencias no solo consiste en alterar el espectro de la señal transmitida, sino también en codificar varios bits de datos. Otra diferencia importante es que los chips se expresan como símbolos QPSK, que es la modulación aplicada a la señal, y que la fase de algunos chips puede variar en función de los bits de datos que codifican en ese instante, pero también en función de valores de bits previos (emplea alguna forma de codificación diferencial). No obstante, aunque cada chip potencialmente podría codificar dos bits, en realidad, para la velocidad de conexión de 5.5 Mbps, cada una de las 16 secuencias de chips utilizadas codifica 4 bits de datos, mientras que para la velocidad de 11 Mbps existen 64 secuencias diferentes que codifican 8 bits de datos cada una.

Alternativamente, puede emplearse la codificación PBCC y las modulaciones BPSK o QPSK para transmitir a 5.5 o a 11 Mbps, respectivamente. Aunque esta última codificación no se puede considerar como una técnica de espectro expandido, comparte algunos objetivos con esta técnica, como mejorar la fiabilidad de la transmisiones, en este caso mediante un código convolucional de tasa 1/2, así como las propiedades espectrales de la señal, para lo cual recurre a una secuencia cíclica de 256 bits, denominada *cover code*, que sirve para seleccionar una de dos constelaciones diferentes de símbolos BPSK, o bien QPSK, mediante las cuales se modulan los bits obtenidos después de la convolución. La mayoría de los fabricantes ignoraron esta codificación opcional, así que fue implementada en un número muy reducido de dispositivos 802.11b. Sin embargo, los dispositivos compatibles con esta enmienda obtuvieron una considerable penetración en el mercado durante varios años después de su aparición.

2.3.3 802.11a

En la enmienda 802.11a, en cambio, se especificaba una nueva capa física que prescindía de la compatibilidad con capas previas y con la capa basada en la enmienda 802.11b, aprobada al mismo tiempo. Esto era debido, principalmente, a que operaba en una región del espectro electromagnético distinta, ubicada en torno a los 5 GHz y en la que transmitían relativamente pocos dispositivos. En EEUU, la organización reguladora de las telecomunicaciones (FCC) habilitó las denominadas bandas *UNII* (*Unlicensed National Information Infraestructure*) para la transmisión sin licencia en esta región del espectro. En concreto, tales bandas fueron denominadas: *UNII-1* (o UNII inferior, comprendida entre 5.150 y 5.250 GHZ), *UNII-2* (o intermedia, entre 5.250 y 5.350 GHz) y *UNII-3* (o superior, entre 5.725 y 5.825 GHz). Muchos países respetaron estas convenciones, pero otros restringieron el uso de parte de estas bandas o incorporaron algunas nuevas (por ejemplo, en torno a los 4.9 GHz) para este fin.

En todo caso, la enmienda recomendaba la división de esta región en canales cuyas frecuencias centrales estuviesen separadas a 5 MHz y la numeración de los mismos partiendo de un canal en una frecuencia inicial, definida por cada dominio regulador o cada país. De esta manera, en muchos países la región de los 5 GHz se dividía en 200 canales, correspondiendo el canal 0 a la frecuencia de 5.000 GHz, el canal 1 a 5.005 GHz y así sucesivamente. En la práctica, muchos de estos canales no se utilizaron porque no pertenecían a las referidas bandas UNII o bien se encontraban dentro pero muy cerca de los límites de estas bandas, por lo que su uso fue restringido.

Por otra parte, las exigencias de esta enmienda, en cuanto al espectro de la señal transmitida, desaconsejaban la transmisión simultánea por canales cuyas frecuencias centrales no estuviesen separadas al menos a 20 MHz. Por esta causa, cada banda UNII contaba tan solo con cuatro canales sin solapamiento (en realidad, si que existe un poco), ofreciendo todas estas bandas un total de doce canales (véase **Figura 3**). Uno de los objetivos prioritarios de esta enmienda era definir una capa física que permitiese alcanzar velocidades de conexión notablemente mayores que las especificadas en la primera versión del estándar. La principal dificultad para conseguir este objetivo residía en la interferencia intersímbolo, que es producida por la propagación multitrayectoria de la señal y se agrava al disminuir el tiempo de símbolo. Por esta razón, se decidió incorporar a esta capa la técnica de procesamiento de la señal llamada OFDM, que ya había sido utilizada con éxito en otras tecnologías, como en la familia *xDSL*. Otras cuestiones, como el uso de bandas menos saturadas o la tolerancia a interferencias en banda estrecha también se tuvieron en cuenta en esta enmienda.

Aplicando la técnica OFDM, no se utiliza todo el ancho de banda disponible para transmitir una señal a la velocidad de modulación más alta posible, sino que este ancho de banda se reparte entre una serie de señales subportadoras de diferentes frecuencias y de menor velocidad de modulación. Concretamente, en esta capa física, esta técnica se aplicó a través de 52 subportadoras, de las cuales 48 transportaban distintos bits de datos en paralelo y las otras cuatro servían para monitorizar el canal. El caso más común consistía en asignar un ancho de banda de 20 MHz (donde se localiza la mayor parte de la energía de la señal) para la transmisión de este tipo de señal, siendo espaciadas las subportadoras a 0.3125 MHz. Pero incluso empleando todo este ancho de banda, la energía de las señales subportadoras adyacentes se solapaba, aunque esto generalmente no suponía un obstáculo para la decodificación de estas señales.

Previamente a la transmisión de los datos, éstos se someten a una codificación convolucional de tasa 1/2, 2/3 o 3/4, para reducir la tasa de errores en la recepción. A continuación, los bits obtenidos, cada uno de los cuales se asigna a una subportadora en función de su posición, se modulan mediante BPSK, QPSK, 16-QAM o 64-QAM y seguidamente se transmiten en sus correspondientes señales subportadoras. Aunque todas las subportadoras utilizan la misma modulación en un instante dado, las diferentes combinaciones de códigos de convolución y modulaciones posibilitan una amplia variedad de velocidades de conexión. Sin embargo, en las especificaciones de esta enmienda solo se establecieron como obligatorias las velocidades de 6 (BPSK, 1/2), 12 (QPSK, 1/2) y 24 (16-QAM, 1/2) Mbps y, opcionalmente, podían ser incorporadas las velocidades de 9, 18, 36 (BPSK, QPSK y 16-QAM, 1/3), 48 y 54 Mbps (64-QAM, 2/3 y 3/4).

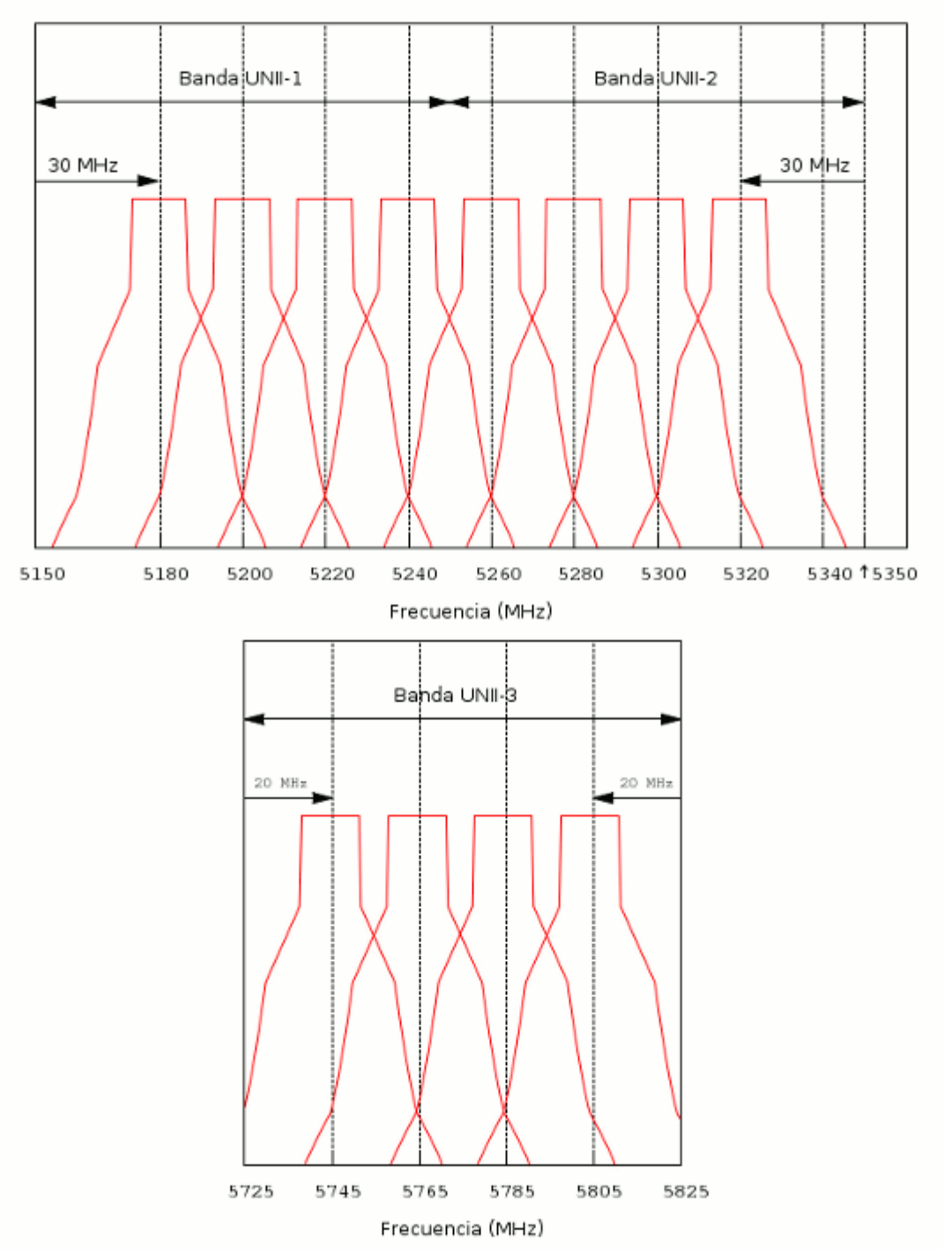

Figura 3: Canales para dispositivos 802.11a en las bandas UNII (5.150 - 5.825 GHz)

Aunque las enmiendas 802.11a y 802.11b fueron aprobadas al mismo tiempo, los productos conformes a la primera enmienda salieron más tarde al mercado por diversas causas, como el mayor coste de sus componentes y la falta de armonización internacional en el uso de estas bandas, lo cual implicaba la prohibición o la existencia de restricciones adicionales en ciertos países para transmitir en las bandas en las que operaban los mencionados productos. Por aquellas fechas, los dispositivos 802.11b ya se habían consolidado, alcanzando una elevada cuota de mercado. Aunque pasado algún tiempo, con la apertura de estas bandas en muchos países y la apuesta de muchas empresas por redes inalámbricas con una mayor capacidad y fiabilidad, los dispositivos compatibles con la capa física 802.11a consiguieron una mayor presencia en el mercado.

2.3.4 802.11g

A mediados del año 2003 era aprobada otra enmienda, denominada 802.11g, en la cual se especificaba otra capa del nivel físico. Esta capa incorporaba algunas características convenientes para muchos usuarios, por ejemplo, el incremento de la velocidad de conexión con respecto a los dispositivos 802.11b, pero sin renunciar a la compatibilidad con los mismos. Para conseguir este propósito, la enmienda en cuestión definía las mismas bandas y canales para la transmisión y soportaba las mismas modulaciones que la enmienda 802.11b. Adicionalmente, para superar las velocidades de conexión estipuladas en la anterior enmienda, utilizó la misma técnica y las mismas modulaciones que la enmienda 802.11a, esto es, OFDM y modulación por desplazamiento en fase, aunque en la banda ya mencionada de los 2.4 GHz.

A pesar de que las restricciones que se fijaron sobre el espectro de la señal para las técnicas DSSS y OFDM eran distintas, se recomendó la misma gestión de canales para las redes integradas por dispositivos 802.11b y las que contaban exclusivamente con dispositivos 802.11g. Por otro lado, para facilitar la interoperabilidad de los dos tipos de dispositivos, cuando pertenecían a un mismo BSS, los que realizaban transmisiones basadas en OFDM debían recurrir a algún mecanismo de protección que evitase la interferencia de las estaciones 802.11b, puesto que éstas son incapaces de reconocer esta clase de transmisiones. Tales mecanismos acarrean siempre algún tipo de sobrecarga, la cual produce cierta pérdida de rendimiento en las transmisiones de las estaciones 802.11g.

Opcionalmente, los dispositivos conformes a esta enmienda podían implementar una versión mejorada de la técnica de transmisión denominada PBCC, que se introdujo en la enmienda 802.11b. Las especificaciones se referían a esta variante como *ERP-PBCC* y se diferenciaba por permitir las velocidades de conexión de 22 y 33 Mbps, correspondientes a las velocidades de modulación de 11 y 16.5 MBaudios, respectivamente, aunque en ambos casos se empleaba la modulación 8-PSK y un código convolucional de tasa 2/3. En consecuencia, las implementaciones de esta capa debían soportar obligatoriamente las velocidades de conexión de 1, 2, 5.5, 6, 11, 12 y 24 Mbps, y de forma opcional las velocidades de 9, 18, 22, 33, 36, 48 y 54 Mbps. Muchos fabricantes apostaron por esta capa física y se apresuraron a salir al mercado con dispositivos compatibles, incluso antes de que fuese ratificado el borrador final de la enmienda 802.11g. Los consumidores también respaldaron a estos productos, de manera que gradualmente han ido reemplazando a sus predecesores, hasta que actualmente la mayoría de dispositivos Wi-Fi en servicio son compatibles con la enmienda 802.11g.

2.3.5 802.11n

La última capa física especificada por el grupo de trabajo 802.11, hasta la fecha, se denominó *High Troughput* y se publicó en la enmienda 802.11n, a finales del año 2009. Dicha enmienda supuso un salto cualitativo y cuantitativo respecto a las velocidades de conexión de las anteriores capas físicas, además incrementó el área cubierta y proporcionó una cobertura más homogénea dentro de dicha área en muchos casos. También se acordó que esta capa física debía ser compatible con las especificadas en las enmiendas 802.11a y 802.11g. Adicionalmente, se incluyó un modo de compatibilidad que posibilitaba la transmisión con las técnicas introducidas en la enmienda 802.11n en presencia de estaciones legadas (previa indicación a éstas).

En lo que concierne a las modulaciones y las codificaciones, no se realizaron muchos cambios con respecto a las anteriores capas basadas en OFDM, aunque se aumentó en cuatro el número de subportadoras de datos y, de forma opcional, podía doblarse el ancho de banda utilizado (en total, 40 MHz) para transmitir a través de 114 subportadoras (de las cuales 108 transportaban datos útiles) o incluso podía reducirse el intervalo de guarda de las subportadoras a 400 ns. También se incorporó una nueva tasa de codificación convolucional (de razón 5/6) y se utilizó, para las velocidades de conexión más altas, dos instancias de la anterior codificación sobre dos secuencias alternas de bits pertenecientes a la MPDU. Alternativamente, podía reemplazarse la codificación convolucional por otro código corrector de errores denominado *LDPC* (*Low Density Parity Check*).

No obstante, las principales diferencias de la capa 802.11n con respecto a sus predecesoras residían en las nuevas técnicas de procesamiento de la señal, disponibles gracias a la adopción de la tecnología *MIMO* (*Multiple Input Multiple Output*). Esta tecnología hizo posible la transmisión o la recepción de múltiples señales simultáneamente, por medio de dos o más antenas con sus propios transceptores (en realidad, la literatura técnica se refiere al término *RF chains*, no a *transceivers*). Sin embargo, la única técnica MIMO cuya implementación exige la enmienda 802.11n se denomina *SDM* (*Spatial Diversity Multiplexing*), o multiplexación espacial, y consiste en la transmisión en paralelo de múltiples señales, emitidas por el mismo canal y por distintas antenas, que pueden ser recibidas por medio de varias antenas en el receptor. Cada una de estas señales transmite datos independientes y recibe el nombre de *spatial stream,* de manera que la enmienda establece que los puntos de accesos compatibles deben soportar la transmisión de al menos dos *streams* y hasta un máximo de cuatro, por lo que, en el mejor caso y en un entorno apropiado, podría cuadriplicarse la tasa de datos.

Otras dos técnicas de esta clase, incluidas opcionalmente en las implementaciones de esta capa, se denominan *Transmit Beamforming* y *STBC* (*Space-Time Block Code*), aunque sus beneficios están más orientados hacia la fiabilidad y la robustez de las transmisiones que hacia el rendimiento, lo que favorece generalmente el alcance de estas transmisiones. Con todas estas variantes de ancho de banda utilizado, modulaciones, tasas de codificación, intervalos de guarda y *streams* espaciales, las especificaciones posibilitan más de 300 velocidades de conexión distintas, aparte de las exigidas por compatibilidad con las estaciones 802.11a/g, sin embargo solo se requiere que 16 de éstas sean soportadas por los puntos de acceso y 8 por las demás estaciones. En las condiciones apropiadas, un punto de acceso, que incorpore las mejores opciones posibles de los parámetros enumerados, podría transmitir una trama de datos a 600 Mbps.

Otra novedad de la enmienda 802.11n es que también especifica nuevas características de la subcapa MAC con el objetivo de incrementar el rendimiento, como un intervalo de espera reducido (*RIFS*) entre las transmisiones de tramas por la misma estación, como la agregación de tramas o de la carga útil de las tramas (*A-MPDU* o *A-MSDU*, respectivamente) o como la confirmación de bloques de tramas (*Block Ack*), en vez de la confirmación individual, además de otras características opcionales que no se abordan en este trabajo. En el aspecto comercial, también salieron al mercado bastantes dispositivos 802.11n antes de la aprobación de la enmienda, que se produjo años después de lo previsto, siendo muchos de estos dispositivos implementados, e incluso certificados, conforme a la segunda versión del borrador de la enmienda 802.11n. Por esta razón, el subcomité del IEEE

encargado de la elaboración de tal enmienda procuró mantener la compatibilidad con este borrador en las sucesivas revisiones. En la actualidad, casi todos los dispositivos Wi-Fi, cuya fabricación se llevó a cabo durante los últimos años, se adhieren a la enmienda 802.11n, de modo que se espera que sustituyan a la base de dispositivos 802.11a/b/g que continúan en producción en la actualidad.

2.3.6 Resumen de las capas del nivel físico

En la cabecera de la **Tabla 1**, mostrada más abajo, se han seleccionado algunas características relevantes del nivel físico implementado por una interfaz de red inalámbrica cualquiera y en las filas siguientes se especifican los valores correspondientes a estas características, o bien las técnicas fundamentales empleadas para la implementación de estas características, para las distintas capas físicas definidas en el estándar IEEE 802.11 y en las enmiendas vigentes actualmente. En la primera columna se especifica la versión del estándar o la enmienda en la que se introdujo dicha capa física. La segunda columna especifica, de forma general, la banda del espectro electromagnético en la que se llevan a cabo las transmisiones de los dispositivos compatibles y también el número máximo de canales disponibles en dicha banda para estos dispositivos (el número real depende del dominio regulador y del país).

En la tercera columna se hace referencia, si es aplicable, a la codificación principal de los datos (por ejemplo, una codificación de canal u otro tipo de codificación característica de la capa física en cuestión) y se ignoran las codificaciones secundarias (como las que se aplican, de forma general, para la corrección de errores, para la eliminación de componente continua o para el entrelazado o el incremento de la aleatoriedad de los bits transmitidos). A continuación, justo en la misma columna, se enumeran todas las técnicas de modulación de la señal que pueden ser implementadas por la correspondiente capa del nivel físico.

A continuación, en la cuarta columna se enumeran las velocidades de conexión o tasas de datos de las correspondientes capas del nivel físico (en Mbps, considerando a los bits de la MPDU como los datos útiles, de hecho, esta PDU constituye la carga de datos transportada por la capa PLCP), excepto para la capa física definida en la enmienda 802.11n, puesto que por falta de espacio no se pueden mostrar todas las tasas de datos que soporta esta capa. Observe que en esta columna se muestra con tipografía resaltada las tasas de datos obligatorias para cada capa física. Finalmente, en la quinta columna se incluyen algunos comentarios pertinentes respecto a ciertas características generales que se han sintetizado de cada capa.

Estándar	Banda / Canales	Codificación / Modulación	Tasa de datos	Comentarios
802.11-1997	300 THz / -	- / 16-PPM, 4-PPM	**1**, 2	Infrarrojos con longitudes de onda entre 850 y 950 nm.
802.11-1997	2.4 GHz / 95	FHSS / GFSK	**1**, 2	Canales separados a 1 MHz. Secuencias de salto de < 80 canales. Para 1 Mbps/2 Mbps hay 2/4 símbolos distintos (frecuencias desplazadas).
802.11-1997	2.4 GHz / 14	DSSS (Barker) / BPSK, QPSK	**1, 2**	Hasta 14 canales separados a 5 MHz, con solapamiento los que están separados por menos de 4 canales. Código de Barker de 11 chips.
802.11b	2.4 GHz / 14	DSSS (CCK) / QPSK	**1, 2, 5.5, 11**	Compatible con 802.11-DSSS. Para 1/2 Mbps usa Barker, para 5.5/11 Mbps usa CCK (4/8 bits por palabra del código de 8 chips complejos).
802.11b	2.4 GHz / 14	PBCC / PSK, QPSK	**5.5, 11**	Codificación opcional basada en un código convolucional de tasa 1/2.
802.11a	5 GHz / 200	OFDM / PSK, QPSK, 16-QAM, 64-QAM	**6**, 9, **12**, 18, **24**, 36, 48, 54	Canales solapados y separados a 5 MHz, aunque no todos están autorizados. Hasta un total de 12 canales sin solapamiento, entre las 3 bandas UNII.
802.11g	2.4 GHz / 14	OFDM / PSK, QPSK, 16-QAM, 64-QAM	**1**, **2**, **5.5**, **11**, **6**, 9, **12**, 18, **24**, 36, 48, 54	Compatible con 802.11b, utilizado en las 4 primeras tasas. Para el resto, usa las mismas modulaciones y codificaciones que 802.11a, pero en la banda y los canales de 802.11b.
802.11g	2.4 GHz / 14	PBCC / 8-PSK	22, 33	Codificación opcional basada en un código convolucional de tasa 2/3 y una velocidad de modulación de 11 o bien 16.5 millones de baudios.
802.11n	2.4 GHz / 14 5 GHz / 200	OFDM / PSK, QPSK, 16-QAM, 64-QAM	Más de 300 tasas de datos. (tasa máxima de 600 Mbps)	Compatible con 802.11a/g , opera en las mismas bandas/canales que éstos. Gracias a la tecnología MIMO y a diversas opciones ofrece numerosas y altas tasas de datos.

Tabla 1: Capas del nivel físico definidas en el estándar IEEE 802.11

2.4 Control de errores

Debido a la escasa fiabilidad del medio inalámbrico, en lo que concierne a las comunicaciones, y a la inexistencia de limitaciones sobre el número de dispositivos que transmiten o que generan interferencias en las bandas establecidas en el estándar 802.11, con frecuencia la tasa de errores por bit promediada por las tramas recibidas será lo suficientemente alta para reducir el rendimiento de las comunicaciones inalámbricas de forma apreciable. Teniendo en cuenta esto, los diseñadores del estándar 802.11 decidieron aplicar alguna técnica de control de errores en la propia capa de enlace. El propósito de esta técnica consistía en realizar un tratamiento de errores más eficiente que los llevados a cabo por las técnicas aplicadas en las capas superiores de la pila de protocolos.

La técnica que escogieron se denomina *Stop & Wait* y, a pesar de ser muy poco eficiente, también es muy simple y muy fácil de implementar. Básicamente, esta técnica consiste en esperar una confirmación positiva (mediante una trama denominada *ACK*) de la estación a la que el emisor envió una trama que requiere confirmación, suponiendo que dicha trama sea recibida sin errores. Si el emisor no recibe esta confirmación dentro de un intervalo de tiempo establecido, retransmite la trama y vuelve a iniciar la espera para recibir la trama de confirmación. Este proceso se repite hasta que recibe la confirmación o bien agota el número máximo de retransmisiones permitido para ese tipo de trama, en cuyo caso la trama es descartada.

Además de las tramas de datos, otras clases de tramas, como las tramas de gestión o las tramas de control de tipo *PS-Poll*, también requieren confirmación (aunque para algunos tipos específicos de tramas no es necesario que se efectúe mediante una trama de *ACK*). Por otra parte, las tramas de datos o de gestión, cuya primera dirección MAC (contenida en el campo de la trama que se denomina *Address1*) es una dirección de broadcast o de multicast, están exentas de ser confirmadas. Por esta razón, una trama enviada a una dirección de difusión o multidifusión por un punto de acceso o una estación perteneciente a un IBSS, no necesita confirmación.

Adicionalmente, la enmienda *802.11e*, que incorpora algunas novedades para mejorar ciertos parámetros relacionados con la calidad de servicio, habilita una opción a través de la cual se indica que ciertas tramas de datos no requieren confirmación. También en esta enmienda, se introduce la posibilidad de confirmar una secuencia de tramas transmitidas durante una o varias TXOPs (abreviatura que, en singular, alude al término *Transmission Opportunity*, esto es, un intervalo de tiempo asignado de forma exclusiva a una estación de un BSS para la transmisión de una o varias tramas), mediante una única trama de confirmación (que se denomina *Block Ack*). Como ya se

mencionó anteriormente, las estaciones que son compatibles con la enmienda 802.11n disponen de una técnica de confirmación en bloque similar, que pueden utilizar para la confirmación de ráfagas de tramas, como las asociadas a una A-MPDU.

2.5 Control de acceso al medio

Las estaciones 802.11 pertenecientes al mismo BSS deben competir, o bien deben coordinarse, para transmitir a través del medio inalámbrico compartido, para lo cual recurren a una de las dos funciones siguientes. La primera función, denominada *DCF* (*Distributed Coordination Function*), se caracteriza por el control de acceso distribuido entre las estaciones vinculadas a una misma instancia de esta función y debe ser implementada por cualquier estación compatible con el estándar 802.11. La segunda función, denominada *PCF* (*Point Coordination Function*), se distingue por la presencia de una entidad que recibe el nombre de *PC* (*Point Coordinator*), encargada de conceder turnos para transmitir a las estaciones de un BSS que lo soliciten.

Para ser más concretos, hay que mencionar que la última función únicamente está disponible en redes que operan en modo *Infraestructura*, en las que el papel del PC es desempeñado por el punto de acceso, aunque el estándar 802.11 también impone la coexistencia de esta función con la DCF. Adicionalmente, se requiere que las estaciones que emplean la DCF puedan asociarse a un BSS con estaciones coordinadas mediante la PCF y que obtengan, al menos periódicamente, la oportunidad de transmitir. Como quiera que el estándar establece el carácter opcional de la operación mediante la PCF, en el resto de esta sección se discutirá la función obligatoria, esto es, la DCF.

Dos características fundamentales relacionadas con la forma en la que funciona la DCF son: los diferentes intervalos de tiempo que debe esperar una estación antes de transmitir una trama y el método de acceso al medio compartido, denominado *CSMA/CA*. Al imponer distintos intervalos de espera a las estaciones antes de acceder al medio (justo después de finalizar una transmisión), en función del tipo de trama que intentan transmitir, se pretende dar preferencia a ciertos tipos de tramas en la competencia por el turno de transmisión. Por lo tanto, las tramas con mayor preferencia son aquellas que pueden ser transmitidas después del intervalo más corto que, en un principio, fue el intervalo denominado *SIFS* (*Short Inter-Frame Space*), aunque posteriormente la enmienda 802.11n definió otro de menor duración, que fue denominado *RIFS* (*Reduced Inter-Frame Space*).

Transcurrido un intervalo SIFS, pueden transmitirse tramas de *ACK*, de *CTS* o los fragmentos, que siguen al primero, de una ráfaga de fragmentos. El siguiente intervalo de tiempo, en cuanto a duración, que se usa cuando se opera bajo la DCF, se denomina *DIFS* (*DCF Inter-Frame Space*). Este intervalo precede generalmente a la transmisión de una trama de datos o de gestión, salvo que antes del comienzo del mismo se haya recibido una trama con errores. En este último caso, la espera se alarga lo suficiente para incluir la posible recepción de una de trama de confirmación dirigida a la estación que envió la trama que se recibió con errores (ya que podría haber llegado a su destino correctamente). Por lo tanto, este intervalo es el de mayor duración en el estándar 802.11 original y se denomina *EIFS* (*Extended Inter-Frame Space*). Gracias a la detención durante el intervalo EIFS, pueden evitarse muchas colisiones que se producen debido al problema de la "*estación oculta*", que se presenta con frecuencia en las redes inalámbricas.

Por otra parte, para arbitrar la competencia por el acceso al medio inalámbrico, las estaciones 802.11 implementan el método de acceso CSMA/CA, el cual obliga a posponer la transmisión de una trama, durante un intervalo de tiempo (que se calculará por medio de la ejecución del algoritmo del retroceso exponencial binario), cuando se detecta una transmisión en curso, esto es, al mismo tiempo que la estación intenta transmitir dicha trama. De esta manera, se puede evitar que ocurra una colisión, aunque para tal fin es necesario que las estaciones detecten la presencia de una señal portadora en el medio antes de iniciar la transmisión. Si no se detecta la portadora durante cierto intervalo de tiempo, que para las estaciones que operan bajo la DCF generalmente es un intervalo DIFS, la transmisión puede iniciarse.

El procedimiento descrito hasta aquí coincide, en líneas generales, con el homólogo utilizado en las redes cableadas basadas en *Ethernet* (CSMA/CD). Sin embargo, existen algunas diferencias, entre las que destaca el mecanismo de detección de la *portadora virtual*. Este mecanismo persigue el mismo propósito y se emplea de la misma forma que el asociado a la portadora física, que ya fue comentado, pero está basado en la observación de ciertos campos de determinadas clases de tramas, los cuales anuncian la duración de un intervalo de tiempo durante el cual se intentará completar un intercambio de tramas. De este modo, las estaciones cercanas son informadas de este evento y, en consecuencia, deberían de abstenerse de transmitir durante ese intervalo. Así pues, la detección de alguna trama válida de la clase anterior por una estación provoca que ésta actualice, cuando sea necesario, una especie de temporizador interno que se denomina *NAV* (*Network Allocation Vector*) y que interviene en el mecanismo de detección de la portadora virtual, causando la inhibición de las transmisiones hasta que expire.

En realidad, cuando la capa física de una estación detecta que el medio se encuentra ocupado, como parte del procedimiento de detección de la portadora física, dicha estación debe esperar a que una nueva detección indique que el medio está libre durante un intervalo DIFS (o un intervalo EIFS, si la última trama detectada fue recibida con errores), antes de invocar el mecanismo de retroceso exponencial binario. Tal mecanismo impone a la estación un nuevo estado de espera, cuya duración dependerá del valor de un contador que se inicializará con un valor aleatorio entre cero y el tamaño de la ventana de contención y que se decrementará en una unidad cada vez que se compruebe la ausencia de transmisiones en el medio durante un intervalo DIFS. Cuando el contador se reduzca a cero, comenzará la transmisión de la trama causante de la invocación del mecanismo de acceso al medio compartido.

Una vez enviada la trama, si necesitaba confirmación pero no se recibió durante el transcurso de un intervalo SIFS, entonces se incrementa en una unidad un contador de reintentos asociado a dicha trama, de manera que si este contador alcanza un valor máximo predefinido, la trama será descartada. En el caso de que se incremente el contador de reintentos, posiblemente aumentará también el tamaño de la ventana de contención (al doble menos una unidad del valor previo), siempre y cuando no exceda un valor máximo predefinido. Este valor máximo de la ventana de contención, junto con el valor mínimo inicial, están determinados por la capa física utilizada por la estación. Por lo tanto, la ausencia de una confirmación de una trama, esperada por la subcapa MAC de una estación, es interpretada por esta subcapa como la ocurrencia de una colisión, suceso que acarrea la invocación del mecanismo de retroceso exponencial binario, empleando los valores actualizados de los contadores que se han mencionado.

En cambio, si la trama fue confirmada o no requería confirmación, ambos contadores (tanto el contador de reintentos como la ventana de contención asociados a la trama transmitida) serán nuevamente inicializados. No obstante, previamente se llevará a cabo una última ejecución de una espera, de nuevo mediante el algoritmo del retroceso exponencial binario, por causa de dicha trama. A pesar de que esta ejecución final parezca superflua, su realización aumenta las posibilidades de que las restantes estaciones obtengan la oportunidad de acceder al medio en ese instante. De esta manera, se reduce la probabilidad de que la última estación en realizar una transmisión vuelva a obtener inmediatamente una oportunidad para transmitir y, a largo plazo, generalmente se evita que acapare el tiempo de transmisión.

2.6 Formatos de trama

En el estándar IEEE 802.11 no existe un único formato de trama, sino que se distinguen varios formatos de trama que tienen algunos campos en común, mientras que otros campos son específicos de algunos tipos de trama. Esto es debido a que en la primera versión del estándar se especifican tres tipos fundamentales de trama, con la particularidad de que el formato de ciertos tipos de trama es fijo y para otros tipos existen ligeras variantes. Uno de estos tipos corresponde a las tramas de datos que, como indica su nombre, se utilizan principalmente para transportar datos útiles, esto es, las MSDUs entregadas por la subcapa LLC.

Sin embargo, también se emplean tramas de datos con otros propósitos, como informar de que no hay datos pendientes para su transmisión, notificar un cambio en el modo de ahorro de energía, conceder el turno para transmitir a una estación que opera mediante la PCF o confirmar la recepción de una trama transmitida mientras está vigente esta función (posiblemente, por medio de otra trama que también transporta datos útiles o notifica la concesión de un turno de transmisión). En total, se definen ocho subtipos de tramas de datos en la primera versión del estándar 802.11, dos de uso general (*Data*, *Null-Function*) y seis restringidos a la operación mediante la PCF (*CF-Poll*, *CF-Ack*, *CF-Poll+CF-Ack*, *Data+CF-Poll*, *Data+CF-Ack*, *Data+CF-Poll+CF-Ack*).

Otro tipo de tramas lo constituyen las tramas de control, que intervienen en los mecanismos de control de errores, control de acceso al medio y ahorro de energía. También en este caso, se aprecian pequeñas diferencias de formato entre las tramas pertenecientes a algunos de los seis subtipos de tramas de control. De estos seis subtipos, cuatro de ellos (*ACK*, *RTS*, *CTS*, *PS-Poll*) corresponden a tramas que deben ser reconocidas por cualquier estación conforme a la primera versión del estándar 802.11, mientras que los dos subtipos restantes (*CF-End*, *CF-End+CF-Ack*) corresponden a tramas que son intercambiadas entre estaciones que se coordinan mediante la PCF.

Finalmente, el último tipo de tramas 802.11 es el que presenta una mayor cantidad de subtipos, debido a que muchas tramas de esta clase participan en la implementación de un amplio número de servicios proporcionados por las redes 802.11. Las tramas de este tipo son denominadas tramas de gestión y están involucradas en operaciones como la autenticación, la asociación, la reasociación, el anuncio y el descubrimiento de redes o el ahorro de energía en redes *Ad hoc*. Debido a que las redes 802.11 que operan bajo el modo *Infraestructura* cuentan con una mayor cantidad de servicios, también disponen de más subtipos de tramas de gestión, de forma que es posible diferenciar hasta una decena de subtipos de estas tramas (de acuerdo con la versión inicial del estándar IEEE 802.11,

estos subtipos de tramas de gestión son los siguientes: *Beacon*, *Probe Request*, *Probe Response*, *Authentication*, *Deaunthentication*, *Association Request*, *Association Response*, *Dissasociation*, *Reassociation Request* y *Reassociation Response*). Sin embargo, la primera mitad de estos subtipos también corresponde a tramas que están presentes en redes en modo *Ad hoc*, como sucede con las tramas de gestión del subtipo *ATIM*, específicas de este modo de operación.

A pesar de todas las diferencias entre tipos y subtipos de tramas, todavía existen algunos campos que se incluyen en todos los formatos de trama y que serán enumerados a continuación. El primer campo que aparece en todas las tramas 802.11 se denomina *Frame Control*, se codifica mediante un par de bytes y contiene diversos subcampos de flags o de tipo numérico con una longitud que varía entre 1 y 4 bits. En la siguiente lista se enumeran estos subcampos (usando la misma denominación que en la versión original del estándar 802.11) y también se describe brevemente el propósito para el que fueron definidos. El orden de enumeración de los anteriores subcampos coincide con el orden creciente del peso de sus bits dentro del campo *Frame Control*, esto es, el orden de transmisión en el tiempo de tales subcampos, que se ilustra también en la **Figura 4**, de izquierda a derecha:

- *Protocol Version*, es un campo numérico con una longitud de 2 bits, que indica la versión del protocolo, especificado por medio del estándar 802.11, al que se ajusta el formato de la trama. Actualmente, este subcampo debe contener el valor cero, correspondiente a la única versión del protocolo definida.
- *Type*, tiene el mismo tamaño que el subcampo anterior, pero se utiliza para codificar el tipo de trama a la que pertenece este campo, ya se trate de una trama de gestión, de control o de datos (característica que se indica mediante los valores: 0, 1 o 2, respectivamente, de este subcampo).
- *Subtype*, es el subcampo de mayor longitud, ya que ocupa 4 bits, los cuales junto con los 2 bits del campo *Type* permiten identificar el subtipo específico de una trama.
- *To DS* y *From DS*, representan dos flags binarios cuyos valores determinan conjuntamente cómo deben interpretarse las direcciones MAC presentes en una trama de datos. En cambio, en los restantes tipos de tramas ambos flags deberían anularse.
- *More Frag*, también es un flag binario, aunque en este caso se activa en aquellas tramas de datos o de gestión que transportan un fragmento (a excepción del último) de una MSDU o de una MMPDU, respectivamente.

- *Retry*, cuando este flag binario se encuentra activado en una trama de datos o en una trama de gestión informa que dicha trama es una retransmisión de otra trama idéntica previamente transmitida.
- *Power Management*, a través de este flag binario, cualquier estación, distinta de un punto de acceso, puede comunicar a las otras estaciones del BSS dentro de su alcance, el estado de ahorro de energía en el cual se encontrará dicha estación cuando finalice el intercambio de tramas al que pertenece la trama en cuestión.
- *More Data*, un punto de acceso activa este flag binario en una trama de broadcast o bien de multicast, cuando tiene pendientes más tramas de estos tipos para transmitir, o bien en una trama dirigida a una estación que opera en el modo de ahorro de energía, cuando almacena una o más tramas destinadas a dicha estación. De forma recíproca, una estación también puede activar este flag en una trama dirigida hacia el punto de acceso cuando la interacción con éste se realiza mediante la PCF y la trama en cuestión responde a otra de tipo *CF-Poll*, suponiendo que almacena más tramas destinadas al punto de acceso.
- *WEP*, se trata de otro flag de un bit que debe estar activado en las tramas de datos protegidas mediante este protocolo de seguridad, pero también en ciertas tramas de autenticación que contienen un texto cifrado, con el mismo algoritmo que emplea WEP, y que son enviadas como respuesta a un desafío planteado por otra estación.
- *Order*, es otro subcampo representado mediante un flag binario cuya activación está limitada a las tramas de datos vinculadas a la clase de servicio *estrictamente en orden*. Por lo tanto, las tramas de broadcast y de multicast asociadas a esta clase de servicio no se reordenan con respecto a las tramas unicast procedentes de la misma estación. Esta situación sucede con frecuencia en BSSs en los que existen una o más estaciones que operan bajo el modo de ahorro de energía.

2	2	4	1	1	1	1	1	1	1	1
Protocol Version	Type	Subtype	To DS	From DS	More Frag	Retry	Pwr Mgt	More Data	WEP	Order

Figura 4: Subcampos del campo Frame Control

Del mismo modo, el campo siguiente, el cual ocupa la segunda posición en una trama 802.11, también está presente en todos los formatos de tramas. Su longitud es de dos bytes y es referido en el estándar con los nombres de *Duration* o bien *ID*, en función de cómo se interprete. En la mayoría de clases de tramas, incluyendo las de datos, expresa un intervalo de tiempo (*Duration*) durante el cual se espera que el medio esté ocupado por un intercambio de tramas en el que participa la propia trama que codifica la duración de este intervalo mediante el citado campo. Por lo tanto, el contador NAV que mantiene cada estación es susceptible de ser alterado en función del valor contenido por dicho campo en una trama detectada en el medio (suponiendo que la estación que transmitió dicha trama pertenece al mismo BSS). Por otro lado, en las tramas de control de tipo *PS-Poll*, este campo representa el identificador de asociación (*ID*) que fue asignado por el punto de acceso a la estación que originó la trama.

Finalmente, el mecanismo de control de errores necesita el valor del último campo de las tramas 802.11, llamado *FCS*. Este campo ocupa 4 bytes y contiene una suma de comprobación de todos los campos de una trama, excluyendo al propio campo FCS pero incluyendo, si existiera, a la carga de datos o cuerpo de la trama. Basado en un código de redundancia cíclico, el valor de este campo puede calcularse de forma algebraica (usando aritmética módulo 2) como el complemento a uno del resto de la división entre dos polinomios. El polinomio dividendo se obtiene al sumar y multiplicar, por un par de polinomios constantes (o que dependen exclusivamente de la longitud de la trama), cierto polinomio cuyos coeficientes se corresponden de forma unívoca con los valores de los bits de dicha trama. En cambio, el polinomio divisor, que recibe el nombre de generador, es fijo y viene determinado por el algoritmo de la suma de comprobación utilizado, en este caso una variante del *CRC-32* (*Autodin*/*ADCCP*/*Ethernet*).

Por último, el polinomio resultado se convierte en una cadena de bits de una forma un tanto peculiar. Ya que, mientras que el estándar 802.11 establece en casi todos los casos que los campos numéricos de varios bytes deben transmitirse en orden creciente de significado (esto es, en orden *Little-Endian*, muy poco común en otros protocolos), así como los bits pertenecientes a cada byte, en el caso del campo FCS especifica, por el contrario, que deben transmitirse en primer lugar los bits correspondientes a los coeficientes de los términos de mayor grado (el autor de este trabajo opina que, en la práctica, no se han respetado estas indicaciones). Todas las operaciones descritas en los párrafos previos y destinadas a calcular el valor del campo FCS mediante el método algebraico se han expresado de forma más condensada en la siguiente fórmula:

$$\mathrm{cp}_1\left(\left(x^k\ \mathrm{C}(x)\ +\ \mathrm{P}(x)\ x^{32}\right)\ \bmod\ \mathrm{G}(x)\right)$$

$k \rightarrow$ Nº de bits de los campos computados.
$\mathrm{cp}_1 \rightarrow$ Complemento a uno (invierte los bits).
$\mathrm{P}(x) \rightarrow$ Polinomio que representa los bits de la trama.
$\mathrm{C}(x) \rightarrow x^{32}+x^{31}+x^{30}+\ ...\ +x^2+x+1$
$\mathrm{G}(x) \rightarrow x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+$
$x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Fórmula 1: Cálculo algebraico del CRC-32

Todos los campos obligatorios de una trama enumerados en esta sección pueden contemplarse en la **Figura 5**, donde se muestra el formato de una trama de datos y se indica, encima de cada campo, su longitud en bytes. Exceptuando el campo *Frame Body*, que se omite en algunas clases de tramas (*Null-Function*, *CF-Poll*, *CF-Ack*, *CF-Ack+CF-Poll*), y el campo *Address 4*, incluido solamente en aquellas tramas de datos que son transmitidas a través de un sistema de distribución inalámbrico (*Wireless Distribution System*), todos los campos mostrados en la figura anterior deben aparecer en cualquier trama de datos. En el caso típico en el que la trama incluye exactamente tres campos de direcciones, dos de estos campos contienen las direcciones MAC de las estaciones de origen y de destino de la trama, mientras que el restante contiene el BSSID del BSS.

La asignación de estas direcciones a los campos *Address 1*, *Address 2* y *Address 3* de una trama de datos depende de los valores que indican los flags *To DS* y *From DS* del campo *Frame Control*. En cambio, cuando la trama incluye cuatro direcciones, las dos primeras corresponden a los dos puntos de acceso por los que la trama abandona e ingresa en el sistema de distribución y las dos siguientes a las estaciones de destino y de origen de la trama, respectivamente. Por último, el campo *Sequence Control* está compuesto de un número de fragmento, definido por los cuatro bits menos significativos de este campo, y de un número de secuencia, codificado mediante los restantes doce bits más significativos, de manera que tales contadores o números de secuencia serán incrementados en las sucesivas transmisiones de diferentes fragmentos o tramas, respectivamente, y se mantendrán iguales en las retransmisiones de los mismos fragmentos o tramas.

2	2	6	6	6	2	6	0-2312	4
Frame Control	Duration	Address 1	Address 2	Address 3	Sequence Control	Address 4	Frame Body	FCS

Figura 5: Formato de las tramas de datos

En cambio, las tramas de gestión tienen un formato homogéneo, que puede ser apreciado en la **Figura 6**, excepto en lo que respecta al cuerpo de la trama. En este formato, las direcciones de la estación de destino de la trama, de origen y el BSSID aparecen siempre en este orden en la trama. Además, los flags *To DS* y *From DS* se anulan en todas las tramas de este tipo. Los demás campos cumplen la misma función que en las tramas de datos, aunque será el contenido del cuerpo de la trama lo que distinga a las diferentes clases de tramas de gestión. El cuerpo de una trama de gestión contiene uno o más parámetros obligatorios, que a su vez pueden ir seguidos por otros parámetros opcionales, cuya aparición depende del uso o de la implementación de determinadas características en la estación de origen, como son algunas capas del nivel físico implementadas por la estación, determinados modos de funcionamiento en los que operan la estaciones o incluso ciertos valores de determinados atributos de la entidad de gestión de la estación (traducción del término inglés: *Station Management Entity*).

No obstante, también existen otros parámetros opcionales, como son los parámetros definidos por el fabricante que, lógicamente, no se describen en el estándar. Sin embargo, a los parámetros definidos en el estándar 802.11 se les asigna un número que determina el orden en el que aparecen (cualquier subconjunto de estos parámetros) en una trama de gestión. Por otro lado, si la trama de gestión contiene parámetros definidos por el fabricante, éstos se ubicarán después de los parámetros predefinidos por el estándar. En lo que respecta al formato, podemos distinguir entre los parámetros de longitud fija (pero con formato y contenido heterogéneos) y los parámetros que se ajustan a un formato general y que pueden contener datos de longitud variable. Los parámetros del último tipo se denominan elementos de información (*Information Element*, abreviadamente *IE*) y se caracterizan por sus dos campos iniciales: el primero ocupa un byte y sirve para identificar el tipo específico del IE, mientras que el segundo tiene el mismo tamaño pero contiene cierto valor que permite calcular la longitud del IE. Los restantes campos de un IE son específicos del tipo concreto de dicho IE.

A continuación, se presentan algunos ejemplos de parámetros que pueden aparecer en las tramas de gestión que se mencionan más abajo. Por ejemplo, el parámetro *Capability Information* se utiliza para advertir o exigir determinadas características de una red o de una estación, como ocurre cuando la red es un IBSS o requiere algún tipo de cifrado, o bien cuando una estación informa que soporta confirmaciones de bloques de tramas o que implementa ciertas técnicas para mejorar la calidad de servicio. Este parámetro tiene una longitud fija y su aparición es obligatoria en los siguientes tipos de tramas: *Beacon*, *Probe Response* y (*Re*)*Association Request* /*Response*.

En cambio, el parámetro *CF Parameter Set*, que anuncia determinada información de utilidad para las estaciones que operan mediante la PCF, es un IE de seis bytes de longitud (sin contar los dos campos iniciales) y está presente solo en las tramas de *Beacon* y *Probe Response* emitidas por el punto de acceso que coordina a un conjunto de estaciones mediante dicha función. Como ejemplo final se hace referencia al *SSID*, que es un IE de longitud variable y que contiene el nombre lógico de la red (esto es, el identificador de un ESS o de un IBSS) y que se encuentra en las tramas de *Beacon*, *Probe Request*/*Response* y (*Re*)*Association Request*.

2	2	6	6	6	2	0-2312	4
Frame Control	Duration	Destination Address	Source Address	BSSID	Sequence Control	Frame Body	FCS

Figura 6: Formato de las tramas de gestión

Para finalizar se abordan las tramas de control, que son las más simples y las de menor longitud, puesto que carecen de cuerpo, esto es, no transportan ninguna información adicional aparte de los campos que tienen en común con los otros tipos de tramas. Aun así, existen diferentes formatos de tramas de control, aunque en este trabajo solo se contemplan las tramas especificadas en la versión inicial del estándar. En la **Figura 7** se representa el formato de una trama de tipo *RTS*, mediante la cual se solicita la reserva del medio de transmisión durante el intervalo de tiempo especificado en el campo *Duration* (en microsegundos). En este formato, el campo *RA* (*Receiver Address*) corresponde a la dirección MAC de la estación receptora (por ejemplo, el punto de acceso en un BSS), mientras que el campo *TA* (*Transmitter Address*) contiene la dirección de la estación que transmite la trama.

También se ajustan a este formato las tramas de tipo *PS-Poll*, que son enviadas por las estaciones pertenecientes a un BSS hacia el punto de acceso que opera en ese BSS. A través de esta clase de tramas, las estaciones reclaman las MPDUs destinadas a las mismas, suponiendo que hubieran sido almacenadas por el punto de acceso mientras que tales estaciones funcionaban en el modo de ahorro de energía, por que lo no estaban operativas para la recepción de tramas. Por esta causa, en este tipo de tramas la dirección que aparece en el campo *RA* corresponde a la dirección MAC del punto de acceso. Por otra parte, el segundo campo de las tramas de tipo *PS-Poll* (denominado genéricamente *Duration/ID*) no representa una cantidad de tiempo, sino el identificador de asociación asignado a la estación emisora. Aún faltan dos clases de tramas que se adhieren a este formato y que pueden ser transmitidas por puntos de acceso operando bajo la PCF. Estas dos clases se denominan *CF-End* y *CF-Ack+CF-End*, anulándose en ambas clases el campo *Duration*, mientras que los campos *TA* y *RA* contienen la dirección MAC del punto de acceso y la dirección de broadcast, respectivamente.

El otro formato de tramas de control, descrito en las especificaciones originales del estándar, se muestra en la **Figura 8** y es idéntico al anterior, salvo porque carece del campo *TA*. Este formato es el que presentan las tramas de los tipos *CTS* o *ACK*, sin que se aprecien grandes diferencias en la semántica de los campos de ambas clases de tramas. De nuevo, el campo *RA* indica la dirección MAC de la estación de destino. Por su parte, el campo *Duration* mantiene el significado ya descrito, aunque en una trama de tipo *CTS* su valor se calcula a partir del campo homólogo de la trama *RTS* a la que responde, restándole el tiempo de transmisión de la propia trama *CTS* y la duración de un intervalo SIFS. El mismo cómputo se efectúa en una trama de tipo *ACK*, aunque con respecto a la trama que confirma y al propio tiempo de transmisión de la trama de *ACK*.

Figura 7: Formato de las tramas de control de tipo RTS y PS-Poll

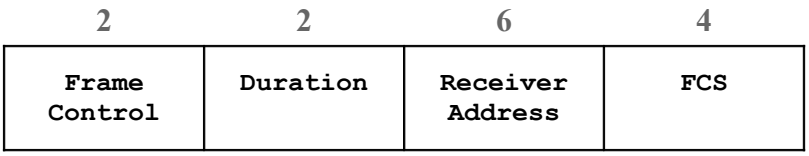

Figura 8: Formato de las tramas de control de tipo CTS y ACK

Capítulo 3

3. Seguridad básica en redes Wi-Fi

Una red inalámbrica es inherentemente menos segura que una red cableada, puesto que la señal no está confinada a un medio de transmisión bien delimitado y protegido físicamente contra el acceso de cualquier atacante potencial. Por el contrario, cualquier atacante pasivo dentro del alcance de la red inalámbrica y provisto con un receptor de radio adecuado puede interceptar y decodificar la señal, si no se toman medidas para evitarlo. En cambio, los ataques activos, particularmente los de denegación de servicio, pueden ser mucho más difíciles de prevenir o incluso en la práctica pueden ser inevitables, debido a la falta de aislamiento del medio de transmisión o de autenticidad en los mensajes correspondientes al protocolo definido en el estándar 802.11.

Generalmente, se considera que la seguridad de una red es suficientemente buena cuando puede garantizar tres características deseables en la comunicación de datos, como son: la autenticación, la privacidad o confidencialidad y la integridad. Mediante la autenticación se pretende comprobar la identidad de algunas o todas las entidades que intercambian mensajes, de forma que sea posible controlar el acceso de estas entidades a la red o a los servicios provistos por la red. La privacidad permite que cierta información contenida en los mensajes pueda ser entendida únicamente por las entidades autorizadas para ello. Por último, la integridad debe asegurar que los receptores de un mensaje sean capaces de distinguir cuando la información recibida es idéntica a la original y cuando ha sido alterada de alguna forma no esperada.

En la versión inicial del estándar 802.11 se especificaron mecanismos para proporcionar todas estas características de seguridad. En este capítulo se describirán los rasgos esenciales de estos mecanismos que, casi en su totalidad, forman parte del protocolo de seguridad denominado WEP (*Wired Equivalent Privacy*). Este protocolo, como indica su nombre, se diseñó para proporcionar a las redes inalámbricas un nivel de seguridad similar al que disfrutan intrínsecamente las redes cableadas. No obstante, aunque parezca un objetivo fundamental para esta clase de redes, en dicho estándar se estipuló que su implementación fuese opcional.

Además, para desgracia de los propietarios de dispositivos que soportaban WEP, a los pocos años de aprobarse el estándar se hicieron públicas algunas debilidades del mismo. Las debilidades descubiertas continuaron aumentando en los años siguientes, de tal modo que consiguieron afectar a todas las características de seguridad que intentaba proporcionar este protocolo. Sin embargo, a pesar de su probada ineficacia, el protocolo WEP todavía se implementa en la mayoría de los dispositivos 802.11, bien por compatibilidad con los dispositivos legados o bien porque no resultan factibles otras alternativas más seguras.

3.1 Autenticación

En un principio se permitían dos formas alternativas de autenticación, a las que se refiere el estándar como autenticación de sistema abierto (*Open System*) o autenticación mediante clave compartida (*Shared Key*), aunque la última solo se exige en los dispositivos que implementan WEP. En el primer caso, el estándar establece que el algoritmo o procedimiento de autenticación es nulo, es decir, solo se requiere que una estación lo solicite para ser autenticada. Aunque no ocurre con frecuencia, una petición de autenticación de esta clase podría ser denegada por causas ajenas a las contempladas en el estándar 802.11.

En cambio, en el segundo caso se emplea un método de autenticación más restrictivo, mediante el cual se intenta comprobar que una estación conoce una clave secreta compartida antes de ser autenticada. Para este propósito se recurre a la técnica de desafío-respuesta que, en esta clase de autenticación, se completa con éxito tras el intercambio de cuatro tramas de gestión con los datos adecuados. Así pues, la estación autenticadora envía una secuencia seudoaleatoria de 128 bytes en la segunda trama, mientras que la estación que inicia la autenticación debe aplicar el procesamiento realizado por WEP sobre dicha secuencia y devolver el resultado a la anterior estación. Cuando las dos estaciones comparten la misma clave WEP, la estación que autentica podrá descifrar el mensaje contenido en la tercera trama, verificar que coincide con la secuencia que generó previamente y notificar a la otra estación, mediante la cuarta trama, que ha sido autenticada.

De acuerdo con el estándar, tras finalizar con éxito el intercambio de tramas correspondiente a alguna de estas dos clases de autenticación, se establece una relación de autenticación mutua entre las dos estaciones (debido a que las tramas de autenticación utilizan direcciones unicast, cada intercambio involucra a un par de estaciones). No obstante, esta relación de autenticación puede ser cancelada por cualquiera de las dos estaciones. También exige el estándar la autenticación

entre las estaciones y el punto de acceso pertenecientes al mismo BSS, previamente al intercambio de datos. Aunque, en la práctica, este procedimiento es unilateral, puesto que son autenticadas las estaciones por el punto de acceso, pero no se efectúa la autenticación recíproca. En cualquier caso, una vez que una estación ha sido autenticada, puede iniciar su asociación con el punto de acceso. Por el contrario, en un IBSS no es obligatoria la autenticación entre las estaciones que forman parte del mismo.

3.2 Integridad

En la primera versión del estándar IEEE 802.11 se especificó un único método para comprobar exclusivamente la integridad de los datos, que además es parte indispensable del protocolo WEP. No obstante, este método no ofrece una protección mucho mejor que la suma de comprobación de una trama (contenida en el campo FCS), puesto que se calcula de la misma forma, esto es, mediante el algoritmo CRC-32, aunque se aplica únicamente a la carga de datos transportada por la trama y se cifra del mismo modo que los datos. Concretamente, en una trama de datos protegida mediante WEP, esta suma de comprobación, que abarca desde la capa LLC hasta la capa de nivel superior contenida en la trama, recibe el nombre de *ICV* (*Integrity Check Value*). Adicionalmente, como parte del encapsulamiento llevado a cabo sobre los datos por el protocolo WEP, el ICV se concatena con los datos, ubicándose a continuación de éstos, y la secuencia resultante se cifra como si fuese una unidad, esto es, completamente y sin distinción entre los datos útiles y el ICV.

En cualquier caso, debemos tener en cuenta que el objetivo que se había planteado para la suma de comprobación basada en el CRC-32 era fundamentalmente la detección de errores en algunos bits aleatorios (o incluso en ráfagas de varios bits consecutivos), por lo que no ofrece una seguridad equiparable a la de un código de autenticación de mensaje (traducción de la expresión en inglés: *Message Authentication Code*), ni tampoco permite verificar la autenticidad del origen del mensaje. Sin embargo, su sencilla y eficiente implementación, junto con el hecho de que no precisa hardware adicional que aumente los costes de producción, posiblemente han sido los factores más valorados para su adopción en el estándar 802.11. Por otro lado, el tipo de encriptación que utiliza el protocolo WEP posibilita que puedan realizarse ciertas alteraciones sobre los datos o el ICV, una vez que han sido cifrados, que no son reveladas por la suma de comprobación del ICV.

3.3 Privacidad

En lo que respecta a la privacidad de los datos, toda la responsabilidad fue confiada al protocolo WEP en la versión inicial del estándar 802.11. Para esta función, WEP utiliza un tipo de cifrado en flujo que recibe el nombre genérico de cifrado *Vernam*. En este tipo de cifrado, cada carácter del texto en claro se combina con un carácter de la clave, mediante la operación *xor*, para producir el correspondiente carácter del texto cifrado. Esta clave, cuya longitud es igual a la del texto en claro, se denomina *keystream* y, cuando consiste en una secuencia realmente aleatoria, es la base de la encriptación denominada *one-time pad*, que se ha demostrado que es irrompible. En la práctica, el keystream se obtiene típicamente a partir de una secuencia seudoaleatoria que puede ser generada tanto por el transmisor como por el receptor del mensaje, a partir de una clave o semilla de longitud reducida. En el caso de WEP, el keystream es generado mediante el algoritmo *RC4* (inventado por *Ron Rivest*) a partir de una clave de 64 bits. No obstante, la mayoría de las implementaciones de este protocolo permiten utilizar también claves de mayor longitud, por ejemplo de 128 o 256 bits (al margen de las especificaciones del estándar).

A pesar de su sencillo diseño, RC4 es uno de los algoritmos de cifrado en flujo más utilizados. Esencialmente, en cada iteración o etapa del algoritmo RC4 se intercambian dos elementos de una permutación de la sucesión de enteros comprendidos entre *0* y *N-1* (incluyendo a ambos números y siendo *N*=256, en el caso de WEP). No obstante, se distinguen dos fases claramente diferentes en este algoritmo. En la fase de inicialización o *KSA* (*Key Scheduling Algorithm*), cada elemento de la sucesión inicial (ordenada de forma creciente) es intercambiado con otro elemento apuntado por un índice seudoaleatorio, cuyo valor depende, en parte, de la clave o semilla que se proporciona como entrada a este algoritmo.

Por lo tanto, al final de esta fase inicial obtenemos una permutación de los *N* números enteros, cuya configuración depende de la clave RC4 empleada. En la siguiente fase, denominada *PRGA* (*Pseudo Random Generation Algorithm*), se procede a la generación del keystream prescindiendo de la mencionada clave. En cada iteración o etapa de esta fase se intercambian igualmente un par de elementos. Además, un elemento de la permutación configurada en cada etapa, que es apuntado por el índice que se obtiene al sumar (módulo *N*) los elementos intercambiados previamente en dicha etapa, constituirá la palabra generada como salida de este algoritmo en esa etapa.

Una particularidad de la encriptación realizada por WEP, es que la clave proporcionada a RC4 como entrada debería ser distinta para el cifrado de sucesivas tramas de datos, de forma que las tramas o MPDUs de esta clase, que son transmitidas consecutivamente por una misma estación, deberían ser cifradas mediante diferentes keystreams. En teoría, esta práctica debería reforzar la privacidad de las MPDUs y facilitar la recuperación de errores de recepción y la comunicación entre varias estaciones. Aunque no se ha postulado la obligatoriedad de esta práctica, se recomienda expresamente en el estándar a través del uso de distintos vectores de inicialización (traducción del término: *Initialization Vector*, en adelante abreviado como *IV*).

Cada IV es una secuencia de 24 bits que desempeña el papel de prefijo de una clave RC4. Así pues, la clave RC4 utilizada para generar un keystream, con el que se cifran los datos de una trama, está compuesta de una clave secreta de 40 bits (o bien, de otro tamaño no contemplado en el estándar original, como 104 o 232 bits) que se concatena y está precedida por un IV de 24 bits, posiblemente distinto para cada trama. Para que la estación receptora pueda descifrar una trama cifrada con WEP, el IV se transmite en claro junto con la propia trama, dentro de una cabecera añadida por el protocolo WEP, la cual precede a los datos cifrados y sigue a la cabecera MAC.

Por otra parte, el protocolo WEP no se ocupa de cuestiones relacionadas con la gestión o la distribución de claves, aunque sí permite que una estación mantenga y pueda utilizar hasta cuatro claves secretas distintas compartidas por las estaciones de un BSS (éstas son las típicas claves WEP que conocen y utilizan habitualmente los usuarios de los dispositivos inalámbricos, aunque algunos autores las llaman claves de broadcast). La clave concreta con la que se encripta una trama se indica mediante un índice de clave de dos bits, que también se incluye sin cifrar en la cabecera WEP, justo después del vector de inicialización. No obstante, no es obligatorio que todas las estaciones de un BSS dispongan de las mismas claves, aunque obviamente dos estaciones podrán intercambiar datos cifrados solamente cuando compartan al menos una clave con el mismo índice.

En este sentido, el estándar también permite definir claves específicas para un par de estaciones transmisora-receptora (al menos diez claves), mediante el atributo *dot11WEPKeyMappings* de la *MIB* (*Management Information Base*) correspondiente a la entidad de gestión de la subcapa MAC, a la que se refiere el estándar mediante el acrónimo *MLME* (*MAC subLayer Management Entity*). A la hora de cifrar o descifrar una trama de tipo unicast, que contenga un par de direcciones MAC asociadas a ciertas estaciones de origen y destino, se escogerá preferentemente la clave específica para ese par de estaciones, en caso de que haya sido definida. Cuando se presenta esta situación,

en la trama cifrada, con la clave específica para el par de estaciones, se debe anular (fijando a cero) el subcampo de la cabecera WEP que denota el índice de la clave compartida. En la práctica, no parece haber muchos dispositivos (si es que hay alguno) que hayan implementado el soporte de las claves WEP específicas para pares de estaciones.

Para finalizar este apartado, se mostrará con mayor detalle el encapsulamiento que realiza el protocolo WEP sobre la carga útil transportada por una trama 802.11 de datos (véase la **Figura 9**), que está contenida en el campo denominado *Frame Body* de dicha trama. Este encapsulamiento es la causa de que el tamaño de este campo, y por lo tanto también de la trama, aumente en ocho bytes. En primer lugar, la cabecera WEP contiene el campo *IV* de tres bytes, que como indica su nombre, contiene el vector de inicialización que se concatena con la clave secreta correspondiente para constituir la clave RC4 empleada para cifrar los datos contenidos en la trama. El siguiente campo (en realidad, en el estándar 802.11 se define como un subcampo del campo IV) ocupa un byte, aunque solo sus dos bits más significativos son útiles (los dos bits del campo que se transmiten en último lugar), puesto que codifican el identificador o índice de la clave WEP. Por esta razón, estos dos bits reciben el nombre de *Key ID*, mientras que los restantes bits del mismo campo se emplean como relleno y se fijan al valor cero.

A continuación, tras la cabecera WEP, viene la carga de datos que le corresponde transportar a la trama, después de haber sido encriptada mediante el algoritmo de cifrado RC4 a partir de la clave compuesta por el IV contenido en la trama y la clave WEP seleccionada. El último lugar, respecto a los campos que son procesados por el protocolo WEP, es ocupado por el campo *ICV*, el cual se concatena al final de los datos útiles y se cifra junto a tales datos, como si ambos constituyesen una única secuencia. Este último campo contiene una suma de comprobación basada en el algoritmo CRC-32, por lo que debe ser codificada mediante cuatro bytes, los cuales, sumados a los cuatro bytes que también requiere la cabecera WEP, suponen en total una sobrecarga de ocho bytes que son añadidos a la longitud de la trama por el protocolo WEP. Finalmente, conviene mencionar que todos los campos que introduce el protocolo WEP, en una trama o MPDU de datos, se ajustan al convenio ya descrito para los restantes campos de una trama 802.11, en cuanto al orden de los bytes y de los bits se refiere (esto es, son transmitidos en orden de significado creciente o *Little-Endian*, tanto los bytes de una trama, como los bits dentro de cada byte).

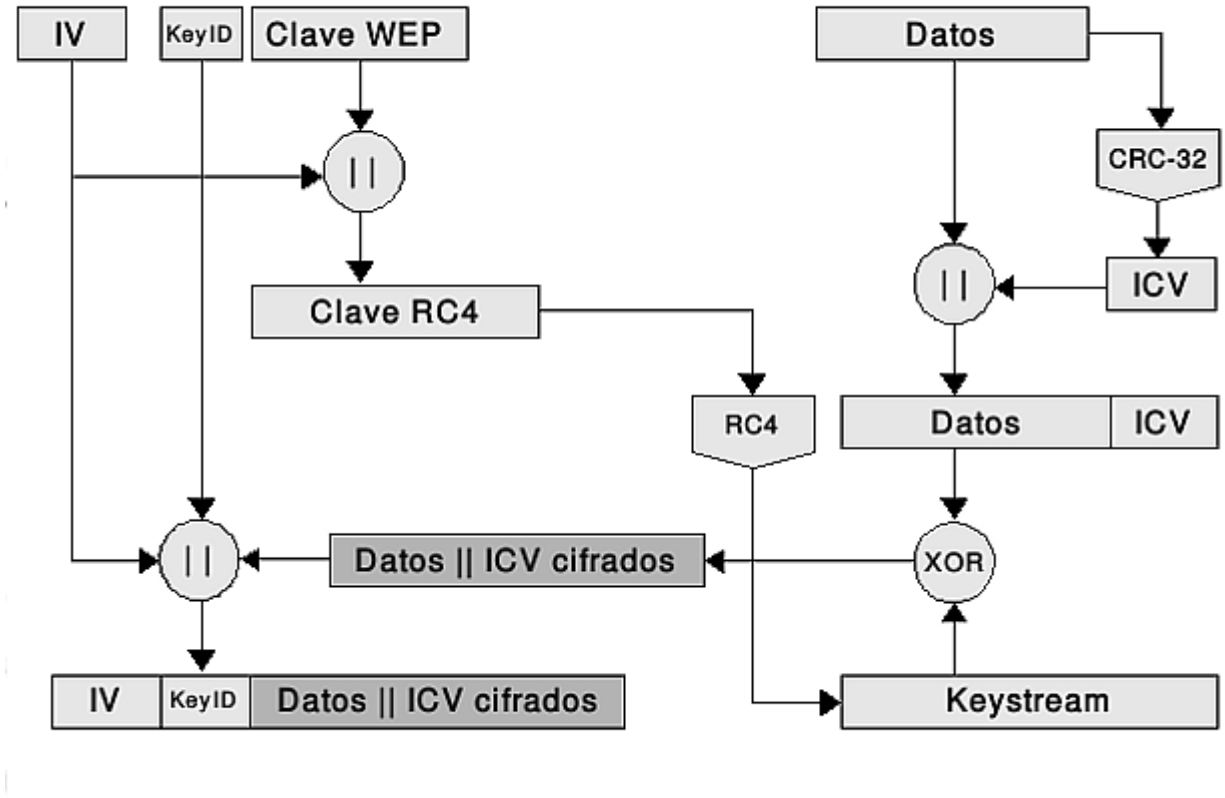

Figura 9: Encapsulamiento de los datos realizado por WEP

3.4 Principales debilidades en la seguridad

En los siguientes apartados se describen algunas de las debilidades más importantes que se han descubierto en los mecanismos de seguridad especificados en la versión inicial del estándar 802.11, los cuales, si excluimos la autenticación de sistema abierto, son todos provistos o están vinculados con el protocolo de seguridad WEP. También se hará mención a ciertos ataques bastante conocidos que aprovechan estas debilidades y se discutirán sus fundamentos, aunque para limitar la extensión de este trabajo no se expondrán muchos detalles sobre éstos, como los algoritmos, la demostración formal o la justificación, o todas las condiciones que deben satisfacerse para que puedan llevarse a cabo tales ataques.

3.4.1 Autenticación

Como ya se comentó, la autenticación de sistema abierto no involucra ningún mecanismo de autenticación, sino tan solo una petición realizada por la estación que solicita ser autenticada y la respuesta de la estación autenticadora. Esta respuesta será normalmente positiva aunque también podría ser denegada por causas ajenas a las credenciales de la estación solicitante. Por el contrario, la autenticación mediante clave compartida requiere la ejecución de un procedimiento sencillo pero muy poco seguro, a través del cual se intenta comprobar que la estación que solicita la autenticación posee una clave secreta que comparte con el punto de acceso o con otras estaciones de un IBSS.

No transcurrió mucho tiempo hasta que se puso de manifiesto la debilidad de la autenticación mediante clave compartida y se hizo público un ataque que superaba esta protección prescindiendo de la clave. Para el ataque descrito en [ASW2001] solamente es necesario que el atacante capture las tramas segunda y tercera de una instancia de este proceso de autenticación que haya finalizado con éxito, siendo entonces capaz de responder correctamente a un desafío propuesto por la estación autenticadora. Esto es debido a que, en tal caso, el atacante puede recuperar el keystream con el que se encriptó la tercera trama enviada durante una autenticación legítima (ya que dispone del texto en claro correspondiente, transmitido en la segunda trama). Desde ese instante, puede reutilizar dicho keystream (y su IV asociado) para cifrar el mensaje con el que la estación autenticadora lo desafía (puede escoger cualquier IV para efectuar tal cifrado y también calcular el ICV del mensaje).

3.4.2 Cifrado Vernam

Este tipo de cifrado está basado en la aplicación de la operación *xor* entre cada byte del texto en claro y su correspondiente byte de la secuencia de cifrado (la cual es denominada keystream). Por las propiedades de esta operación, si se aplica dos veces este cifrado sobre datos cualesquiera usando el mismo keystream, ambas operaciones de cifrado se cancelan, obteniéndose de nuevo los datos originales. Luego, si recuperamos un keystream de determinada longitud (y su IV asociado), estaremos capacitados para descifrar la misma cantidad de datos que han sido cifrados con el mismo keystream (por ejemplo, cuando distintos datos son cifrados y empaquetados con la misma clave WEP y el mismo IV, respectivamente). Adicionalmente, cualquier keystream válido nos permite encriptar datos arbitrarios con una longitud similar (para falsificar una trama protegida con WEP también hay que cifrar el ICV).

Ambos casos son claros ejemplos de ataques de reutilización del keystream, los cuales suponen la principal amenaza para esta clase de cifrado. Además, el protocolo WEP carece de protecciones efectivas contra tales ataques, como se explicará más adelante. Por otra parte, es posible obtener de forma directa un fragmento de keystream realizando la operación *xor* entre ciertos datos cifrados, que podemos extraer de una trama 802.11 protegida con WEP, y los correspondientes datos en claro. De hecho, en la mayoría de redes Wi-Fi, podemos suponer que la carga útil de una trama 802.11 de datos comienza con una cabecera LLC/SNAP, que precede a un paquete ARP o a un paquete IP. Distinguiendo ambos casos (por ejemplo, por medio del tamaño de la trama capturada), es posible deducir el valor de 8 a 16 bytes iniciales de los datos en claro. Refinando la técnica de detección para descubrir paquetes DHCP o diferenciar entre peticiones y respuestas ARP (todas las cuales tienen un tamaño fijo), podríamos deducir un prefijo aún mayor de los datos en claro.

De otro modo, si disponemos de información adicional sobre los servicios accesibles en la red protegida con WEP (por ejemplo: *telnet*, *ftp*, *http*, etc) o sobre los mensajes enviados o recibidos a través de ésta (por ejemplo, enviando *spam* a los usuarios de las estaciones desde una red externa), sería más fácil descubrir parte del texto en claro de algunos mensajes intercambiados en esa red. Incluso si no tenemos la certeza de ningún fragmento de texto en claro que se transmita en dicha red inalámbrica, pero disponemos de varios mensajes encriptados con el mismo keystream, entonces operando un par de ellos mediante la función *xor*, obtendremos el resultado de aplicar esta función a los datos en claro correspondientes a tales mensajes. Así pues, estos mensajes en claro combinados mediante la operación *xor* podrían ser vulnerables a ataques estadísticos basados en la frecuencia de aparición de los valores de los bytes, dependiendo de la naturaleza de estos mensajes (por ejemplo, cuando se trata del lenguaje natural, de código fuente, etc).

Alternativamente, existe otra clase de ataques para recuperar el keystream asociado a una trama 802.11 cifrada o para aumentar el tamaño de cierto prefijo de un keystream que se ha obtenido de alguna forma. No obstante, este tipo de ataques es posible gracias a ciertas debilidades que exhibe el mecanismo del protocolo WEP para verificar la integridad de las tramas, por lo que se describirán en el apartado siguiente. Con la intervención de alguno de los anteriores ataques, puede reunirse una colección de prefijos de cierto tamaño de keystreams diferentes, que se puede utilizar para construir un diccionario formado a partir de pares de IVs y sus correspondientes keystreams. De esta manera, si capturamos un paquete cifrado mediante un IV contenido en el diccionario, podemos descifrarlo total o parcialmente empleando el keystream asociado a dicho IV en el diccionario.

Otras propiedades que presenta la operación *xor*, como son la conmutatividad y la asociatividad, posibilitan realizar modificaciones sobre los datos cifrados (aplicándoles la operación *xor* con una secuencia determinada) que producen el mismo resultado que si se efectuasen tales modificaciones sobre los datos en claro, antes de ser cifrados. Por otro lado, la correspondencia unívoca entre cada byte del texto en claro y cada byte del texto cifrado habilita los ataques de fuerza bruta sobre bytes individuales del texto cifrado, suponiendo que conocemos los valores en claro de los restantes bytes y que contamos con algún procedimiento que permita determinar si el valor supuesto es el correcto.

3.4.3 Integridad

Anteriormente fue expuesta la forma en la cual se emplea el campo ICV para comprobar la integridad de una trama protegida con WEP y también se indicaron algunas carencias del cómputo, mediante la técnica del CRC-32, del valor contenido en dicho campo. Posiblemente, la debilidad más fácil de atacar del protocolo WEP sea la falta de autenticidad que conlleva este mecanismo, lo que permite a cualquier atacante calcular el valor del campo ICV a partir de datos arbitrarios. Por esta razón, si un atacante consigue un keystream suficientemente largo, podría cifrar una trama cualquiera y enviarla a una estación, sin que esta estación advierta que no es una trama legítima. Tanto el ataque de fragmentación de Bittau [BHL2006], como el ataque inductivo de texto en claro escogido de Arbaugh [Arb2001] se sirven de esta debilidad para aumentar el tamaño de un prefijo dado de un keystream.

El ataque de fragmentación requiere que alguna estación reensamble una secuencia de datos, enviados mediante sucesivos fragmentos encriptados con el mismo keystream, y que los retransmita por el medio inalámbrico tras ser reensamblados (por ejemplo, un punto de acceso suele hacer las dos cosas con las tramas de broadcast fragmentadas). Como el estándar 802.11 establece un límite máximo de 16 fragmentos para transmitir una MSDU, un atacante que obtenga un keystream de *N* bytes de longitud podría emplearlo para transmitir 16 fragmentos, cada uno con *N-4* bytes de datos útiles (descontando los 4 bytes del ICV), y capturar la trama retransmitida, una vez reensamblada, para obtener un keystream de longitud *16*(N-4)*, a partir de los datos cifrados de dicha trama y los datos en claro correspondientes a los fragmentos. De este modo, si el atacante cuenta con un prefijo de 8 bytes de un keystream y realiza el ataque tres veces seguidas con éxito, será capaz de recuperar un keystream que supere el tamaño máximo del cuerpo de una trama 802.11 (esto es, 2312 bytes, aunque en la práctica se impone la menor MTU de las redes Ethernet, que alcanza los 1500 bytes).

Por otro lado, el ataque inductivo de Arbaugh posibilita incrementar en un byte el prefijo de un keystream en cada paso del ataque. Así pues, un atacante que disponga de un keystream de *N* bytes puede seleccionar un mensaje en claro de *N-3* bytes, calcular su ICV y concatenarlo con el mensaje anterior, con lo que obtiene una secuencia de *N+1* bytes. Por tanto, el keystream disponible bastará para cifrar todos los bytes de dicha secuencia excepto el último. Para determinar el valor cifrado de este último byte, el atacante recurre a una búsqueda por fuerza bruta, esto es, selecciona un valor arbitrario para este byte y envía la trama con el mensaje resultante a alguna estación que muestre algún indicio en caso de que el mensaje sea válido, o bien cuando no lo sea, desde el punto de vista del procesamiento del mensaje mediante el protocolo WEP.

Por ejemplo, es posible enviar dicha trama a la dirección de difusión en un BSS que funcione en modo Infraestructura, con el propósito de que el punto de acceso la reciba, la descifre y verifique la suma de comprobación del campo ICV. Si el valor seleccionado para el último byte es el adecuado, el ICV será descifrado correctamente y se comprobará que es válido, lo que provocará que el punto de acceso (salvo que implemente restricciones adicionales a las definidas en el estándar) reenvíe la trama dentro del propio BSS. En consecuencia, si no hay errores de transmisión, será necesario enviar a lo sumo 256 tramas para encontrar el valor correcto de dicho byte. Repitiendo este proceso, podemos extender un prefijo de un keystream hasta que alcance el máximo tamaño posible de la carga de datos de una MPDU.

Otro inconveniente importante del cómputo mediante el algoritmo CRC-32 del valor presente en el campo ICV es su linealidad respecto al operador *xor*. Esta es la causa de que los datos y el ICV puedan ser modificados (incluso cuando están encriptados mediante cifrado Vernam) aplicándoles la operación *xor* con ciertas secuencias predefinidas, de manera que la estación receptora no pueda detectar tales alteraciones. Para conseguir esto, se calcula el valor del CRC-32 de la secuencia con la cual se combinan los datos vía operación *xor* y, a continuación, se aplica la misma operación al ICV junto con este valor calculado. Esta propiedad del CRC-32 es la que origina los ataques de tipo *bit-flip*, gracias a los cuales algunos bits de los datos cifrados pueden ser conmutados, de la misma forma que sucede con los bits correspondientes de los datos en claro al ser descifrados, siendo el receptor incapaz de advertir la inversión de estos bits mediante la suma de comprobación contenida en el campo ICV.

Este tipo de ataques puede conducir a la recuperación de los datos en claro de una trama cifrada con WEP, suponiendo que algunas estaciones tengan acceso a un host disponible en una red externa (por ejemplo, un computador conectado a Internet) y que este host se encuentre bajo el control del atacante (como es sugerido en [BGW2001]). En tal caso, si la trama transporta un paquete IP y el atacante puede predecir la dirección de destino del paquete (por ejemplo, descifrando un paquete previo perteneciente a una secuencia intercambiada entre los mismos hosts), solo tendría que alterar ciertos bits del cuerpo de la trama, correspondientes al campo que codifica la dirección IP adecuada, para cambiar el destino del paquete y dirigirlo hacia el host controlado por el atacante. Si todas las condiciones enunciadas se verifican, entonces el paquete debe llegar descifrado al host controlado por el atacante, puesto que el cifrado WEP solo se aplica al enlace inalámbrico entre dos estaciones.

Posteriormente, un hacker de identidad desconocida, que adoptó el apodo de *KoreK*, fue capaz de refinar aún más los anteriores ataques basados en el CRC-32. Este hacker consiguió modificar el ICV, aunque preservando su correcta verificación, gracias a la alteración no solo de ciertos bytes correspondientes a los datos, sino también del tamaño de la carga de datos, mediante el ataque al que llamó *Chop-Chop*. Este ataque permite descifrar una trama (y recuperar el keystream con el que fue cifrada), obviamente, sin la clave WEP, pero también sin ningún keystream previamente obtenido. Para esto, es necesario eliminar el último byte de la carga de datos encriptada y modificar los restantes datos en función del valor en claro de dicho byte que se ha supuesto. A continuación, se precisa enviar la trama modificada a cualquier estación que indique si el ICV de dicha trama es válido, en cuyo caso ha sido descifrado el valor de dicho byte (en otro caso, puede ser descubierto enviando, como mucho, 256 tramas distintas, siempre que no se produzcan errores de transmisión). Aplicando este procedimiento a los siguientes bytes de la carga de datos modificada (cuyo tamaño se reduce en un byte en cada etapa del ataque), éstos pueden ser descifrados en orden inverso.

Las transformaciones que concibió KoreK para modificar los datos están basadas en el método para calcular el CRC-32 mediante la división de polinomios. Probablemente, este hacker advirtió que una secuencia de datos, seguida por los datos que representan el valor de su CRC-32, produce un resto constante cuando su correspondiente polinomio es dividido por el polinomio generador del CRC-32. Por lo tanto, cuando se elimina el ultimo byte de la mencionada secuencia, su polinomio asociado ya no es congruente con el resto anterior. No obstante, como pudo comprobar el autor del ataque, es posible encontrar otro polinomio, que depende del valor del byte suprimido, tal que si se suma al polinomio que representa a los datos truncados, el resultado produce el resto característico de una secuencia *Datos-ICV* válida cuando se divide por el mismo polinomio generador.

3.4.4 Algoritmo RC4

Ron Rivest diseñó el algoritmo de cifrado en flujo denominado RC4 en 1987, desde entonces muchos investigadores han estudiado la seguridad de este algoritmo, en parte por sus numerosas implementaciones pero también por su sencillo diseño. Muchos de estos estudios analizaban estados internos del algoritmo o salidas producidas por el mismo que se apartaban de un comportamiento aleatorio y que, en ocasiones, consiguieron describir mediante correlaciones formalmente definidas entre las entradas, los estados internos o las salidas. En este sentido, existen estudios estadísticos sobre las secuencias generadas por RC4, que permitieron a varios autores elaborar algoritmos para diferenciar estas secuencias de bits de las que son realmente aleatorias, empleando para tal fin un keystream de suficiente longitud o numerosos keystreams generados a partir de claves distintas.

Un ejemplo puede hallarse en [FMS2002], artículo donde se citan dos autores que encontraron desviaciones estadísticas en los triples consistentes en las palabras producidas por RC4 en las etapas t y $t+1$, y el valor de t módulo N (siendo $N=256$ en el caso de WEP). Por medio de esta desviación, los autores pudieron distinguir una secuencia generada por RC4 de 2^{30} palabras de una secuencia aleatoria. Del mismo modo, otros dos autores del anterior artículo descubrieron una desviación más fácil de apreciar, que consiste en que la segunda palabra generada por RC4 toma el valor cero con el doble de la probabilidad esperada. Por tanto, con tan solo 2^8 keystreams generados por RC4 a partir de claves distintas y que no guardan ninguna relación entre sí, afirmaron ser capaces de diferenciar este algoritmo de un auténtico generador de secuencias aleatorias.

Debido a la gran difusión del artículo [FMS2001], ha adquirido mucha notoriedad la debilidad de la *invarianza*, a la que los autores le atribuyen la causa de una correlación probabilística entre un numeroso conjunto de claves, que llamaron claves débiles, y la permutación final configurada por la inicialización del algoritmo RC4 (KSA). De esta forma, una pequeña cantidad de bits de una clave débil (los menos significativos de cada byte de la clave) influye, con determinada probabilidad, sobre un número mayor de bits de las palabras de esta permutación. Además, cuando estos patrones de bits se manifiestan en la permutación anterior, existe cierta probabilidad de que se repitan en las palabras iniciales (en lo que respecta a WEP, cada palabra contiene un solo byte) de un keystream generado por RC4. Basándose en esta correlación, los autores idearon un método para diferenciar los keystreams generados por RC4 de las secuencias producidas por un generador de bits aleatorio, para lo cual necesitaban aproximadamente 2^{21} keystreams diferentes producidos por RC4 a partir de claves aleatorias de 64 bits.

A pesar de lo interesantes que puedan resultar todas estas particularidades del algoritmo RC4 no parecen tener mucha utilidad práctica a la hora de sobrepasar los procedimientos de seguridad que utiliza WEP. Sin embargo, en uno de los artículos mencionados ([FMS2001]), los autores abrieron la puerta a los ataques de criptoanálisis contra una clave WEP. En dicho artículo, identificaron una clase de vectores de inicialización, a los que denominaron *IVs débiles*, que verifican una condición llamada *condición resuelta*. Para este conjunto de IVs, descubrieron una correlación probabilística entre ciertos estados de la fase de inicialización de RC4, la primera palabra del keystream generado por este algoritmo y una palabra específica de la clave WEP (que depende del IV concreto), aunque tal correlación puede generalizarse a otros prefijos conocidos, de mayor longitud, de una clave RC4. Por lo tanto, dado un IV débil y aplicando esta correlación, puede pronosticarse el valor de un byte determinado de la clave WEP con una probabilidad que estimaron cercana al 5%.

Aunque, en principio, esta debilidad no parece muy grave, reuniendo suficientes IVs débiles que ataquen el mismo byte de la clave, debería observarse una desviación estadística de los pronósticos hacia el valor auténtico de dicho byte de la clave. Por esta razón, el ataque ideado por estos autores, que fue denominado con sus iniciales (FMS), se define como un ataque estadístico contra la clave WEP. En cambio, en su mencionado artículo lo califican como un ataque de claves relacionadas (*related-key attack*), puesto que requiere los keystreams (en realidad, tan solo la palabra inicial de éstos) generados por claves RC4 distintas pero que guarden cierta relación entre ellas (en este caso, tienen en común la clave WEP). No obstante, hay que advertir que dos requisitos indispensables para la realización del ataque, como son que el atacante conozca cierto prefijo de las claves RC4 y las palabras iniciales de los keystreams correspondientes, son ajenos al cómputo efectuado por RC4, aunque el ataque se trata en este apartado por estar basado en el criptoanálisis de este algoritmo.

Adicionalmente, en los apéndices del citado artículo, incluyeron una estimación del número de paquetes distintos (y con diferentes IVs) que deberían capturarse para que el ataque tuviera éxito contra una clave WEP (recordemos que el tamaño de esta clave es de 40 bits y que los IVs que la preceden aportan los restantes 24 bits de la clave RC4), oscilando este número entre 1 y 4 millones. Sin embargo, conviene aclarar que en esta estimación solo tuvieron en cuenta los IVs débiles que se ajustaban al patrón: $<I, 255, K>$ (siendo $3 <= I < 8$, el índice de un byte de la clave WEP y $0 <= K < 256$, un byte cualquiera) y además asumieron que los IVs eran generados de forma secuencial en orden *Little-Endian* o *Big-Endian*.

Al poco tiempo de publicarse este ataque, se confirma su efectividad, en la práctica, mediante diferentes implementaciones. Posiblemente, la primera implementación fue realizada por los autores del artículo [IRS2001], en el cual alardeaban de haberla completado en pocas horas. En la misma, los implementadores tuvieron en cuenta todos los IVs que satisfacían alguna condición resuelta para algún byte de la clave, haciendo referencia aquí el término IV a cualquier prefijo de la clave RC4 conocido o pronosticado. Como quiera que el acierto en el pronóstico de los primeros bytes de la clave WEP permitía descubrir nuevos IVs débiles que, potencialmente, podían revelar los bytes posteriores de la clave WEP con mayor probabilidad que los IVs de menor tamaño, en primer lugar examinaron y ordenaron (en función de su probabilidad de acierto) los valores candidatos para el primer byte de la clave WEP, empleando todos los IVs disponibles para esto. Después utilizarían una búsqueda en profundidad para comprobar los valores candidatos de los restantes bytes de la clave, dando mayor prioridad al valor más probable para cada byte de la clave.

Para estimar la probabilidad de acierto de cada candidato (es decir, cada valor pronosticado para un byte de la clave WEP por un IV débil apropiado), tuvieron en cuenta el número de pronósticos a favor del mismo, pero también si dicho valor pertenecía a cierta clase de caracteres del código ASCII, ya que muchos dispositivos inalámbricos generaban la clave WEP por medio de este tipo de codificación, a partir de una contraseña introducida por el usuario y restringida a un subconjunto de caracteres de este código. Por último, también otorgarían un mayor peso a los valores pronosticados por los IVs débiles correspondientes a *condiciones resueltas especiales*, ya que, según *Adi Shamir* (presunto descubridor de estos IVs), la probabilidad de acierto de esta clase de IVs aumentaba aproximadamente hasta el 13% (por lo que se conocerían más tarde como IVs de alta probabilidad). Con estas mejoras, la implementación demostró una alta efectividad para recuperar una clave WEP cuando se acumulaba una cantidad de paquetes (con diferentes IVs) comprendida entre 500.000 y 1.000.000 (el doble para claves de 104 bits).

Aunque los autores de la anterior implementación decidieron no publicar el código de la misma, en el año siguiente ya estaban disponibles públicamente varias aplicaciones que implementaban el ataque FMS. En su artículo [Hul2002], David Hulton (apodado *h1kari*) reconocía la autoría de una de estas implementaciones, a la vez que hacía públicos los resultados de sus indagaciones sobre esta materia. Posiblemente, la principal novedad que expuso en su artículo fue que consiguió modificar la correlación ya comentada, en la que se basa el ataque FMS, para involucrar al segundo byte del keystream, en vez del primero.

Por desgracia, los IVs que posibilitaban esta nueva correlación eran muy escasos y no ofrecían muchas posibilidades de acertar en el pronóstico (la probabilidad de acierto era menor respecto a los IVs débiles anteriores, aproximadamente de un 2%). No obstante, algunos de los nuevos IVs débiles conducían a ciertos estados del algoritmo RC4, en las primeras etapas de la fase de inicialización, que permitían realizar un pronóstico sobre la palabra inicial del keystream generado por este mismo algoritmo. En este caso, si la predicción anterior se verificaba (se supone que el atacante podía recuperar el prefijo de un keystream generado a partir de un IV cualquiera), la correlación respecto a la segunda palabra del keystream ganaba evidencias a su favor y los IVs débiles referidos pasaban a ser considerados como IVs de alta probabilidad (esto es, aquellos IVs cuya probabilidad de acierto está próxima al 13%).

Por otro lado, Hulton observó que la búsqueda exhaustiva de IVs débiles exigía una cantidad de procesamiento y un tiempo de cómputo considerables para la mayoría de computadores de aquella época. Por esta razón, recopiló una serie de patrones de IVs, que expresó de forma algorítmica y que caracterizaban a una buena parte de los IVs débiles existentes (entendiendo por IV cualquier prefijo de una clave RC4). De esta forma consiguió acelerar el procedimiento para discernir si un determinado IV era apropiado para atacar cierto byte de la clave WEP. Además, calculó diversas probabilidades y compiló algunas estadísticas relacionadas con los IVs débiles que se ajustaban a esos patrones. También propuso reducir la amplitud de la búsqueda (que fue llamada *fudge*, esto es, el número de candidatos considerados para cada byte de la clave) para los últimos bytes de la clave. Para contrastar los hallazgos o las mejoras citados, implementó el ataque en cuestión y, después de comprobar su rendimiento, difundió su testimonio sobre la ruptura de claves WEP con un número de paquetes que oscilaba entre 500.000 y 2.000.000, invirtiendo para tal propósito un tiempo de procesamiento, en ocasiones, inferior a un minuto.

Entrado el año 2004, el hacker que se dio a conocer bajo el seudónimo de KoreK sorprendió a los usuarios de cierto foro de Internet dedicado a una aplicación para la detección de redes Wi-Fi, enviando una implementación de 17 ataques de criptoanálisis contra una clave WEP, influenciados por la técnica esencial del ataque FMS. Aunque un par de estos ataques eran versiones casi idénticas del ataque FMS y del ataque de Hulton (sobre el segundo byte del keystream), y casi la totalidad de los restantes ataques estaban basados en los mismos fundamentos que los dos anteriores, destapó un abundante grupo de ataques de criptoanálisis sobre el algoritmo RC4 que, incluso en la actualidad, parece no haberse agotado (según se afirma en [Cha2006]). En líneas generales, cada uno de estos 17 ataques contra la clave WEP se puede clasificar en una de las tres siguientes categorías:

- La primera categoría se compone de ocho ataques que dependen únicamente de la primera palabra (esto es, el byte inicial) del keystream, entre los cuales se incluye una variante del ataque FMS.
- La segunda categoría aglutina el mismo número de ataques que la primera, aunque dichos ataques recurren principalmente a la segunda palabra del keystream, tal y como sucede con el ataque de Hulton.
- La tercera categoría incluye tan solo un ataque, aunque este ataque contempla cuatro casos distintos, tales que si se verifica alguno de ellos entonces pueden descartarse, con una alta probabilidad de acierto, determinados valores para ciertas palabras de la clave. Por lo tanto, el propósito de este ataque consiste en reducir el número de candidatos, o bien el espacio de búsqueda de los valores, que pueden presentar ciertas palabras de la clave WEP.

No obstante, la mayoría de los ataques de las dos primeras clases se diferencian de los ataques predecesores (esto es, FMS y Hulton) en la mayor complejidad de las condiciones iniciales que se evalúan para determinar la viabilidad de los ataques más recientes. En general, esto es debido a que los ataques de KoreK dependen de un mayor número de cláusulas que las exigidas por la *condición resuelta* o por la precondición para el ataque de Hulton. Por ejemplo, frecuentemente se incluyen cláusulas en las que intervienen elementos o índices de elementos concretos del vector de estado de RC4. En ocasiones, también presentan cláusulas que afectan a posiciones concretas del vector de estado o de la clave RC4 (algunos ataques solo afectan a determinadas palabras de la clave).

Para complicar más las cosas, algunos ataques presuponen el cumplimiento de ciertas clausulas cuando se observan ciertos valores en determinadas palabras del keystream generado por la víctima, aunque no puede comprobarse la veracidad de estas suposiciones a partir del fragmento de la clave RC4 disponible (por tanto, no parece lógico estimar la probabilidad de éxito de uno de estos ataques de la misma forma que se hizo para el ataque FMS). Finalmente, debido a la manera tan informal con la que fueron presentados los ataques de KoreK, no existen demasiados estudios al respecto, aunque han sido analizados en [Cha2006], mientras que en [Tew2007] se estima que son necesarios 700.000 paquetes, con distintos IVs, para que la probabilidad de recuperar una clave WEP de 104 bits supere el 50%. En cambio, una clave WEP de 40 bits puede romperse con aproximadamente 300.000 paquetes.

Actualmente, el ataque más eficiente contra una clave WEP, así como el que requiere una menor cantidad de paquetes, fue denominado mediante las iniciales de los autores del artículo [PTW2007]. No obstante, este ataque está basado en una propiedad del vector de estado del algoritmo RC4 que fue descubierta por R. J. Jenkins. Posteriormente, esta propiedad sería demostrada y generalizada a cualquier permutación aleatoria de enteros entre *0* y *N-1* por A. Klein. Gracias a esta propiedad, Klein pudo advertir la existencia de una correlación probabilística entre el vector de estado de RC4 en la iteración *i-ésima* de su inicialización, la *i-ésima* palabra de la clave RC4 y la *i-ésima* palabra del keystream generado por RC4 a partir de dicha clave. Aunque la probabilidad de pronosticar el valor auténtico de una palabra de una clave WEP, por medio de esta correlación, es notablemente inferior respecto a la del ataque FMS (aproximadamente del 0,5%), esta correlación presenta una ventaja crucial con respecto a la del anterior ataque y es que cada IV es susceptible de ser utilizado para dicho pronóstico. Por tanto, desde el punto de vista del ataque PTW, cualquier IV se considera *"débil"* (para el ataque FMS solo hay 768 IVs débiles para el primer byte de la clave WEP).

Además de describir con mayor detalle el ataque de Klein sobre una clave WEP y de expresar formalmente la correlación que explota dicho ataque, los autores del referido artículo introdujeron una optimización para mejorar su eficiencia y propusieron algunos métodos para llevarlo a cabo en la práctica, incluyendo técnicas para recuperar prefijos de keystreams de diversos tamaños, estrategias de búsqueda y de selección de los valores candidatos para las palabras de la clave, etc. Mediante la optimización que aplicaron, el ataque PTW podía realizar la búsqueda de cada palabra de la clave de forma independiente y en paralelo, en vez de secuencialmente, como era habitual hasta entonces. No obstante, esta optimización acarreaba el inconveniente de reducir la probabilidad de acierto (así que, en general, es necesaria una mayor cantidad de paquetes para obtener la clave), aunque normalmente es preferible esta desventaja a la computación y el tiempo desperdiciados en la búsqueda de la clave al seleccionar un valor incorrecto para alguna palabra inicial de la clave.

La efectividad del ataque PTW resulta evidente si prestamos atención a la cantidad de paquetes necesaria para romper la clave, puesto que ésta se reduce en, al menos, un orden de magnitud con respecto a los ataques previos. En su trabajo [Tew2007], Erik Tews estima que, obteniendo 35.000 paquetes con distintos IVs, el ataque PTW tiene una probabilidad del 50% de romper una clave WEP de 104 bits, mientras que esta probabilidad asciende al 85% si tal cantidad alcanza los 45.000 paquetes. En cambio, para una clave WEP de 40 bits (tamaño especificado en el estándar 802.11), muchos expertos y aficionados con experiencia en romper claves de este tipo, coinciden en que un número de paquetes entre 10.000 y 20.000 es suficiente en la mayoría de los casos.

3.4.5 Protocolo WEP

Algunas debilidades del protocolo WEP no se pueden atribuir exclusivamente a las primitivas de seguridad empleadas por éste, como el algoritmo de cifrado, la función escogida para verificar la integridad o el procedimiento de autenticación. En cambio, la elección de determinados parámetros de operación o incluso la forma en la que se utilizan o interactúan estas primitivas pueden contribuir notablemente a la aparición o la explotación de estas debilidades. Un caso ejemplar es el tamaño de la clave WEP, ya que al limitarse su tamaño a 40 bits (probablemente debido a presiones para que cumpliese con la normativa de exportación de EEUU vigente en ese momento) resulta vulnerable a ataques de fuerza bruta, que incluso pueden llevarse a cabo en equipos convencionales. El experto en seguridad Tim Newsham anunció que pudo romper mediante fuerza bruta una clave WEP en 210 días con su portátil de 500 MHz (de manera que podría tardar semanas con un equipo actualizado, al alcance de cualquier consumidor hoy en día).

En ocasiones, una clave WEP es generada a través de un método más restrictivo, por ejemplo, empleando directamente los códigos ASCII de una contraseña de cinco caracteres o utilizando un generador de claves, a partir de una contraseña en ASCII, con una entropía reducida (Newsham comprobó que algunos generadores apenas alcanzaban una entropía de 21 bits). En tales casos, el espacio de búsqueda se reduce drásticamente, de forma que puede explorarse en cuestión de días o incluso, en el mejor caso, en tan solo unos pocos segundos. Por el contrario, para una clave WEP de 104 bits, incluso cuando es generada por medio de alguno de estos métodos, el ataque por fuerza bruta aún resulta impracticable, aunque un ataque de diccionario podría ser efectivo contra claves de ambos tamaños, suponiendo que la contraseña escogida fuese poco original (palabras en cierto idioma, números, etc).

Por otro lado, el cifrado mediante una clave compuesta de un IV público y variable que precede a una clave secreta fija posibilita los ataques de claves relacionadas. Todos los ataques contra la clave WEP mencionados en este trabajo (que son todos los que conoce el autor del mismo) son de este tipo y están basados en correlaciones que dependen del conocimiento de un prefijo de la clave RC4 (no obstante, también se han descubierto ataques contra la clave en el caso hipotético de que ésta precediese al IV, aunque son bastante menos efectivos y suelen tener restricciones adicionales, como que son aplicables solamente a ciertas clases de claves o bien que el atacante debe conseguir o suponer correctamente cierta información secreta, etc).

Aparte de los ataques contra la clave WEP, los IVs favorecen otros tipos de ataques a causa de su reducido tamaño (24 bits) y de que se transmiten en claro junto a los datos en cuyo cifrado intervienen. Por lo tanto, en una red con tráfico intenso, el espacio de IVs puede agotarse en unas cuantas horas, facilitando a un atacante la construcción de un diccionario que asocie a determinados IVs sus correspondientes keystreams (siempre que sea posible recuperar los keystreams a través del texto en claro conocido o por otros medios). De esta forma, el atacante podría descifrar aquellos paquetes cuyos IVs estuviesen contenidos en el diccionario, suponiendo que el tamaño de la carga de datos de estos paquetes no fuese superior a la longitud de los keystreams almacenados en dicho diccionario y que se corresponden con los mismos IVs.

Todavía tiene peores consecuencias el hecho de que el protocolo WEP no establezca ninguna restricción sobre cómo deben ser generados los IVs. Aunque el estándar IEEE 802.11 recomienda que se cambien en cada trama transmitida por el mismo emisor (sin embargo, en una secuencia de tramas suficientemente larga no podrá evitarse la repetición de algún IV), no prohíbe a una estación reutilizar un IV cualquiera con la misma clave WEP. Por esta razón, un atacante puede reenviar una trama capturada, consiguiendo que sea procesada de nuevo por el receptor, o bien puede utilizar un IV y su keystream asociado, que haya obtenido de algún modo, para cifrar y enviar a la víctima todas las tramas que le plazca, mientras ésta utilice la misma clave WEP.

Finalmente, otra limitación importante del protocolo WEP es que no aborda la gestión de claves, de modo que operaciones como la generación, la distribución o la renovación de claves, en general, debe realizarlas manualmente el administrador de la red. En una red de tamaño mediano o grande, estas tareas pueden requerir un tiempo, un esfuerzo o un coste elevados, así que su ejecución puede restringirse por debajo de la periodicidad adecuada. Por otro lado, la actitud de algunos usuarios de redes Wi-Fi domésticas, que no están preocupados por la administración de su red o que ignoran cómo realizar cualquier tarea relacionada con esta cuestión, es muy común.

Por razones como éstas, podemos encontrar muchas redes Wi-Fi en las que tan solo se configura una clave WEP para todos los dispositivos inalámbricos asociados a dicha red, siendo además dicha clave reemplazada de forma muy esporádica. Una desventaja obvia de esta configuración es que la privacidad de las comunicaciones disminuye con la difusión de la clave, puesto que cada uno de los usuarios que comparten una clave WEP puede descifrar el tráfico de los otros. Además, cuanto más tiempo se conserve y se utilice una clave secreta compartida, mayores serán las posibilidades de que sea descubierta.

3.4.6 Alternativas para mejorar la seguridad de WEP

A medida que se iban descubriendo las numerosas debilidades del protocolo WEP, los expertos comenzaron a proponer soluciones para solventar o, al menos, para paliar una alta proporción de estas debilidades. Algunas de estas medidas serían incorporadas posteriormente en las versiones más recientes del estándar IEEE 802.11 o en sus sucesivas enmiendas. Al mismo tiempo, algunos fabricantes de dispositivos inalámbricos implementaron sus propias soluciones para mitigar estas debilidades, independientemente de lo establecido en las especificaciones del estándar. Así ocurrió con la solución denominada *WEP Plus*, nombre comercial de una técnica implementada por varios fabricantes (y por algunos sistemas operativos de fuentes abiertas, como los citados en [Tew2007]), cuyo objetivo consiste en filtrar los IVs empleados para encriptar las comunicaciones, descartando determinados IVs débiles utilizados en ataques contra la clave WEP. Sin embargo, la efectividad de las implementaciones es bastante limitada, puesto que no resulta práctico filtrar todos los posibles IVs débiles, ni siquiera para el ataque FMS. Por lo tanto, esta técnica ofrece muy poca protección contra los ataques de KoreK y ninguna contra el ataque PTW.

Otra solución, con escasa aceptación por parte de la industria, fue propuesta por el grupo de trabajo 802.11 y recibió el nombre de *WEP2* (aunque finalmente no se incorporó en la enmienda 802.11i). Entre las novedades que introdujo estaba la extensión de la clave y también del IV hasta los 128 bits. Con este tamaño de la clave y del IV, los ataques de fuerza bruta y de reutilización del keystream (con objeto de descifrar tramas), respectivamente, contaban con unas posibilidades de éxito despreciables. Sin embargo, es muy probable que los miembros del correspondiente comité del IEEE no advirtieran que, mediante esta composición de la clave RC4, los ataques como el FMS (según se afirma en [FMS2002]) aumentaban su eficacia, ya que es posible descubrir un estado más tardío de la inicialización del algoritmo RC4 gracias a que el IV conocido es de mayor tamaño.

Posiblemente, la solución más segura, anterior a la implementación de los estándares WPA y WPA2, fue denominada *WEP dinámico* y consistía esencialmente en la renovación automática y periódica de las claves WEP (que en este caso eran específicas para cada sesión y cada usuario). Este procedimiento generalmente se efectuaba por medio de los mecanismos especificados en el estándar IEEE 802.1X y en las diversas variantes del protocolo EAP. No obstante, en muchos casos las implementaciones no ofrecían la seguridad esperada, puesto que realizaban la renovación de las claves de manera eventual, por ejemplo, tras la (re)asociación de la estación al punto de acceso o durante la itinerancia.

De cualquier forma, una red Wi-Fi que genere tráfico, principalmente tramas 802.11 de datos, con la suficiente intensidad, facilitará la ruptura de una clave WEP de sesión en muy pocos minutos (o incluso segundos, como sugiere el título del artículo [PTW2007]). Por otra parte, los ataques de reutilización del keystream todavía resultan efectivos durante el tiempo de vida de una clave de sesión que, como pudieron comprobar los autores del artículo [BHL2006], suele ser suficiente para efectuar uno de estos ataques con cierta holgura. Además, en [SIR2002] proponen renovar la clave WEP antes de agotar el espacio de IVs disponible, evitando así el descifrado de tramas encriptadas mediante el mismo keystream.

En cuanto a la integridad, prácticamente todos los expertos coincidían en que el valor del campo ICV calculado mediante el CRC-32 debía ser reemplazado por el valor computado por una función hash criptográfica que permitiese detectar cualquier modificación de los datos cifrados realizada por un atacante (que no conozca el mensaje en claro asociado a estos datos). Además, muchos de estos expertos aconsejaban que dicha función debía generar también un MAC (*Message Authentication Code*), mediante el cual es posible verificar la autenticidad de un mensaje y rechazar los ataques de texto en claro escogido. Por desgracia, la complejidad de estas funciones criptográficas estaba fuera del alcance de muchos de los dispositivos legados, que únicamente implementaban los mecanismos de seguridad requeridos por el protocolo WEP. Por este motivo, los autores de [SIR2002] sugirieron la generación de un MAC a partir de una función hash universal basada en clave, de modo que esta función pudiera ser implementada por el firmware de tales dispositivos, previa actualización del mismo, y ofreciese un nivel de seguridad adecuado bajo un cifrado seguro.

En dicho artículo, los autores también afirman que la autenticación mediante clave compartida, vinculada al protocolo WEP, es suficientemente segura si se garantiza la privacidad y la integridad de los datos. Así mismo, se expone la necesidad de proteger la negociación de estos mecanismos frente a ataques del intermediario (traducción de la expresión en inglés: *man-in-the-middle attacks*), mediante los cuales podrían suplantarse a las dos partes e inducirlas a escoger alternativas menos seguras (probablemente, esto se evitaría con un método de autenticación fuerte y bilateral). A pesar de todo, en los estándares WPA, WPA2 y en algunas soluciones propietarias anteriores, como en el caso de WEP dinámico, se ha apostado por el binomio 802.1X/EAP para aumentar la seguridad de la autenticación, así como para desempeñar otras tareas complementarias, como la generación y la distribución de claves.

Capítulo 4

4. Seguridad avanzada en redes Wi-Fi

Desde el año 2001 la falta de seguridad del protocolo WEP queda patente, descubriéndose en los años siguientes nuevas manifestaciones de esta inseguridad, así como diversos métodos de ataque que las explotan. Sin embargo, en ese mismo año, el grupo de tareas *TGi* (*Task Group i*) del IEEE comienza a indagar nuevas técnicas que mejoren la seguridad de las redes 802.11, tras obtener la autorización para acometer el correspondiente proyecto después de un breve lapso de tiempo. Algunas de estas técnicas serían incorporadas más tarde en la enmienda *802.11i*, cuya aprobación se produjo en Junio del 2004. No obstante, algunos fabricantes de dispositivos Wi-Fi no estaban dispuestos a esperar, hasta la incierta fecha de aprobación de la enmienda, para proporcionar a sus dispositivos medidas de seguridad más efectivas.

Ante esta coyuntura, la *Wi-Fi Alliance* precipitó la estandarización de una solución de seguridad en Octubre del año 2002, que recibió el nombre comercial de *WPA* (*Wi-Fi Protected Access*). Aunque se trataba de una solución interina de esta organización (no estandarizada por el IEEE), estaba basada en la versión 3.0 del borrador de la enmienda 802.11i e incorporó un subconjunto de las características de esta enmienda. No obstante, el grupo TGi del IEEE se esforzó posteriormente por mantener la compatibilidad hacia atrás con WPA (sin embargo, existen pequeñas diferencias en las especificaciones y en la implementación de algunas características). Por otra parte, las nuevas protecciones de seguridad introducidas por WPA no añadían demasiada complejidad técnica con respecto a WEP, por lo que en muchos casos era posible la actualización del firmware de los dispositivos legados que implementaban WEP para reemplazar este protocolo por WPA.

Varios meses después de la aprobación de la enmienda 802.11i, la *Wi-Fi Alliance* anunció una nueva certificación basada en dicha enmienda, a la que denominaron *WPA2*. A partir del año 2006, la certificación base de los productos Wi-Fi (que deben superar todos los productos para recibir la denominación comercial Wi-Fi) incluye la certificación WPA2. Los limitados recursos de los dispositivos sobre los que debía funcionar WPA y la temprana fase en la que se encontraba el desarrollo de la enmienda 802.11i, cuando se elaboró el estándar del protocolo WPA, propiciaron

que determinadas características de la enmienda 802.11i fuesen descartadas en las especificaciones de WPA, o bien que no hubiesen sido concebidas durante la elaboración del mencionado estándar. Entre las principales características que diferencian a los estándares WPA y WPA2 podemos destacar las siguientes:

- Cifrado y protección de la integridad y de la autenticidad mediante CCMP. En el momento de la publicación del estándar sobre WPA no se había especificado este mecanismo en su totalidad, el cual exige, además, una capacidad de cómputo bastante superior a la de WEP.
- Preautenticación y almacenamiento en cache de una PMKSA. Se trata de técnicas que permiten reducir el tiempo invertido en la itinerancia, ya que puede ser significativo para determinadas aplicaciones (por ejemplo, de tiempo real, multimedia, etc), especialmente cuando se utilizan ciertos procedimientos de autenticación basados en el estándar 802.1X.
- Seguridad robusta entre estaciones (que no son puntos de acceso). Los nuevos mecanismos de seguridad que introdujo WPA protegían la comunicación entre una estación y el punto de acceso asociado, pero no la comunicación directa entre estaciones (por ejemplo, la que tiene lugar en un IBSS).

El resto de este capítulo estará enfocado en las especificaciones de la enmienda 802.11i, debido a que ésta reemplaza y amplía el estándar WPA, no obstante el término WPA2 se utilizará indistintamente para referirse a los procedimientos de seguridad introducidos en dicha enmienda. Un concepto importante en esta enmienda es el de *asociación de seguridad*, que es definido como un conjunto de políticas, claves y otros datos (como contadores, espacios de secuencias, etc) compartidos y mantenidos por dos estaciones de un mismo BSS, con el propósito de intercambiar información de forma segura.

Una asociación de seguridad puede establecerse a través de un intercambio de tramas, mediante el cual son autenticadas las estaciones, se negocian determinados parámetros de seguridad o se comprueba que ambas estaciones comparten cierta información secreta. Alternativamente, también se puede crear una asociación de seguridad configurando la misma clave secreta (o bien la misma contraseña) en las dos estaciones. En cualquier caso, en la enmienda 802.11i se distinguen dos clases fundamentales de asociaciones de seguridad, que se describen a continuación:

- **Pre-RSNA**, que hace referencia a una asociación que utiliza mecanismos de seguridad especificados (en alguna versión antigua del estándar 802.11) antes de la introducción de las redes de seguridad robusta (esto es, redes 802.11 que se ajustan a la definición de una *Robust Security Network*, tal y como se describe en la enmienda 802.11i), como son, por ejemplo: la autenticación mediante clave compartida o el cifrado y la protección de la integridad mediante WEP.
- **RSNA**, que denota una asociación de seguridad basada en los mecanismos introducidos en la enmienda 802.11i. Por lo tanto, habilita el cifrado, así como la comprobación de la integridad y la autenticidad mediante los protocolos TKIP o CCMP, o bien permite derivar las claves, además de cualquier otro parámetro que sea necesario, para la operación de alguno de estos protocolos. La mayor parte de las asociaciones de seguridad de esta clase son establecidas mediante la autenticación 802.1X, una clave o contraseña predefinida y compartida entre dos estaciones o el *4-Way Handshake*.

A partir de estos dos tipos de asociaciones podemos diferenciar dos clases de redes 802.11, atendiendo a los mecanismos de seguridad que utilizan las estaciones que las componen. Así pues, una red de seguridad robusta (*Robust Security Network,* abrev. *RSN*) solo permite la creación de asociaciones de seguridad robusta (abrev. *RSNAs*), mientras que una red de seguridad transicional (*Transition Security Network,* abrev. *TSN*) permite ambos tipos de asociaciones (esto es, tanto *pre-RSNAs*, como *RSNAs*). En el resto de este capítulo únicamente serán aludidas las RSNs, puesto que su implementación debe limitarse a las técnicas de seguridad introducidas en la enmienda 802.11i.

Una parte considerable de este capítulo se dedicará a describir las características esenciales de aquellas técnicas que sustentan la privacidad, la integridad y la autenticidad (del origen de los datos) de las comunicaciones de datos en redes inalámbricas de seguridad robusta. Sin embargo, debido a la estrecha relación que existe entre los mecanismos que proporcionan estas características de seguridad, se estudiarán conjuntamente en este capítulo, dentro del contexto de cada uno de los protocolos de seguridad introducidos en la enmienda 802.11i, los cuales son denominados *TKIP* y CCMP, respectivamente.

Por otro lado, la autenticación de usuarios o dispositivos se tratará en un apartado previo de este capítulo, debido a su independencia respecto a los anteriores protocolos de seguridad (aunque mantienen cierta relación a través de los procedimientos de generación y renovación de claves). No obstante, antes de adentrarnos en la materia principal de este capítulo, se describirán las fases fundamentales que podemos distinguir en la operación de una estación perteneciente a una RSN (ejemplificadas en el caso específico de una red en modo Infraestructura).

4.1 Fases de operación en una RSN

La interacción entre una estación y un punto de acceso vinculados al mismo BSS, que forma parte o constituye por sí mismo una RSN, consta típicamente de las siguientes etapas:

- *Descubrimiento*, que se realiza mediante las tramas de *Beacon* difundidas por un punto de acceso o las tramas de tipo *Probe Response* con las que éste responde a un sondeo iniciado por una estación. Además, mediante esta clase de tramas se advierten los mecanismos de seguridad soportados por el punto de acceso (a través del campo denominado *RSN Information Element*), incluyendo la suite de cifrado para el tráfico de broadcast/multicast (por ejemplo, el protocolo responsable de la confidencialidad y la autenticidad de los datos, junto a cualquier otro parámetro requerido para su operación), así como una o más suites de cifrado disponibles para el tráfico unicast y también los procedimientos soportados para la autenticación de estaciones y la gestión de claves. Una vez que ha detectado una red, una estación puede unirse al BSS correspondiente iniciando la autenticación de sistema abierto (por compatibilidad con versiones previas del estándar 802.11). A continuación, para completar la asociación, debe realizar la negociación con el punto de acceso de los mecanismos de seguridad y conseguir que concluya con éxito (debe seleccionar una suite de cifrado unicast y un método de autenticación de los que éste oferta)
- *Autenticación*, incluso cuando un punto de acceso de una RSN acepta la asociación de una estación, éste restringirá el acceso de dicha estación a la red hasta que se lleve a cabo la autenticación mutua entre ambos dispositivos. De esta manera, puede evitarse el acceso de estaciones no autorizadas a la red y también que un punto de acceso no autorizado pueda suplantar a uno legítimo para obtener información privada sobre una estación autorizada. En consecuencia, la estación y el punto de acceso (o una entidad que preste el servicio de

autenticación en la red) deben intercambiar información con el objetivo de demostrar sus credenciales a la otra parte. Para este fin, ambas estaciones deberían acordar previamente algún procedimiento que facilitase este intercambio y que protegiese los datos sensibles involucrados en el mismo. Sin embargo, este proceso entraña una complejidad que no siempre está justificada por las necesidades de las infraestructuras que utilizan estas redes. Por esta razón, existe otra alternativa en la que se omite por completo esta fase y la sustituye por un método de autenticación implícito y "fuera de banda", basado en la configuración de una clave secreta predefinida que comparten la estación y el punto de acceso.

- *Generación y distribución de claves*, cuando la fase de autenticación concluye con éxito se genera una clave maestra que compartirán posteriormente la estación y el punto de acceso. Alternativamente, una clave del mismo tipo puede ser configurada con antelación por el administrador de la red, en cuyo caso se prescinde de la citada autenticación. Seguidamente, en esta fase se realiza un intercambio de tramas mediante el cual se comprueba que ambos dispositivos poseen la misma clave maestra. Si la validación tiene éxito, los dispositivos generan varias claves temporales de sesión, como parte del anterior intercambio, las cuales permitirán cifrar y comprobar la integridad y la autenticidad de parte de las comunicaciones de datos efectuadas posteriormente. No obstante, la restricción de acceso continuará vigente hasta que las entidades de cada dispositivo involucradas en esta fase transfieran la clave temporal correspondiente a la capa MAC y se sincronicen para la utilización de dicha clave. Adicionalmente, algunas de las tramas transmitidas en este intercambio transportan ciertos datos que permiten confirmar la suite de cifrado unicast acordada en la primera fase.
- *Intercambio de datos protegidos*, durante esta fase las estaciones pueden intercambiar datos libremente a través de uno o más puntos de acceso. Estos datos estarán amparados por los mecanismos de seguridad negociados en la primera etapa, tanto en lo que respecta a la privacidad, como la integridad y la autenticidad del origen, para lo cual se recurre a las claves temporales generadas en la fase anterior o a nuevas instancias originadas durante algún proceso de renovación de claves posterior. No obstante, conviene recordar que estos mecanismos solamente tienen efecto sobre el enlace entre la estación y el punto de acceso, por lo que no se aplican a través del sistema de distribución y, en general, tampoco es suficiente una única instancia de los mismos para proteger las comunicaciones unicast entre dos estaciones (excluyendo a los puntos de acceso) del mismo BSS.

- *Terminación de la conexión*, cuando un usuario ha terminado de enviar y recibir datos puede requerir la transición a esta fase de la estación que utiliza. Otras causas posibles son: la pérdida de operatividad del enlace de radio, algún error durante el proceso de generación de la clave temporal o que el propio usuario desactive la interfaz inalámbrica o interrumpa el funcionamiento de la estación. Durante esta fase suelen ocurrir los siguientes eventos: el punto de acceso desautentica a la estación, las asociaciones de seguridad entre ambos se anulan y se suprimen, y por último, vuelve a restringirse el acceso de la estación a la red por medio del bloqueo selectivo del tráfico impuesto por el punto de acceso.

4.2 Autenticación

En la enmienda 802.11i se definen dos métodos de autenticación, uno basado en el estándar *IEEE 802.1X* y otro basado en una clave predefinida y compartida entre dos o más estaciones. En el primer caso, se ejecuta explícitamente un procedimiento de autenticación mediante una secuencia de paquetes intercambiados entre la estación, el punto de acceso y posiblemente otro dispositivo más, sin embargo el mecanismo de autenticación específico no entra dentro del ámbito del estándar 802.1X. En el segundo caso, la autenticación se realiza de forma implícita al configurar expresamente en el punto de acceso y en la estación vinculada a éste una clave maestra compartida entre ambos dispositivos (denominada *Pre-Shared Key*, abrev. PSK), aunque como efecto lateral del proceso de generación y de distribución de claves se comprueba que ambos dispositivos poseen la misma clave. En el supuesto de que tuviesen distintas claves maestras, en la fase correspondiente se anularía la asociación entre la estación y el punto de acceso. En cualquier caso, la fase previa, esto es, la fase de autenticación, se omite por completo en la autenticación mediante PSK.

Sin embargo, el control de acceso basado en el estándar IEEE 802.1X se aplica en ambas formas de autenticación, evitando así que una estación consiga acceder a los servicios de la red (salvo, posiblemente, al servicio de autenticación) hasta que dicha estación y el punto de acceso generen e instalen una clave temporal de sesión. Concretamente, la *Wi-Fi Alliance* ha promocionado estos dos métodos de autenticación bajo los nombres de *WPA*(*WPA2*)-*Personal* y *WPA*(*WPA2*)-*Enterprise*, que hacen referencia a la autenticación mediante PSK y mediante el estándar IEEE 802.1X, respectivamente. De modo que, recomiendan el primero para redes domésticas o redes de pequeñas empresas, y el segundo para empresas o para organizaciones de mayor envergadura.

4.2.1 Autenticación basada en el estándar 802.1X

En el estándar IEEE 802.1X se especifica un mecanismo de control de acceso a redes basado en el concepto de Puerto, así como un marco de trabajo para la autenticación y para otros procesos relacionados con la autenticación (gestión de claves, autorización, etc). En este estándar, la noción de Puerto se refiere a un punto de conexión de un sistema con una LAN, ya sea físico, como una conexión física a un segmento de una LAN, o lógico, como una asociación entre una estación y un punto de acceso. Cada Puerto es controlado por una entidad de acceso a puerto o *PAE* (acrónimo de *Port Access Entity*) aunque, desde el punto de vista del control de la operación de un Puerto, podemos considerar que se divide en dos puertos conceptuales, denominados: *Puerto Controlado* y *Puerto no Controlado*. Cada trama recibida por un Puerto 802.1X estará disponible en ambos puertos conceptuales, pero generalmente las únicas entidades de protocolo que tienen acceso al Puerto no Controlado son las PAEs.

Por lo tanto, un par de PAEs, implementadas por dos dispositivos distintos situados en cada extremo de un enlace, pueden intercambiar tráfico de autenticación sin restricciones a través del Puerto no Controlado. Normalmente, las restantes entidades de protocolo de red son accesibles a través del Puerto Controlado, cuando éste se encuentra en estado *Autorizado*. Sin embargo, el Puerto Controlado se encuentra inicialmente en estado *No Autorizado*, aunque como resultado de una autenticación correcta la PAE asociada al Puerto 802.1X puede provocar la transición de éste hacia el estado *Autorizado*. Por consiguiente, todo el tráfico útil es bloqueado hasta la autenticación del dispositivo cliente contra el dispositivo que controla el acceso a la red (aunque también es posible debilitar esta restricción para permitir el intercambio de determinada clase de tráfico de red, como podría suceder, por ejemplo, con los paquetes DHCP). La enmienda 802.11i también exige el control de acceso por parte del dispositivo cliente, de manera que esta medida debería frustrar los ataques contra una estación basados en la suplantación del punto de acceso.

Como se afirmó anteriormente, el estándar IEEE 802.1X propone un marco de trabajo para la autenticación de usuarios o dispositivos. En este marco de trabajo se describen las entidades que participan en la autenticación, los esquemas principales de la interacción entre estas entidades, el formato genérico de los mensajes que intercambian, el comportamiento de estas entidades frente a ciertos tipos de mensajes y cómo pueden ser transportados estos mensajes por determinados protocolos de red. Sin embargo, los mecanismos de autenticación concretos, mediante los cuales se validan las credenciales de los usuarios o las estaciones, no son abordados en dicho estándar,

sino que generalmente se describen en RFCs específicos y son ejecutados por medio de protocolos especializados que se ajustan a las especificaciones de un protocolo más general, denominado *EAP* (acrónimo de *Extensible Authentication Protocol*).

El protocolo EAP se describe en el RFC 3748 y establece los fundamentos de la autenticación basada en el estándar IEEE 802.1X, de hecho, el modelo arquitectónico que propone este estándar es muy parecido al que se expone en el mencionado RFC. La característica más importante de este protocolo es su extensibilidad, gracias a la cual puede incorporar nuevos procedimientos de autenticación no definidos en el RFC original. Esto resulta evidente enumerando la extensa lista de métodos de autenticación que han sido especificados posteriormente, incluso algunos de ellos por empresas u organizaciones privadas, de manera que no todos han sido estandarizados por el IETF. En el RFC 3748 se define también un marco de trabajo, aunque está enfocado sobre la autenticación de entidades, en vez del control de acceso, como ocurre con el estándar IEEE 802.1X. No obstante, todas las cuestiones que se mencionaron con respecto al anterior estándar son abordadas también en las especificaciones de EAP, junto a otras nuevas, como son la negociación de los métodos de autenticación, la forma en la que son encapsulados los datos de autenticación intercambiados, la retransmisión de paquetes perdidos o la eliminación de paquetes duplicados.

Aunque el RFC 3748, que contiene las especificaciones del protocolo EAP, detalla algunos procedimientos de autenticación concretos, como: *EAP-MD5*, *EAP-OTP* y *EAP-GTC*, la mayoría de los restantes procedimientos, denominados métodos conforme a la terminología de este estándar, son especificados en sus propios RFCs. Las técnicas empleadas por estos métodos para validar las credenciales de los usuarios o de los dispositivos son muy diversas. Así pues, algunos métodos utilizan pares de identificadores de usuario y de contraseñas, otros están basados en protocolos de desafío-respuesta que emplean una función hash o secuencias seudoaleatorias de un solo uso (esto es, lo que se conoce como *One-Time Passwords*) y, en general, los métodos más seguros recurren a la criptografía de clave asimétrica por medio de certificados digitales de clave pública. No obstante, la enmienda IEEE 802.11i restringe explícitamente la utilización de métodos en una RSN solamente a aquellos que garantizan una autenticación fuerte y bilateral. Partiendo de esta premisa, es posible la generación de una clave segura que sea conocida únicamente por las estaciones autorizadas para ello (por lo tanto, los métodos detallados en el RFC 3748 no son apropiados para este cometido).

El marco de trabajo presentado en el estándar IEEE 802.1X plantea una arquitectura (que se asemeja a la se expone en el RFC 3748) en la que participan tres entidades fundamentales:

- **Cliente** (traducción libre del término inglés: *Supplicant*), es una PAE que reside en el dispositivo por medio del cual un usuario pretende acceder a la red y, por lo tanto, es la entidad que desencadena el proceso de autenticación (por ejemplo, este rol puede desempeñarlo una estación 802.11).
- **Autenticador** (en inglés: *Authenticator*), también es una PAE, pero es implementada por un dispositivo que controla el acceso a la red y que, en consecuencia, puede admitir o bloquear el tráfico de un Cliente (por ejemplo: un NAS o un servidor de acceso a una red, o también un punto de acceso, pueden realizar esta función).
- **Servidor de Autenticación** (*Authentication Server*, en adelante: AS), es la entidad que implementa el servicio de autenticación en una red (puesto que el Autenticador tiene que delegar esta función en dicha entidad) y que, por tanto, determina si debe autorizar el acceso de un Cliente a la red o, por el contrario, debe denegar la correspondiente petición de acceso (por ejemplo, un servidor *RADIUS* o un servidor *Diameter*, generalmente, serán los encargados de llevar a cabo esta tarea).

Aunque es posible que el Autenticador y el AS sean implementados por el mismo dispositivo, típicamente un AP, debido a los recursos limitados de los que disponen estos dispositivos de red, la opción de destinar un equipo exclusivamente para la función de AS resulta más adecuada en muchas situaciones. Esta alternativa ofrece una mayor escalabilidad, además de otras ventajas como la centralización de la autenticación y la simplificación o la reducción de costes y de esfuerzo en la administración. Adicionalmente, los servidores de autenticación pueden desempeñar otras tareas relacionadas con la autorización y la auditoría, en cuyo caso reciben el nombre más específico de servidores *AAA* (acrónimo que hace referencia a las funciones de autenticación, autorización y auditoría). Por lo que se ha afirmado anteriormente, no resulta difícil intuir que gran parte de la complejidad de la autenticación, en lo que respecta a la implementación, reside en el AS, puesto que éste debe soportar, en función de las circunstancias del entorno en el que opera, un repertorio más o menos variado de métodos de autenticación para dar servicio a distintos tipos de clientes.

En lo que respecta al Cliente, éste realiza un intercambio de paquetes EAP con el AS durante el proceso de autenticación. En cambio, el Autenticador no participa directamente en este proceso, aunque se ocupa de desencapsular y de volver a encapsular, mediante los paquetes o las tramas adecuados para el destinatario, los paquetes EAP que recibe del Cliente o del AS. El Autenticador también reenvía los paquetes EAP dirigidos al Cliente, cuando requieren una respuesta que no fue recibida, y suele iniciar el intercambio de paquetes EAP (y por tanto, la autenticación) enviando un paquete de tipo *EAP-Request/Identity* al Cliente. En la **Figura 10** se muestra un esquema de los protocolos involucrados típicamente en la autenticación 802.1X, suponiendo que el Autenticador y el AS residen en diferentes dispositivos interconectados mediante una red TCP/IP (nótese que, por razones estéticas, no se ha respetado la correspondencia con el mismo nivel del modelo OSI de los protocolos que aparecen a la misma altura en las torres izquierda y derecha del Autenticador).

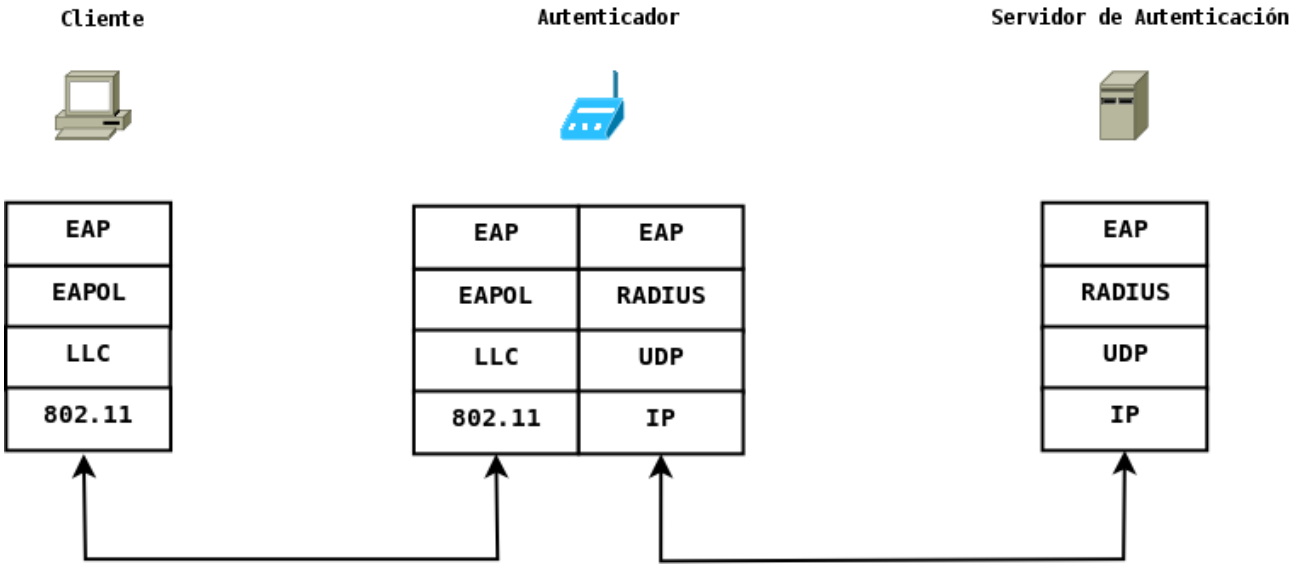

Figura 10: Torre de protocolos típica en la autenticación 802.1X

El protocolo EAP permite que el Cliente y el AS negocien un método de autenticación y también facilita el intercambio de los datos de autenticación requeridos por el método acordado. Para todas estas funciones, se utilizan paquetes *EAP-Request* enviados por el AS hacia el Cliente y paquetes *EAP-Response* con los que responde el Cliente. Este intercambio de paquetes EAP es totalmente secuencial (como se afirma en el RFC 3748, EAP es un protocolo de tipo "*lock-step*"),

de tal manera que no puede enviarse una nueva petición hasta que la anterior haya sido resuelta. Adicionalmente, el protocolo EAP soporta la retransmisión de paquetes perdidos y la eliminación de paquetes duplicados, aunque otras facilidades, como son la entrega en orden y la detección de errores, deben ser provistas por los protocolos subyacentes o bien por métodos de autenticación particulares. Si la autenticación se desarrolla normalmente, el AS notificará al Cliente el resultado de la misma (aunque esta notificación también afecta al Autenticador), mediante un paquete de tipo *EAP-Success*, cuando termina con éxito, o mediante un paquete de tipo *EAP-Failure*, en otro caso.

En cuanto al transporte de los paquetes EAP que son transferidos entre el Autenticador y el AS, la enmienda 802.11i no impone ningún protocolo en particular, aunque en la práctica el protocolo *RADIUS* (*Remote Authentication Dial-In User Service*) es casi un estándar "*de facto"* (no obstante, el IETF también ha especificado otro protocolo más sofisticado llamado *Diameter*). Este protocolo define cuatro tipos de paquetes, que cumplen un cometido similar a los distintos tipos de paquetes EAP. Durante una autenticación 802.1X típica, el Autenticador recibe paquetes *EAP-Response* del Cliente, los encapsula en paquetes *RADIUS-Access-Request* y los reenvía al AS. Recíprocamente, el AS envía paquetes *EAP-Request* al Cliente encapsulados en paquetes *RADIUS-Access-Challenge*, siendo los anteriores paquetes EAP desencapsulados y reenviados a su destino por el Autenticador. Este intercambio de paquetes continúa hasta que el AS recibe los datos necesarios para autenticar al Cliente, suceso que comunica al Autenticador por medio de un paquete *RADIUS-Access-Accept* que contiene un paquete *EAP-Success* dirigido al Cliente o, por el contrario, hasta que se produce algún error durante la autenticación, en cuyo caso el AS lo notifica al Cliente mediante un paquete *EAP-Failure* encapsulado dentro de un paquete *RADIUS-Access-Reject* que envía al Autenticador.

Del mismo modo que el protocolo EAP, el protocolo RADIUS cuenta con facilidades para la retransmisión de paquetes perdidos y la eliminación de paquetes duplicados, pero también incorpora ciertos mecanismos para comprobar la integridad, la autenticidad y la "frescura" (traducción del término inglés: "*freshness"*, que en este contexto expresa la cualidad de ser nuevo, en vez de una instancia repetida) de los paquetes transmitidos por el AS, así como para encriptar las contraseñas de usuario recibidas por el Autenticador, el cual las reenviará después hacia el AS por medio de paquetes *RADIUS-Access-Request*. En cualquier caso, para que dichos mecanismos puedan proveer las anteriores medidas de seguridad, previamente se requiere el establecimiento de una clave secreta compartida entre cada Autenticador y su correspondiente AS. Además, un Autenticador debe incluir en cada paquete que envía al AS un valor aleatorio, que servirá para comprobar la "frescura" del paquete de respuesta del AS (para más detalles consulte [Tuo2003]).

Otra característica destacable del protocolo RADIUS es que cada paquete debe ser transportado en un único paquete UDP y también que expresa la información útil que contiene mediante pares de atributos y valores. Cada uno de estos pares se codifica por medio de una tripla con formato *TLV*, esto es, tres subcampos adyacentes denominados: *Type*, *Length* y *Value*. De este modo, un paquete EAP encapsulado dentro de un paquete RADIUS estará asociado al valor de un atributo de tipo *EAP-Message* (*Type*=79). Aparte del tipo anterior, existen muchos otros tipos de atributos RADIUS que tienen diferentes propósitos, como el de informar sobre determinadas características del Cliente, del Autenticador o del AS, el de establecer el valor de ciertas propiedades ligadas a estas entidades, o incluso modificar su comportamiento (por ejemplo: la dirección IP del Autenticador, la duración de una sesión de usuario o la activación de la auditoría de ciertas acciones). Además, se concedió cierta flexibilidad a las implementaciones de este protocolo, gracias a la reserva de un intervalo de identificadores de atributos para su definición por los fabricantes. Sin embargo, algunos fabricantes se apresuraron a implementar sus propios atributos RADIUS, antes incluso de que fuese reservado el anterior rango, lo que ha originado, en ocasiones, problemas de interoperabilidad.

Por su parte, el Cliente y el Autenticador intercambian tráfico EAP mediante paquetes *EAPOL* (*EAP Over LAN*). Precisamente en el estándar *IEEE 802.1X-2004* se especifica como encapsular los paquetes EAP en paquetes EAPOL, así como los paquetes EAPOL en tramas *Ethernet*, *Token-Ring* o incluso a través de los servicios de la subcapa *LLC* definida en el estándar *IEEE 802.2* (esto es lo que ocurre en las tramas 802.11 de datos, puesto que su carga útil comienza normalmente con una cabecera *LLC/SNAP*). Esencialmente, un paquete EAPOL contiene tres campos: uno que indica la versión del protocolo EAPOL correspondiente al paquete (*Protocol Version*, que ocupa un byte y, actualmente, se fija al valor 1), después viene un campo que identifica el tipo de paquete EAPOL (*Packet Type*, que también ocupa 1 byte, ya que solo existen cinco tipos de paquetes EAPOL en la actualidad) y, por último, el campo que indica el tamaño de la carga útil del paquete (*Packet Body Length*, que ocupa dos bytes), al cual se le asigna el valor 0 si el paquete carece de carga de datos.

Adicionalmente, tres tipos de paquetes EAPOL, denominados *EAPOL-Packet*, *EAPOL-Key* y *EAPOL-Encapsulated-ASF-Alert*, respectivamente, contienen un campo más (referido como *Packet Body*), que aloja la carga útil del paquete EAPOL y cuyo tamaño es indicado por el campo previo. En el caso de un paquete de tipo *EAPOL-Packet*, la carga útil es un paquete EAP, mientras que un paquete de tipo *EAPOL-Key* transporta material de claves u otra información relacionada con la generación o el establecimiento de una clave de sesión. El último tipo de paquete EAPOL referido, esto es: *EAPOL-Encapsulated-ASF-Alert*, no se utiliza en la autenticación ni en la gestión de claves

de las estaciones de una RSN (según se afirma en [AE 2004]). Finalmente, los paquetes de tipo *EAPOL-Start* y *EAPOL-Logoff* son transmitidos por el Cliente y destinados al Autenticador para solicitarle que inicie el proceso de autenticación o que finalice la sesión actual, respectivamente.

Un esquema resumido de un intercambio típico de paquetes, que sucede en una RSN cuando se ejecuta una instancia de la autenticación basada en el estándar 802.1X, se ha representado en la **Figura 11**. No obstante, es frecuente que se prescinda del paquete inicial (*EAPOL-Start*), ya que la entidad de gestión del nivel de enlace del Autenticador puede advertir la conexión de un nuevo Cliente, lo que induciría al Autenticador a solicitar la identificación de dicho Cliente. Sin embargo, el mensaje *EAP-Request/Identity* enviado por el Autenticador tampoco es imprescindible, ya que la identidad del Cliente podría ser conocida a priori o determinada de alguna otra forma. No obstante, en la práctica podemos comprobar que en la mayoría de redes de la clase mencionada, el mensaje anterior inicia la autenticación. Por otro lado, este mensaje puede considerarse como una excepción al normal intercambio de paquetes EAP, en el que intervienen exclusivamente el Cliente y el AS, sin embargo, su uso se justifica porque conlleva diferentes ventajas, como la reducción del tráfico o del procesamiento que debe atender o realizar el AS.

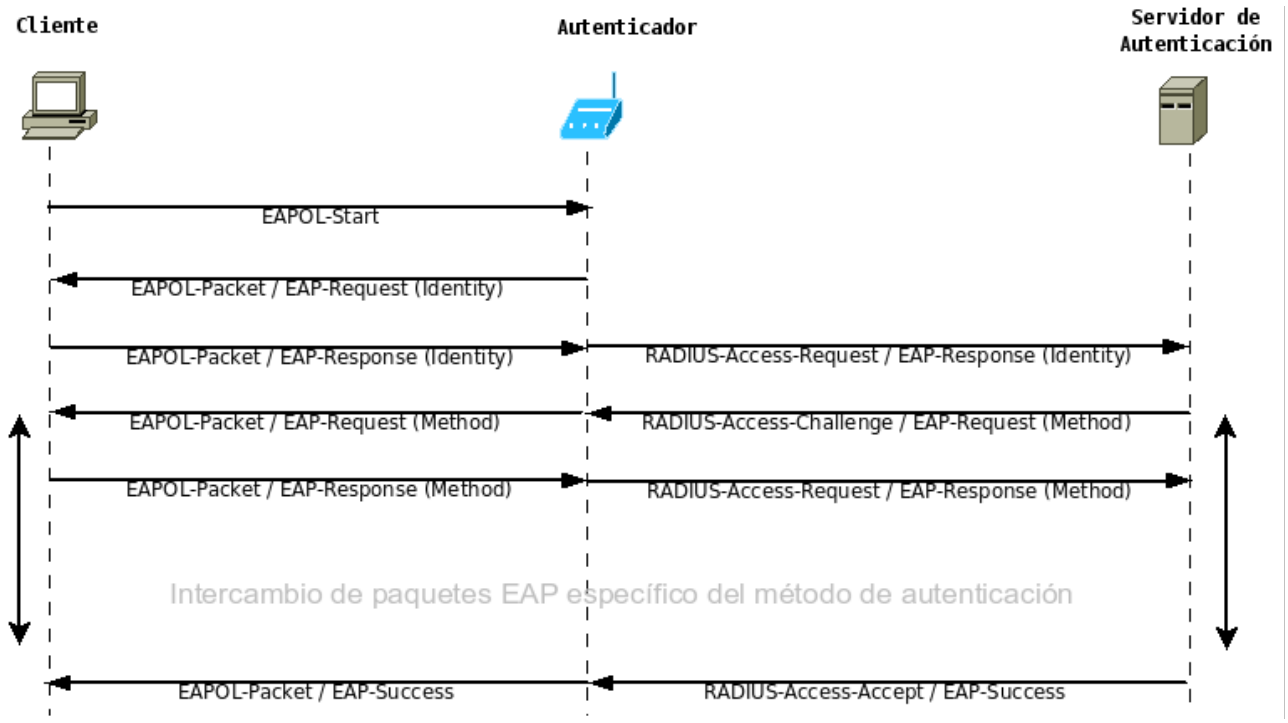

Figura 11: Esquema de una autenticación 802.1X típica

Como respuesta al mensaje EAP inicial del Autenticador, el Cliente devuelve un paquete de tipo *EAP-Response/Identity*, mediante el cual indica su identidad. Sin embargo, debido a que los datos incluidos en este tipo de paquete no están cifrados, un Cliente puede optar por ocultar su autentica identidad hasta una etapa más tardía de la autenticación, en la que transmite esta identidad protegida por medio de los mecanismos de seguridad provistos por algún método de autenticación específico. Así que, en lugar de la auténtica identidad del Cliente, dicho paquete *EAP-Response/Identity* puede transportar un identificador genérico o un seudónimo. Después de la etapa de identificación inicial, comienza el intercambio de paquetes EAP entre el Cliente y el AS, durante el cual la función y el formato de la mayoría de estos paquetes EAP intercambiados dependen del método de autenticación concreto que fue acordado por estas dos entidades.

Alternativamente, y aunque no se muestra en el esquema anterior, el Cliente podría rechazar el método de autenticación seleccionado por el AS respondiendo (al primer paquete enviado por el AS correspondiente a dicho método) con un paquete de tipo *EAP-Response/Nak* o incluso de tipo *EAP-Reponse/Expanded-Nak*. Mediante estas dos clases de paquetes, el Cliente puede sugerir otros métodos de autenticación alternativos, mientras que el AS puede responder aceptando uno de los métodos ofertados por el Cliente, abortando la autenticación o continuando la negociación con otros métodos EAP distintos a los ya propuestos. Por último, como ya fue comentado, el AS finaliza la autenticación cuando notifica al Cliente el éxito o el fracaso de ésta mediante un paquete de tipo *EAP-Success* o de tipo *EAP-Failure*, respectivamente.

La consecuencia directa de una autenticación finalizada con éxito es que el AS estará capacitado para generar cierto material de claves de alguna forma que depende del método de autenticación concreto que se ha ejecutado. El RFC 3748 se refiere a dicho material de claves como clave maestra de sesión (*Master Session Key*, abrev. *MSK*) y afirma que a partir del mismo se deriva la clave *AAA*. No obstante, existe mucha confusión en el anterior RFC sobre la existencia, el propósito y la forma en la cual se genera la clave AAA, como se analiza en [Aki2005]. Por un lado, este RFC menciona que dicha clave puede ser generada a partir de la MSK por el AS, o bien por el AS y por el Cliente de forma simultánea, cuando la autenticación es mutua (como sucede en las RSNs), para después ser transportada desde el AS hasta el Autenticador (en el primer caso, también debería ser entregada al Cliente). Bajo este escenario, podría interpretarse que la clave AAA desempeña el mismo papel que la clave PMK (o bien, que esta última clave es generada directamente a partir de la anterior), la cual se discutirá próximamente en esta sección.

Otra posibilidad que contempla el RFC 3748 es que la clave AAA sea idéntica a la clave MSK, como presupone que ocurrirá en muchos métodos de autenticación. De hecho, en la versión actual del estándar IEEE 802.11 (que fue publicada en el año 2007 e incluye la enmienda 802.11i) no se menciona a la clave AAA, en cambio, sí que se hace referencia múltiples veces a la clave MSK, por lo que puede deducirse que dicho estándar no establece distinción alguna entre ambas claves. En todo caso, e independientemente de que se denomine MSK o AAA a la clave anterior, el objetivo final de ésta es la derivación de otra clave maestra, que compartirán más tarde un par de entidades (concretamente, dos PAEs: un Cliente y un Autenticador) y que se denomina *Pairwise Master Key* (abreviadamente: *PMK*).

De acuerdo con la sección 8.5.1.2 del estándar IEEE 802.11-2007, en esta clase de autenticación la PMK debe obtenerse truncando la MSK, de tal forma que la primera clave estará compuesta por los primeros 256 bits de la MSK (exigencia que entra en conflicto con lo que se afirma en la sección 8.4.8 del mismo estándar, con respecto a que el procedimiento de generación de la PMK depende del método EAP específico empleado en la autenticación). En cambio, este estándar no especifica cómo debe ser transferida la PMK desde el AS hasta el Autenticador, aunque en el caso de que el AS sea implementado por un servidor RADIUS sugiere la opción de que sea transmitida mediante un atributo de tipo *MS-MPPE-Recv-Key*.

Una vez que el Cliente y el Autenticador disponen de la misma instancia de la clave PMK serán capaces de completar con éxito un intercambio de paquetes *EAPOL-Key* que se denomina *4-Way Handshake*. El propósito fundamental de este intercambio de paquetes es la generación de una clave temporal nueva, la cual facilitará la protección del tráfico unicast (esto es, el tráfico intercambiado entre un par de estaciones) durante toda o parte de la sesión en la que está vigente. Aunque para ser más exactos, hay que advertir que esta clave temporal solamente es una porción de un conjunto de claves de mayor longitud, que ha sido denominado *Pairwise Transient Key* (abrev. *PTK*).

Por su parte, la PTK se origina como consecuencia de una instancia del *4-Way Handshake* que llega a buen término y es imprescindible para que un Autenticador desactive el control de acceso. Por lo tanto, toda la sucesión de claves, que se ha mencionado en este apartado, desemboca en la derivación de la PTK. A su vez, podemos considerar que esta sucesión forma parte de una jerarquía de claves más grande, que está encabezada por la PTK. Por lo tanto, todas las claves que pertenecen a esta jerarquía más amplia, representada en la **Figura 12**, están relacionadas de una u otra forma con la protección del tráfico unicast en una red 802.11.

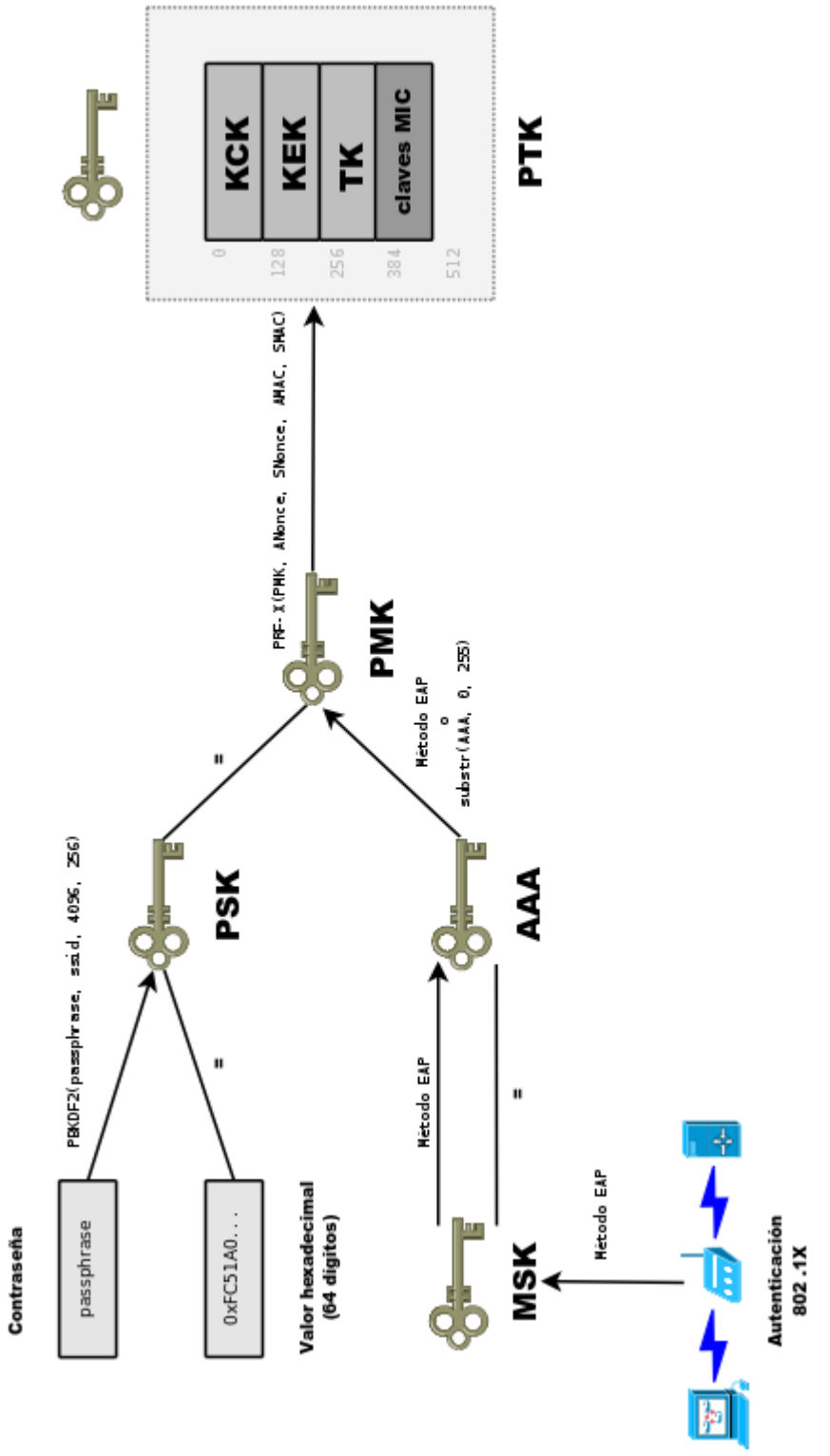

Figura 12: Jerarquía de claves para el tráfico unicast

4.2.2 Algunos métodos de autenticación comunes

En un principio, las certificaciones WPA/WPA2 de la *Wi-Fi Alliance* contaban tan solo con el método *EAP-TLS* (RFC 5216) para evaluar las implementaciones de la autenticación basada en el estándar IEEE 802.1X (véase [Ou2005]). Probablemente por esta causa, es actualmente el método soportado por más dispositivos de cliente y los sistemas operativos que los gestionan, pero además es considerado por los expertos como uno de los métodos EAP más seguros. Esto es debido a que su seguridad se basa en la utilización de certificados digitales de clave pública para la autenticación del Cliente y del AS, y a que, durante dicha autenticación, se ejecuta el *TLS-Handshake*, vinculado al protocolo *TLS* (versión revisada y actualizada del protocolo *SSL*), haciendo uso de la opción de la autenticación mutua y a través del encapsulamiento de los mensajes TLS dentro de paquetes EAP.

El principal inconveniente de este método es que requiere, en mayor medida que los restantes métodos tratados más adelante, una infraestructura de clave pública (*Public Key Infraestructure*, abrev. *PKI*), no solo para la verificación y para la firma de los certificados de cliente y de servidor, sino también para comprobar la validez de los certificados de cliente, puesto que presumiblemente algunos de éstos serán revocados. Sin embargo, el despliegue o el uso de los servicios de una PKI requiere un esfuerzo de administración o una inversión económica que no todas las organizaciones están dispuestas a afrontar. Otra desventaja de este método, generalmente de escasa importancia, es que la identidad de un Cliente puede ser descubierta por un atacante a través de cierto campo de un paquete *EAP-Response/Identity* o de un certificado *X.509* de cliente, ya que en ambos casos los datos correspondientes son transmitidos en claro por el Cliente durante la autenticación.

En el año 2005, la *Wi-Fi Alliance* anuncia que procederá a certificar otros cuatro métodos EAP: *EAP-TTLS/MSCHAPv2*, *PEAPv0/EAP-MSCHAPv2*, *PEAPv1/EAP-GTC* y *EAP-SIM*. Entre estos métodos EAP, los más utilizados son *EAP-TTLS* (RFC 5281) y *PEAP* (draft), los cuales tienen en común con *EAP-TLS* que utilizan certificados de servidor para la autenticación del AS, para lo cual recurren a la autenticación unilateral provista por el *TLS-Handshake*. En cambio, estos dos métodos se diferencian de *EAP-TLS* en que si el AS logra ser autenticado, justo después del *TLS-Handshake* se produce el establecimiento de una sesión TLS, gracias a la cual puede crearse un túnel seguro que protege la autenticación del Cliente (autenticación interna). A través de este túnel, el método *PEAP* (*Protected EAP*) habilita una nueva autenticación EAP mediante algún otro método, como: *EAP-MSCHAP*, *EAP-MD5* o *EAP-GTC*, con el propósito de validar las credenciales del Cliente.

En cambio, *EAP-TTLS* es más flexible, ya que no exige la transmisión de paquetes EAP durante la autenticación interna y en su lugar emplea pares de atributos y valores similares a los contenidos en los paquetes RADIUS. Por esta razón, *EAP-TTLS* no solo permite la autenticación del Cliente del mismo modo que con un método EAP, sino también mediante métodos ajenos al protocolo EAP, como son *PAP* o *CHAP*. Comparados con *EAP-TLS*, los métodos *EAP-TTLS* y *PEAP* cuentan con la ventaja de su mayor facilidad de implementación y administración, puesto que solo requieren un certificado de servidor para cada AS, que debe ser firmado por la Autoridad de Certificación cuyo certificado está presente en cada dispositivo cliente (alternativamente, el Cliente puede disponer del certificado de servidor del AS contra el cual se autentica). Sin embargo, ambos métodos presentan el inconveniente de que muchas implementaciones de los mismos son proclives a ciertos ataques que, en cambio, no afectan a *EAP-TLS*.

Finalmente, a mediados del 2009 los métodos *EAP-FAST* y *EAP-AKA* serían los últimos en ser incorporados al repertorio de métodos de autenticación que certifica la *Wi-Fi Alliance*. El primero de éstos, es decir, *EAP-FAST* (*Flexible Authentication via Secure Tunneling*) fue desarrollado por *Cisco Systems*, por lo que en un principio fue implementado exclusivamente por dispositivos de este fabricante (véase [EFO2007]), aunque más tarde sería especificado en el RFC 4851, obteniendo mayor aceptación. Tiene algunas características en común con *PEAP*, como el establecimiento de un túnel mediante certificados de clave pública para proteger la autenticación interna del Cliente, la cual se efectúa también mediante un método EAP. Alternativamente, admite la configuración de una clave compartida (*Protected Access Credential*) para la creación del túnel, lo que le permite prescindir del uso de certificados y de los servicios de una PKI. También cuenta con mecanismos adicionales para prevenir algunos ataques del intermediario (esto es, de tipo "*man-in-the-middle*").

En cambio, *EAP-SIM* (*Subscriber Identity Module*, especificado en el RFC 4186) y *EAP-AKA* (*Authentication and Key Agreement*, especificado en el RFC 4187) se escogieron ante la necesidad de adaptar los mecanismos de autenticación de las redes celulares *GSM* y *UMTS*, respectivamente, para la autenticación en redes Wi-Fi. Para esto, los dispositivos cliente necesitan una tarjeta *SIM* o *USIM*, la cual contiene las credenciales del usuario. El aumento de los escenarios en los que tienen lugar la convergencia de las redes celulares y las redes Wi-Fi ha favorecido el desarrollo de estos métodos. Así pues, hoy en día no resulta difícil encontrar terminales móviles de redes celulares que también pueden funcionar como dispositivos Wi-Fi, así como redes de acceso Wi-Fi desplegadas por operadores de redes de telefonía móvil (por ejemplo, las redes Wi-Fi de algunos proveedores del servicio de Internet móvil), gracias al bajo coste y a la facilidad de despliegue de las redes Wi-Fi.

4.2.3 Autenticación mediante clave pre-compartida

Como ya se expuso anteriormente, este tipo de autenticación se efectúa implícitamente cuando dos estaciones son configuradas con la misma clave secreta, la cual en este contexto se denomina clave pre-compartida (*Pre-Shared Key*, abrev. *PSK*). La enmienda 802.11i no establece cómo deben ser distribuidas estas claves, aunque recomienda un método para que sean generadas a partir de una contraseña. Por tanto, la distribución se realizará mediante algún procedimiento "*fuera de banda*", típicamente mediante la configuración manual por parte del administrador de la red. No obstante, dependiendo del número de estaciones pertenecientes a una red concreta, esta opción podría resultar impracticable. Por otro lado, aunque el estándar permite que cada par de estaciones utilice una clave pre-compartida distinta, en la práctica las implementaciones no soportan esta característica, así que se configura la misma clave en todas las estaciones del mismo BSS.

Como consecuencia de que en este tipo de autenticación las estaciones no participan en ningún intercambio de tramas con la finalidad exclusiva de la validación de sus credenciales, los miembros del grupo de tareas TGi decidieron que la propia clave pre-compartida (en adelante referida como *PSK*) hiciera las funciones de la PMK. Por tanto, esta clave se emplea durante el *4-Way Handshake* para derivar la clave temporal de sesión. No obstante, si un par de estaciones de una RSN (una de las cuales puede ser un punto de acceso) intentan comunicarse pero no disponen de la misma PSK, se producirá el fallo de alguna etapa del *4-Way Handshake*, lo que a su vez provocará que se anulen la autenticación y la asociación (como son definidas en el estándar 802.11 inicial) que previamente se establecieron entre ambas estaciones.

Debido a que el tamaño de la PSK es de 256 bits, al igual que el de la PMK, no sería difícil que un usuario cometiese un error al configurar el valor de esta clave. Por esta causa, en la enmienda 802.11i se ofrece la opción de generar el valor de esta clave a partir de una contraseña consistente en una cadena de caracteres ASCII cuya longitud debe estar comprendida entre 8 y 63 caracteres. Para este propósito, se propuso la función de generación de claves denominada *PBKDF2* (acrónimo que proviene de *Password-Based Key Derivation Function*), que se define en el estándar *PKCS#5*. Esta función, a su vez, aplica de manera sucesiva una función seudoaleatoria sobre cada bloque de la clave que genera. Dicha función seudoaleatoria recibe como parámetro la contraseña elegida por el usuario y el valor inicial de un bloque, o bien un bloque resultado de aplicar esta función una o más veces a un bloque inicial. Por otro lado, un bloque inicial contiene una cadena de caracteres arbitraria (a la que se llama "*sal*") concatenada con el índice del bloque de la clave.

En lo que concierne a las especificaciones de la enmienda 802.11i, la función seudoaleatoria seleccionada fue *HMAC-SHA1*, la cual se aplica sobre dos bloques iniciales distintos (cuyos índices toman los valores 1 y 2, respectivamente) para producir una clave de 256 bits (son necesarios un par de bloques debido a que esta función genera un valor de solo 160 bits de longitud). Por otra parte, cada bloque inicial contiene el nombre lógico de la red (esto es, el SSID), que se utiliza a modo de "sal" (para dificultar los ataques basados en tablas de valores precomputados), seguido por su índice de bloque correspondiente (que se codifica como un entero de 4 bytes en orden Big-Endian).

Para obtener un bloque determinado de la clave, partiendo del correspondiente bloque inicial, se aplica la función seudoaleatoria un total de 4096 veces sobre la contraseña de usuario y sobre el bloque inicial y los sucesivos valores computados a partir de este bloque. Además, el resultado de la *n-ésima* aplicación de esta función sobre el bloque inicial se combina mediante la operación *xor* con todos los resultados obtenidos de aplicar dicha función desde 1 hasta *n-1* veces al mismo bloque inicial. Por último, los dos bloques obtenidos mediante este proceso se concatenan (respetando el orden que indican sus índices de bloque) y la secuencia resultante se trunca hasta los primeros 256 bits, que serán los que constituyan la PSK.

4.2.4 Generación y distribución de claves

Después de que la autenticación 802.1X o la asociación 802.11 (cuando se usa la autenticación basada en la PSK) entre dos estaciones pertenecientes a una RSN concluya con éxito, se ejecutará el proceso de generación y distribución de claves. Este proceso cumple diferentes objetivos, siendo los más importantes los que se mencionan a continuación:

- Comprobar que ambas estaciones disponen a priori, o bien han sido provistas a través de la autenticación 802.1X, de una PMK idéntica y actualizada.
- Generar una clave (o un conjunto de claves) para proteger el tráfico unicast y facilitar la coordinación de ambas estaciones para la utilización de esta(s) clave(s).
- Distribuir, si está disponible, una clave (o un conjunto de claves) para la protección del tráfico tanto de tipo broadcast como de tipo multicast.
- Confirmar los parámetros de seguridad ofertados o seleccionados por cada estación y, si es requerido por el Autenticador, modificar la elección inicial.

Los objetivos previamente enumerados pueden alcanzarse a través de dos intercambios distintos de paquetes EAPOL, a los que el estándar 802.11 se refiere como: *4-Way Handshake* y *Group Key Handshake*, respectivamente. La realización del primer intercambio es obligatoria durante la etapa de generación y distribución de claves, puesto que éste posibilita el cumplimiento de todos o casi todos los objetivos citados. Además, cuando dos estaciones 802.11 consiguen completar con éxito este intercambio, se genera un conjunto de claves temporales de sesión que serán compartidas por ambas estaciones. Esta colección de claves se denomina *Pairwise Transient Key* (abrev. *PTK*) y contiene, entre otras claves, las claves que se utilizan para resguardar la privacidad, la integridad y la autenticidad de los datos que intercambian las dos estaciones durante toda o parte de una sesión.

En cambio, una instancia del *Group Key Handshake* puede ocurrir inmediatamente después de una instancia exitosa del *4-Way Handshake* o también, en cualquier momento, durante la fase de intercambio protegido de datos, siempre que sea necesaria la renovación de las claves dedicadas a la protección del tráfico de broadcast y de multicast. Estas claves, en conjunto, reciben el nombre de *Group Transient Key* (abrev. *GTK*, aunque también es posible que este término designe a una única clave) y suponen la causa de la existencia del *Group Key Handshake*, esto es, el procedimiento para la transmisión de una GTK desde el Autenticador hasta los Clientes. Por lo tanto, podemos afirmar que los dos tipos de intercambios involucran a las mismas entidades: un Cliente y un Autenticador (aunque en este trabajo se asume que el segundo rol lo desempeña un AP, también podría realizarlo cualquier estación perteneciente a un IBSS), y también a los mismos protocolos y tipos de paquetes: *802.11*, *LLC* y *EAPOL-Key*.

Los paquetes *EAPOL-Key* merecen ser mencionados aparte, puesto que el formato de su cuerpo depende de su aplicación específica, identificada a través del campo *Descriptor Type* (en el caso de las redes 802.11, se asigna el valor 2 a dicho campo). Tanto es así, que el formato de los paquetes de tipo *EAPOL-Key* intercambiados en esta clase de redes no se describe en el estándar IEEE 802.1X, sino en la propia enmienda 802.11i. Por razones de espacio no se ha detallado dicho formato en este trabajo, aunque es suficiente mencionar que, además de un campo para transportar claves y otros datos de propósito general (denominado *Key Data*), contiene campos específicos para albergar un contador de secuencia (o de repetición) asociado al paquete y otro asociado a la suite de cifrado en grupo negociada (cuando el paquete transporta una GTK), así como para un IV utilizado para la encriptación de la clave y de los restantes datos contenidos en el campo *Key Data*, para un nonce y para un código de verificación de la integridad/autenticidad del paquete *EAPOL-Key*, el cual ocupa 128 bits y se transmite en el campo llamado *Key MIC*.

Adicionalmente, un campo de flags (*Key Type*) indica cuales campos, de entre los mencionados anteriormente, transportan información útil (esto es, cuales son utilizados en un paquete concreto), también indica si el campo *Key Data* está encriptado y si el paquete está vinculado a una instancia del *4-Way Handshake* o bien del *Group Key Handshake*. Para finalizar, también puede destacarse un subcampo de dos bits (*Key Descriptor Version*) del anterior campo, que indica los algoritmos para el cifrado y la verificación de la autenticidad/integridad que se aplican a determinados paquetes del intercambio en cuestión (esto es, alguno de los dos tipos de "*handshake*"). Así pues, dependiendo del valor que indique este subcampo (alguno de los valores 1 o 2), la pareja de algoritmos utilizada para el cifrado/autenticación de los paquetes *EAPOL-Key* será: *ARC4/HMAC-MD5* o bien *AES Key Wrap/HMAC-SHA1-128*, respectivamente.

Como se explicó anteriormente, la PTK representa el último eslabón originado como parte de la jerarquía de claves destinada a la protección del tráfico unicast y su generación constituye un requisito imprescindible para la desactivación de las restricciones de acceso impuestas sobre un Puerto 802.1X. Cada instancia de una PTK es compartida por un par de estaciones que mantienen, o bien adoptan para su generación, los roles de Cliente y Autenticador, respectivamente, durante la instancia del *4-Way Handshake* en la que se genera la citada instancia de la PTK. Por medio del intercambio de paquetes estipulado en el *4-Way Handshake*, un par de estaciones derivan una PTK a partir de la PMK que comparten y también a partir de ciertos datos vinculados a sus entidades de la subcapa MAC y de cierta información producida específicamente para cada instancia de este tipo de intercambio.

Para aportar más detalles, es necesario mencionar la función seudoaleatoria (que se denota con el acrónimo *PRF*) que las estaciones invocan para el cómputo de la PTK. Concretamente, utilizan alguna de las dos versiones de esta función, denominadas *PRF-384* y *PRF-512*, respectivamente. La versión utilizada depende de que el tamaño exigido de la PTK sea de 384 o bien de 512 bits, requisito que viene impuesto por la suite de cifrado unicast establecida, según se trate de CCMP o TKIP, respectivamente. Por otra parte, la secuencia seudoaleatoria que devuelve esta función se obtiene mediante sucesivas invocaciones a la función hash basada en clave *HMAC-SHA-1*. En cada aplicación de esta función hash, la PMK juega el papel de clave, mientras que los datos de entrada son los siguientes: una cadena de caracteres predefinida, las direcciones MAC de ambas estaciones, los dos nonces generados durante el *4-Way Handshake* y el valor de un contador que se incrementa en uno en cada invocación de esta función hash correspondiente al cómputo de la misma PTK.

También es necesario aclarar que la PTK, en realidad, está compuesta por al menos tres claves, que son utilizadas con diferentes propósitos. A continuación, se enumeran estas claves en orden de aparición (esto es, en orden creciente de significado de sus bits) dentro de la PTK:

- **Key Confirmation Key** (abrev. **KCK**), es una clave de 128 bits utilizada por la función hash aplicada (HMAC-MD5 o HMAC-SHA1-128) para la generación del valor del campo *Key MIC* de ciertos paquetes de tipo *EAPOL-Key* que participan en una instancia del *4-Way Handshake* o del *Group Key Handshake*. De esta manera, cualquier estación que disponga de esta clave puede comprobar la integridad y la autenticidad de tales paquetes. Además, esta comprobación debe ser realizada por las dos estaciones que intervienen en uno de estos intercambios (esto es, un "*handshake*") para validar la legitimidad del mismo.
- **Key Encryption Key** (abrev. **KEK**), se trata también de una clave secreta de 128 bits cuya finalidad es encriptar la clave de grupo (esto es, la GTK), así como otros datos que pueden acompañar a la GTK dentro del campo *Key Data* de algunos paquetes de tipo *EAPOL-Key* intercambiados durante el *4-Way Handshake* o el *Group Key Handshake*. Como ya se dijo, hay dos algoritmos que fueron establecidos para el cifrado del campo *Key Data*, a los que el estándar se refiere como: *ARC4* y *AES Key Wrap*, respectivamente.
- **Temporal Key** (abrev. **TK**), el tamaño de esta clave (así como el de la PTK) depende de la suite de cifrado unicast empleada, siendo de 256 bits para TKIP y de 128 bits para CCMP. Esto es debido a que en el caso de CCMP, la TK representa una única clave que se utiliza con diferentes cometidos, mientras que en el caso de TKIP, la TK contiene una clave de 128 bits para el cifrado (a veces referida también con el nombre de TK) y dos claves de 64 bits para la verificación de la integridad/autenticidad (claves MIC o Michael).

En la **Figura 13** se muestra un esquema, incluyendo algunas variantes posibles, de los mensajes intercambiados durante una instancia típica de un *4-Way Handshake* que finaliza de forma exitosa. Como su nombre indica, mediante este proceso se puede alcanzar un acuerdo entre dos estaciones, siempre que se complete un intercambio válido de cuatro mensajes. En cada punto de la siguiente lista se describen algunas características destacables de un mensaje distinto del *4-Way Handshake*:

- El primer mensaje es enviado por el Autenticador después de que haya finalizado con éxito la asociación o la autenticación 802.1X del Cliente. Este mensaje contiene un nonce de 32 bytes generado por el Autenticador (referido como *ANonce*), el cual será utilizado para el

cómputo de la PTK, y también, de acuerdo con el estándar, el identificador de la PMK (abrev. *PMKID*, que se obtiene como resultado de aplicar la función *HMAC-SHA1-128* a la PMK y a otros datos). No obstante, en la práctica muchas implementaciones no incluyen el PMKID en el primer mensaje, salvo cuando se emplea la autenticación 802.1X y además el Autenticador mantiene en su caché una copia de una PMK que comparte con algún Cliente. En tal caso, el Autenticador puede ofrecer a dicho Cliente la posibilidad de prescindir de la autenticación 802.1X enviándole este mensaje con el identificador de esta PMK.

- En el segundo mensaje se incorpora otro nonce similar al anterior, aunque generado por el propio Cliente (que se denota como *SNonce*), además del campo *RSN Information Element*, transmitido anteriormente durante la (re)-asociación del Cliente, gracias al cual éste indicaba sus preferencias con respecto a las suites de autenticación y de cifrado unicast. Previamente al envío de este mensaje, el Cliente debe haber computado la PTK, supuesto que conoce la PMK, las direcciones MAC de ambas estaciones y los dos nonces. Luego, puede recuperar la KCK perteneciente a dicha PTK y utilizarla para calcular el valor del campo *Key MIC* incluido en el paquete *EAPOL-Key* correspondiente a este mensaje.
- Antes de construir el tercer mensaje, el Autenticador debe generar una PTK idéntica a la del Cliente, gracias al nonce recibido en el mensaje previo. A continuación, comprueba el MIC del segundo mensaje y después compara el campo *RSN IE* de este mismo mensaje con el que recibió durante la (re)-asociación del Cliente. Si alguna de estas comprobaciones produce un resultado negativo, el mensaje es descartado o la asociación es cancelada, respectivamente. En cambio, si el Autenticador no detecta ningún error, procede a elaborar el tercer mensaje, el cual contiene, al menos, el nonce generado por éste (*ANonce*), el *RSN IE* difundido en las tramas de tipo *Beacon* y *Probe Response*, así como el valor oportuno del campo *Key MIC*. Opcionalmente, puede incluir un segundo campo *RSN IE* para establecer la suite de cifrado unicast empleada para la comunicación con el Cliente, revocando la elección realizada por éste. Por último, si dispone de la clave de grupo (esto es, la GTK) debería incluirla en este mensaje acompañada de un índice (para distinguirla de instancias previas). Adicionalmente, cuando la GTK es transmitida en este mensaje, se exige su encriptación mediante la KEK y el algoritmo seleccionado para este propósito, del mismo modo que ocurre con el campo *Key Data* del paquete *EAPOL-Key* asociado a este mensaje (en tal caso, este campo contiene la GTK y todos los *RSN IEs* que sean necesarios, mientras que el valor del campo *Key MIC* se calcula después de esta operación de cifrado).

- Gracias al cuarto mensaje, el Cliente notifica al Autenticador que ha recibido y ha verificado el tercer mensaje (por ejemplo, que ha comprobado el valor del campo *Key MIC* y que tanto el nonce como el *RSN IE* inicial contenidos en el mensaje coinciden con los que ha recibido anteriormente). Además, el Autenticador puede asumir, cuando recibe el cuarto mensaje, que el Cliente ha puesto la PTK a disposición de su capa MAC (y posiblemente también la GTK) y se encuentra preparado para iniciar el intercambio seguro de datos. En tal caso, el Autenticador repetirá esta misma acción, siempre que valide correctamente la integridad y la autenticidad del mensaje final del *4-Way Handshake*.

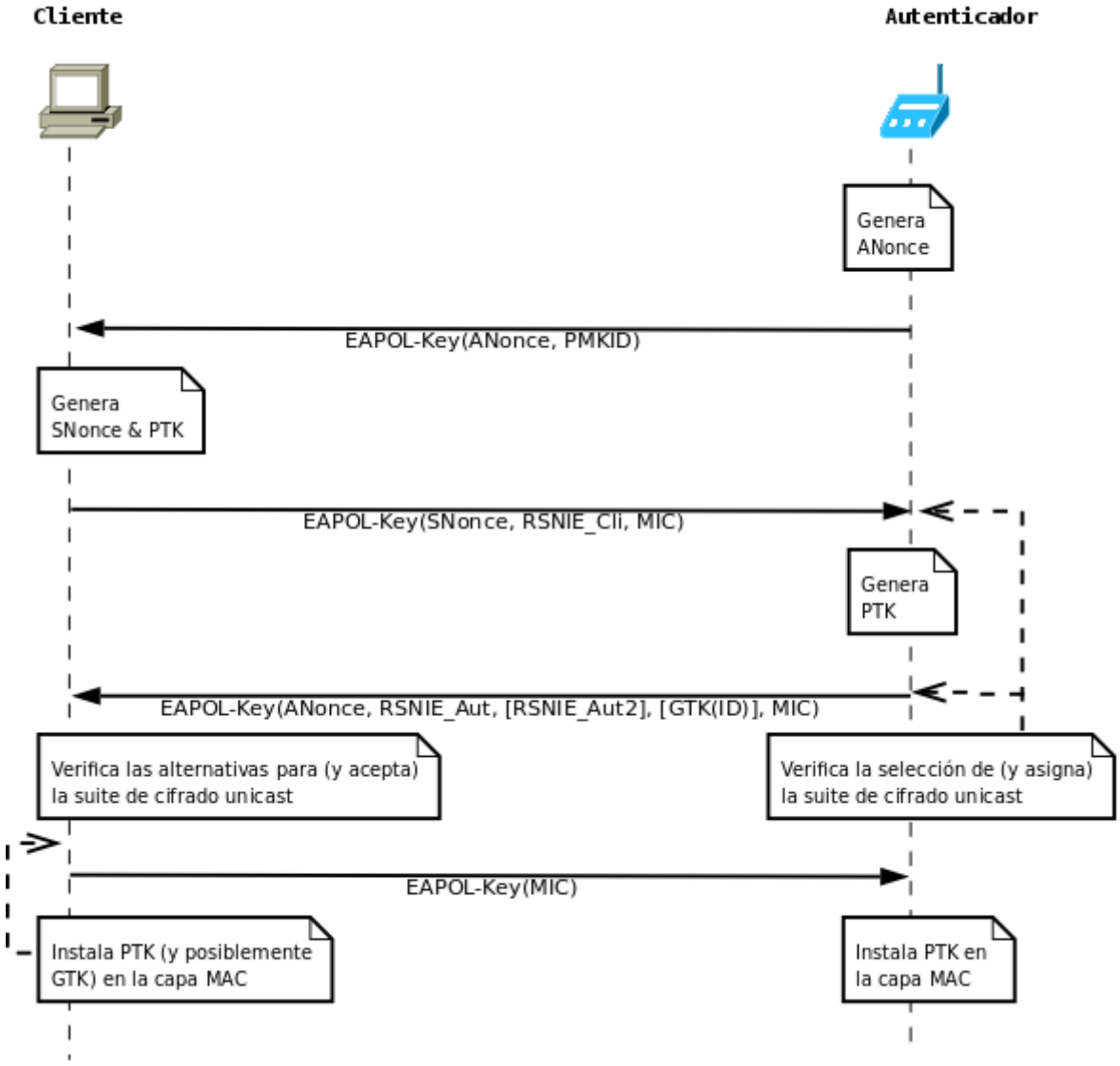

Figura 13: Esquema de un 4-Way Handshake

En lo que respecta a la jerarquía de claves para la protección del tráfico de broadcast/multicast, podemos apreciar que es mucho más simple que la correspondiente al tráfico unicast, ya que solo contiene dos miembros: la *GMK* y la *GTK*. La GMK (*Group Master Key*) es el punto de partida de esta jerarquía y, aunque el estándar 802.11-2007 no especifica su origen, en la práctica podría ser originada por el Autenticador a partir de una secuencia seudoaleatoria (tal vez exista una errata en la primera línea de la sección 8.5.1.3 del mencionado estándar). Sin embargo, en el anterior estándar se recomienda la opción de la renovación periódica de la GMK, reduciendo así la cantidad de datos expuestos en caso de que resulte comprometida.

También se recomienda en el estándar la generación de la GTK a partir de la GMK, mediante la misma función seudoaleatoria que se utiliza para producir la PTK, aunque en este caso el número de posibilidades es mayor para el tamaño de la secuencia generada (para una clave *WEP* sugiere *PRF-40* o *PRF-104*, mientras que para *TKIP* y *CCMP* las funciones apropiadas son: *PRF-256* y *PRF-128*, respectivamente). Adicionalmente, para el cómputo de la GTK, la función seudoaleatoria emplea la GMK como la clave de la función hash invocada (esto es, *HMAC-SHA-1*), mientras que como datos de entrada entrega a esta función el resultado de concatenar una cadena de caracteres predefinida, la dirección MAC del Autenticador y, por último, un número aleatorio o seudoaleatorio (denominado *GNonce*). Según lo afirmado previamente, se deduce que no se precisa la intervención de ningún Cliente para derivar la GTK. Por otro lado, la GTK tampoco contiene otras claves aparte de la TK. Sin embargo, dependiendo de la suite de cifrado escogida para el tráfico de broadcast y multicast, la TK podría descomponerse, del mismo modo que su homóloga para el tráfico unicast, en varias claves distintas (como ocurre con la suite de cifrado TKIP).

Una vez concluido el *4-Way Handshake*, si el Cliente no recibió en el tercer mensaje de este intercambio la GTK, la fase de generación y distribución de claves debería proseguir con el *Group Key Handshake*. Además, por diversas circunstancias puede ser necesaria la renovación de la GTK, como sucede, según se estipula en el estándar, cuando se cancela la asociación entre un Cliente y un Autenticador (por ejemplo, una estación abandona o es expulsada de un BSS). Alternativamente, una estación puede implementar sus propias políticas de seguridad que impongan la caducidad de la GMK (por ejemplo, como permiten algunos puntos de acceso). En ambos casos, el Autenticador debería producir una nueva GTK y efectuar paulatinamente un *Group Key Handshake* con cada uno de sus Clientes hasta reemplazar las versiones antiguas de la GTK que mantienen todos estos.

Para que una instancia del *Group Key Handshake* alcance su objetivo no necesita más que el intercambio de dos mensajes, uno por parte del Cliente y otro por parte del Autenticador, tal como se muestra en el esquema de la **Figura 14**. Sin embargo, también es posible que un Cliente solicite su ejecución mediante un paquete de tipo *EAPOL-Key* dirigido al Autenticador, cuyos flags *Request* y *Group Key*, pertenecientes al campo *Key Information*, están activados. Sin embargo, en el caso típico, como se ilustra en la anterior figura, el Autenticador desencadena el *Group Key Handshake* transmitiendo el mensaje inicial destinado al Cliente.

El primer mensaje contiene la GTK y su índice correspondiente encapsulados en una estructura denominada *KDE* (acrónimo que proviene de *Key Data cryptographic Encapsulation*) presente dentro del campo *Key Data* del paquete *EAPOL-Key* (del mismo modo que es transportada durante un *4-Way Handshake*). También en este caso, el campo *Key Data* contenido en este paquete se cifra con la KEK de la PTK y el valor del campo *Key MIC* se calcula a partir de todos los campos del paquete *EAPOL-Key* y la KCK perteneciente a la PTK, recurriendo en ambas operaciones a los algoritmos indicados por el subcampo *Key Descriptor Version*. Por otra parte, otro campo de este paquete, denominado *Key RSC* (acrónimo de *Key Receive Sequence Counter*), codifica el número de secuencia (correspondiente a la suite de cifrado) de la última trama protegida con dicha GTK que fue transmitida por el Autenticador. Adicionalmente, cuando el algoritmo utilizado para encriptar la GTK es *ARC4* (*Key Descriptor Version=1*), el contenido del campo *EAPOL-Key IV* se concatena con la KEK para construir la clave de cifrado, aunque previamente a esta operación de cifrado se descartan los primeros 256 bytes del keystream generado por este algoritmo.

Por su parte, después de recibir el mensaje anterior, el Cliente verifica que el correspondiente paquete *EAPOL-Key* no está duplicado, ni tampoco ha sido recibido en desorden, por medio del campo *Key Replay Counter*. A continuación, verifica la integridad y la autenticidad del paquete aplicando la función hash seleccionada (*HMAC-MD5*, o bien, *HMAC-SHA1-128*) y la clave KCK. Si el valor calculado mediante esta función no coincide con el que presenta el campo *Key MIC* de este paquete, entonces dicho paquete será descartado. En caso contrario, el Cliente descifrará la KDE y entregará, a la entidad de la subcapa MAC de la propia estación, la GTK con su índice asociado, así como el número de secuencia contenido en el campo *Key RSC* del paquete. A partir de ese momento, el Cliente estará capacitado para descifrar el tráfico de broadcast/multicast encriptado con la anterior GTK y, posiblemente, también podrá transmitir tráfico de la misma clase cifrado con dicha GTK.

Finalmente, por mediación del segundo mensaje, el Cliente confirma al Autenticador que está preparado para la utilización de la GTK que recibió en el primer mensaje. El paquete *EAPOL-Key* correspondiente al segundo mensaje no contiene datos adicionales (en el campo *Key Data*), ya que su único propósito es la confirmación de la recepción y de la validación del mensaje inicial por parte del Cliente. En cualquier caso, el Autenticador debe comprobar la integridad/autenticidad de dicho paquete, a través del valor del campo *Key MIC*, antes de aceptar el final del intercambio. No obstante, antes debe asegurarse de que el valor del contador de repetición (*Key Replay Counter*) de este paquete coincide con el valor del campo homólogo del paquete correspondiente al mensaje inicial, evitando así emparejamientos incorrectos entre paquetes vinculados a diferentes instancias del *Group Key Handshake*.

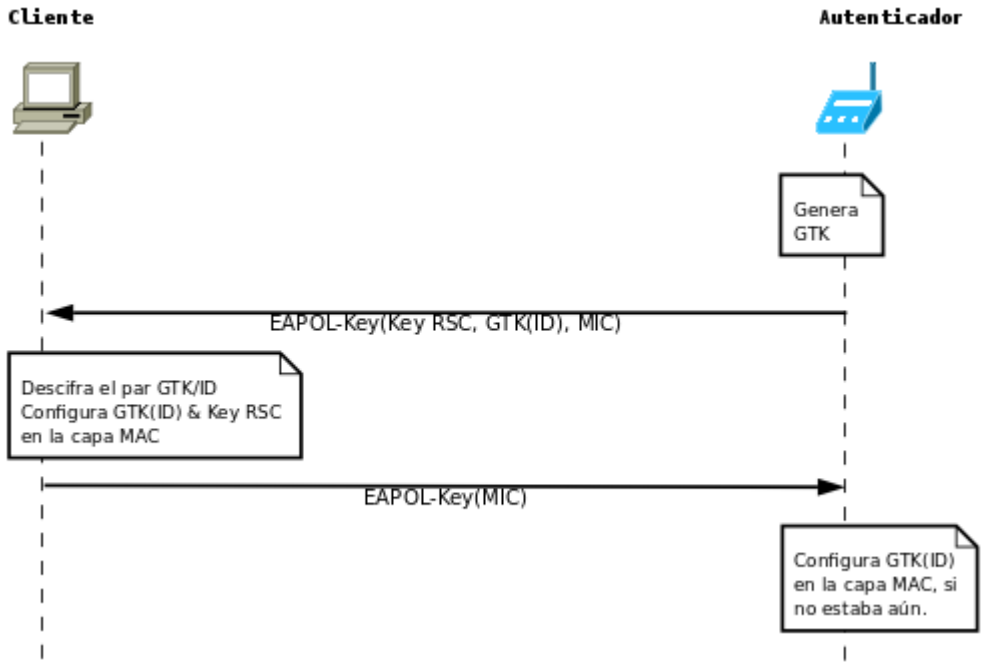

Figura 14: Esquema de un Group Key Handshake

4.3 El protocolo de seguridad TKIP

Temporal Key Integrity Protocol (abrev. *TKIP*) es el protocolo sobre el cual se asienta gran parte de la seguridad abordada en la certificación WPA, incluyendo la confidencialidad, la integridad y la autenticidad del origen de los datos. Uno de los principales objetivos del diseño de este protocolo y, a la vez, la principal restricción con respecto a las primitivas de seguridad que podía incorporar, fue que pudiese ser implementado por una amplia proporción de los dispositivos Wi-Fi existentes en ese momento (los cuales únicamente soportaban el protocolo WEP) mediante la actualización del firmware o del software. De hecho, TKIP conserva el mismo algoritmo de cifrado que WEP, esto es: RC4, que en aquel tiempo muchas estaciones implementaban mediante hardware, descargando así a la CPU de una considerable carga de trabajo (véase [Wal2002]).

Sin embargo, el objetivo más importante fue la elaboración de nuevos diseños que facilitasen la implementación de las características de seguridad abordadas y que permitiesen eliminar o paliar las numerosas debilidades, que ya se conocían, del protocolo WEP. Como se mostrará más adelante, el grado de cumplimiento de este último objetivo ha sido bastante aceptable. Sin embargo, TKIP todavía presenta algunas debilidades que, aunque son considerablemente menos importantes que las del protocolo WEP, han favorecido su adopción como una solución opcional en una RSN, mientras que CCMP se ha convertido en la solución cuya implementación es obligatoria. Finalmente, puede apreciarse que los mecanismos que implementan las medidas de seguridad asociadas al protocolo TKIP y relacionadas con la confidencialidad, por un lado, y con la integridad/autenticidad, por otro lado, son claramente diferentes y se exponen en las dos secciones que vienen a continuación.

4.3.1 Confidencialidad provista por TKIP

Al igual que WEP, TKIP se sirve del algoritmo RC4 para cifrar los datos. No obstante, podemos apreciar diferencias significativas en la forma en la cual se genera la semilla o clave por paquete proporcionada como entrada a dicho algoritmo. En primer lugar, el vector de inicialización con el que cuenta TKIP tiene un tamaño de 48 bits (frente a los 24 bits del IV de WEP), por lo que es necesario un campo adicional (además del campo *IV/Key ID* original) para contener dicho vector. Este campo se denomina *Extended IV* y contiene los cuatros bytes más significativos del nuevo tipo de IV (transmitidos en orden creciente de significado). La presencia de este tipo de IV en una trama 802.11 de datos es advertida mediante el flag *Ext IV*, que se añade en TKIP al campo *IV/Key ID*,

que ya existía en las cabeceras añadidas por WEP. Como se muestra en la **Figura 15**, los dos bytes menos significativos del IV son transmitidos mediante el campo *IV/Key ID*, separados por un valor especial, referido como *WEP_Seed[1]*, que se deriva del segundo byte (menos significativo) del IV.

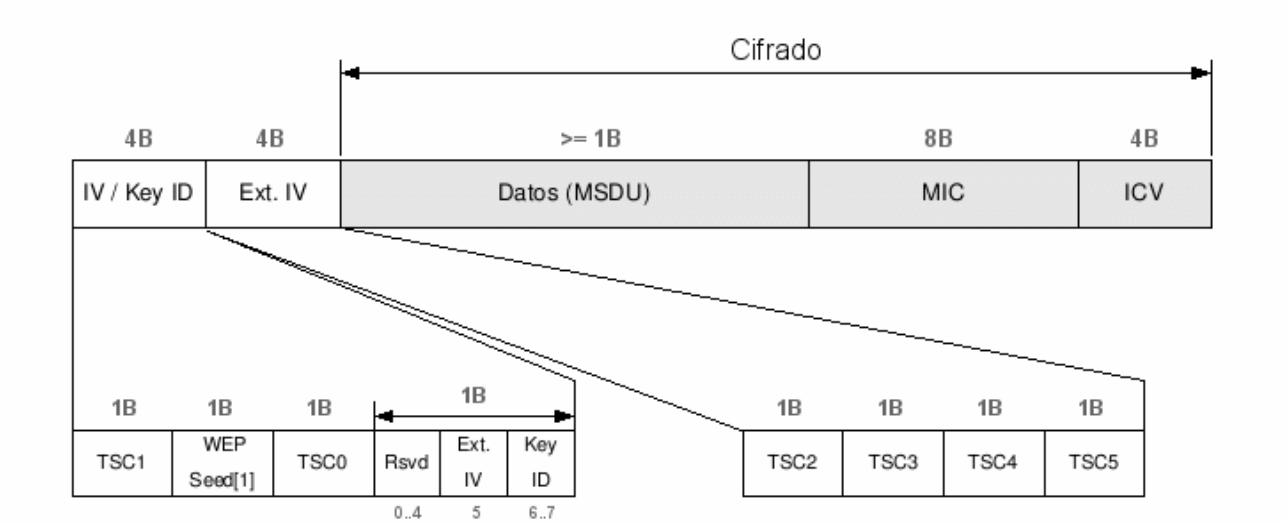

Figura 15: Encapsulamiento de los datos realizado por TKIP

Para ser más exactos, el vector de inicialización extendido incluido en las tramas protegidas mediante TKIP recibe el nombre de *TKIP Sequence Counter* (abrev. *TSC*), y no solo se utiliza para la generación de la clave RC4 por paquete, sino que también ejerce la función de un contador de secuencia para esta clase de tramas. De esta manera, cualquier estación receptora que comparta con otra estación una asociación de seguridad basada en el protocolo TKIP (como podría ocurrir con una PTKSA o una GTKSA), debe mantener un contador de recepción (en realidad, un contador por cada valor de prioridad de las tramas soportado y por cada asociación de seguridad de esta clase). Este contador se incrementa de forma monótona, puesto que se sincroniza con el TSC de las tramas recibidas, adquiriendo el valor de la última de éstas, a menos que dicho valor sea igual o menor al del contador de la estación, en cuyo caso no se modifica tal contador y la trama es descartada.

Adicionalmente, cada estación que actúa como transmisora mantiene, como parámetro de una asociación de seguridad basada en TKIP, un contador de envío cuyo valor se incluye en la cabecera TKIP de cada trama de datos (esto es, cada MPDU de datos) transmitida que esté vinculada a dicha asociación. En consecuencia, cada transmisión de una trama de este tipo provocará que la estación transmisora incremente el contador de envío de forma monótona. Por lo tanto, esta técnica debería

proteger efectivamente contra los ataques de repetición de tramas y de reutilización del keystream (asumiendo que el proceso de generación de la clave RC4 para cada trama cuenta con la suficiente entropía), siempre que la clave temporal sea renovada, o bien finalice la comunicación, antes de que se agote el espacio de valores del contador, como exige de forma expresa la enmienda 802.11i.

En segundo lugar, la clave empleada para el cifrado/descifrado mediante RC4 será calculada a partir de un fragmento de la clave temporal (esto es, los primeros 128 bits de la TK perteneciente a una PTK o a una GTK, aunque para simplificar esta sección se refiere también a este fragmento como TK), el TSC transmitido en claro en la cabecera añadida por TKIP y la dirección MAC del transmisor (abrev. *TA*, que corresponde al campo *Address 2* de una trama 802.11). Por consiguiente, se genera una clave RC4 específica para cada sesión, para cada estación, para cada trama (a veces llamada paquete) y posiblemente para cada usuario. Estos datos son combinados por medio de una función criptográfica de mezcla (parecida a una función HMAC que opera sobre varios datos de entrada) que consta de dos etapas. Dicha función debe producir una clave única para cada trama, de tal modo que dicha clave no revele ningún indicio sobre los datos de entrada.

Para conseguir este objetivo, en la primera fase de esta función se mezcla la *TK*, los 32 bits más significativos del *TSC* y la *TA* mediante operaciones *xor* entre fragmentos de los datos anteriores, sustituciones no lineales de los resultados anteriores (por medio de una S-Box) y sumas modulo 2^{16} entre los resultados de las sustituciones y los valores acumulados en sumas previas. Desde el punto de vista de una estación transmisora, los datos de entrada de la primera fase no deberían cambiar con mucha frecuencia (por ejemplo, tan solo cuando sea renovada la *TK* o cuando los incrementos sucesivos del TSC superen una diferencia de 2^{16}), por lo que es posible evitar la repetición de los cálculos anteriores, siempre que los datos de entrada no cambien, almacenando en una memoria caché el resultado de la ejecución de esta primera fase.

A continuación, la segunda fase recibe como entradas la *TK*, los 16 bits menos significativos del *TSC* y la salida producida por la primera fase (denominada *TTAK*, en la notación del estándar). El procesamiento efectuado sobre estos datos de entrada incluye operaciones muy similares a las de la primera fase, aunque también se realizan rotaciones a la derecha de un bit de algunos registros internos antes de ser sumados. Además, gracias a la participación de los bits de menor peso del TSC (que son los que cambian con más frecuencia) y a la forma tan compleja en la que se combinan con los restantes datos de entrada, podemos suponer que no existirán correlaciones apreciables entre el valor del TSC y la clave generada. Asumiendo que esta suposición es cierta, tampoco deben existir

correlaciones entre las claves generadas sucesivamente, debido a que el TSC varía en cada trama distinta, frustrando de esta manera los ataques de claves relacionadas, que representan la principal amenaza contra la seguridad provista por WEP.

Siendo rigurosos, advertiremos que no todos los bytes de la clave RC4 (que se denota como *WEP seed* en el estándar) son producidos por la función de mezcla, puesto que los tres primeros bytes de la anterior clave, como si se tratase del IV característico del protocolo WEP, se obtienen directamente del TSC y se concatenan con los restantes trece bytes generados mediante la función de mezcla (el tamaño de la clave RC4 empleada por TKIP es de 128 bits). Concretamente, el primer byte de la clave RC4 se corresponde con el segundo byte del TSC y el segundo byte de la clave RC4 depende del mismo valor que el primero, aunque en este caso a tal valor se le aplican ciertas operaciones a nivel de bit para mitigar la generación de claves débiles. Por último, el valor asignado al tercer byte de la clave RC4 es el correspondiente al primer byte del TSC (esto es, el byte menos significativo). En la **Figura 16** se muestra un diagrama del flujo de datos asociado a la generación de la clave RC4 por paquete, así como de la cabecera TKIP añadida durante el encapsulamiento de los datos realizado conforme a este protocolo.

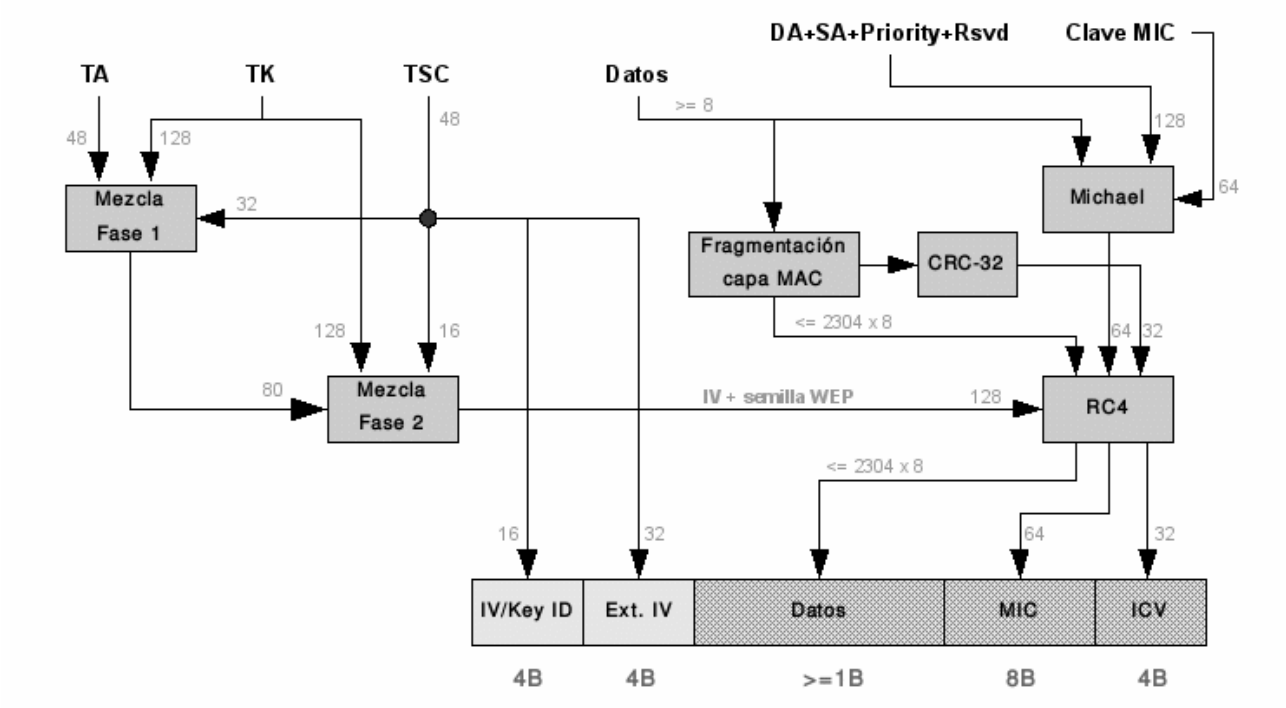

Figura 16: Procesamiento realizado por TKIP

4.3.2 Integridad y autenticidad provista por TKIP

En la **Figura 15** se mostraba precediendo al campo ICV, que se mantiene por compatibilidad con el hardware legado basado en WEP, otro campo denominado *MIC* (acrónimo que corresponde al término: *Message Integrity Code*, escogido para evitar confusiones con las múltiples acepciones del acrónimo MAC). Este campo, cuya longitud es de 64 bits, debe contener un valor computado mediante una función hash basada en clave (traducción libre del término: "*keyed hash function*") denominada *Michael*, que fue diseñada por *Niels Ferguson*. Esta función hash se aplica al mensaje formado por los campos: *SA*, *DA*, *Priority* y un campo reservado de tres bytes de longitud que se fija a cero en la actualidad, así como a la MSDU entregada a la subcapa MAC. Todos estos datos son concatenados en el orden en el cual se han enumerado y después se les añade una secuencia de relleno que consiste en un byte con el valor *0x5A* seguido por entre cuatro a siete bytes adicionales fijados a cero, de forma que el mensaje resultante presente una longitud múltiplo de 32 bits.

Los campos *SA* y *DA* se corresponden con las direcciones MAC de la estación de origen y de destino de la trama, respectivamente. Por otra parte, el valor del campo *Priority* viene indicado por medio de un único byte, aunque en realidad este campo no existe como tal en una MPDU, sino que se define como el valor de prioridad especificado a través de la primitiva de servicio de la subcapa MAC *MA-UNITDATA.request* (actualmente, este valor se fija a cero en muchas implementaciones). Por último, la MSDU hace referencia a los datos entregados por la subcapa LLC a la subcapa MAC y que intervienen en el cómputo del MIC antes de ser fragmentados por la subcapa MAC (en caso de que sea necesario) y de ser encriptados mediante RC4. Por tanto, el protocolo TKIP realiza una doble comprobación de la integridad de la carga de datos mediante los campos ICV y MIC, además de comprobar la autenticidad del origen de ésta (y de algunos otros campos de la MPDU) mediante el campo MIC. Este par de campos, esto es: el ICV y el MIC, se añaden al final de una MPDU y de una MSDU, respectivamente, y se encriptan como si formaran parte de la carga de datos.

La clave utilizada para la generación del MIC tiene un tamaño de 64 bits, aunque una asociación de seguridad protegida mediante TKIP requiere dos claves de esta clase. Una de estas dos claves (ubicada entre los bits 128 y 191 de la TK) está destinada a la validación del tráfico enviado desde el Autenticador al Cliente, mientras que la otra (que se encuentra entre los bits 192 y 255 de la TK) se aplica al tráfico en el sentido inverso. En cualquier caso, el algoritmo *Michael* divide la clave de entrada en dos fragmentos de 32 bits que se combinan entre ellos y también con bloques de 32 bits del mensaje, por medio de operaciones *xor* y sumas módulo 2^{32}. Adicionalmente, los registros que

almacenan los resultados intermedios de estas operaciones son modificados mediante rotaciones a la derecha o a la izquierda, así como a través de algunos intercambios de posiciones específicas de tales registros.

Con el secuenciamiento de estas operaciones tan simples, es posible el cómputo de un valor para el MIC, asociado a una secuencia de datos protegida mediante el protocolo TKIP, de manera que permita eliminar o reducir en gran medida la posibilidad de que tengan éxito los ataques de tipo *bit-flip*, los que efectúan el truncamiento o la división de la carga de datos (concatenándola o no con cadenas arbitrarias), los que suplantan a la estación de origen o de destino, los que intentan redirigir los paquetes modificando su dirección IP de destino, los ataques de fragmentación y los ataques iterativos que permiten revelar algunos bytes de la clave o de los datos en claro.

A pesar de todo, la transformación realizada por el algoritmo *Michael* no es criptográficamente segura, hecho que se reconoce en la enmienda 802.11i y que posteriormente ha sido ratificado con la demostración de varias debilidades. No obstante, en la elección de este algoritmo para verificar la integridad/autenticidad, han sido factores determinantes tanto la facilidad de implementación, como la eficiencia en el uso de recursos. Por lo tanto, con esta decisión se ha pretendido que TKIP se convierta en una opción viable para ser implementada por muchos dispositivos que soportaban WEP. Sin embargo, hay que asumir la posibilidad de que sucedan ataques activos contra TKIP, que ni siquiera el cifrado del campo MIC o el contador de repetición TSC puedan impedir.

En previsión de dicha posibilidad, este protocolo fue dotado con una serie de contramedidas destinadas a paliar la efectividad de estos ataques, que se activan cuando se detectan dos errores del MIC en un período de 60 segundos. En este contexto, un error del MIC se refiere a un fallo en la verificación del campo MIC de una trama recibida por una estación, siempre que los valores del TSC y del ICV hayan sido comprobados correctamente, o bien se refiere a una notificación de un error del MIC enviada por una estación a un punto de acceso a través de un paquete *EAPOL-Key* especial, denominado *MIC Failure Report* (que se caracteriza por activar los flags: *Request* y *Error*, así como por indicar el valor del TSC de la trama errónea mediante el campo *Key RSC*).

Como parte de estas contramedidas e independientemente del origen de los errores del MIC, un punto de acceso (o en general, un Autenticador) que detecte dos errores del MIC en un minuto debe eliminar todas las PTKSAs establecidas para las que se acordó TKIP como suite de cifrado unicast, así como desautenticar a las estaciones con las que comparte tales PTKSAs y reinicializar el estado de los Puertos 802.1X asociados a dichas estaciones, en caso de emplear la autenticación

802.1X. En cambio, cuando se despliega TKIP como suite de cifrado para el tráfico de broadcast/multicast, estas medidas afectan a todas las estaciones asociadas con el punto de acceso. Además, cuando se cumple esta condición, la GTKSA vigente debe ser descartada y reemplazada por otra nueva que no podrá utilizarse hasta transcurridos 60 segundos. En cualquier caso, una vez que las contramedidas han sido iniciadas por el punto de acceso, se impide que las estaciones establezcan nuevas PTKSAs basadas en TKIP durante 60 segundos.

Por otro lado, una estación que reciba una trama con un valor del MIC erróneo notificará al punto de acceso este evento mediante un paquete de tipo *MIC Failure Report*, tras descartar dicha trama. Entonces, si la estación detecta otro error del MIC dentro de los 60 segundos posteriores a este evento, se activan las contramedidas TKIP de la estación (la cual actúa como Cliente), debido a las cuales su entidad de gestión de la subcapa MAC (esto es, la MLME) reacciona provocando la eliminación de la PTKSA y la GTKSA, que fueron adquiridas como resultado de su incorporación al BSS. Seguidamente, la propia estación fuerza su desautenticación e impide la creación de nuevas asociaciones de seguridad (ya se trate de PMKSAs o de GTKSAs) con el mismo punto de acceso durante 60 segundos, siempre que tales asociaciones pretendan utilizar TKIP como suite de cifrado.

4.4 El protocolo de seguridad CCMP

Al contrario que TKIP, el protocolo *CCMP* debe ser implementado por toda estación compatible con la seguridad RSN, introducida en la enmienda 802.11i. *CCMP* es el acrónimo de *Counter Mode with CBC-MAC Protocol*, término que proviene del modo de cifrado en bloque denominado *CCM* (descrito en el RFC 3610), que no solo está destinado a proteger la confidencialidad, sino también la integridad y la autenticidad del origen de los datos. Además de estas características, el protocolo CCMP cuenta con mecanismos para mitigar la repetición de MPDUs, para lo que usa un contador de secuencia parecido al TSC del protocolo TKIP. Por su parte, el modo CCM recurre a otros dos modos de cifrado en bloque distintos (aplicados al mismo algoritmo de cifrado): el modo *Contador*, para cifrar los datos, y el modo *CBC-MAC*, para generar un código de verificación de la integridad y la autenticidad del mensaje, que también se denomina MIC en las especificaciones de CCMP.

En el protocolo CCMP, el modo CCM es aplicado sobre el algoritmo de cifrado en bloque de clave simétrica denominado *AES*, que fue especificado en el estándar *FIPS PUB 197-2001* del *NIST* (organización del gobierno de los EEUU que promueve estándares). Este algoritmo fue diseñado

para encriptar mensajes mediante bloques con un tamaño fijo de 128 bits. Sin embargo, soporta diferentes tamaños de clave (concretamente de 128, de 192 y de 256 bits), aunque en el caso de CCMP se estipuló una longitud de clave de 128 bits. A pesar de que CCM hace uso de la misma clave para cifrar y para comprobar la autenticidad/integridad de los datos, en el RFC 3610 se afirma que esta característica no acarrea ninguna debilidad, siempre que el algoritmo de cifrado subyacente genere la suficiente entropía (esto es, cuando dicho algoritmo se comporta aparentemente como una permutación aleatoria de cada entrada). Además, posteriores investigaciones han demostrado que proporciona un nivel de seguridad equiparable a la de otros modos de cifrado en bloque.

En el anterior RFC se requiere, para la aplicación del modo CCM sobre un algoritmo de cifrado concreto, la especificación de dos parámetros: el tamaño del campo que contiene el *MIC* (a mayor tamaño, mayor garantía de la integridad de los datos, aunque también mayor sobrecarga añadida a la MPDU) y el tamaño del campo que contiene la *longitud del mensaje* (a mayor tamaño de este campo, menor es el tamaño del campo que contiene el nonce, lo que permite cifrar mensajes de mayor longitud a costa de reducir el número máximo de bloques que pueden ser cifrados con la misma clave). En las especificaciones de CCMP, los valores asignados a estos dos parámetros son: 8 bytes para el campo MIC y 2 bytes para el campo que contiene la longitud del mensaje (en este contexto, la longitud de la carga de datos transportada por la MPDU).

4.4.1 Confidencialidad provista por CCMP

Las protecciones de la confidencialidad que incorpora CCMP afectan tanto a la carga de datos transportada por la MPDU, como al MIC calculado sobre dicha MPDU en claro y que se concatena al final de los datos previamente a la encriptación de la MPDU, como se representa en la **Figura 17**. Con esta finalidad, se llevan a cabo una o más operaciones de cifrado por cada MPDU mediante el algoritmo AES, aunque al operar en modo Contador tales operaciones no se aplican directamente a los datos, sino a uno o más bloques de contador, estando constituido cada uno de estos bloques por:

- Un campo de flags que ocupa un byte y que codifica mediante tres bits el tamaño en bytes del campo que indica la longitud del mensaje (tamaño que coincide con el del campo que contiene el contador de bloque, que se describe más abajo). Debido a que los restantes bits del campo de flags se fijan a cero y a que el tamaño del campo que denota la longitud del mensaje es invariable en CCMP, el valor de este campo de flags es constante.

- Un campo que contiene un *nonce*, esto es, un número muy grande que debe ser diferente en cada mensaje cifrado con la misma clave (en este protocolo, cada MPDU se considera un mensaje distinto). En el caso de CCMP, el valor del nonce no cambia en los bloques de contador asociados a la misma MPDU y tiene un tamaño fijo de 13 bytes.
- Un campo utilizado para numerar los bloques de contador correspondientes a un mensaje determinado. Este campo tiene un tamaño de dos bytes en CCMP y el valor que contiene, esto es, el índice o contador de bloque, se codifica en orden *Big-Endian*.

Entonces, para el cifrado de un mensaje en claro, éste se divide en bloques de 128 bits y cada uno de éstos se combina con su correspondiente bloque de contador, previamente encriptado con la clave temporal vigente en ese momento, mediante la operación *xor*. Por tanto, esta forma de operar corresponde a un tipo de cifrado aditivo, en esencia similar al cifrado *Vernam*, aunque en el modo Contador la semilla de la secuencia seudoaleatoria, mediante la cual se encripta cada bloque del mensaje, incluye tanto a la clave de cifrado, como a los bloques de contador. No obstante, debido a que la única parte de la semilla que varía, cuando se cifran diferentes bloques de contador asociados a un mismo mensaje, es el contador de bloque, el algoritmo de cifrado subyacente (que en este caso es AES) debe generar la suficiente entropía para garantizar la independencia de todas las secuencias seudoaleatorias producidas a partir de los distintos bloques de contador.

Para ser más exactos, el primer bloque de contador (el bloque cuyo índice es cero) se reserva para encriptar el MIC (descartando los bytes sobrantes del bloque), de manera que el bloque cuyo valor del contador es uno será el que se utilice para encriptar el bloque inicial del mensaje en claro. En cuanto al *nonce* incluido en un bloque de contador, éste no se obtiene generando una secuencia seudoaleatoria, sino que se construye a partir de tres campos disponibles en una MPDU protegida mediante CCMP (sin embargo, ninguno de estos campos está encriptado, así que cualquiera podría examinar sus respectivos valores). A continuación, estos tres campos son enumerados en el orden de aparición dentro del nonce:

- Un campo que denota la prioridad de la trama y que tiene una longitud de un byte. El valor que presenta se deriva del campo *QoS Control* (en concreto, de un subcampo del anterior denominado *TID*), incluido en las tramas que cuentan con soporte para calidad del servicio conforme a la enmienda 802.11e. En las restantes tramas (esto es, las tramas que carecen de soporte para calidad del servicio), el valor de este campo se fija a cero.

- El campo *Address 2* de la MPDU, que típicamente indica la dirección MAC de la estación que transmite la trama a través del medio inalámbrico y que, en este protocolo, es integrado también en el nonce y define el valor de los siguientes seis bytes de éste.
- El campo denominado *Packet Number* (abrev. *PN*), perteneciente a la cabecera CCMP que precede a los datos cifrados de una MPDU. Este campo desempeña también la función de contador de secuencia en CCMP y tiene un tamaño de seis bytes.

En consecuencia, el *nonce* empleado para cifrar una MPDU se puede reconstruir a partir de los campos contenidos en la cabecera MAC y en la cabecera CCMP de dicha MPDU, las cuales son transmitidas ambas en claro. El formato de la cabecera añadida por el protocolo CCMP se ilustra en la **Figura 17**, donde apreciamos seis bytes con etiquetas que van desde *PN0* hasta *PN5* y que pertenecen al número de secuencia CCMP, siendo los más significativos los que tienen un número mayor en su etiqueta. Por otro lado, podemos observar que los subcampos *Extended IV* y *Key ID* ocupan la misma posición dentro de la cabecera CCMP que en la cabecera TKIP, desempeñando la misma función que en el protocolo TKIP.

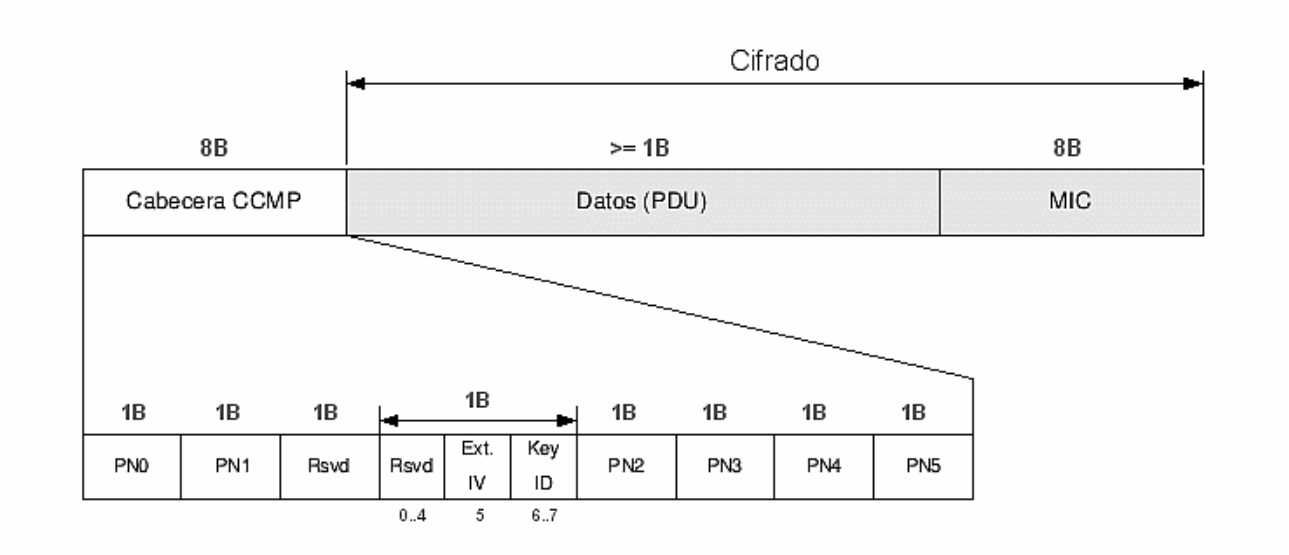

Figura 17: Encapsulamiento de los datos realizado por CCMP

Además, cada estación transmisora debe mantener un contador de secuencia, por cada PTKSA y cada GTKSA basadas en CCMP y asociadas a dicha estación, que almacene el valor del campo PN de la última MPDU de datos transmitida y vinculada a una de estas asociaciones de seguridad. Este contador se incrementa de forma monótona en cada transmisión de una MPDU de datos y se

inicializa al valor uno cuando se instala la primera instancia o cuando se renueva la clave temporal. Del mismo modo, cada estación receptora debería disponer de un contador de repetición por cada PTKSA y cada GTKSA basadas en CCMP, pero también por cada valor de prioridad que la estación pueda distinguir en las tramas recibidas (si no dispone de suficientes contadores, cualquier estación transmisora debería de abstenerse de enviarle tramas en las cuales se especifique la prioridad).

Por tanto, una MPDU protegida con CCMP y recibida por una estación será descartada siempre que el valor del PN de la misma sea menor o igual que el contador de repetición correspondiente mantenido por la estación. En caso contrario, la estación procesará dicha MPDU y actualizará el contador apropiado con el valor del PN de la trama, suponiendo que valide su integridad. Además, el contador de recepción se inicializará a cero cuando se actualice la clave temporal vinculada a la misma asociación de seguridad que el contador. Finalmente, en la **Figura 18** se muestra el diagrama de bloques del procesamiento realizado por CCMP sobre los datos y otros campos de una MPDU. Concretamente, el procesamiento llevado a cabo por los dos bloques de cifrado AES, sobre sus respectivas entradas, es referido en el estándar como *CCM originator processing*.

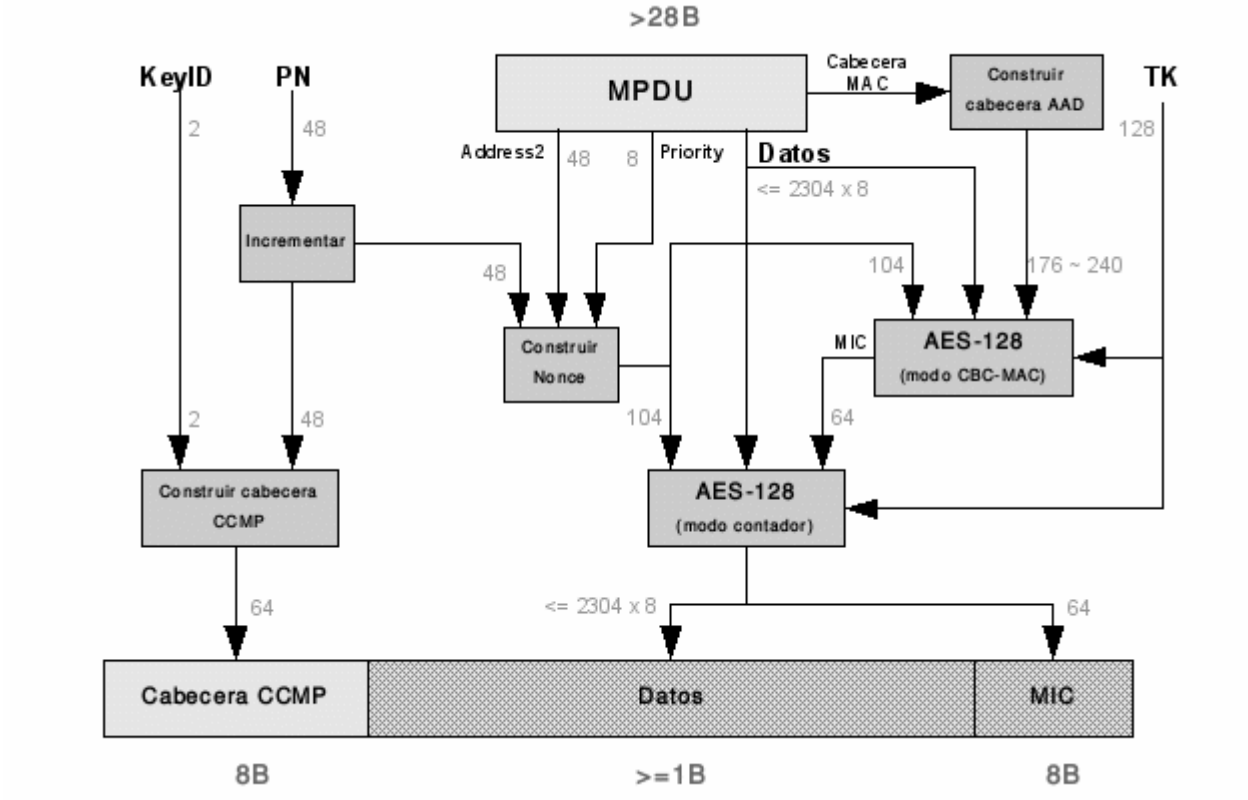

Figura 18: Procesamiento realizado por CCMP

4.4.2 Integridad/Autenticidad provista por CCMP

Estas dos características son verificadas no solo sobre los datos útiles de una trama de datos, sino también sobre un buen número de campos de la cabecera MAC de la MPDU. Estos campos, cuya integridad y autenticidad se resguarda pero no su privacidad, integran los, así denominados, datos de autenticación adicionales (traducción del término inglés: *Additional Authentication Data*, abrev. *AAD*), los cuales, en conjunto, tienen una longitud que oscila entre 22 y 30 bytes, en función de la clase y el formato específicos de la trama de datos. Sin embargo, algunos subcampos de los anteriores campos, en especial aquellos que pueden ser alterados durante sucesivas retransmisiones, no se tienen en cuentan para calcular el MIC sino que, a efectos de este cómputo, son reemplazados por ceros. En la siguiente lista se enumeran todos los campos que, cuando aparecen en una trama de datos, son incluidos en los mencionados datos de autenticación adicionales:

- *Frame Control*, excluyendo los bits que indican el subtipo de trama (subcampo *Subtype*) y los bits asociados a los flags: *Retry*, *PowerMgt* y *MoreData*.
- *Sequence Control*, concretamente el subcampo que numera los fragmentos de una MSDU (esto es, el subcampo *Fragment Number*), puesto que el otro subcampo perteneciente a este campo (que se denomina *Sequence Number*) se enmascara mediante ceros.
- *QoS Control*, en las tramas que incluyen este campo y que, por tanto, especifican parámetros de calidad del servicio, se recuperan los bits del subcampo *TID* (abrev. de *Traffic Identifier*) del citado campo para ser procesados como parte de la cabecera AAD.
- Todas las direcciones MAC de la MPDU (incluyendo el campo *Address 4*, si lo hubiera) se integran en la cabecera AAD y participan en el cómputo del MIC.

Una vez construida la cabecera AAD, se aplica una función de formato a dicha cabecera y a otra información adicional (flags, nonce, longitud de la cabecera y longitud del mensaje) que transforma estos datos en una secuencia de bloques de 128 bits, siendo rellenado con ceros el último bloque, si es necesario, para que esta secuencia tenga un tamaño múltiplo de 16 bytes (véase [Dwo2004]). Por su parte, la carga de datos se divide en bloques de 128 bits y se rellena del mismo modo que la cabecera AAD, hasta que alcance una longitud que sea múltiplo del tamaño de bloque. Después, las secuencias de bloques correspondientes a la cabecera AAD y a la carga de datos se concatenan (en este orden) para producir el mensaje final cuya integridad/autenticidad será comprobada.

El bloque inicial de este mensaje contiene, en primer lugar, un campo de flags de un byte de longitud que informa de la existencia de la cabecera AAD y que codifica los dos parámetros para la aplicación del modo CCM (esto es, los tamaños en bytes de los campos que contienen la longitud del mensaje y el código para la autenticación del mensaje o MIC, respectivamente). A continuación, se ubica el mismo *nonce* que se utiliza para el cifrado en modo Contador y, por último, se reserva un campo de dos bytes para codificar la longitud de la carga de datos de la MPDU. En el siguiente bloque, un campo del mismo tamaño codifica la longitud de la cabecera AAD, a la cual precede. Los restantes datos de esta cabecera ocupan también el tercer bloque de la secuencia, cuyo espacio sobrante, si existe, es rellenado con ceros. Tras el tercer bloque, los siguientes bloques contienen la carga de datos de la MPDU.

Los bloques del mensaje anterior son cifrados mediante el algoritmo AES operando en el modo de encadenamiento de bloques (traducción libre del término: *Cipher Block Chaining*), que consiste en que cada bloque, exceptuando al primero, es transformado aplicándole la operación *xor* con el resultado de cifrar el bloque previo, realizándose tal operación previamente al procesamiento del bloque referido mediante el cifrado AES. Cuando se ha completado el cifrado del último bloque del mensaje, los primeros ocho bytes de dicho bloque son designados como el código de autenticación del mensaje (traducción del término en inglés: *Message Authentication Code*, a cuyo acrónimo hace referencia el modo de cifrado para la autenticación de los datos descrito en esta sección, esto es, el modo *CBC-MAC*).

Como ya se afirmó, CCMP utiliza la misma clave para encriptar un mensaje y para verificar la integridad/autenticidad de dicho mensaje y de los datos de autenticación adicionales, a diferencia de la forma en la que opera TKIP, que divide la TK en tres claves que cumplen distintas funciones. En cambio, CCMP emplea la TK entera, contenida en una PTK o en una GTK, para el desempeño de las funciones anteriores. Otra diferencia notable con respecto a TKIP, es que CCMP comprueba la integridad/autenticidad de cada MPDU individualmente, mientras que el primer protocolo realiza esta verificación sobre una MSDU entera, después de que ésta sea entregada a la subcapa MAC, la cual podría fragmentar la MSDU en varias MPDUs previamente a su transmisión.

4.5 Algunos ataques contra WPA/WPA2

Salvo en contados casos que afectan al protocolo de seguridad TKIP, hasta ahora no se han descubierto vulnerabilidades significativas que atenten contra la confidencialidad, la autenticidad o la integridad provistas mediante los procedimientos especificados en los estándares WPA/WPA2. Casi todos los ataques que, en la práctica, vulneran estas características de seguridad son posibles gracias a una configuración inadecuada de los dispositivos o a una implementación deficiente de los mecanismos que intervienen en los procedimientos que soportan tales características. Por lo tanto, en la mayoría de los casos, la causa de estas debilidades no debería ser achacada directamente a las especificaciones de WPA o del propio estándar 802.11.

Por el contrario, en este estándar se ha prestado poca atención a la disponibilidad del protocolo especificado, hecho que evidencian los numerosos ataques de denegación de servicio descubiertos (por ejemplo, contra la autenticación y la asociación 802.11, la autenticación basada en el estándar IEEE 802.1X, el *4-Way Handshake* o las asociaciones de seguridad basadas en TKIP). Sin embargo, este tipo de ataques no se aborda en este trabajo, ni tampoco los dirigidos a la interacción entre el Autenticador y el AS, puesto que el estándar presupone una comunicación segura entre ambos y que la entidad que desempeña el papel del AS es confiable. De todos modos, se enumeran una serie de ataques que, con mayor o menor efectividad, actúan sobre algunos de los mecanismos clave de la seguridad provista por los estándares WPA/WPA2, como son la autenticación mediante PSK o mediante el estándar 802.1X, el *4-Way Handshake* y el cifrado de los datos efectuado a través de los protocolos de seguridad TKIP o CCMP.

4.5.1 Ataques contra la autenticación mediante PSK

La generación de la PSK mediante una contraseña facilita a los usuarios la configuración de esta clave en un dispositivo 802.11, aunque habilita un ataque de diccionario offline contra dicha clave que, en caso de tener éxito, otorga al atacante el mismo nivel de acceso que a un usuario legítimo que disponga de dicha clave. Naturalmente, solo es posible efectuar el ataque con éxito contra las contraseñas incluidas en el diccionario del atacante, por lo que la probabilidad de éxito del ataque será prácticamente nula contra aquellas contraseñas con la suficiente entropía (por ejemplo, las que contienen letras minúsculas y mayúsculas, números y signos de puntuación) o las que alcanzan una longitud mínima (en el estándar 802.11 se recomienda que contengan al menos 20 caracteres).

Además, para realizar este ataque es necesario conocer el ESSID de la red atacada, de modo que el atacante pueda calcular la PSK a partir de la contraseña escogida y dicho ESSID, empleando para ello la función *PBKDF2*. También son necesarios los dos primeros mensajes de una instancia del *4-Way Handshake* ejecutada por una estación que utilice esta forma de autenticación en dicha red. Por lo tanto, empleando las direcciones MAC del Cliente y del Autenticador, los dos nonces que intercambian ambas entidades (datos que están todos presentes en los dos mensajes anteriores) y la PMK (idéntica a la PSK en este tipo de autenticación) computada en el paso previo, el atacante podrá calcular, por medio de la función PRF correspondiente, el valor de la PTK que se deriva de la contraseña elegida.

A continuación, el atacante puede comprobar si la PTK generada es la correcta (en cuyo caso también lo es, con toda probabilidad, la contraseña de partida) calculando el MIC del segundo o de otro mensaje posterior del *4-Way Handshake*, con la función hash apropiada y la KCK perteneciente a dicha PTK, y verificando seguidamente si este valor coincide con el contenido en el paquete de tipo *EAPOL-Key* asociado a dicho mensaje. Puesto que el cómputo de la PMK conlleva un coste computacional mucho mayor que el de la PTK a partir de la PMK (8192 invocaciones a la función *HMAC-SHA-1*, en el primer caso, frente a 3 o 4 en el segundo), puede incrementarse drásticamente la velocidad del ataque mediante tablas con valores precomputados de la PMK que se obtienen a partir de identificadores de red frecuentemente utilizados (por ejemplo, los ESSIDs preconfigurados por diversos fabricantes de puntos de acceso) y de las contraseñas de un diccionario (algunas tablas de esta clase han sido denominadas erróneamente tablas *Rainbow*, ya que este nombre hace alusión a una variante de ataques TMTO, los cuales están basados en otras técnicas diferentes).

Así que, las 300 claves por segundo que, aproximadamente, puede comprobar un computador de escritorio convencional, o incluso las 1.000 claves por segundo que pueden alcanzar los de mayores prestaciones (estaciones de trabajo, servidores, etc) en el ataque de diccionario, están muy lejos de las cerca de 50.000 claves por segundo que pueden comprobarse gracias al uso de tablas de PMKs precalculadas. No obstante, esta última técnica presenta algunos inconvenientes con respecto a la empleada originalmente, como su aplicabilidad restringida a redes con identificadores específicos, esto es, los ESSIDs utilizados en la construcción de la tablas, así como la duplicidad del espacio de almacenamiento que supone la existencia de entradas en diferentes tablas que corresponden a una misma contraseña.

En el artículo [LT 2010] se propone una mejora para el ataque de diccionario contra la PSK, aplicable a enlaces Wi-Fi protegidos mediante el protocolo CCMP, aunque su efecto prácticamente no resulta apreciable a menos que se efectúe dicho ataque mediante tablas de PMKs precomputadas. Con esta mejora se evita comprobar la corrección de una PMK candidata mediante el cómputo del MIC de un paquete de tipo *EAPOL-Key* transmitido en una instancia exitosa del *4-Way Handshake.* Para este cómputo se requiere la KCK derivada a partir de dicha PMK, en cambio, esta variante nueva del ataque necesita la TK generada a partir de la anterior PMK, proceso que admite un par de optimizaciones, según afirman los autores del citado artículo. En primer lugar, se puede prescindir de la primera iteración de la función *PRF-384*, puesto que no interviene en el cómputo de la TK y, en segundo lugar, se pueden reutilizar ciertos estados internos de la función *HMAC-SHA-1* que son idénticos y que se alcanzan durante la segunda y la tercera iteración de la función *PRF-384*.

Una vez que se ha computado una TK candidata, y dado que es posible reconstruir un bloque de contador cualquiera (involucrado en el cifrado de una MPDU mediante CCMP) a partir de las cabeceras MAC y CCMP de la correspondiente MPDU, se procede a encriptar el segundo bloque de contador (esto es, el bloque cuyo valor del contador es uno) mediante el algoritmo AES y la TK candidata, obteniéndose un keystream con el cual podría haberse cifrado el primer bloque de los datos en claro de dicha MPDU. Esta posibilidad puede comprobarse aplicando la operación *xor* al bloque inicial de los datos cifrados junto con los correspondientes datos en claro supuestamente conocidos (esto es, la cabecera *LLC/SNAP*), recuperando de esta manera un prefijo del auténtico keystream (o, al menos, presuntamente auténtico) que se compara con el keystream producido a partir de la TK calculada.

Si los prefijos de estos dos keystreams coinciden (incluso aunque sus longitudes no alcancen el tamaño de un bloque de cifrado), el atacante puede estar prácticamente seguro de que la clave que ha calculado es idéntica a la clave temporal de sesión con la que se encriptó la MPDU protegida mediante CCMP (esto es, la TK vigente en ese momento). En consecuencia, la PMK a partir de la cual el atacante obtuvo dicha clave también debe coincidir con la PSK configurada por la víctima. En la práctica, este método para verificar una PMK candidata puede ahorrarnos una o incluso dos iteraciones de la función hash *SHA-1* con respecto al método basado en la comprobación del MIC (dependiendo del tamaño del paquete *EAPOL-Key*). No obstante, esta nueva alternativa introduce una operación de cifrado AES adicional, a pesar de la cual, los autores del anterior artículo afirman que sigue siendo más eficiente que el método utilizado previamente.

4.5.2 Ataques contra la autenticación mediante 802.1X

En la enmienda 802.11i se requiere explícitamente que cualquier método EAP que participe en la autenticación 802.1X entre estaciones 802.11 esté basado en algún tipo de autenticación fuerte (por ejemplo, que impida que los ataques del intermediario sean computacionalmente viables) y bilateral, esto es, que exija la autenticación mutua de ambas estaciones. No obstante, en la práctica las implementaciones de estos métodos pueden no ser todo lo estrictas que debieran o también es posible que la configuración de los Clientes o de los Servidores de Autenticación albergue errores u omita ciertas medidas de seguridad, originándose brechas en la seguridad de la red susceptibles de ser explotadas con los ataques apropiados. Esta última posibilidad no es inverosímil ni siquiera en los protocolos de autenticación basados en certificados digitales, ya que una validación insuficiente de un certificado de un Cliente o de un Servidor de Autenticación (por ejemplo, en lo que respecta a la caducidad, la revocación, la cadena de certificación o la identidad de la entidad acreditada) puede habilitar la suplantación de una de las partes por un atacante activo.

Un caso particular de los protocolos de autenticación mediante certificados digitales es el de algunos protocolos que utilizan túneles ya comentados (por ejemplo, PEAP o EAP-TTLS). En estos protocolos, la autenticación del AS se lleva a cabo mediante un certificado digital, gracias al cual se produce, cuando esta autenticación finaliza con éxito, el establecimiento de un túnel entre el Cliente y el AS, que protege la autenticación del Cliente y, en algunos casos, también su identificación. En el artículo [ANN2002] se expone una debilidad que se manifiesta en determinados entornos que emplean algún tipo de autenticación basada en túnel y que posibilita a un atacante la realización de un ataque del intermediario a través del cual puede suplantar a un Cliente.

Para que este ataque pueda llevarse a cabo, el material de claves generado como resultado de una autenticación correcta debe ser originado exclusivamente a partir de la autenticación del AS. Además, el método de autenticación del Cliente debe ser independiente del método de autenticación del AS y no debe existir ningún tipo de interacción entre ambos, como ocurre típicamente con los protocolos de autenticación basados en túnel utilizados en redes Wi-Fi. Finalmente, se requiere un entorno donde el método de autenticación del Cliente pueda ejecutarse de forma aislada, es decir, sin ser precedido por el método de autenticación del AS y sin contar con la protección del túnel que se establece gracias a la contribución del protocolo para la autenticación del AS.

El último artículo citado plantea un escenario vulnerable a un ataque de este tipo, aplicado al caso particular de los protocolos PEAP y EAP-AKA. Sin embargo, los requisitos para llevarlo a la práctica dificultan en gran medida su aplicación a una red Wi-Fi. Por ejemplo, es necesaria una infraestructura de autenticación característica de redes celulares, que debe ser accesible a través de dos tecnologías inalámbricas distintas, como lo son Wi-Fi y alguna otra tecnología propia de redes celulares, como puede ser UMTS. Además, el Cliente objetivo del ataque debe utilizar la tecnología celular junto con el método EAP-AKA para su autenticación, mientras que el atacante-intermediario debe suplantar a la estación base con la que se conecta el Cliente, pero también a dicho Cliente con respecto a la red, con el propósito de reenviar el tráfico de autenticación de éste a través de un túnel establecido con el AS de dicha red mediante PEAP. De esta forma, el atacante podría completar con éxito el intercambio de paquetes correspondiente al método de autenticación PEAP/EAP-AKA y, en consecuencia, podría ser autenticado por la red.

Sin embargo, es mucho más difícil realizar este ataque contra una única red Wi-Fi, puesto que los métodos de autenticación del Cliente que se ejecutan a través de un túnel seguro generalmente no son adecuados para la autenticación de estaciones 802.11 (por ejemplo, no son métodos EAP o no se ajustan a los requisitos de este estándar, como ocurre con PAP o con EAP-GTC). Además, aunque el método de autenticación del Cliente sea seguro, de acuerdo con las especificaciones del estándar debe producir 256 o más bits de material de claves (esto es, la MSK), que solo debe ser conocido por el Cliente y el AS (hasta que se entrega al Autenticador), así que el atacante no podrá obtener la TK derivada de forma indirecta de la MSK y, por tanto, tampoco podrá cifrar/descifrar paquetes cuyo destino/origen es el Cliente (aunque consiga ser autenticado suplantando al Cliente).

Para contrarrestar esta vulnerabilidad, el último artículo mencionado propone varias soluciones. La solución más obvia consiste en impedir el funcionamiento de forma simultánea de los protocolos de autenticación del Cliente en ambos modos, esto es, de forma independiente y mediante un túnel. Otras opciones menos restrictivas se decantan por la vinculación de los métodos de autenticación del Cliente y del AS asociados al mismo protocolo de autenticación basado en túnel, por ejemplo, generando la MSK a partir del material de claves originado por cada método, o bien a partir del material de claves aportado por un método y la clave compartida por el Cliente y el AS para el otro método. Otra alternativa más compleja para conseguir esta vinculación requiere que tanto el Cliente como el AS computen un valor que depende de la clave o del material de claves aportado por cada método, a continuación, ambos entregarán los valores que han computado a una entidad encargada de verificar que son iguales, determinando así el resultado de la autenticación.

4.5.3 Ataques contra el 4-Way Handshake

Este mecanismo ha demostrado su seguridad frente a los ataques del intermediario, de repetición y de falsificación de mensajes, entre otros ataques, gracias a la verificación de un MIC calculado mediante una clave de 128 bits y a la utilización de dos nonces de 256 bits generados por ambas entidades. No obstante, en el artículo [HM 2005] se describe un ataque de reflexión por medio del cual un atacante puede conseguir ser autenticado por una estación 802.11 que desempeñe los roles de Autenticador y de Cliente simultáneamente, en caso de que dicha estación utilice la autenticación mediante PSK y también la misma instancia de esta clave en ambos roles. Este escenario es típico de las redes 802.11 que operan en modo *Ad-Hoc*, donde cada estación asume ambos roles durante la autenticación 802.1X (en caso de que sea necesaria) y la generación de claves, siendo esta última etapa realizada mediante la ejecución concurrente de dos instancias del *4-Way Handshake*, cada una de las cuales es iniciada por una estación distinta.

Bajo estas circunstancias, un atacante podría suplantar a una estación del IBSS con la cual la víctima iniciará una instancia del *4-Way Handshake*. En tal caso, el atacante se limitará a reenviar a la víctima los mensajes que recibe de ésta (realizando las modificaciones apropiadas en la cabecera MAC), adoptando el mismo rol que la víctima, esto es, Autenticador o Cliente, pero como parte de la otra instancia del *4-Way Handshake*. Así pues, repitiendo y alternando de forma apropiada los mensajes de las dos instancias del *4-Way Handshake* en las que participan el atacante y la víctima, éste puede conseguir que las dos instancias sean completadas con éxito, ya que la víctima debería validar correctamente los paquetes de tipo *EAPOL-Key* recibidos del atacante, puesto que fueron originados por ella.

No obstante, aunque el atacante logre autenticarse no conseguirá ninguna información sobre la PMK utilizada por la víctima. Por esta razón, el atacante no está capacitado para calcular la PTK que se deriva de ésta y, por tanto, el ataque resulta inocuo en lo que respecta a la confidencialidad y la integridad de los datos transmitidos por la víctima durante la sesión que comparte con el atacante. Una solución sencilla para evitar este ataque, sugerida en el anterior artículo, consiste en configurar dos instancias diferentes de la PMK en cada estación, de modo que cada instancia de este conjunto de claves intervenga únicamente en una de las dos instancias del *4-Way Handshake*, en la cual la estación actúa como Autenticador o como Cliente.

4.5.4 Ataques contra TKIP

En el año 2008 fue publicado el primer ataque efectivo contra la privacidad provista por TKIP, conocido como ataque *Beck-Tews*, aunque solo afecta a redes Wi-Fi que soportan diferentes niveles de prioridad para las tramas, como las redes compatibles con la enmienda 802.11e. Mediante este ataque puede descifrarse una trama enviada por el punto de acceso a una estación, de forma similar a como lo hace el ataque *Chop-Chop* contra WEP (ya que TKIP también añade al final de la carga de datos el ICV basado en el algoritmo CRC-32), aunque empleando paquetes *MIC Failure Report* para detectar un pronóstico correcto del último byte de los datos. Sin embargo, la velocidad a la que puede descifrarse una trama con este ataque está limitada a un byte por minuto, evitando así activar las contramedidas TKIP que provocarían la renovación de la clave temporal e invalidarían la clave con la que se cifró la trama. Por este motivo, y dado que suele imponerse la caducidad de la clave temporal, este ataque afecta típicamente a tramas de pequeño tamaño o a tramas de las cuales puede deducirse el valor de muchos campos de su carga útil (P. Ej. las que contienen paquetes ARP).

Después de descifrar una trama de datos es posible calcular la clave MIC por medio de la cual se valida la autenticidad de dicha trama (esto es, la clave que protege los datos enviados desde el AP a una estación) gracias al ataque de inversión sobre el algoritmo Michael descrito en [Woo2004]. Entonces, a partir del keystream obtenido al descifrar la trama y su correspondiente IV, el atacante puede encriptar otras tramas arbitrarias construidas por él (suponiendo que no superen el tamaño de la trama descifrada). A su vez, estas tramas falsificadas pueden transmitirse por (hasta un máximo de) siete canales adicionales para el soporte de calidad de servicio (esto es, canales QoS asignados a tramas con distintas prioridades), siempre que estos canales presenten menor cantidad de tráfico que el canal donde se capturó la trama descifrada y, por lo tanto, el valor del TSC contenido en el campo correspondiente al IV de dicha trama no haya sido utilizado aún en estos canales.

En el artículo [Bec2010] se propone combinar el citado ataque con un ataque de fragmentación para obtener keystreams de mayor longitud. Para conseguir esto, se recuperan algunos prefijos de keystreams correspondientes a las cabeceras de varias tramas distintas cuyo texto en claro inicial es conocido. A continuación, el atacante construye una trama de datos con el objetivo de provocar una respuesta previsible en una estación y la transmite en fragmentos, cada uno de los cuales es cifrado con un keystream diferente, por un canal con distinta prioridad y con menor tráfico. En caso de que el atacante pueda inferir el valor de todos los campos de la trama de respuesta, estará capacitado para calcular la clave MIC empleada por dicha estación para la comunicación en sentido STA-AP.

Adicionalmente, la trama de respuesta de dicha estación podría causar el mismo efecto sobre el AP, de forma que el texto en claro de la respuesta del AP pudiese ser anticipado por el atacante, lo que permitiría, en general, obtener un keystream con una longitud mayor que la de cada fragmento.

También en el anterior artículo se detalla el ataque de reinicialización del algoritmo Michael, por medio del cual un atacante, que conozca la clave MIC apropiada, puede concatenar una trama cifrada (en realidad, su carga de datos), que fue enviada por el AP a una estación, con un mensaje escogido por el atacante, de manera que el MIC calculado por la estación receptora sobre los datos resultantes coincida con el MIC de la trama capturada. En muchos casos, esta situación es posible insertando dos palabras (de 32 bits cada una), con los valores adecuados, que se ubicarán entre el mensaje arbitrario y los datos de la trama capturada. Típicamente, estos valores dependen del estado inicial del algoritmo Michael para esta trama (que depende exclusivamente de la clave Michael) y del estado alcanzado por este algoritmo tras procesar con la misma clave el mensaje construido por el atacante. No obstante, existen formas alternativas para calcular los valores de estas dos palabras.

Por consiguiente, una vez que el atacante dispone de un keystream válido y de la clave Michael correspondientes al mismo enlace protegido mediante TKIP (y al mismo sentido de la comunicación en dicho enlace), puede construir una trama arbitraria, encriptarla con ese keystream y transmitirla como si fuese el fragmento inicial de un mensaje (esto es, de una MSDU). Entonces, construye el siguiente fragmento a partir de la trama ya capturada, suponiendo que su IV sea mayor que el IV del fragmento previo, y ambos fragmentos los transmite por un canal para calidad de servicio con menor tráfico que el de la trama capturada (esto es, un canal por el cual no se haya transmitido aún, durante la sesión vigente, el IV asociado al fragmento inicial del mensaje creado por el atacante).

Algunos meses después de la publicación del ataque de Beck-Tews, dos investigadores plantean en su artículo [MO 2009] un nuevo escenario donde puede llevarse a cabo este ataque, así como los ataques relacionados descritos en esta sección. En este nuevo escenario no se requiere la distinción de prioridades, ni los diferentes contadores de secuencia asociados (esto es, distintos contadores de recepción vinculados a la misma clave de cifrado), como ocurre con las estaciones que implementan el protocolo TKIP y el soporte para calidad de servicio conforme a la enmienda 802.11e. Por tanto, un atacante puede prescindir de esta característica cuando se establece como un intermediario de las comunicaciones entre las dos estaciones (ataque MITM) y controla completamente el intercambio de tramas entre ambas (P. Ej. cuando reenvía las tramas de dos estaciones ubicadas más allá de sus respectivos alcances o cuando puede interferir en la recepción de cada trama por cada estación).

En estas circunstancias, el atacante puede descifrar una trama de pequeño tamaño de la cual conoce parcialmente su contenido y obtener la clave Michael utilizada en el MIC que incluye esta trama. Para conseguir esto, el atacante ejecuta el ataque de Beck-Tews e impide la recepción en la estación atacada de otras tramas procedentes de la misma estación que la trama que intenta descifrar (para que no se incremente el contador de recepción y frustre el ataque de Beck-Tews). Si el número de bytes desconocidos de la trama es pequeño, el atacante puede encontrar sus respectivos valores mediante una búsqueda por fuerza bruta, empleando el ICV de la trama capturada (que ha sido previamente descifrado) para evaluar a los candidatos generados en esta búsqueda.

Además, según los autores del anterior artículo, existen situaciones en las que únicamente se desconoce el valor de un byte de los datos útiles de una trama protegida mediante TKIP, por lo que podría efectuarse la verificación de una trama candidata, en un porcentaje alto de casos, con tan solo recuperar el valor del último byte del ICV de la trama capturada y compararlo con el valor del byte homólogo de la trama candidata (reduciendo sensiblemente el tiempo de ejecución del ataque con respecto al ataque de Beck-Tews). Finalmente, tras descifrar la trama capturada, descubrir la clave Michael y recuperar el keystream correspondiente, el atacante puede enviar una versión falsificada y convenientemente modificada de dicha trama a la estación receptora.

4.5.5 Ataques contra CCMP

Tanto en el caso de AES, que fue estandarizado por el NIST tras ser estudiado exhaustivamente por la comunidad criptográfica, como en el caso del modo CCM, cuya seguridad es equiparable a la de otros modos de cifrado en bloque (como demostró Jakob Jonsson), no se han encontrado ataques, debilidades u otras evidencias que comprometan de forma significativa su seguridad. No obstante, en el artículo [JMU2005] se afirma que CCMP es vulnerable a un ataque de tipo *TMTO* (acrónimo de *Time-Memory Trade-Off*). Esta clase de ataques de texto en claro escogido fue ideada por M. E. Hellman, aunque para muchos tipos de cifrado (por ejemplo, cifrado en bloque en modo ECB) es aplicable igualmente en el caso de texto en claro conocido. Esencialmente, un ataque de esta clase consiste en construir una tabla de valores precomputados a partir de un fragmento de texto en claro (típicamente, del tamaño de un bloque del algoritmo de cifrado) que se supone que aparece cifrado en algún mensaje enviado por la víctima. Entonces, el atacante cifra el bloque de texto en claro con diferentes claves de su elección y confía en encontrar alguno de los bloques cifrados resultantes en algún mensaje encriptado por la víctima, deduciendo así la clave de cifrado de dicho mensaje.

Adicionalmente, para reducir el tamaño de la tabla anterior (a costa de aumentar el tiempo de ejecución del ataque) se utiliza una técnica que consiste en transformar un bloque encriptado por el atacante con una clave arbitraria en otra clave de cifrado diferente mediante la, así denominada, función de reducción. Repitiendo estos pasos sucesivamente, se genera una sucesión de claves de las cuales el atacante almacena, en una entrada de la tabla, únicamente las claves primera y última. De esta forma, cuando el atacante obtiene un bloque que ha sido cifrado con una clave desconocida, pero cuyo texto en claro supone que es el mismo que utilizó para construir la tabla, puede aplicar la función de reducción una o más veces a este bloque encriptado y comparar las claves producidas con las últimas claves de cada sucesión que están indexadas en la tabla. Si ocurre una coincidencia, el atacante podrá deducir la clave, con la cual se encriptó dicho bloque, a partir de la clave inicial correspondiente a la clave final coincidente de la mencionada tabla.

Dado que en el modo Contador un bloque de texto en claro no es encriptado directamente por el algoritmo de cifrado subyacente (por ejemplo, por medio de AES), sino aplicándole la función *xor* con el resultado de encriptar un bloque de contador mediante dicho algoritmo, otra premisa para llevar a cabo un ataque TMTO sobre este modo de cifrado es que se pueda predecir el contenido de algún bloque de contador empleado para cifrar el mensaje cuyo texto en claro se conoce. Esta es la cuestión fundamental de la que trata el artículo [JMU2005], donde se explica detalladamente como reconstruir un bloque de contador asociado a una MPDU protegida mediante CCMP, usando para tal fin un conjunto de campos, pertenecientes a dicha MPDU, que son transmitidos en claro.

En este protocolo cada bloque de contador contiene un campo de flags con un valor constante, mientras que el campo que contiene el índice o contador de bloque se inicializa a uno en el bloque inicial utilizado para cifrar los datos de una MPDU cualquiera y se incrementa en una unidad en cada uno de los sucesivos bloques de contador que se emplean para el mismo propósito. Por tanto, el único campo que difiere, en los bloques de contador con el mismo índice asociados a diferentes MPDUs, es el *nonce*. Por su parte, el *nonce* se compone de un valor que representa la prioridad de la MPDU (subcampo *TID,* o bien se fija a cero en las tramas sin prioridad) seguido por la segunda dirección MAC de la MPDU. Ambos datos están presentes en dos campos de la cabecera MAC de la MPDU, la cual se transmite en claro. Del mismo modo se transmite el campo *PN* (el contador de secuencia empleado por CCMP) incluido en la cabecera CCMP, cuyo valor también se incorpora al *nonce* en último lugar. En consecuencia, un atacante puede reconstruir el contenido de un bloque de contador cualquiera asociado a una MPDU protegida mediante CCMP que haya capturado.

En cuanto a la complejidad de esta clase de ataques, Hellman estimó (en su artículo [Hel1980]) una solución cercana a la óptima, con respecto a la probabilidad de éxito del ataque frente a su coste computacional, consistente en emplear $N^{1/3}$ tablas diferentes con distintas funciones de reducción, acumulando aproximadamente $N^{2/3}$ entradas en total. Esta configuración conlleva una complejidad temporal (en términos de operaciones de cifrado y de reducción) de $O(N^{2/3})$, siendo N el tamaño del espacio de claves. Como consecuencia de esto, un ataque de esta clase contra la clave temporal de 128 bits, correspondiente a una sesión protegida mediante CCMP, demandaría una gran cantidad de tiempo y de recursos (del orden de 2^{85} operaciones y el mismo número total de entradas entre todas las tablas), lo que significa que prácticamente sería imposible llevarlo a cabo en la actualidad.

Pero incluso aunque fuese viable en la práctica, dicho ataque solo podría ser efectuado contra el tráfico generado por una única estación, debido a que el *nonce* de cada bloque de contador incluye la segunda dirección MAC de la trama para cuyo cifrado se usa tal bloque (típicamente, la dirección MAC de la estación que transmite la trama al medio inalámbrico). Por la misma razón, solo tienen interés para el atacante las MPDUs que incorporan el mismo valor del campo PN que el *nonce* del bloque de contador a partir del cual fueron construidas las tablas para el ataque TMTO (recordemos que el PN se inicializa a uno siempre que la clave temporal es instalada o bien cuando es renovada y que, a continuación, se incrementa de forma monótona en las sucesivas MPDUs correspondientes a la misma sesión y transmitidas por la misma estación). Además, aunque el ataque fuese ejecutado con éxito contra una MPDU, únicamente permitiría descifrar el tráfico correspondiente a la misma sesión de dicha MPDU, esto es, el tráfico encriptado con la misma clave temporal.

Otro ataque de texto en claro conocido aplicable al cifrado aditivo y, en particular, al modo Contador se denomina ataque de colisión de claves y fue inventado por E. Biham. Para llevarlo a cabo, un atacante debe cifrar el mismo mensaje con una gran cantidad de claves distintas generadas aleatoriamente o escogidas arbitrariamente por él mismo. Además, dicho atacante debe reunir una colección numerosa de textos cifrados por la víctima con diferentes claves (que son desconocidas por el atacante) correspondientes al mensaje en claro anterior. Cuando el producto de los cardinales de ambos conjuntos de claves (esto es, las claves conocidas, escogidas por el atacante, y las claves utilizadas por la víctima y desconocidas por el atacante) supera el cardinal del espacio de claves, existe una alta probabilidad de que ambos conjuntos tengan una o más claves en común. En cuyo caso, deberían coincidir algunos textos cifrados por el atacante con otros textos encriptados por la víctima, lo que permitiría al atacante descubrir una o más claves de cifrado usadas por la víctima con tan solo encontrar esos textos idénticos.

En cambio, en el artículo [FM 2000] se propone reemplazar la precondición del texto en claro conocido para el ataque de colisión de claves de Biham por una precondición más débil que también comprende a los mensajes cuyo texto en claro es generado por una fuente con redundancia lineal conocida. Dicho de otra manera, se sugiere aprovechar la probable circunstancia de que el texto en claro correspondiente a los mensajes encriptados por la víctima satisfaga determinadas ecuaciones lineales a nivel de bits. En la práctica, esta condición es verificada por casi todos los mensajes cuya información es transportada mediante paquetes correspondientes a la familia de protocolos TCP/IP, ya que estos paquetes generalmente presentan valores conocidos en diversos campos pertenecientes a las cabeceras de estos protocolos, o bien determinados bits de algunos campos pueden deducirse de los valores de otros campos.

En consecuencia, dos vectores que representan a dos mensajes en claro que se ajustan al mismo patrón, respecto a estas dependencias lineales, producirán el mismo resultado al ser multiplicados por una matriz característica de dichas relaciones lineales. Además, el resultado de la multiplicación de un vector, correspondiente a un mensaje en claro, por la anterior matriz característica es referido, por los autores del mencionado artículo, como el *hallmark* del mensaje (en el sentido de una marca o sello distintivo). Ampliando este concepto a cualquier mensaje o secuencia de datos, también debe verificarse que dos mensajes cifrados con el mismo *keystream* (o, en este caso, bloques de contador idénticos encriptados con la misma clave AES) producirán la misma marca cuando ocurre lo mismo con sus respectivos textos en claro.

Por otra parte, combinando mediante la operación *xor* una marca conocida de algún mensaje de texto en claro, que el atacante supone que corresponde a cierto mensaje encriptado con una clave desconocida, y la marca producida por este mensaje cifrado se obtendrá la marca del keystream generado por la anterior clave desconocida. Por otro lado, el atacante puede calcular la marca de un fragmento de keystream, originado a partir de una clave arbitraria, y compararla con la marca del keystream de un mensaje cifrado por la víctima. En caso de que las dos marcas coincidan, es muy probable (dependiendo de la cantidad y de la especificidad de las relaciones lineales) que las claves que originaron ambos keystreams sean idénticas.

Entonces, comparando las marcas de numerosos keystreams, generados por el atacante a partir de claves distintas, con las marcas de los keystreams de muchos paquetes encriptados por la víctima (cuyos textos en claro verifican ciertas relaciones lineales) con diferentes claves de sesión es posible encontrar algunas coincidencias que revelen algunas claves empleadas por la víctima. Disponiendo

de un número muy grande de ejemplares de las dos clases (por ejemplo, 2^{64} keystreams y la misma cantidad de paquetes encriptados con CCMP), la probabilidad de que al menos dos keystreams de ambas clases hayan sido producidos por una misma clave es alta. Sin embargo, el atacante no puede prever las claves generadas por él que serán comunes con las de la víctima y posiblemente tampoco podrá almacenar todos los paquetes correspondientes a todas las sesiones en las cuales la víctima utilizó alguna clave de cifrado asociada a algún paquete capturado por el atacante.

Finalmente, en el mismo artículo se pretende ampliar todavía más la aplicabilidad de este ataque incluyendo los casos en los cuales las relaciones lineales no se verifican siempre. Estos casos son abordados por los autores a través de un enfoque que considera a las anteriores relaciones lineales como ecuaciones lineales probabilísticas. Además, estos mismos autores proponen paliar la pérdida de efectividad de esta variante del ataque de colisión de claves, en los casos mencionados, mediante el uso de algún código corrector de errores aplicado a las marcas de los keystreams. No obstante, aunque no resultaría difícil para un atacante realizar dicha codificación a partir de los keystreams generados por él mismo, el proceso inverso, esto es, la obtención de palabras de este código a partir de los mensajes encriptados por la víctima podría no ser viable. En opinión del autor de este trabajo, esta posibilidad es la más plausible si consideramos como la fuente de los errores a los mensajes de texto en claro que no satisfacen las anteriores relaciones lineales y asumimos que estos errores se propagarán inexorablemente a los mensajes cifrados por la víctima.

Capítulo 5

5. Aplicación práctica

Para aprovechar los conocimientos teóricos adquiridos sobre seguridad en redes Wi-Fi durante la elaboración de este trabajo, se ha desarrollado una aplicación que implementa un ataque sencillo, aunque no muy conocido, sobre esta clase de redes. Este ataque está dirigido contra el protocolo de seguridad WEP, cuyas debilidades ya han sido puestas de manifiesto y explotadas de forma práctica mediante numerosas herramientas. No obstante, el ataque seleccionado para su implementación en esta aplicación, descrito por su inventor: *W. A. Arbaugh* como un ataque inductivo de texto en claro escogido, ha tenido muy poca repercusión a nivel de publicaciones científicas y también en lo que respecta a las aplicaciones más populares destinadas a atacar el protocolo WEP. Este hecho resulta increíblemente sorprendente cuando se comprueba la simplicidad, la facilidad de implementación y la eficiencia de este ataque comparado con otras alternativas similares (como el ataque *Chop-Chop* de *KoreK* o el ataque de fragmentación de *Bittau*).

En concreto, el ataque inductivo de *Arbaugh* es un ataque de falsificación de mensajes (esto es, un ataque que facilita la inserción o fabricación de mensajes falsos), lo que faculta al atacante para construir un mensaje con un texto en claro arbitrario, de tal forma que dicho mensaje pueda superar los mecanismos incorporados por el protocolo WEP para proteger la privacidad y la integridad de los datos, evitando así que la víctima descubra que se trata de una falsificación. La razón por la que el citado ataque recibe el calificativo de inductivo se debe a que posibilita la extensión de un prefijo conocido de N bytes de longitud de un keystream válido, consiguiendo que este prefijo alcance los $N+1$ bytes de longitud en un paso exitoso del ataque. Por lo tanto, en cada paso del ataque se intenta determinar el valor del *(N+1)-ésimo* byte del keystream, gracias a un procedimiento de prueba-error mediante el cual se comprueba, en el peor caso, cada uno de los 256 valores posibles para tal byte. Además, para que esta comprobación pueda llevarse a cabo, se requiere que alguna estación actúe a modo de oráculo o, más concretamente, que reaccione de forma distinta cuando reciba un mensaje con el valor correcto para ese byte.

La aplicación desarrollada, que se ha denominado ***Injection Wizard*** y se ha publicado en la Web bajo licencia GPL ver. 2, no se limita a implementar una prueba de concepto del ataque inductivo de *Arbaugh*, sino que incluye un completo repertorio de funciones destinadas a facilitar la inyección de tramas de datos en una red Wi-Fi protegida mediante WEP. Gracias a la inyección de tráfico de esta clase, se generan nuevas claves RC4 para cada trama, a partir de los correspondientes IVs que están contenidos en las cabeceras WEP de tales tramas. A su vez, estas claves RC4 originan nuevos keystreams para el cifrado de los datos transportados por estas tramas. De esta manera, los ataques estadísticos contra la clave WEP basados en el conocimiento de tales IVs, que implementan muchas herramientas conocidas y de libre distribución, pueden ser acelerados considerablemente.

Algunos aspectos que se han priorizado durante el diseño de la aplicación han sido la **facilidad de uso**, gracias a la interacción con el usuario a través de una interfaz gráfica y a la automatización de todos los pasos previos a la inyección, la **robustez** frente a los errores de envío y de recepción, gracias a la configuración de determinados parámetros de transmisión y también a diversas técnicas para la recuperación ante ciertos errores de comunicación y, finalmente, la **separación** entre clases, procedimientos y módulos correspondientes a la interfaz de usuario y los estrictamente vinculados a la ejecución del ataque o a ciertas funciones directamente relacionadas con dicho ataque. Entre estas funciones asociadas al ataque podemos mencionar: la configuración de los parámetros del ataque (por ejemplo, la interfaz de red y ciertas particularidades de su driver, la tolerancia a los errores de transmisión, etc), la selección de la red inalámbrica atacada y la búsqueda de una estación que facilite la inyección de tráfico nuevo en dicha red. De todas estas funciones, las más importantes se describen en el modelo de casos de uso de la siguiente sección.

5.1 Modelo de casos de uso

El modelo de casos de uso de un sistema especifica la funcionalidad esencial de dicho sistema, la cual se deriva de los requisitos funcionales del sistema. Por lo tanto, este modelo está enfocado en la descripción del comportamiento del sistema tal y como es percibido por los usuarios finales, analistas y encargados de las pruebas de validación. Además, se utiliza como un instrumento para facilitar la conceptualización y la comprensión de un sistema, así como la comunicación entre los desarrolladores y los expertos del negocio. Concretamente, en [BJR1999] se define un **caso de uso** como la descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable y de interés para un actor particular.

En el contexto de este tipo de modelado, un **actor** representa un conjunto coherente de roles que desempeñan los usuarios finales o bien los sistemas externos cuando interaccionan con el sistema en cuestión del modo especificado en algún caso de uso. Preferentemente, un modelo de casos de uso enumera los actores principales del sistema, que son aquellos que persiguen un objetivo que puede ser satisfecho por el sistema, siendo además este objetivo la razón principal por la cual se define y se especifica el correspondiente caso de uso. Los restantes actores que pueden participar en un caso de uso, ya se trate de usuarios finales o de sistemas externos, son denominados actores secundarios. No obstante, aunque los actores se consideran entidades externas al sistema en cuestión pueden ser representados mediante clases en los modelos de análisis o de diseño.

Otros aspectos generales a tener en cuenta en el modelado de casos de uso son el **ámbito** y el **nivel de abstracción** o **granularidad** de un caso de uso. En cuanto al ámbito, la definición de caso de uso expuesta anteriormente se ajusta a la descripción de una aplicación o de un sistema basado en software, ya que se centra en la funcionalidad de éste, por lo que el ámbito se establece en el nivel de **sistema**. No obstante, en ocasiones es necesario describir procesos del negocio en los que está involucrado el sistema, pero que también implican a individuos, a entidades o a otros sistemas ajenos al sistema en cuestión o que no interactúan directamente con éste. En este caso, el ámbito abarcado trasciende al sistema en cuestión y corresponde al nivel de **organización**. Por el contrario, también es posible restringir el estudio a cierta parte o a cierto subsistema del sistema en cuestión, en cuyo caso estará limitado al ámbito del nivel de **componente**.

Además de la taxonomía referida al ámbito, *Alistair Cockburn* estableció distinciones entre los casos de usos basándose en su nivel de abstracción o granularidad. Así pues, en lo que respecta a la granularidad, los casos de uso más comunes, esto es, los que se ajustan a la definición ya enunciada, son los de nivel de **usuario**, ya que están enfocados en los objetivos de los actores que interactúan con el sistema en cuestión. En cambio, los casos de uso de nivel de **resumen** presentan un nivel de abstracción mayor y suelen utilizarse para cometidos como: organizar o agrupar casos de nivel de usuario afines, indicar el secuenciamiento de los casos de uso de nivel de usuario correspondientes al ciclo de vida de un conjunto de objetivos relacionados, etc. Por el contrario, los casos de uso de menor nivel de abstracción generalmente se clasifican dentro del nivel de **subfunción** y típicamente expresan objetivos o procedimientos que se requieren, en ocasiones, para llevar a cabo uno o varios casos de uso de nivel de usuario.

Mediante un **diagrama de casos de uso** se pueden representar y visualizar algunos aspectos relevantes de un modelo de casos de uso, como los actores que intervienen en la ejecución de los casos de uso y las relaciones semánticas entre los casos de uso del modelo. La participación de un actor en un caso de uso se representa mediante una relación de **asociación** que conecta a estas dos entidades (en algunos diagramas esta relación se ilustra con una flecha que parte desde la entidad que inicia la interacción). La **especialización** o **generalización** de casos de usos se muestra de la misma forma y tiene la misma semántica que la relación equivalente entre clases. Adicionalmente, cuando un caso de uso incluye el comportamiento de otro caso de uso (generalmente el caso de uso incluido se especifica con menor granularidad) se representa mediante una relación de **dependencia** con el estereotipo *include*, que parte desde el caso de uso incluyente hasta el caso de uso incluido. Mientras que si un caso de uso describe un comportamiento opcional o alternativo y está incluido dentro de otro caso de uso más abstracto, se enlaza con éste mediante una relación de dependencia con el estereotipo *extends*, la cual apunta hacia el caso de uso extendido por el primero.

La secuencia de acciones que describe el comportamiento típico o esperado de un caso de uso constituye el **escenario principal de éxito** (abreviadamente E.P.E.) de éste, que debería incluirse en la especificación textual de ese caso de uso. Los casos de uso incluidos dentro de otro deberían ser referenciados en diferentes pasos de esta secuencia, mientras que los casos de uso que extienden a otro mayor se detallan como subsecuencias alternativas de la secuencia principal, apareciendo justo después del E.P.E. Otras características de un caso de uso que pueden incluirse en su especificación textual son las condiciones que deben satisfacerse antes o después de la ejecución del caso de uso, de manera que cuando son especificadas se exponen en los apartados denominados **precondición** y **garantía** del caso de uso, respectivamente.

El diagrama de casos de uso de la aplicación, desarrollada como parte de este trabajo, se ilustra en la **Figura 19**, aunque solo muestra las extensiones comunes a varios casos de uso. También se ha omitido el ámbito en la especificación textual de los casos de uso, puesto que para todos ellos es de nivel de sistema, así como el actor principal, debido a que el único actor considerado es el usuario que ejecuta la aplicación. En cambio, al final de cada caso de uso se muestra su garantía de éxito, esto es, la condición que se verifica cuando el E.P.E. del caso de uso concluye satisfactoriamente. Finalmente, cada alternativa a un paso del E.P.E. (esto es, cada extensión) se denota con una letra distinta, mientras que los pasos correspondientes a una misma alternativa utilizan la misma letra. Además, la notación *N..M* se aplica a un paso opcional de un caso de uso que reemplaza a cualquier paso del E.P.E. cuyo número se encuentra entre *N* y *M* y también a los pasos que le siguen.

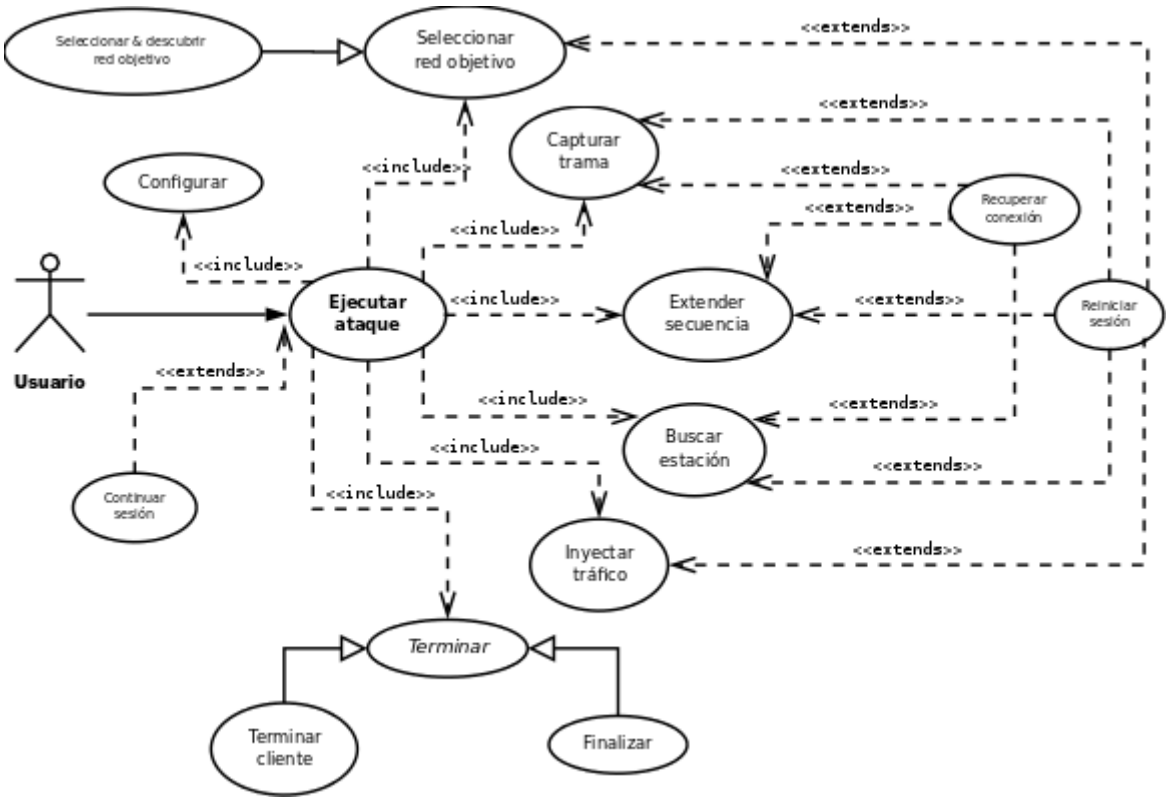

Figura 19: Diagrama de casos de uso

Nombre	**Ejecutar ataque**
Nivel	Resumen
Descripción	Agrupa las acciones esenciales de las que consta una sesión típica en la que se utiliza la aplicación para realizar un ataque de inyección de tráfico en una red Wi-Fi protegida mediante WEP.
Precondición	Ninguna.
E.P.E.	1. *Configurar* 2. *Seleccionar red objetivo* 3. *Capturar trama* 4. *Extender secuencia* 5. *Buscar estación* 6. *Inyectar tráfico* 7. *Terminar*
Extensiones	1..7 || : *Continuar sesión* (véase más adelante)
Garantía	Inyección de tráfico WEP en la red atacada.

Nombre	**Configurar**
Nivel	Usuario
Descripción	Configuración de diversos parámetros de operación de la aplicación relacionados con la ejecución del ataque.
Precondición	Despliegue y ejecución de forma adecuada de los componentes ejecutables que forman parte de la aplicación.
E.P.E.	1. El usuario selecciona la interfaz de red inalámbrica por medio de la cual se realiza el ataque y varias características relacionadas con los datos procesados por su controlador. 2. El usuario indica un rango de direcciones de red en el cual es posible que estén incluidas las direcciones de los equipos de la red atacada y también escoge una estrategia de búsqueda de tales equipos. 3. El usuario establece los parámetros relativos a la gestión de los errores de envío y recepción de tramas, como son el tiempo de espera para una trama y el número de reintentos en la transmisión de una trama sin respuesta.
Extensiones	4. La aplicación almacena en memoria secundaria, a petición del usuario, las opciones de configuración establecidas, de forma que estén disponibles en posteriores sesiones.
Garantía	La aplicación conservará y utilizará los parámetros especificados por el usuario en las siguientes fases del ataque.

Nombre	**Seleccionar red objetivo**
Nivel	Usuario
Descripción	El usuario selecciona una de las redes Wi-Fi protegidas mediante WEP, que han sido detectadas por la aplicación, con el propósito de que dicha red se convierta en el objetivo del ataque y la aplicación se una a ella.
Precondición	Caso de uso *Configurar* completado con éxito.
E.P.E.	1. El usuario solicita la detección de redes Wi-Fi. 2. La aplicación muestra una lista de redes Wi-Fi protegidas con WEP que ha detectado mediante la interfaz de red configurada. 3. El usuario elige una red de esta lista para que se convierta en el objetivo del ataque. 4. La aplicación se une a la red seleccionada.
Extensiones	2.a La aplicación no detecta ninguna red Wi-Fi protegida con WEP, por lo que vuelve al estado previo justo antes del inicio del E.P.E. 3.b El usuario no está satisfecho con las redes detectadas, así que vuelve a iniciar el caso de uso. 4.c La aplicación no recibe confirmación de la unión con la red objetivo, así que vuelve al estado correspondiente al 2º paso del E.P.E. 1..4 || : *Reiniciar sesión* (véase más adelante).
Garantía	La aplicación únicamente interacciona con la red objetivo seleccionada, salvo que vuelva a modificarse la configuración de la aplicación.

Nombre	**Seleccionar y descubrir red objetivo**
Nivel	Usuario
Descripción	El usuario selecciona una red con ESSID oculto de entre las redes Wi-Fi protegidas con WEP detectadas por la aplicación, de forma que la aplicación intente unirse a esta red y se convierta en el objetivo del ataque.
Precondición	La misma que el caso de uso *Seleccionar red objetivo*.
E.P.E.	1. Igual que en el caso de uso *Seleccionar red objetivo*. 2. Igual que en el caso de uso *Seleccionar red objetivo*. 3. El usuario elige una red, cuyo ESSID está oculto, de la lista de redes que muestra la aplicación para que se convierta en el objetivo del ataque. 4. La aplicación comienza la detección del ESSID de la red objetivo, por lo que permanece a la espera de que dicho ESSID sea difundido. 5. La aplicación descubre el ESSID de la red objetivo, a continuación lo notifica al usuario y se une a esta red.
Extensiones	2.a Igual que en el caso de uso *Seleccionar red objetivo*.. 3.b Igual que en el caso de uso *Seleccionar red objetivo*. 4.c El usuario interrumpe la detección del ESSID y la interacción continúa a partir del 2º paso del E.P.E. 5.d Igual que el paso *4.c* del caso de uso *Seleccionar red objetivo*. 1..5 || : *Reiniciar sesión* (véase más adelante).
Garantía	Igual que en el caso de uso *Seleccionar red objetivo*.

Nombre	**Capturar trama**
Nivel	Usuario
Descripción	El usuario escoge una trama de datos que le ofrece la aplicación con el propósito de recuperar un prefijo de la secuencia seudoaletoria con la que se ha cifrado dicha trama
Precondición	Caso de uso *Seleccionar red objetivo* completado con éxito.
E.P.E.	1. El usuario activa la captura de tramas que ofrece la aplicación. 2. La aplicación muestra al usuario algunos detalles de una trama de datos cifrada con WEP que ha capturado y solicita su aprobación para usar la trama en el siguiente paso del ataque. 3. El usuario confirma la solicitud de la aplicación para que la trama sea utilizada en el siguiente paso. 4. La aplicación utiliza la trama anterior para extraer un prefijo de la secuencia seudoaleatoria de cifrado y lo notifica al usuario.
Extensiones	2.a El usuario cancela la captura de tramas y la aplicación vuelve al estado inicial de este caso de uso. 3.b El usuario rechaza la trama ofrecida por la aplicación y ésta continúa la captura del mismo modo que tras el primer paso del E.P.E. 4.c La aplicación no recibe la confirmación de que el prefijo de la secuencia seudoaleatoria extraído es correcto, así que lo notifica al usuario y vuelve al estado inicial de este caso de uso.

	4.d *Recuperar conexión* (véase más adelante). 1..4 || : *Reiniciar sesión* (véase más adelante).
Garantía	La aplicación establece la secuencia seudoaleatoria, cuyo prefijo acaba de obtener, como base para llevar a cabo el ataque, hasta el momento en el que vuelva a ejecutarse este caso de uso.

Nombre	**Extender secuencia**
Nivel	Usuario
Descripción	La aplicación ejecuta el ataque inductivo de *Arbaugh* para aumentar la longitud del prefijo obtenido de la secuencia seudoaleatoria, prolongando este ataque hasta que la longitud de la secuencia descubierta permita el cifrado de un paquete ARP (junto con la cabecera *LLC/SNAP* y el *ICV*).
Precondición	Caso de uso *Capturar trama* completado con éxito.
E.P.E.	1. El usuario desencadena el inicio del ataque mediante su interacción con la interfaz de usuario de la forma adecuada. 2. La aplicación aumenta la longitud de la secuencia seudoaleatoria obtenida en un byte en cada paso del ataque, a la vez que muestra el valor de cada byte descubierto al usuario. 3. La aplicación detiene el ataque y lleva a cabo su finalización cuando la secuencia seudoaleatoria descubierta alcanza una longitud de 40 bytes.
Extensiones	2.a El usuario cancela el ataque y la aplicación reacciona retrocediendo hasta el estado inicial del caso de uso, aunque mantiene la secuencia de cifrado recuperada hasta este momento. 2.b *Recuperar conexión* (véase más adelante). 1..3 || : *Reiniciar sesión* (véase más adelante).
Garantía	La longitud del prefijo de la secuencia seudoaleatoria, que la aplicación obtuvo antes de iniciar este caso de uso, es incrementada hasta igualar el tamaño de la carga de datos de una trama, protegida con WEP, que transporta un paquete ARP.

Nombre	**Buscar estación**
Nivel	Usuario
Descripción	La aplicación intenta descubrir una estación conectada a la red objetivo con una dirección IP perteneciente al rango configurado previamente, para lo cual realiza un sondeo con paquetes ARP.
Precondición	Caso de uso *Extender secuencia* completado con éxito.
E.P.E.	1. El usuario transmite la orden de inicio de la búsqueda mediante la interfaz de usuario. 2. La aplicación realiza una búsqueda de estaciones con una dirección IP comprendida dentro del rango establecido, empleando para ello el método de búsqueda configurado, y al mismo tiempo informa al usuario sobre el progreso de la búsqueda. 3. La aplicación finaliza la búsqueda tras detectar a una estación en la red y registrar su dirección IP, informando al usuario de este suceso.

Extensiones	2.a El usuario cancela la búsqueda en curso, lo que acarrea el restablecimiento del estado en el que se encontraba la aplicación al iniciarse el caso de uso. 2.b *Recuperar conexión* (véase más adelante). 3.c La aplicación no detecta a ninguna estación después de sondear el espacio de búsqueda configurado, por lo que informa al usuario de esta situación y vuelve al estado en el que se inició el caso de uso. 1..3 || : *Reiniciar sesión* (véase más adelante).
Garantía	La aplicación descubre una dirección IP que es muy probable que corresponda a una estación perteneciente a la red que responde a peticiones ARP.

Nombre	**Inyectar tráfico**
Nivel	Usuario
Descripción	La aplicación construye e inyecta paquetes de tipo ARP, destinados a la dirección IP descubierta anteriormente, con el objetivo de que una estación responda a estos paquetes y genere tráfico encriptado con diferentes secuencias de cifrado.
Precondición	Caso de uso *Buscar estación* completado con éxito.
E.P.E.	1. El usuario inicia la inyección de tráfico a través de su interacción con la interfaz de usuario. 2. La aplicación inyecta paquetes de tipo ARP dirigidos a la dirección IP de una presunta estación de la red objetivo y muestra la cantidad de paquetes de este tipo recibidos frente a los enviados.
Extensiones	2.a El usuario cancela la inyección de paquetes después de haberse iniciado y el estado de la aplicación se revierte hasta el estado inicial del caso de uso. 1..3 || : *Reiniciar sesión* (véase más adelante).
Garantía	La aplicación inyecta paquetes ARP que, suponiendo que sean recibidos por una estación con la dirección IP apropiada, deberían incrementar el tráfico de la red objetivo y provocar la generación de nuevas secuencias de cifrado.

Nombre	***Terminar*** (caso de uso abstracto)
Nivel	Usuario
Descripción	El usuario completa su interacción con la aplicación, así que solicita que finalice la ejecución de la interfaz de usuario y, posiblemente también, la sesión que es mantenida por la aplicación.
Precondición	Todos los componentes ejecutables de la aplicación se encuentran en ejecución en este momento.
E.P.E.	1. El usuario lleva a cabo la acción de la interfaz de usuario que invoca este caso de uso. 2. La interfaz de usuario notifica a los restantes componentes ejecutables las instrucciones para su finalización o su desconexión con respecto a dicha interfaz y, a continuación, procede a terminar su ejecución.
Extensiones	Ninguna
Garantía	La interfaz de usuario y, posiblemente otros componentes, terminan su ejecución.

Nombre	**Terminar cliente**
Nivel	Usuario
Descripción	El usuario cesa temporalmente su interacción con la aplicación, pero desea que la aplicación conserve los datos de la sesión actual y que continúe con el cómputo realizado en ese momento.
Precondición	Igual que en el caso de uso *Terminar*.
E.P.E.	1. Igual que en el caso de uso *Terminar*. 2. La interfaz de usuario informa a los otros componentes ejecutables de su próxima terminación, se desvincula de éstos y los mensajes que estaban destinados a la interfaz de usuario se vuelcan por la salida estándar. 3. La interfaz de usuario termina su ejecución.
Extensiones	Ninguna
Garantía	Finaliza la ejecución de la interfaz de usuario pero el procesamiento en curso y los datos de la sesión actual se mantienen.

Nombre	**Finalizar**
Nivel	Usuario
Descripción	El usuario solicita que finalice la ejecución de todos los componentes ejecutables de la aplicación y que se liberen los recursos utilizados por éstos.
Precondición	Igual que en el caso de uso *Terminar*.
E.P.E.	1. Igual que en el caso de uso *Terminar*. 2. La interfaz de usuario solicita a los otros componentes ejecutables que concluyan la comunicación con ésta y que finalicen su ejecución, tras esta acción, la interfaz de usuario hace lo mismo. 3. Los restantes componentes ejecutables acaban su ejecución, bien de forma inmediata o cuando alcancen un estado que permita su cancelación.
Extensiones	Ninguna
Garantía	Todos los componentes ejecutables de la aplicación finalizan su ejecución.

Nombre	**Continuar sesión**
Nivel	Subfunción
Descripción	La interfaz de usuario se sincroniza con una sesión iniciada previamente.
Precondición	Se está ejecutando una sesión de la aplicación.
E.P.E.	1. El usuario inicia la ejecución de la interfaz de usuario. 2. La interfaz de usuario detecta la sesión iniciada, se sincroniza con ella y continúa la ejecución en la misma etapa en que se encuentra dicha sesión.
Extensiones	Ninguna
Garantía	El usuario continúa la interacción con la aplicación en el mismo punto en que se encontraba la sesión previamente iniciada.

Nombre	**Reiniciar sesión**
Nivel	Subfunción
Descripción	El usuario solicita abortar la operación en curso y volver a ejecutar el caso de uso *Configurar*.
Precondición	Se está ejecutando algún caso de uso de nivel de usuario distinto de *Configurar* y *Terminar*.
E.P.E.	1. El usuario invoca esta función a través de alguna forma de interacción con la interfaz de usuario, que debe estar disponible después de la ejecución del caso de uso *Configurar*. 2. La aplicación aborta la operación en curso y seguidamente descarta todos los datos correspondientes a la sesión actual, excepto los datos que fueron establecidos mediante el caso de uso *Configurar*, el cual se convierte en el nuevo escenario de la interacción con el usuario.
Extensiones	Ninguna
Garantía	El usuario puede modificar nuevamente la configuración de la aplicación gracias a la restauración del caso de uso *Configurar*.

Nombre	**Recuperar conexión**
Nivel	Subfunción
Descripción	La aplicación detecta la desconexión con el punto de acceso de la red inalámbrica, por lo que intenta recuperar dicha conexión.
Precondición	La aplicación no ha recibido ninguna trama de respuesta, o bien ha recibido una notificación de desconexión, por parte del punto de acceso
E.P.E.	1. La aplicación detecta la interrupción de la conexión y solicita permiso al usuario para intentar restablecerla. 2. El usuario accede a que se efectúe el intento para recuperar la conexión. 3. La aplicación restablece la conexión con el punto de acceso y continúa su ejecución tras el punto de extensión del E.P.E. desde el cual fue invocado este caso de uso.
Extensiones	2.a El usuario renuncia a recuperar la conexión, provocando que la aplicación reanude la interacción con el usuario desde el primer paso del E.P.E. del caso de uso desde el cual se invoca a éste.
Garantía	Restablecimiento de la conexión con el punto de acceso.

5.2 Modelo de diseño

Durante la fase del diseño de la aplicación, los requisitos del sistema proporcionan el punto de partida para elaborar una descripción de la estructura interna del software que facilite, en las fases siguientes, el desarrollo de una implementación que permita satisfacer estos requisitos. Además de los requisitos funcionales, compilados en los casos de uso, y las entidades abstraídas por medio del estudio del dominio del problema o del modelo de negocio, representadas en sus correspondientes modelos de análisis, los **requisitos no funcionales** y ciertas **restricciones de la implementación** pueden influir, incluso en mayor medida que los modelos y las especificaciones elaborados durante el análisis, en la definición de la arquitectura, de la organización y de las abstracciones plasmadas en el modelo de diseño.

En este sentido, el soporte de múltiples idiomas para los mensajes y la leyenda mostrados por la interfaz gráfica y de múltiples plataformas para la ejecución de dicha interfaz (lo cual resulta una tarea muy compleja sin una librería de componentes gráficos implementada de forma consistente en todas las plataformas soportadas) han condicionado algunas decisiones del diseño, por ejemplo: establecer de forma estricta la **separación** entre la **capa de presentación** y la **capa de la lógica de la aplicación**, así como adoptar el estilo de arquitectura **Cliente-Servidor**, gracias al cual se facilita también la separación de capas, el desarrollo del cliente y del servidor en diferentes lenguajes de programación e incluso la implementación independiente de la capa de la lógica de la aplicación por medio de un servidor más ligero que una hipotética versión no distribuida de esta aplicación.

Los **aspectos estáticos** más importantes del modelo de diseño de un sistema pueden visualizarse gracias a una o más vistas de este modelo representadas gráficamente mediante diagramas de clases. Generalmente, un **diagrama de clases** muestra un conjunto de clases (en este caso, clases a nivel de diseño), interfaces y las relaciones que existen entre todas estas entidades (como las relaciones de asociación, dependencia, generalización y realización). También los **diagramas de objetos** pueden ilustrar algunos aspectos estáticos de los modelos de diseño, aunque los objetos que son instancias de clases de diseño aparecen con más frecuencia en los **diagramas de interacción** que, en cambio, permiten documentar los aspectos dinámicos de los modelos de diseño y en la mayoría de los casos estarán vinculados a una colaboración. Por su parte, a través de una **colaboración** puede detallarse cómo se lleva a cabo una interacción concreta entre un actor y el sistema, la cual está asociada a algún caso de uso.

Las abstracciones elementales presentes en el modelo de diseño (incluyendo clases, interfaces y las relaciones entre éstas) pertenecen al **dominio de la solución** en vez de al dominio del problema, al contrario de lo que ocurre con los modelos de análisis. En consecuencia, cabe esperar un mayor refinamiento de tales abstracciones, así como un mayor nivel de detalle en la especificación de sus propiedades. Aunque no se van a describir en este trabajo las propiedades que caracterizan a cada tipo de abstracción, se comentarán aquellas propiedades específicas que distingan a una abstracción de sus homólogas. Por ejemplo, mediante el estereotipo *active* se han distinguido las clases activas, esto es, las clases cuyos objetos pueden originar un flujo de control. Además, algunas librerías de funciones (como las que soportan los lenguajes de programación híbridos, basados en el paradigma orientado a objetos e imperativo) se han modelado como clases con el estereotipo *utility*, esto es, clases no instanciables cuyos atributos y operaciones tienen alcance de clase.

Del mismo modo, cada uno de los atributos pertenecientes a ciertas clases que está precedido por el estereotipo *property* (creado específicamente para el propósito mencionado a continuación) denota una "*propiedad*", entendida de forma análoga a las propiedades de los *JavaBeans*, esto es, atributos que son accedidos por operaciones públicas que permiten recuperar y establecer el valor de dichos atributos y que son denominadas empleando ciertas convenciones. Por lo tanto, dada una propiedad de nombre *X*, se asume que existen dos operaciones, cuyas signaturas son: *getX(): TipoX* y *setX(TipoX x): void*, que están destinadas a recuperar y a modificar, respectivamente, el valor del atributo asociado a la propiedad *X*, aunque tales operaciones no se muestran en las interfaces de las clases representadas en los diagramas de este trabajo.

Por último, conviene advertir que aunque el diseño corresponde al de una aplicación distribuida que se adhiere al estilo de arquitectura **Cliente-Servidor**, la interacción entre los subsistemas del cliente y del servidor no se ajusta totalmente al esquema **Petición-Respuesta**, sino que a causa del tiempo de ejecución indeterminado y no acotado de muchas operaciones del servidor, se ha decidido que el resultado de tales operaciones sea notificado por el servidor de forma asíncrona con respecto a la petición del cliente que originó dicho cómputo. En cualquier caso, las siguientes subsecciones describirán las clases, las operaciones de estas clases y los parámetros de estas operaciones que se consideran más importantes con respecto al diseño de la aplicación. Esta descripción comienza con las clases pertenecientes al subsistema del cliente y continúa con las clases vinculadas al subsistema del servidor, estando organizadas todas estas clases en dos paquetes que se corresponden con ambos subsistemas.

5.2.1 Diseño del cliente

El subsistema del cliente comprende una colección de clases agrupadas de forma lógica en el paquete denominado *cliente* y de forma física en un componente ejecutable con idéntico nombre. Este conjunto de clases implementa la interfaz gráfica que interactúa con el usuario, además lleva a cabo la codificación y el envío al servidor de los mensajes que invocan los servicios fundamentales de la aplicación. Recíprocamente, también realiza la recepción y la decodificación de los mensajes de respuesta del servidor y de cualquier otra notificación asociada a alguna función invocada por el cliente. Adicionalmente, permite guardar y recuperar la configuración de la aplicación y participa, aunque a un nivel muy básico, en la ejecución de la lógica de control de la aplicación. El diagrama de la **Figura 20** representa esquemáticamente las clases más importantes del cliente y sus relaciones (observe que el estereotipo «*instantiate*» de algunas relaciones de dependencia indica que la clase dependiente puede crear instancias de la clase independiente).

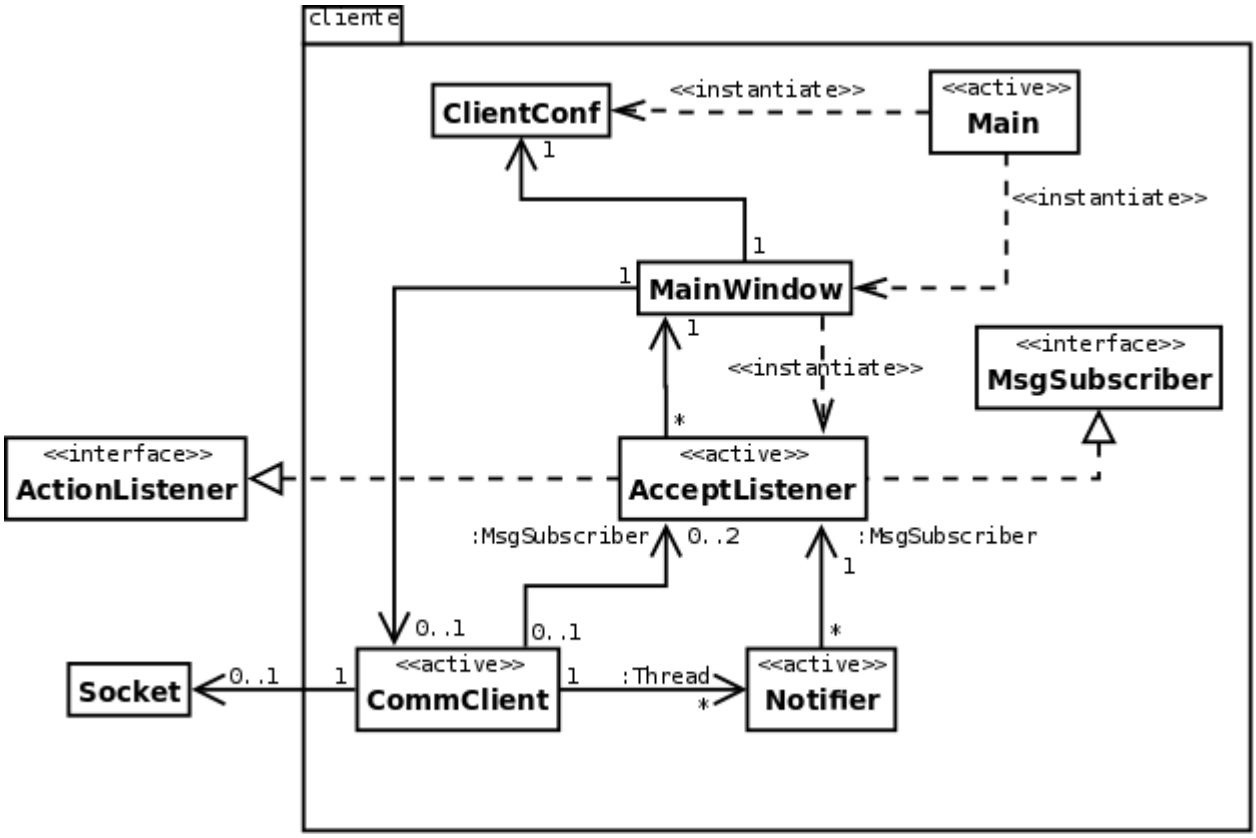

Figura 20: Diagrama de clases del cliente

A continuación, en la **Figura 21** se muestra la clase *Main*, la cual contiene la operación cuyo método implementa el punto de entrada de la aplicación cliente, por lo que su signatura se ajusta a la exigida por el lenguaje *Java* para esta clase de métodos. Desde este método se crea una instancia de la clase *ClientConf*, la cual contiene los parámetros de configuración del cliente (por ejemplo: el idioma de los mensajes mostrados por la interfaz gráfica o la dirección IP del computador y el número de puerto TCP que habilitan la conexión con el servidor de esta aplicación) y también los parámetros de configuración del servidor (por ejemplo, el nombre y otros parámetros de operación de la interfaz de red inalámbrica utilizada en el ataque, el número máximo de reenvíos de una trama que no fue respondida o el tiempo de espera máximo para recibir la respuesta a una trama). Además, esta clase dispone de métodos para recuperar/almacenar estos parámetros de configuración desde/en disco y actualizar sus valores en una instancia de la clase a partir de una tabla hash.

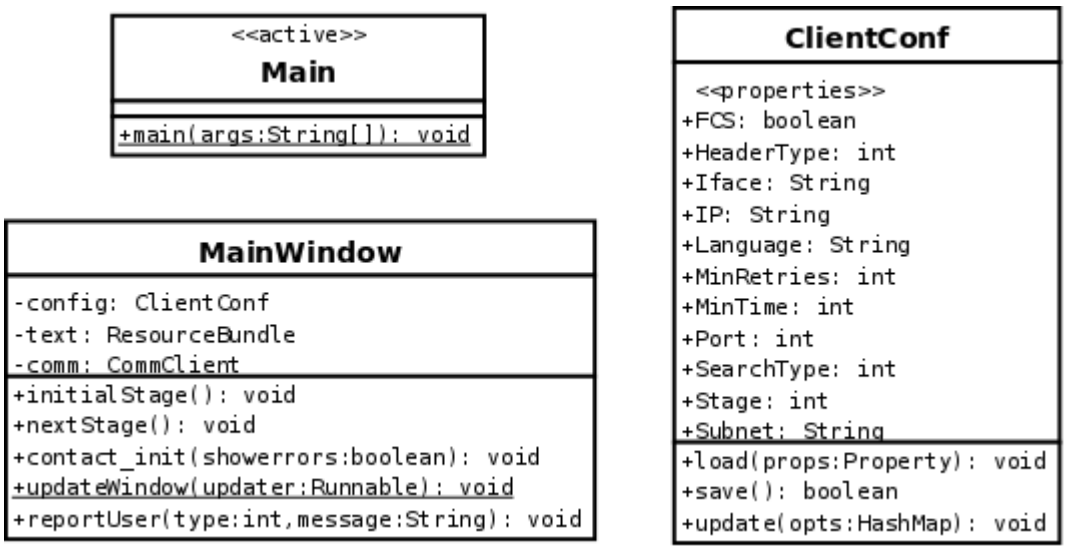

Figura 21: Interfaces de clases del cliente 1

La instancia de la clase *MainWindow* construye y muestra la ventana principal de la aplicación, que se encarga de gestionar casi toda la interacción con el usuario. Dicho objeto debe tener acceso a la configuración de la aplicación almacenada en disco y a los mensajes mostrados por la interfaz gráfica, en el idioma correspondiente, durante su creación. Por tanto, la clase *Main* crea este objeto y lo inicializa mediante la operación *initialStage()*, después intenta que dicho objeto establezca contacto con el servidor invocando la operación *contact_init(...)*. Si la anterior operación logra su cometido, la instancia de *MainWindow* se comunicará con el servidor a través de un objeto de la

clase *CommClient*. Por otro lado, la operación *reportUser(...)* permite notificar al usuario cierto tipo de información, pero debe ser invocada exclusivamente desde un método para la gestión de eventos de la interfaz gráfica. En cambio, la operación con alcance de clase denominada *updateWindow(...)* permite modificar el aspecto de la ventana principal desde cualquier método. También es necesario mencionar la operación *nextStage(...)*, puesto que desencadena la transición a una nueva etapa del ataque, siendo invocada desde un objeto asociado al control de la ventana principal.

Entre las entidades que muestra la **Figura 22**, destaca la clase activa *CommClient*, gracias a la cual la ventana principal puede intercambiar mensajes con el servidor de la aplicación. Para cumplir con esta función, un objeto de la clase anterior requiere la dirección IP y el puerto TCP mediante los cuales puede establecer una conexión con el servidor. Ambos parámetros deben ser especificados por medio de la operación *SetServerAddress(...)*, encargada del establecimiento de la conexión y de la inicialización del socket TCP generado durante este proceso y accesible a través del atributo *sock*. Si la operación anterior cumple su objetivo, la ventana principal podrá enviar mensajes al servidor mediante la operación *sendMsg(...)* y recibir las respuestas correspondientes mediante la operación *getResponse()*. Entre ambos eventos, el objeto que posibilita a la ventana principal el acceso a estas operaciones debe guardar los mensajes de respuesta recibidos en una cola.

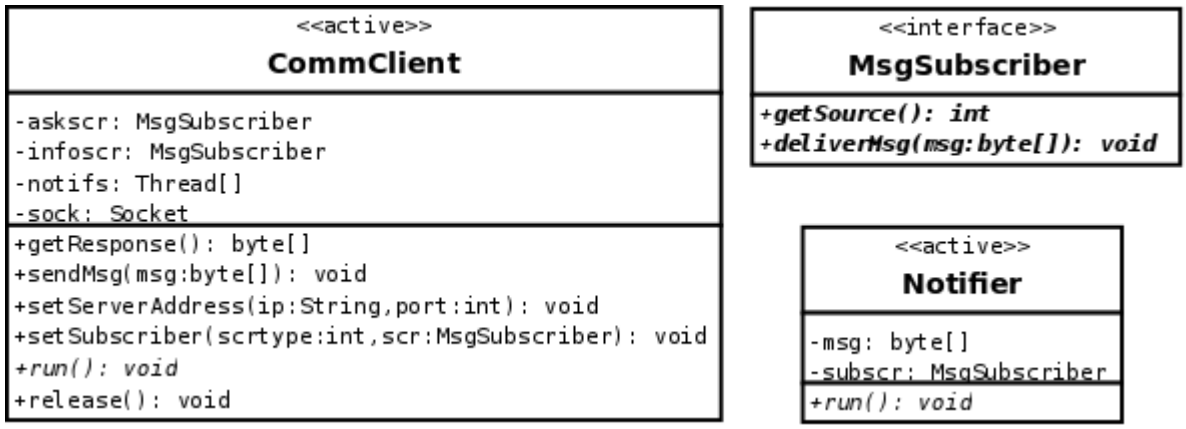

Figura 22: Interfaces de clases del cliente 2

Además, una instancia de la clase activa *CommClient* puede recibir notificaciones por iniciativa del servidor destinadas a la ventana principal. Los objetos designados como receptores directos de estas notificaciones deben ser especificados como parámetros de la operación *setSubscriber(...)* de la citada instancia, aunque también debe indicarse el tipo específico de notificación en la que están interesados tales objetos. Por su parte, una instancia de la clase *CommClient* mantiene el acceso a

los objetos suscritos a estas notificaciones a través de sus atributos: *askscr* e *infoscr*, para lo cual tales objetos deben implementar las operaciones de la interfaz *MsgSubscriber*. Para ser más exactos, estas notificaciones son realizadas por objetos de la clase activa *Notifier*, que son instanciados por la anterior instancia de la clase *CommClient*, la cual les entrega el mensaje a notificar y una referencia al objeto receptor del mensaje.

Por exigencia del lenguaje de programación utilizado, el flujo de control generado por un objeto de la clase *Notifier* (así como los flujos generados por los objetos que son instancias de casi todas las clases activas mostradas en los diagramas de diseño) debe ser especificado mediante el método correspondiente a la operación *run()* de esta clase (y en general, como el método de una operación con la misma signatura que la anterior). Por otra parte, los objetos que son susceptibles de recibir una notificación del servidor deben implementar la interfaz *MsgSubscriber*, la cual especifica una operación para recibir los mensajes: *deliverMsg(...)* y otra operación para identificar a los auténticos destinatarios de los mensajes: *getSource()*. Finalmente, cabe aclarar que la operación *release()* de la clase *CommClient* detiene el flujo de control iniciado por un objeto que implementa esta operación y después libera los recursos utilizados por este objeto.

5.2.2 Diseño del servidor

Del mismo modo que en el cliente, las clases que integran el servidor (y algunos módulos que han sido representados como clases para que aparezcan reflejados en el modelo de diseño orientado a objetos) se han agrupado en un paquete homónimo, con respecto a este subsistema, y después se han desplegado como un **componente ejecutable**, que es capaz de completar un procesamiento ya iniciado prescindiendo de la ejecución del cliente. Esto es debido, al menos en parte, a que el diseño se ajusta al de un **servidor concurrente basado en hebras**, aunque a consecuencia de que el cliente restringe la invocación de determinadas operaciones del servidor de forma concurrente, no deberían ejecutarse varias hebras del servidor simultáneamente, salvo durante breves y puntuales instantes de tiempo. Por otro lado, las operaciones más sencillas, o las que requieren muy poco tiempo para ser completadas, pueden ser ejecutadas por la hebra principal del servidor, esto es, la hebra que atiende las peticiones del cliente. En la **Figura 23** se ilustra un diagrama de clases simplificado que incluye las clases más importantes del servidor y sus relaciones (observe que el estereotipo «*instantiate*», que califica a algunas relaciones de dependencia, ha sido abreviado como «*inst*»).

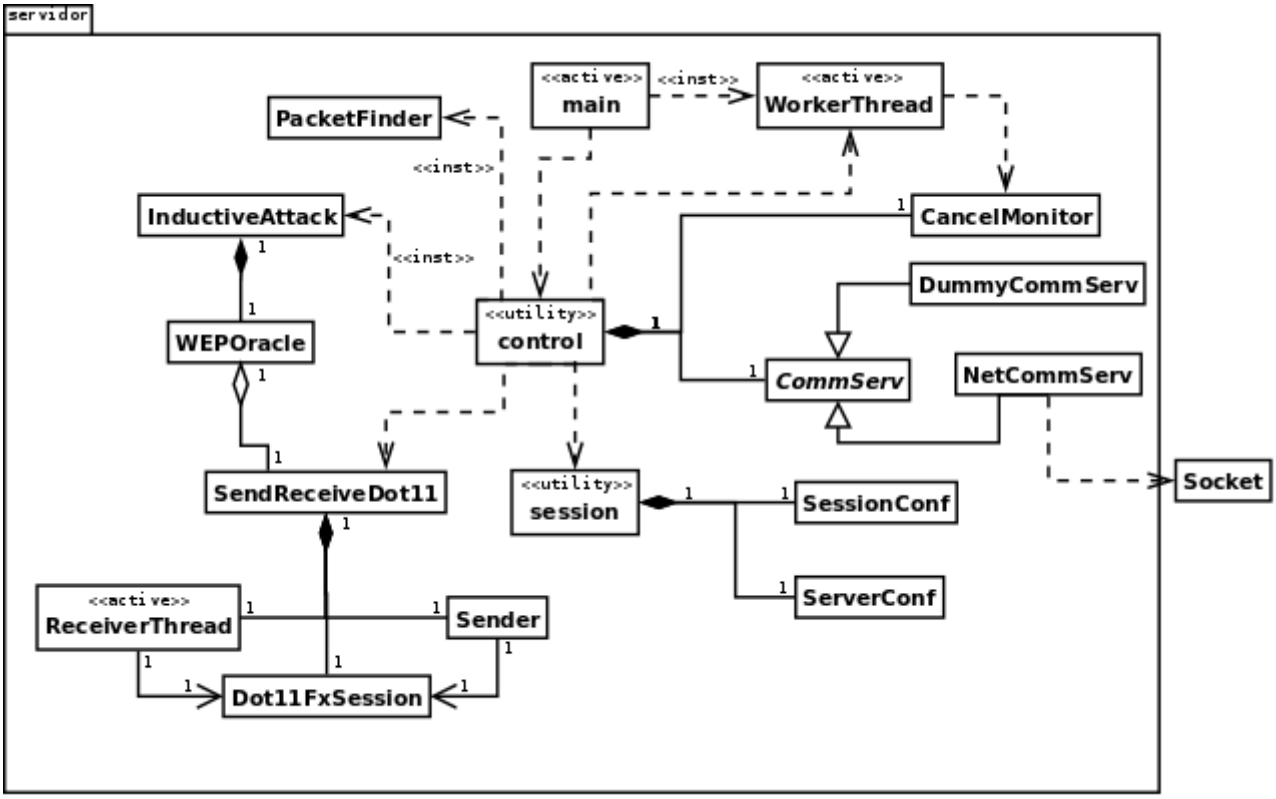

Figura 23: Diagrama de clases del servidor

A continuación, en la **Figura 24** se han esbozado los rasgos más importantes de las interfaces de las tres clases que implementan la mayor parte de la lógica de control del servidor. Por medio de la clase *main* se ha representado un módulo ejecutable (en vez de un módulo de librería) que contiene el punto de entrada a la aplicación servidor. Asimismo, dicho punto de entrada se ha representado mediante una operación ficticia, cuya signatura es la siguiente: *__init__()*. Por otra parte, el atributo *workerthread* referencia a la hebra que ejecuta la operación solicitada por el cliente, cuando dicha operación es realizada por una hebra adicional, mientras que el atributo *thread_at_work* indica si existe una hebra activa referenciada por el atributo anterior.

Como respuesta a ciertas peticiones del cliente, la hebra principal puede ejecutar algunas de las operaciones que se ilustran en la clase *main*. Por ejemplo, mediante la operación *request_cancel()* se solicita la cancelación de la última hebra asignada al atributo *workerthread*, en caso de que siga activa. Por su parte, la operación *disconnect()* cierra la conexión con el cliente, sin que esto afecte a cualquier cómputo ya iniciado, en cambio, la operación *ready_to_exit()* actúa como la anterior pero también procede a cancelar cualquier hebra activa del servidor y a terminar la ejecución de éste.

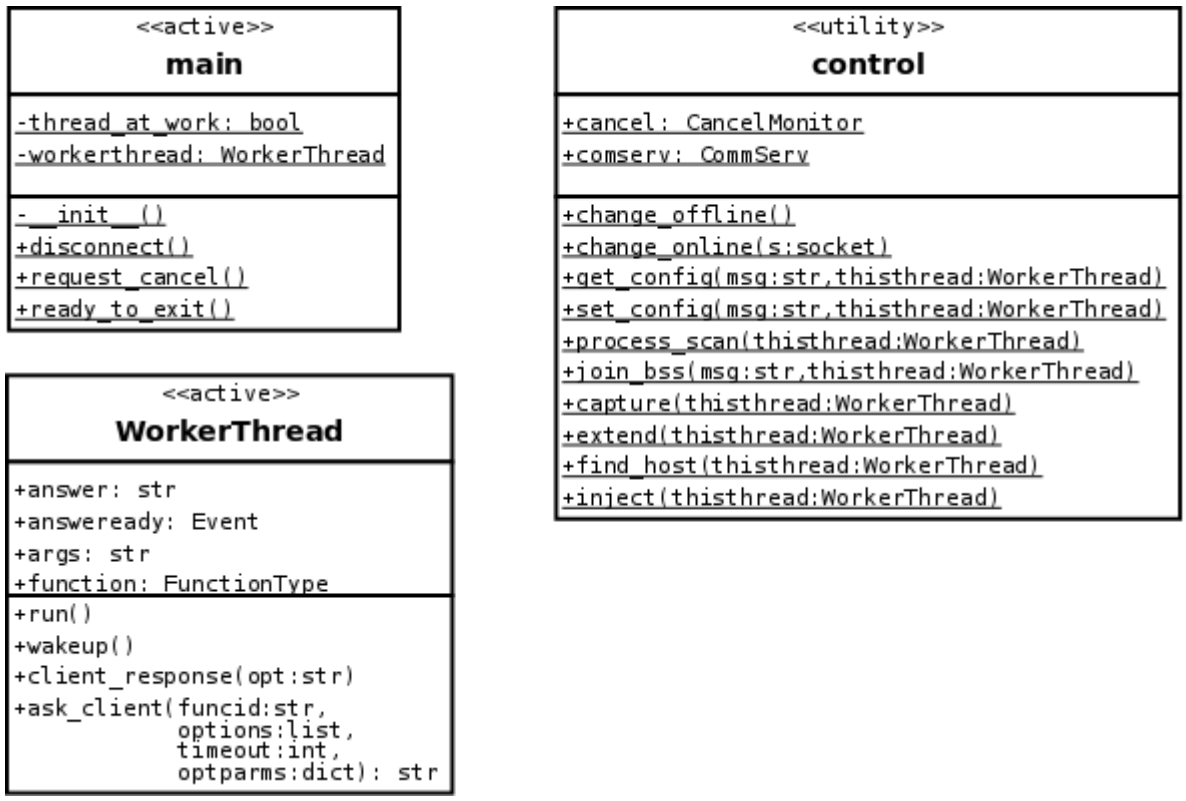

Figura 24: Interfaces de clases del servidor 1

Como era previsible, las instancias de la clase *WorkerThread* son hebras que ejecutan las tareas más complejas del servidor, a la vez que la hebra principal del mismo sigue a la espera de nuevas instrucciones del cliente. La tarea que lleva a cabo un objeto de esta clase viene especificada como una función, a la que acompaña una lista de argumentos, siendo ambos datos entregados antes de la inicialización del objeto, aunque después de la misma continúan siendo accesibles por medio de los atributos *function* y *args*, respectivamente. En concreto, dicha tarea se ejecuta mediante el método asociado a la operación *run()* de la clase mencionada. Otra operación, denominada *ask_client(...)*, envía un mensaje al cliente para solicitarle que seleccione una opción de una lista que se le ofrece. Seguidamente, el cliente puede transmitir su elección al servidor y el módulo *main* la recibirá y la notificará a la hebra interesada mediante la operación *client_response(...)*, encargada de asignar la respuesta al atributo *answer* y de asertar la variable de condición asociada al atributo *answeready*. Finalmente, la operación *wakeup(...)* permite interrumpir la espera de una hebra de la clase anterior, que es provocada por la operación *ask_client(...)*, cuando la primera operación es invocada durante dicha espera.

Por otro lado, el módulo *control* (representado como una clase de utilidades del mismo nombre) aglutina una gran parte de la funcionalidad esencial del servidor y exporta dos objetos accesibles a través de las variables globales *cancel* y *comserv*, la primera de las cuales permite monitorizar una orden de cancelación de una tarea en ejecución, mientras que la segunda de estas variables habilita el/la envío/recepción de mensajes hacia/desde el destino/origen asociado al objeto referenciado por esta variable. Precisamente, las operaciones *change_offline()* y *change_online(...)* llevan a cabo el reemplazo del objeto referenciado por la variable *comserv* por otro objeto que habilita la redirección de los mensajes del servidor hacia la salida estándar o hacia el cliente, respectivamente.

Las siguientes operaciones de este módulo cuentan con un parámetro (*thisthread*) instanciado con el objeto correspondiente a la hebra desde la cual se ejecuta el servicio invocado por el cliente. De esta forma, el método que implementa este servicio puede acceder a las operaciones de la clase *WorkerThread*, a pesar de que no son requeridas por todas las operaciones del módulo *control* que incluyen este parámetro formal. Este es el caso de las operaciones: *get_config(...)* y *set_config(...)*, las cuales permiten recuperar y alterar la configuración del servidor, respectivamente. En cambio, las operaciones *process_scan()* y *join_bss()* cumplen los mismos objetivos que dos pasos del E.P.E. del caso de uso *Seleccionar red objetivo*, como son la búsqueda de redes Wi-Fi protegidas con WEP y la asociación con la red Wi-Fi seleccionada. Por su parte, las operaciones: *capture(...)*, *extend(...)*, *find_host(...)* e *inject(...)* se encargan de desempeñar la funcionalidad esencial de los casos de uso: *Capturar trama*, *Extender secuencia*, *Buscar estación* e *Inyectar tráfico*, respectivamente.

En la **Figura 25** se muestran las clases a partir de las cuales son instanciados los dos objetos que son exportados por el modulo *control*. La variable *cancel* mantiene la única instancia de la clase *CancelMonitor* que necesita el servidor. El atributo *state* de esta instancia indica si alguna operación está pendiente de ser cancelada, mientras que el atributo *lock* es manipulado por los métodos que implementan las operaciones mostradas de esta clase para garantizar el acceso en exclusión mutua de éstos al primer atributo. En concreto, la operación *is_cancelled()* devuelve el valor del atributo *state* y las operaciones *set_cancelled()* y *clear_cancelled()* activan y desactivan, respectivamente, una solicitud de cancelación. Por lo tanto, el objeto de la clase *CancelMonitor* permite que la hebra principal y otra hebra, la cual ejecuta alguna función del servidor, se coordinen para monitorizar y alterar la ejecución de una tarea. De esta manera, la hebra principal puede solicitar la cancelación de una tarea y la otra hebra acatar esta petición, una vez que alcance cierto punto en la ejecución de la operación correspondiente.

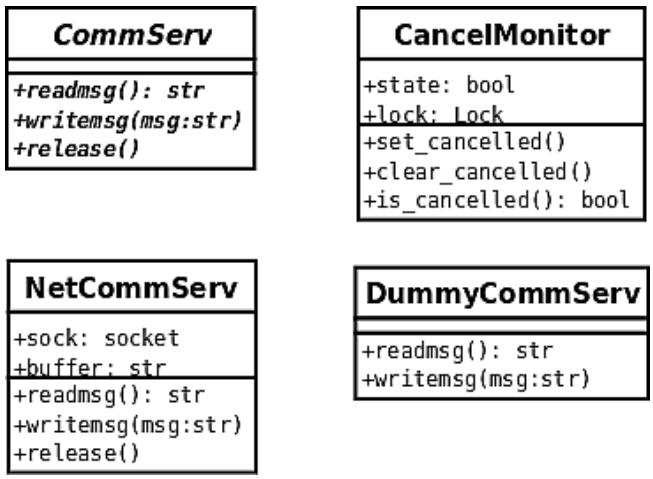

Figura 25: Interfaces de clases del servidor 2

En cuanto a la variable *comserv*, ésta puede referenciar a diferentes objetos durante una sesión del servidor, siempre que éstos sean instancias de una subclase de *CommServ*, puesto que esta clase es abstracta. Esto es lo que ocurre con los objetos de la subclase *NetCommServ*, la cual implementa todas las operaciones de la clase abstracta *CommServ*, esto es: *writemsg(...)*, *readmsg()* y *release()*. Mediante estas operaciones, estos objetos ofrecen diferentes servicios, como el envío de mensajes al cliente o la entrega de mensajes procedentes del cliente (después de la recepción y la reconstrucción de estos mensajes en el espacio de almacenamiento proporcionado por el atributo *buffer*), así como la liberación de los recursos asignados a tales objetos (incluyendo el socket TCP que es referenciado por el atributo *sock*), respectivamente.

La otra subclase de la clase *CommServ*, denominada *DummyCommServ*, desempeña un papel de una importancia marginal, puesto que solo se crea una instancia de dicha subclase para reemplazar al último objeto instanciado a partir de la subclase *NetCommServ*, en caso de que se interrumpa la comunicación con el cliente. Por lo tanto, las operaciones de la subclase *DummyCommServ* han sido redefinidas de forma distinta a las operaciones de la otra subclase. Para concretar más, la operación *writemsg(...)* vuelca los mensajes que recibe como argumento por la salida estándar, por lo que no tiene ningún interés, salvo para mostrar información que puede resultar útil durante la depuración. Por el contrario, la operación *readmsg()* no realiza ningún cómputo útil, sino que se bloquea hasta agotar el tiempo de espera sin devolver mensaje alguno. Lo mismo puede afirmarse de la operación *release()*, que ni siquiera es redefinida porque no tiene ningún efecto en esta subclase.

Dirigiendo ahora la atención al módulo *session*, puede comprobarse que contiene la declaración de las clases *ServerConf* y *SessionConf*, y que crea una instancia de cada una de estas clases para ponerlas a disposición de los otros módulos del servidor. El objeto de la clase *ServerConf* contiene información sobre el estado y la configuración del servidor, por lo que afecta al funcionamiento de la mayor parte de las operaciones básicas de éste. Por ejemplo, este objeto registra la fase del ataque en la que se encuentra el servidor en cada momento (propiedad *stage*), así como ciertos parámetros de operación del servidor establecidos por el usuario, incluyendo la interfaz inalámbrica utilizada en el ataque (propiedad *if_mon*), el tipo de cabecera que devuelve dicha interfaz cuando opera en modo monitor (propiedad *mon_head_type*), el prefijo de la dirección IP de la subred atacada supuesto por el atacante (propiedad *subnet*), etc.

Otras propiedades, como el identificador de la red inalámbrica y del BSS perteneciente a esta red, que se denominan *ssid* y *ap_MAC*, respectivamente, y que determinan el objetivo del ataque, no se definen hasta etapas más tardías de la ejecución de este ataque. Por otro lado, las operaciones *is_updated()* y *get_cli_config()* (cuyos nombres han sido abreviados) tratan de determinar si se les ha asignado un valor a todos los parámetros de configuración del servidor definidos por el cliente y, en tal caso, la segunda operación devuelve una cadena de caracteres que codifica tales valores en un formato conocido por el cliente. Los rasgos más importantes de la interfaz de la clase *ServerConf* pueden apreciarse en la **Figura 26**.

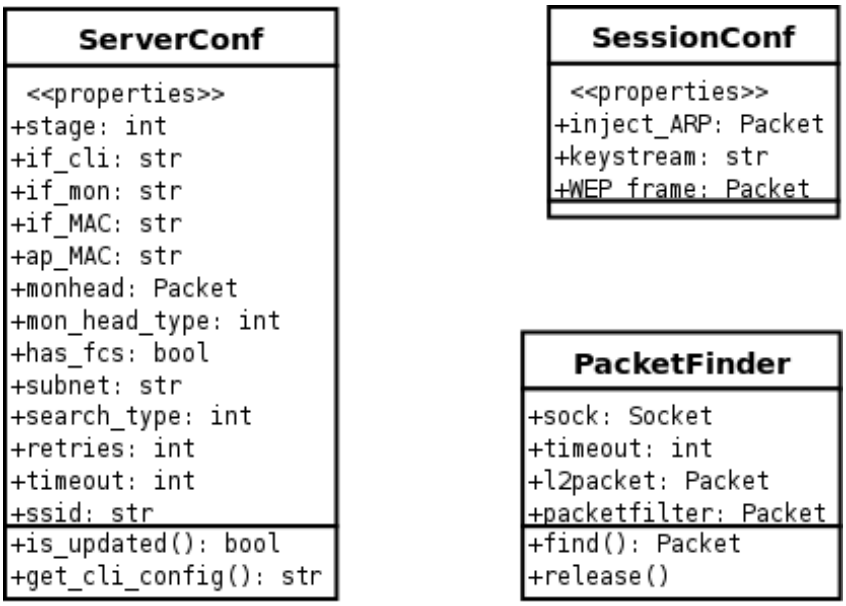

Figura 26: Interfaces de clases del servidor 3

En la misma figura se representa la interfaz de la clase *SessionConf*. Al igual que con la clase anterior, el servidor crea una única instancia de esta clase, la cual se encarga de almacenar algunos datos obtenidos en fases posteriores de una sesión de ataque, como por ejemplo, la carga de datos de la trama cifrada con WEP a partir de la cual se extrae el prefijo del keystream que permite iniciar el ataque inductivo (propiedad *WEP_frame*), la versión extendida del anterior keystream que se ha obtenido mediante el ataque inductivo (propiedad *keystream*), o bien, el paquete ARP que provocó la transmisión de una trama de respuesta durante la búsqueda de estaciones pertenecientes a la red atacada (propiedad *inject_ARP*).

Por último, en esta figura también se ha incluido una vista simplificada de la interfaz de la clase *PacketFinder*, a pesar de que pertenece a otro módulo. Esta última clase se instancia desde varios módulos con el objetivo de encontrar cierta clase de tramas, en caso de que sean recibidas a través del socket asociado al atributo *sock*. Además, las tramas recibidas deberían ajustarse al formato que indica el atributo *l2packet* y su recepción no puede demorarse más allá del intervalo de tiempo que representa el atributo *timeout*. De todas las tramas recibidas, se devuelve la primera que satisfaga alguna condición o filtro, evaluado sobre los valores de algunos de sus campos, que está incluido en una lista de filtros mantenida por el atributo *packetfilter*. Esta última función es la que desempeña el método correspondiente a la operación *find()*, en cambio, el método correspondiente a la operación *release()* cierra el socket ya mencionado.

En la siguiente figura, esto es, en la **Figura 27** se muestra una clase compuesta denominada *SendReceiveDot11*, que fue diseñada para enviar una trama y para recibir, de forma concurrente con la operación de envío, una o más tramas de respuesta. Para conseguir esto, cada instancia de la clase compuesta se sirve de una instancia de cada una de las clases componentes, esto es, las restantes clases que aparecen en el diagrama. Antes de su inicialización, cada instancia de la clase compuesta debe recibir los siguientes argumentos: el nombre de la interfaz de red inalámbrica destinada a la transmisión/recepción, un objeto que representa a la clase cuyas instancias encapsularán las tramas recibidas y un filtro que se aplicará a la recepción de tramas a través del socket creado para el envío y la recepción de tramas mediante la anterior interfaz de red (argumentos que se corresponden con los parámetros formales: *iface*, *packetype* y *sockfilter* de la operación *__init__*(), respectivamente). No obstante, toda instancia de la citada clase entregará estos argumentos para inicializar otro objeto, que será instanciado mediante la clase *Dot11FXSession*, delegando en este último objeto la creación del socket.

Figura 27: Interfaces de clases del servidor 4

Para especificar el número de reintentos que realizará un objeto de la clase *SendReceiveDot11* para la transmisión de una trama, en caso de que no reciba la(s) trama(s) de respuesta adecuada(s), se define la operación *set_retries(...)*, que establece el valor del atributo *retries* perteneciente a dicho objeto. En cambio, la operación *send(...)* de esta clase delega en la clase *Sender* para la transmisión de tramas. Del mismo modo, las operaciones *set_timeout(...)* y *set_filters(...)*, las cuales establecen el tiempo máximo de espera para la(s) trama(s) de respuesta y las condiciones o filtros evaluados sobre los objetos que representan las tramas capturadas, respectivamente, también delegan en las operaciones homónimas de la clase *ReceiverThread*. Por lo tanto, una vez conocida la delegación de las operaciones de la clase *SendReceiveDot11*, se desprende que la clase *Sender* es la encargada de enviar las tramas, mientras que la clase *ReceiverThread* se ocupa de capturar y filtrar la(s) trama(s) de respuesta oportuna(s), cuando son transmitidas por el medio inalámbrico.

Por otro lado, la clase *Dot11FXSession*, además de crear el socket para la transmisión/recepción de tramas, pone a disposición de las clases *Sender* y *ReceiverThread* un *pipe* para su comunicación y coordinación, que es accedido por las instancias de la primera clase para la escritura (a través de la operación *get_wrpipe()*, la cual devuelve el objeto referenciado por el atributo *wrpipe*) y por las instancias de la segunda clase para la lectura (por medio de la operación *get_readpipe()*, que hace lo mismo con el atributo *rdpipe*). Sin embargo, para conseguir una sincronización más fina durante el

envío y la recepción de tramas (evitando que la recepción comience después de que se transmitan la(s) trama(s) de respuesta), la clase *Dot11FXSession* proporciona a las dos clases mencionadas tres variables de condición, que pueden ser accedidas por medio del atributo *conditions*. Cada variable de condición puede ser manipulada para su activación, para su desactivación o bien para forzar el bloqueo o la espera de una hebra a que la condición asociada a la variable sea activada, por medio de las operaciones: *set_condition()*, *clear_condition()* y *wait_condition()*, respectivamente.

Con respecto a la clase *ReceiverThread*, cada instancia de ésta determina la clase, a partir de la cual debe originar los objetos que representan a las tramas capturadas, por medio de la operación *get_packet_type()* perteneciente a la clase *Dot11FXSession* (la cual devuelve el valor del atributo *packetype*). Más tarde, la instancia de *ReceiverThread* pondrá a disposición de los otros objetos las tramas capturadas (que superen los filtros definidos) gracias al atributo *frames* de la instancia de la clase *Dot11FXSession*, accesible mediante la operación *get_frames()* de esta clase. Por otro lado, tanto la clase *ReceiverThread* como la clase *Sender* disponen de un atributo denominado *ctrlchan*, que encapsula el extremo correspondiente del pipe accedido por sus respectivas instancias.

Por su parte, la clase *Sender* envía una o más réplicas de una trama (esto es, envía más de una trama cuando no se produce la recepción de la(s) trama(s) esperada(s)), proporcionadas por la clase *SendReceiveDot11*, a través de la operación *send_dot11()*. Mientras tanto, una instancia de la clase *ReceiverThread* debe estar ejecutando el método correspondiente a la operación polimórfica *run()*, a la espera de recibir una secuencia de tramas que se ajusten, en orden de recepción, a la secuencia de filtros recopilada en su atributo *filters*. Por lo tanto, las tramas capturadas inducen la generación de objetos de la clase especificada por el atributo *packetype* y sobre éstos se evalúan los filtros que están incluidos en la secuencia mantenida por el atributo *filters*.

Una vez que se captura una sucesión de tramas que satisface la secuencia de filtros establecida, se alcanza el objetivo perseguido por la instancia de la clase *SendReceiveDot11* y, en consecuencia, dicha instancia procede a finalizar el flujo de control originado por ella. A continuación, conviene invocar a la operación *release()* de la anterior instancia, puesto que su invocación desencadenará la ejecución de las operaciones homónimas de los objetos componentes de esta instancia, provocando la liberación de los recursos consumidos por este conglomerado de objetos, así como la detención de la hebra iniciada por la instancia de la clase *ReceiverThread*, tras asertar el atributo *stop* asociado a este objeto.

Finalmente, las dos clases principales que intervienen en la ejecución del ataque inductivo de *Arbaugh* se muestran en la **Figura 28**, aunque también participan de forma subordinada a este par todas las clases representadas en la figura anterior. También la clase *WEPOracle* desempeña este rol de subordinada respecto a la clase *InductiveAttack*, que ofrece la interfaz mediante la cual interactúa con las clases o las funciones que utilizan este ataque. Además, esta última clase funciona de forma similar a un iterador, esto es, en cada invocación a la operación *next()* devuelve un nuevo elemento, salvo que lo impida algún error en el enlace de comunicaciones. Este elemento, en el caso concreto de esta clase, es el valor descubierto del byte que sigue al prefijo del keystream obtenido.

Para lograr el propósito encomendado a la clase *InductiveAttack*, un objeto de esta clase debe ser inicializado por medio de la operación *__init__(...)* con los siguientes argumentos: el objeto que contiene la configuración del servidor, un valor que indica el tipo de estación que desempeñará el papel de oráculo (esto es, la estación que permite determinar si la predicción del valor del siguiente byte del keystream es correcta), el prefijo conocido por el atacante del keystream, el texto en claro escogido por el atacante para que sea cifrado con el keystream atacado, el vector de inicialización que interviene en la generación de este keystream y, por último, el índice de la clave WEP utilizada también por la víctima para generar el mismo keystream.

Los atributos *keystr* y *wepayload* de un objeto de la clase *InductiveAttack* mantienen los valores iniciales, así como la semántica, con respecto a los parámetros formales homónimos de la operación *__init__(...)*. En cambio, el atributo *wepkt* representa la carga de datos encriptada y encapsulada del mismo modo que lo haría el protocolo WEP a partir de los datos en claro, el vector de inicialización y el índice de la clave WEP especificados como argumentos de la anterior operación. Sin embargo, los dos primeros argumentos de la operación *__init__(...)* son entregados intactos a un objeto de la clase *WEPOracle*, mediante su operación homónima. Esta clase puede instanciar un objeto que se utiliza como prototipo de una trama 802.11 y que está disponible mediante el atributo *dot11frame*.

A través de la operación *test_WEP_frame(...)*, una instancia de la clase *WEPOracle* recibe como argumento otro objeto que representa la carga de datos de una trama 802.11 protegida mediante el protocolo WEP. Seguidamente, el método correspondiente a esta operación genera la representación binaria de la trama 802.11, después de ensamblar el prototipo que referencia el atributo *dot11frame* con la carga de datos recibida como argumento. Una vez que este método dispone de la secuencia de bytes correspondiente a la trama 802.11, la transmitirá a través del medio inalámbrico para que pueda recibirla su destinatario, esto es, la estación designada como oráculo.

Dependiendo de la respuesta del oráculo a una trama, el método anterior puede determinar si la carga de datos transportada por la trama 802.11 es válida, desde el punto de vista del procesamiento realizado por el protocolo WEP. En caso de que reciba una respuesta afirmativa, el método devuelve la constante asociada al atributo *OK*, mientras que en el caso opuesto debería devolver la constante *NO_RESPONSE*. No obstante, un error en el enlace de comunicaciones puede impedir que la trama elaborada por este método sea recibida por el oráculo o que la confirmación del oráculo regrese a la estación que transmitió la trama confirmada. En tal caso, el valor devuelto por este método es el que corresponde al atributo *NO_CONNECTION*. Por último, las dos operaciones denominadas *release()* de las dos clases mostradas en la siguiente figura desempeñan una función análoga a la descrita en las clases mencionadas anteriormente.

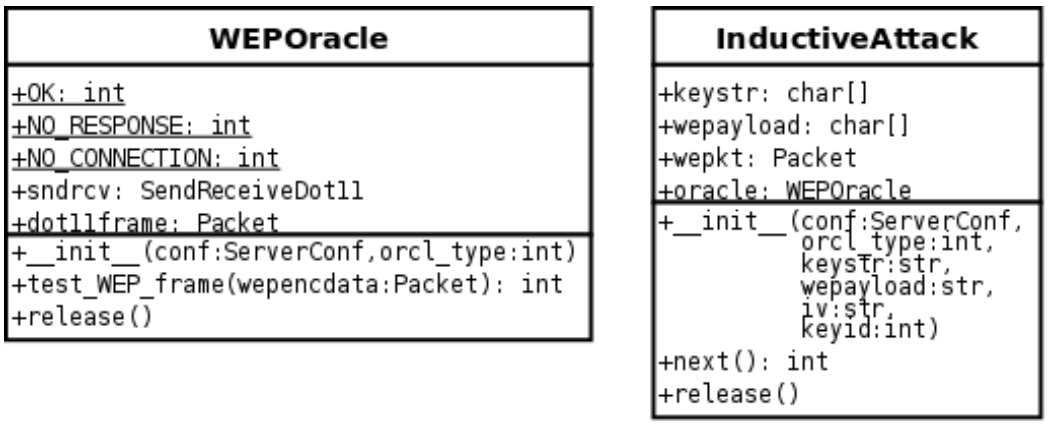

Figura 28: Interfaces de clases del servidor 5

5.2.3 Aspectos dinámicos del diseño

Los aspectos dinámicos de un modelo de diseño no pueden describirse mediante diagramas de clases o diagramas de objetos, no obstante, esta función puede ser desempeñada por elementos de comportamiento. Concretamente, las interacciones representan la alternativa más común empleada para realizar esta tarea en el contexto de los modelos de diseño. De hecho, una interacción describe una serie de mensajes intercambiados entre un conjunto de objetos, dentro de un contexto particular, para alcanzar un propósito específico. Gráficamente, las interacciones son representadas mediante diagramas de interacción, en los cuales aparecen típicamente objetos, sus relaciones y los mensajes intercambiados entre ellos.

Para ser más específicos, debemos mencionar que existen dos tipos de diagramas de interacción: los diagramas de secuencia y los diagramas de colaboración. Un diagrama de secuencia destaca la ordenación temporal de los mensajes intercambiados entre los objetos, mientras que un diagrama de colaboración hace un mayor énfasis en la organización estructural de los objetos que intercambian mensajes, esto es, en las relaciones existentes entre tales objetos. Aunque, en general, son diagramas semánticamente equivalentes, por lo que es posible transformar un ejemplar de un tipo de diagrama en un ejemplar del otro tipo sin pérdida de información. No obstante, esta sección se limita al uso de diagramas de secuencia, de manera que resalta el orden de invocación y el anidamiento en el flujo de control de las operaciones.

En cualquier caso, la cantidad de interacciones que tienen lugar entre un conjunto de objetos, especialmente si existen varios objetos activos, generalmente es muy grande, por ejemplo, pueden producirse tantas interacciones como flujos de control distintos pueden ocurrir entre estos objetos en el contexto de la funcionalidad de la aplicación que implementan. Esta es la causa de que en este apartado solo hayan sido incluidos un par de diagramas de secuencia típicos, esto es, dos diagramas que ilustran y constituyen ejemplos representativos de los dos tipos de interacciones más frecuentes que suceden entre los principales objetos integrados en el cliente y en el servidor. Adicionalmente, se ha alterado la semántica habitual de los focos de control anidados, ya que en ambos diagramas se utilizan para representar operaciones invocadas por hebras distintas, de manera que la hebra iniciada por el propio objeto activo es la responsable del foco de control principal.

El primer diagrama, ilustrado en la **Figura 29**, representa la ejecución de un servicio requerido por el cliente, que es ejecutado en el servidor por medio de una hebra instanciada a partir de la clase *WorkerThread*. En este caso particular, la operación requerida tiene como objetivo la asociación con una red inalámbrica, y en concreto, con el BSS especificado por el cliente en el mensaje de petición. Ante esta petición, el servidor crea la hebra que la atiende y esta hebra notifica al cliente que se ha iniciado el procesamiento de la operación, por medio del mensaje enviado en la primera invocación a la operación *writemsg*. Durante el procesamiento de la operación, la hebra comprueba si el cliente ha solicitado la cancelación del servicio, invocando la operación *is_cancelled*. En caso contrario, tras terminar el procesamiento, la hebra actualiza el estado del servidor, si procede. En el ejemplo presentado en la siguiente figura, la hebra anterior realiza esta tarea por medio del objeto de la clase *ServerConf*. Por último, la hebra devuelve el resultado de la operación que ha ejecutado, a través del segundo mensaje vinculado a la operación *writemsg*.

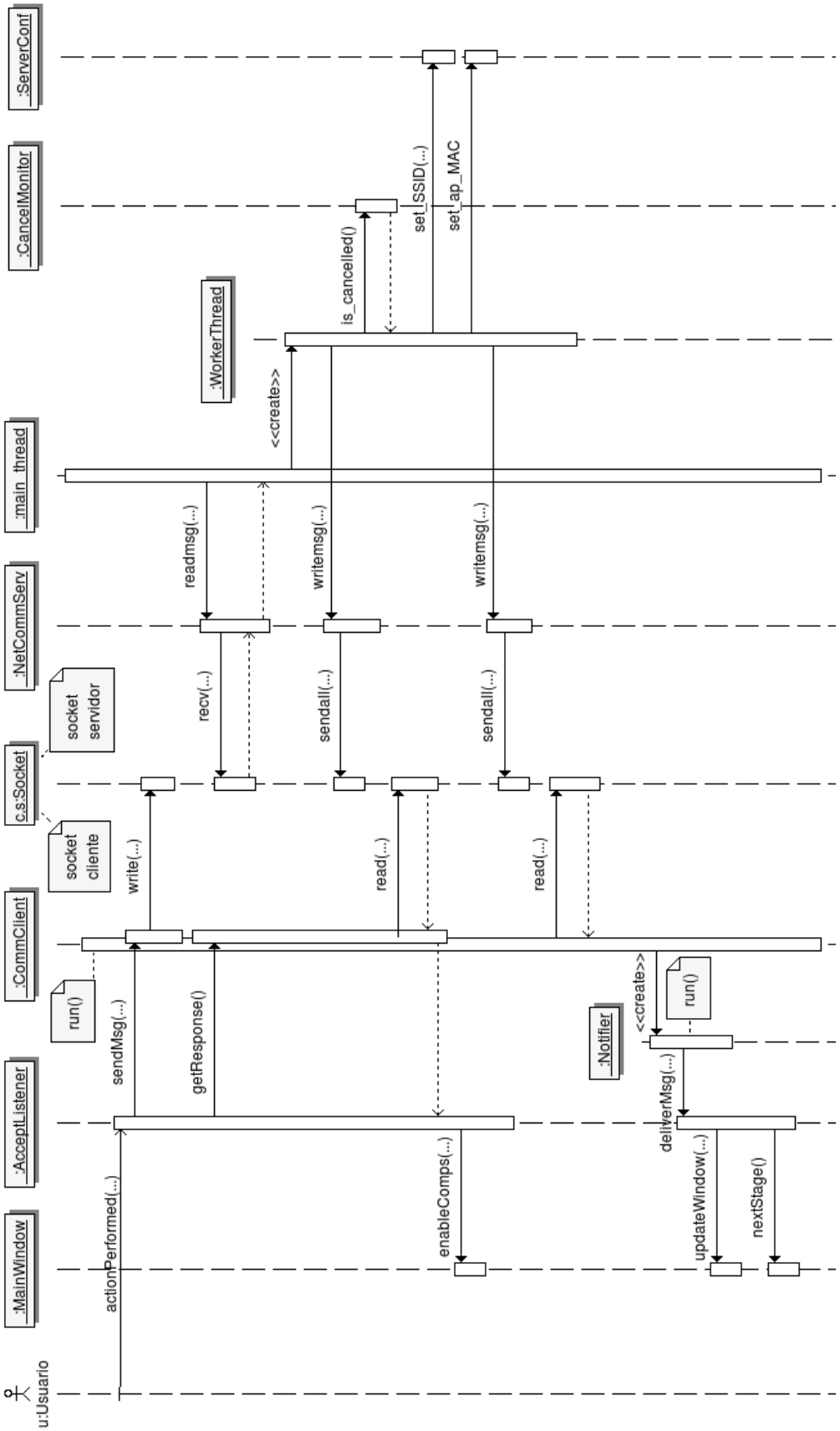

Figura 29: Diagrama de secuencia de un servicio requerido por el cliente

Por el contrario, en el segundo diagrama, que se aprecia en la **Figura 30**, la interacción la inicia el servidor con objeto de solicitar al usuario que seleccione una opción de un lista que previamente debe haber recibido. Esta interacción comienza cuando se invoca una operación de la misma hebra que lleva a cabo la función requerida del servidor, en concreto, se trata de la operación denominada *ask_client*. La lista de opciones, de entre las cuales puede escoger una el cliente, es entregada como argumento de esta operación y, tras ser codificada, es enviada al cliente por medio de la operación *writemsg*. También hay que advertir que en ambos diagramas de secuencia se ha utilizado el mismo objeto para representar a los sockets instanciados por el cliente y por el servidor, respectivamente, tal y como aparece indicado en las correspondientes notas de texto. En consecuencia, este elemento, presente en ambos diagramas, permite delimitar la frontera que separa a los objetos pertenecientes al cliente y al servidor.

A continuación, el mensaje codificado por el servidor será recibido por el cliente por medio del socket creado por este último, de manera que la hebra instanciada a partir de la clase *CommClient* podrá recogerlo y cederlo a otra hebra asociada a un objeto de la clase *Notifier*. Este objeto entrega el mensaje a su destinatario, esto es, algún objeto de control de la ventana principal capacitado para modificar el estado de dicha ventana y que muestra una ventana de diálogo para interactuar con el usuario. Una vez que el usuario selecciona una opción, ésta es transmitida al servidor a través del mecanismo usual y, finalmente, la hebra principal del servidor la entrega a la hebra que ejecuta el servicio demandado por el cliente. Para completar este último paso, la hebra principal invoca la operación *client_response* de la hebra que inició la interacción y le comunica la opción devuelta por el cliente a través del correspondiente parámetro de esta operación.

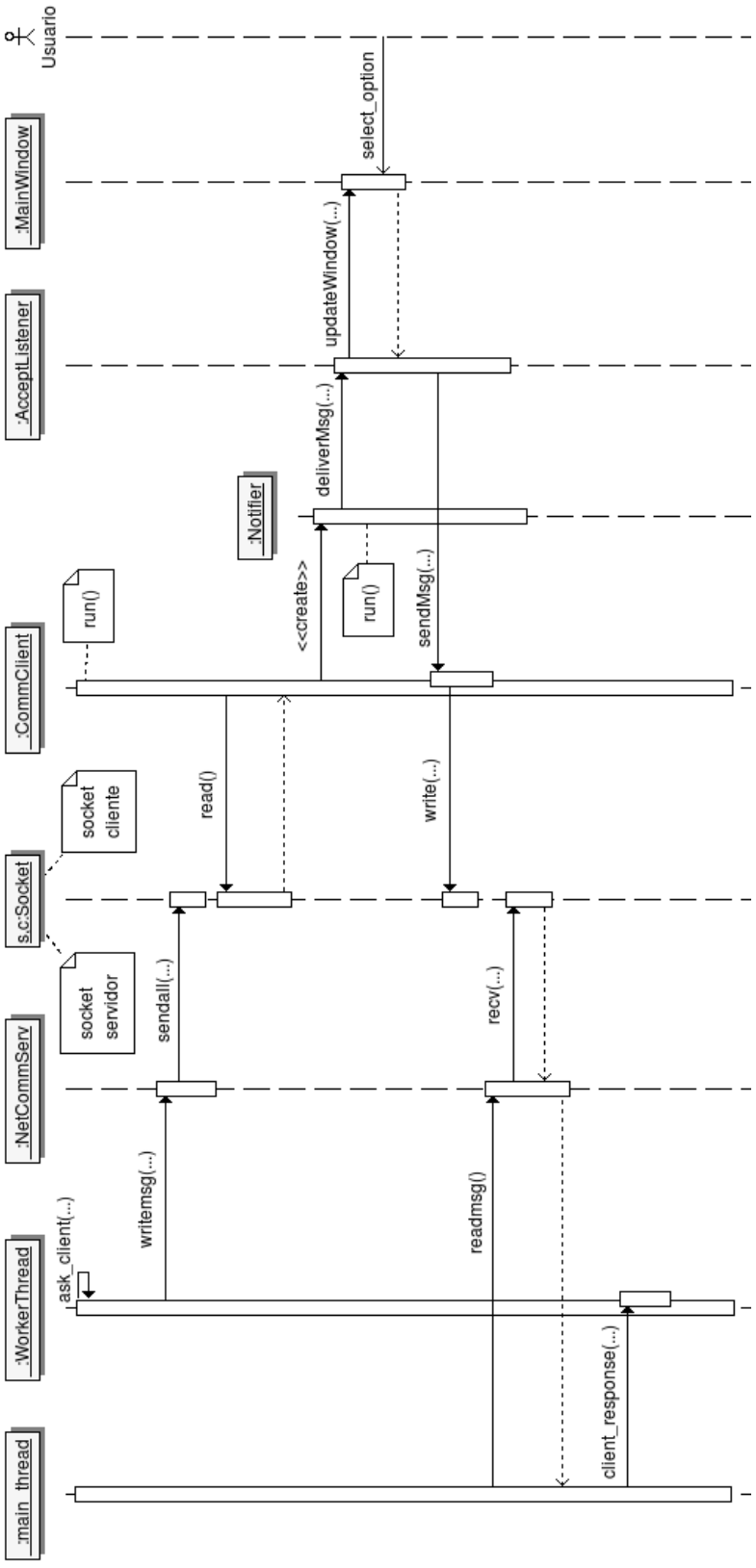

Figura 30: Diagrama de secuencia de una solicitud iniciada por el servidor

5.3 Cuestiones sobre la implementación

En esta sección se comentan algunas decisiones importantes que se han tomado con respecto a ciertas cuestiones relacionadas con la implementación, bien durante esta misma etapa del desarrollo, o bien previamente al comienzo de la misma, estando este último caso motivado por la estimación del riesgo que implicaban para el desarrollo de la aplicación o porque involucraban a determinados requisitos indispensables para acometer las tareas de esta etapa. Asimismo, se explican las razones más importantes que han motivado la adopción de ciertas soluciones que abordan estas cuestiones y, cuando resulte suficientemente interesante, también se concretarán las razones fundamentales por las que se han descartado otras alternativas viables a las soluciones adoptadas.

Adicionalmente, se hará mención a ciertos detalles técnicos cuyo conocimiento ha demostrado ejercer una notable influencia sobre el nivel de corrección, de eficiencia y de efectividad alcanzados por la implementación realizada. Finalmente, todas estas cuestiones serán abordadas en el contexto de diferentes aspectos de la implementación, como son las particularidades de los controladores software (también llamados drivers) o del sistema operativo que presta sus servicios a la aplicación, los lenguajes de programación y las librerías empleados durante la codificación de la aplicación o incluso el intérprete que transforma el código fuente de un programa en algún tipo de código objeto intermedio y lo interpreta dentro de su entorno de ejecución particular.

5.3.1 Sistema operativo

Posiblemente, la cuestión más importante que se abordó en lo que respecta a la implementación fue la adquisición y el procesamiento de tramas 802.11 **“en crudo”** (traducción del término inglés: *raw*), esto es, sin que hayan sido modificadas por el software o por el firmware subyacentes, tras ser recibidas por la estación y antes de ser entregadas a la aplicación de usuario, ni tampoco después de que la aplicación las ponga a disposición del sistema para su transmisión. En consecuencia, como lo que se pretende es la transmisión y la recepción de tramas 802.11 sin la intervención de las capas superiores de la pila de protocolos de red, sería conveniente que el sistema operativo contase con las primitivas necesarias para llevar a cabo estas funciones (independientemente de fabricantes, marcas y modelos de interfaces de red inalámbrica).

Por desgracia, en la práctica, tanto los fabricantes de dispositivos, como los desarrolladores de sistemas operativos propietarios se resisten a revelar información sobre la implementación o sobre el funcionamiento interno de sus productos. Por esta causa, los programadores difícilmente tienen acceso a primitivas de tan bajo nivel. No obstante, existen interfaces inalámbricas, que se venden junto con drivers y aplicaciones propietarios, que disponen de facilidades para el procesamiento de tramas 802.11 en crudo, o incluso drivers propietarios, desarrollados para adaptadores inalámbricos que ya estaban al alcance del consumidor, con prestaciones similares. Sin embargo, estos productos suponen una excepción frente a la gran mayoría de dispositivos y de controladores de software para el hardware (es decir, drivers) que no cuentan con esta capacidad.

Por el contrario, algunos sistemas operativos de fuentes abiertas o que están basados en software libre ofrecen esta facilidad casi de forma ordinaria. En general, esto es debido a que tales sistemas operativos son desarrollados por una comunidad abierta de programadores que no está sometida a las presiones de la industria. Incluso es posible que esta comunidad consiga influenciar a algunos fabricantes de dispositivos para que implementen ciertas características, ya sea a nivel del driver o del firmware del dispositivo, acordes con la funcionalidad que presentan otros dispositivos similares o bien con las exigencias del sistema en el cual se integran. En este sentido, los sistemas operativos basados en distribuciones de *GNU/Linux* han demostrado estar provistos de excelentes facilidades para el desarrollo de aplicaciones de este tipo.

Concretamente, una enorme cantidad de drivers de dispositivos Wi-Fi para Linux permiten la captura de tramas 802.11 en crudo cuando operan en **modo monitor** (es más, estos drivers ofrecen esta facilidad exclusivamente en modo monitor). En este modo, una interfaz Wi-Fi funciona como un receptor de radio pasivo, el cual permite la captura de tramas 802.11 cuando son transmitidas por el mismo canal en el que opera dicha interfaz. Adicionalmente, algunos de estos drivers para Linux soportan la **inyección en modo monitor**, gracias a la cual cuentan con la posibilidad de transmitir tramas 802.11 en crudo a la vez que operan en modo monitor. Actualmente, muchos drivers para Linux basados en el módulo que se denomina *mac80211* soportan la inyección en modo monitor de forma nativa, esto es, sin necesidad de parchear el driver. Como, en lo que concierne a este trabajo, la única opción disponible para procesar tramas 802.11 en crudo consistía en alguna distribución de GNU/Linux, nos ceñiremos a este sistema en el resto de este apartado.

Conviene aclarar, sin embargo, que el procesamiento de tramas 802.11 en crudo llevado a cabo por los drivers, en la práctica, no es tan estricto como ha sido descrito. Por ejemplo, se ha podido comprobar que algunos drivers de interfaces Wi-Fi no incluyen el campo *FCS* en las tramas 802.11 en crudo que entregan a las aplicaciones de usuario, mientras que otros drivers sí lo hacen. Por otro lado, todos los drivers para Linux de este tipo que se han probado tampoco precisan dicho campo para la transmisión de tramas 802.11 en crudo, por lo que será generado por el propio driver o por el firmware del dispositivo. Incluso esta situación puede reproducirse con otros campos de una trama, por ejemplo: el campo *Duration*, cuando el valor asignado por una aplicación de usuario al citado campo es modificado por el firmware de algún dispositivo Wi-Fi previamente a su transmisión.

Una vez conocidas algunas de las particularidades del procesamiento de tramas 802.11 en crudo bajo Linux, es el momento de mencionar la primitiva de este sistema operativo que posibilita este procesamiento. Esta primitiva se denomina en Linux **"socket de paquetes"** (traducción del término inglés: *packet socket*) y no solo se aplica a tramas 802.11, sino a cualquier tipo de trama utilizada en el nivel de enlace. Por tanto, se trata de un tipo de socket que opera a nivel de enlace en la pila de protocolos de red y que interactúa directamente con el driver de la interfaz de red (según se afirma en [Ins2001]). Sin embargo, esta cualidad (esto es, la de ser una primitiva de programación de bajo nivel), en ocasiones, acarrea la dependencia de esta clase de sockets con respecto a ciertos detalles de las implementaciones, como ya se ha mencionado para el caso de las tramas 802.11 en crudo.

Otra dificultad añadida para la utilización de este tipo de sockets es la ausencia de **buffering**, que típicamente asiste en la recepción de tramas o de paquetes en casi todos los tipos de sockets y que es desempeñado por el núcleo del sistema operativo. En consecuencia, sin la intervención de este mecanismo todos los paquetes o las tramas que son transmitidos mientras no está ejecutando la operación de recepción del socket se perderán, esto es, no podrán ser recuperados en las siguientes invocaciones a dicha operación. Afortunadamente, mientras que el desarrollo de la primera versión estable de Linux progresaba, fue presentada una solución llamada *BSD Packet Filter* (abrev. *BPF*) en un congreso sobre UNIX, la cual tenía como objetivo mejorar el rendimiento de las herramientas de monitorización de redes empleadas en sistemas UNIX. Esta solución estaba basada en el filtrado de tramas o de paquetes previamente a que el driver de la interfaz de red los entregase a la pila de protocolos del sistema operativo. De esta forma, un proceso de usuario podía disponer de una copia de las tramas o los paquetes que superasen los filtros, sin que esto afectase al normal procesamiento de las tramas o de los paquetes originales.

Posteriormente, esta facilidad fue incorporada en el núcleo de Linux, tras experimentar algunas modificaciones, bajo la denominación de *Linux Socket Filter*. Precisamente, estos filtros aplicados a los sockets de paquetes habilitan el buffering de las tramas o de los paquetes que hayan superado las condiciones de filtrado. A pesar de esto, durante el desarrollo de las pruebas de la aplicación se descubrió que algunos filtros provocaban que ciertas tramas, que deberían haber sido admitidas tras el proceso de filtrado, en la práctica fuesen descartadas, excepto cuando se establecían unas reglas de filtrado muy sencillas. Posiblemente, este comportamiento anómalo sea debido a que se aborta la recepción de nuevas tramas mientras se evalúan los filtros sobre una trama recién capturada.

5.3.2 Librerías

Una vez disponibles un dispositivo Wi-Fi, un driver para tal dispositivo y un sistema operativo que permitan el procesamiento de tramas 802.11 en crudo, podemos emprender la búsqueda de una librería que facilite la decodificación, la construcción y el envío de este tipo de tramas. En este caso, la única opción disponible, en lo que respecta a los sistemas operativos con esta facilidad, consistía en una distribución de GNU/Linux, por lo que únicamente se contemplaron librerías desarrolladas para este sistema operativo. Dos alternativas que satisfacen los anteriores requisitos son las librerías *LORCON* y *Radiate.* Sin embargo, estas librerías presentan ciertos inconvenientes, algunos de los cuales se enumeran a continuación:

- Están escritas en el lenguaje C, lo que implica que no pueden aprovechar los beneficios de los lenguajes orientados a objetos y, en cambio, presentan las desventajas habituales de este lenguaje en lo que respecta a la gestión de errores, gestión de memoria, comprobación de tipos de datos, etc.
- Su funcionamiento está restringido a un conjunto de drivers para interfaces Wi-Fi específico, por ejemplo, la versión 0.3 de Radiate está restringida a interfaces Wi-Fi que cuentan con el chipset *Prism2*, no obstante, la segunda versión de LORCON soporta un mayor número de drivers, incluyendo los drivers: *wlan-ng*, *hostap*, *madwifi*, *rt73*, *rt2500*, etc.
- Las operaciones proporcionadas por estas librerías están a un nivel muy bajo de abstracción, sobrecargando al programador con abundantes detalles relacionados con la implementación, por ejemplo, características dependientes de los drivers, sockets para la comunicación con el núcleo (sockets *netlink*), etc.

- Carecen de funciones o procedimientos para construir tramas 802.11, como ocurre en el caso de LORCON, lo que obliga a trabajar con la representación binaria de las tramas, o bien cuentan con estas funciones pero no permiten especificar el valor de algunos campos de las tramas, como sucede en el caso de Radiate. No obstante, ambas librerías tienen en común la ausencia de funciones para el procesamiento de paquetes correspondientes a los protocolos de las capas de nivel superior, como ocurre con los protocolos *LLC*, *IP*, *ICMP*, etc.

Todos estos inconvenientes han sido solventados por una librería escrita en Python y publicada bajo licencia *GPL v2*, llamada **Scapy**, que finalmente ha sido la escogida para la manipulación de las tramas 802.11 y que, en consecuencia, ha propiciado la elección de Python como lenguaje de programación para el desarrollo del servidor. En este caso, la citada librería se ha utilizado sobre un sistema GNU/Linux, aunque según afirma el autor de la misma, también debería funcionar en otros entornos, por ejemplo, en ciertos sistemas derivados de Unix e incluso en Windows. Naturalmente, en lo que respecta a las aplicaciones que trabajan directamente con tramas 802.11, sigue siendo un requisito indispensable que tanto el sistema operativo, como el driver de la interfaz de red Wi-Fi subyacentes posibiliten el procesamiento de tramas 802.11 en crudo. Para conseguir este propósito, dicha librería recurre a los sockets de paquetes, ya comentados, cuando ejecuta sobre un sistema basado en Linux.

Otras características que favorecen a *Scapy* sobre las restantes alternativas son su **potencia** y su **facilidad de uso**. Tan extremadamente potente es esta librería que, con una sola sentencia en la que se instancia un objeto, mediante una secuencia de bytes correspondiente a una trama del nivel de enlace, puede reconstruir y representar mediante este objeto, compuesto a su vez de otros objetos, todos los paquetes contenidos en dicha trama, los cuales corresponden a los distintos protocolos de red involucrados en la comunicación de datos, incluyendo los protocolos de nivel de enlace, de red, de transporte y de aplicación. Por otra parte, su facilidad de uso viene avalada por una colección de funciones muy amplia, que implementa las operaciones más comunes. No menos importantes son las características que facilitan la **extensibilidad** de esta librería, tanto a nivel del formato de las tramas o paquetes, como de los protocolos soportados. En este sentido, podemos considerar también a Scapy como un **framework**, ya que permite la introducción de nuevos tipos de paquetes mediante la redefinición de una serie de primitivas existentes.

Si tuviésemos que mencionar alguna desventaja de Scapy, podríamos decir que es una librería extensa, compleja y que requiere un esfuerzo considerable para aprovechar todas las facilidades que nos ofrece. A nivel más práctico, también se advirtió que la clase usada para representar las tramas 802.11 carecía de un atributo para albergar el valor del campo *FCS* de tales tramas, si bien no todos los drivers de dispositivos Wi-Fi incluyen este campo como parte de los datos correspondientes a las tramas que entregan a las librerías o al módulo del núcleo apropiado. Por esta causa, el autor de este trabajo ha propuesto un parche para Scapy que corrige este defecto, según se ha observado en las pruebas realizadas (respondiendo así al *ticket 109* que fue publicado en el sitio para la gestión del proyecto y el seguimiento de errores de Scapy). Durante estas pruebas, también se pusieron de manifiesto otros defectos de poca importancia, achacables a la implementación de la inyección en modo monitor realizada por ciertos drivers.

5.3.3 Lenguajes de programación

Las considerables ventajas que ofrece Scapy frente a las librerías alternativas han decantado la elección del lenguaje de programación utilizado para implementar el servidor, que es el responsable de materializar gran parte de la lógica de la aplicación. Por tanto, se ha acometido la programación del **servidor** empleando el lenguaje **Python**, gracias a lo cual se ha facilitado su desarrollo no solo con los beneficios de la programación orientada a objetos, sino también con las ventajas adicionales que ofrece un lenguaje de *script* como Python. Entre estas ventajas se incluyen su facilidad de uso y de aprendizaje, así como su potencia expresiva, la cual conlleva generalmente una reducción en el tiempo de desarrollo de las aplicaciones que no requieren una gran envergadura. Además, Python es un lenguaje interpretado, implementado en múltiples plataformas y dotado de excelentes facilidades para el tratamiento de errores en tiempo de ejecución (dispone de un mecanismo para la gestión de excepciones similar al de Java y al de C++).

No obstante, Python también adolece de los inconvenientes característicos de los lenguajes de scripts, como los que provienen de la ausencia de declaraciones de tipos o de mecanismos para la ocultación de la información (esto es, mecanismos que restringen la visibilidad de las operaciones y de los atributos). Algunos de estos inconvenientes son: una mayor dificultad para la depuración de errores y para el mantenimiento del código. Por otra parte, una característica particular de Python, como es la delimitación de los bloques de código mediante el uso del mismo nivel de sangrado de las sentencias componentes, puede disuadir a muchos programadores de realizar el anidamiento de

múltiples estructuras de control distintas. Además, actualmente la implementación de referencia del intérprete de Python (codificada en C y denominada *CPython*) también impone limitaciones sobre la concurrencia de las hebras, ya que no permite la ejecución paralela de varias hebras, ni siquiera en sistemas multiprocesador.

Otro inconveniente de este lenguaje es que recurre a librerías externas, implementadas en otros lenguajes de programación, para facilitar el desarrollo de las interfaces gráficas de usuario (abrev. *GUIs*) de las aplicaciones. De hecho, la librería gráfica que está incluida de forma predeterminada en muchas implementaciones de Python (llamada *Tkinter*) no es más que una adaptación (término mediante el cual se ha traducido la expresión del inglés: "*binding*") de la librería *Tk*, originalmente concebida para el desarrollo de GUIs bajo el lenguaje de script *Tcl*. Otras librerías diseñadas para construir GUIs también han sido adaptadas al lenguaje Python y estaban escritas en lenguajes tales como: *C*, *C++* o *Java.* Además, algunas de estas librerías han sido portadas a diferentes sistemas operativos, como: *Windows*, *Mac OS*, *Linux*, *FreeBSD* u otros derivados de Unix, presentando un mayor o menor grado de homogeneidad en todas estas plataformas.

Por el contrario, el intérprete para la versión *Standard Edition* de Java incluye desde versiones muy tempranas las APIs *AWT* y *SWING* para el desarrollo de GUIs. Este intérprete ha sido portado a numerosas plataformas gracias al apoyo de diferentes empresas, organizaciones y comunidades de programadores, aunque son las implementaciones de la propia empresa que concibió este lenguaje (*Sun Microsystems*) las que han demostrado mayor coherencia en las diferentes plataformas para las que fueron destinadas. Por tanto, la portabilidad y la coherencia de las diferentes implementaciones del intérprete, junto con el sencillo diseño, la facilidad de uso y la abundante documentación de las mencionadas APIs para la creación de GUIs han determinado la elección de **Java** como el lenguaje para el desarrollo del **cliente** de esta aplicación.

Capítulo 6

6. Conclusiones y líneas futuras

Uno de los principales objetivos propuestos al iniciar este proyecto fue describir, documentar y demostrar, a nivel práctico, la debilidad del protocolo WEP. Como resultado de estas actividades se han recopilado suficientes evidencias para afirmar que todas las características de seguridad que se intentaron implantar mediante el protocolo WEP han sido violadas. Por lo tanto, cualquier atacante, que disponga de los medios y de las habilidades necesarios, puede descifrar o alterar todos o parte de los datos transmitidos a través del medio inalámbrico y protegidos mediante el protocolo WEP, incluso puede encriptar datos arbitrarios que sean validados por su destinatario, en lo que respecta a su procesamiento por el protocolo WEP. En última instancia y reuniendo la cantidad suficiente de tramas cifradas con la misma clave WEP, el atacante será capaz de descubrir dicha clave, lo que en muchos casos le facultará para descifrar todo el tráfico WEP transmitido por las estaciones de una red Wi-Fi y para intercambiar tramas de datos con cualquiera de estas estaciones.

Aunque existen muchas técnicas para dificultar esta clase de ataques, por ejemplo: prescindir de la difusión del ESSID de la red inalámbrica en las tramas de *Beacon* o inhabilitar la comunicación entre las estaciones pertenecientes al mismo BSS, ninguna resulta infalible ante un atacante con la suficiente destreza, ni siquiera la renovación automática de la clave WEP (esto es, mediante el uso de WEP dinámico) cuando la red alcanza un volumen de tráfico suficientemente alto. Por lo tanto, los mecanismos de seguridad especificados en la primera versión del estándar IEEE 802.11, esto es, el protocolo WEP y la autenticación mediante una clave WEP compartida, han quedado obsoletos y su uso está desaconsejado cuando se dispone de alternativas más seguras, como son WPA y WPA2.

Con el estándar correspondiente a WPA se introdujo el protocolo TKIP, destinado a reemplazar al protocolo WEP en los dispositivos que implementaban este último y a resolver sus abundantes fallos de seguridad. Desde que comenzó a implementarse hasta la actualidad, TKIP ha cumplido con este objetivo de forma bastante aceptable, proporcionando un nivel de seguridad adecuado para la mayoría de usuarios, a pesar de que fueron descubiertas algunas debilidades en este protocolo que posibilitan determinados ataques contra la privacidad, la integridad y la autenticidad de cierta clase

de tramas protegidas mediante este protocolo. No obstante, los ataques que explotan tales brechas en la seguridad de TKIP, en la práctica, están restringidos a tramas de tamaño reducido, así como a redes 802.11 que implementan determinadas características opcionales o bien que son susceptibles de albergar determinados escenarios de ataque especialmente complejos y sofisticados. En cambio, la enmienda 802.11i, en la cual se basa la certificación WPA2, originó y otorgó mayor relevancia al protocolo de seguridad CCMP, el único de obligada implementación por las estaciones compatibles con la anterior enmienda, cuya seguridad no se ha visto amenazada, hasta este momento, por ningún ataque efectivo que sea viable en la práctica.

En cuanto a la autenticación de estaciones de una red Wi-Fi, tanto WPA como WPA2 incorporan un método de autenticación bastante simple y sencillo de implementar (denominado autenticación mediante PSK), apropiado para redes domésticas o de pequeñas empresas y basado en la posesión de una clave compartida por dos o más estaciones de un mismo BSS autorizadas para intercambiar datos entre ellas. No obstante, este método de autenticación es vulnerable a un ataque de diccionario "*offline*", aunque dicho ataque solo es efectivo cuando la clave compartida es generada a partir de una contraseña relativamente simple. Por lo tanto, este ataque es fácilmente evitable prescindiendo de la contraseña empleada para la generación de la clave o bien escogiendo una contraseña con la suficiente entropía para que sea prácticamente imposible que esté incluida en algún diccionario.

El otro tipo de autenticación soportado por los estándares WPA/WPA2, esto es, la autenticación basada en el estándar IEEE 802.1X, requiere una infraestructura más compleja que, generalmente, solo se amortiza en redes de un tamaño considerable, como las que despliegan las grandes empresas u organizaciones. Por otro lado, este tipo de autenticación no especifica un método concreto para validar las credenciales aportadas por las estaciones, sino que permite cierto grado de libertad en la elección de dicho método. En consecuencia, la seguridad de este tipo de autenticación dependerá del método de autenticación específico empleado, aunque los métodos más comunes (por ejemplo: *EAP-TLS*, *EAP-TTLS* y *PEAP*) proporcionan un nivel de seguridad suficientemente alto cuando son implementados correctamente.

Finalmente, en opinión del autor de este trabajo, es posible abrir nuevas líneas de investigación para refinar algunas técnicas que fueron aplicadas mediante el protocolo TKIP con objeto de mitigar las debilidades del protocolo WEP. Aunque, en este caso, estas técnicas refinadas estarían dirigidas a enmendar las debilidades persistentes en el protocolo TKIP que, presumiblemente, están ligadas de forma inherente al legado del protocolo al que sucede, esto es, el mencionado protocolo WEP.

Por ejemplo, podría evitarse la reutilización de un fragmento de keystream en un canal para calidad del servicio, distinto al canal donde fue capturada la trama correspondiente (como propone el ataque de *Beck-Tews*), utilizando la prioridad de la trama en el proceso de generación del keystream (dando por supuesto que a las tramas transmitidas por diferentes canales para calidad del servicio también se les asigna distintas prioridades). De este modo, si se mantienen iguales el resto de parámetros que intervienen en el proceso de generación del keystream, dos tramas emitidas con distintas prioridades deberían ser encriptadas mediante diferentes keystreams.

Otra alternativa más drástica para abordar esta cuestión, por cuanto supone anular una medida estipulada en la enmienda 802.11i, aunque también más efectiva que la solución previa, consiste en prescindir de los distintos contadores de recepción para las tramas protegidas mediante el protocolo TKIP, los cuales se corresponden con los distintos valores de prioridad de las tramas soportados por la estación receptora. Por tanto, si se utiliza un único contador de recepción para todas las tramas recibidas, independientemente de su prioridad, puede descartarse cualquier trama cuyo IV no supere el valor del contador de recepción, impidiendo la reutilización de cualquier fragmento de keystream vinculado a un valor obsoleto del IV.

Sin embargo, incluso adoptando alguna de estas dos soluciones propuestas, la falsificación de tramas, descrita en [MO 2009], así como el descifrado de tramas, referido en [Bec2010], continúan siendo factibles gracias a la variante del ataque Chop-Chop que introdujo el ataque de Beck-Tews. Una solución obvia para rechazar esta variante del ataque concebido originalmente para atacar el protocolo WEP, esto es, el ataque Chop-Chop, requiere solamente suprimir el campo ICV en las tramas protegidas mediante TKIP. En consecuencia, tal medida supone modificar el encapsulado de los datos realizado por este protocolo, lo que inevitablemente implica renunciar a la compatibilidad con las implementaciones actuales del mencionado protocolo. En cambio, este inconveniente podría evitarse recurriendo a otras técnicas que, en vez de alterar el formato de los datos encapsulados por TKIP, modificasen los procedimientos para validar la integridad de las tramas o las condiciones que deben verificarse para que se desencadenen las contramedidas TKIP, con el objetivo fundamental, en todo caso, de impedir el progreso del ataque contra TKIP derivado del ataque Chop-Chop.

En cuanto al protocolo CCMP, el autor de este trabajo coincide con la aseveración realizada en el artículo [JMU2005] sobre la posibilidad (más teórica que práctica) de llevar a cabo un ataque de tipo TMTO sobre este protocolo (no obstante, mantiene la postura de que el citado artículo contiene algunos errores de comprensión del estándar IEEE 802.11-2007 y, en particular, en lo que respecta

al protocolo CCMP, por ejemplo, cuando confunde el bloque inicial de contador utilizado para cifrar la carga de datos de una trama, esto es, el bloque cuyo valor del índice es uno, con el bloque inicial devuelto por la función de formato aplicada a los datos cuya integridad se valida, es decir, los datos de entrada que son procesados por medio del modo CBC-MAC del algoritmo AES).

Sin embargo, en el último artículo citado no se concreta cómo podría efectuarse un ataque de esta clase, esto es: un ataque de tipo TMTO, específicamente contra el protocolo CCMP. Por tanto, podría profundizarse en la investigación sobre métodos para el despliegue de este tipo de ataques contra redes protegidas mediante CCMP y, especialmente, en alternativas al enfoque tradicional que propugna la construcción de una tabla de valores precomputados dependiente de un bloque de texto en claro conocido (esto es, un fragmento de un mensaje de texto en claro conocido del tamaño de un bloque de cifrado) y de su correspondiente bloque de contador encriptado, gracias al cual se cifra el anterior bloque en claro, mediante el modo Contador, empleando numerosas claves distintas.

Por ejemplo, suponiendo que un atacante pueda deducir parte de los datos en claro de una o más tramas cifradas mediante CCMP, entonces será capaz de recuperar algún bloque de contador cifrado (no necesariamente entero) correspondiente a los datos encriptados de estas tramas, cuyo texto en claro supuestamente conoce. Este bloque de contador, o cualquier otro bloque idéntico cifrado con una clave distinta, puede convertirse en el objetivo de un ataque TMTO mediante la generación de una tabla de valores precomputados a partir de los datos en claro del anterior bloque de contador (en el artículo [JMU2005] se detalla como reconstruir los datos en claro de un bloque de contador por medio de la trama en cuyo cifrado interviene dicho bloque).

Generalmente, resulta más fácil obtener un bloque de contador específico, encriptado con una clave cualquiera, que un bloque cifrado, a partir de un bloque de datos en claro y de un bloque de contador específicos, perteneciente a una trama de datos protegida mediante CCMP y generada por una estación concreta. Por lo tanto, la aplicabilidad de esta variante de ataque TMTO aumenta con respecto al enfoque tradicional. Del mismo modo, la tabla de valores precomputados construida por el atacante dependerá solamente de los datos en claro de un bloque de contador específico y de las claves de cifrado atacadas, pero no de un bloque de texto en claro específico cifrado mediante cierto bloque de contador específico, lo que favorece la genericidad del ataque.

Apéndice A: Contenido de los CD-ROMS

Además de la memoria del proyecto impresa y una copia por triplicado de la misma, conforme a la reglamentación del Proyecto Fin de Carrera, se entregan dos CDs en cuyas etiquetas se indica si contienen la memoria o el código fuente del proyecto, respectivamente. El primer CD, además de la memoria, contiene un resumen de la misma, ubicándose en el directorio raíz del CD los ficheros correspondientes a los dos documentos anteriores, que se denominan **memoria.pdf** y **resumen.pdf**, respectivamente. En cambio, el CD con las fuentes del programa contiene los siguientes directorios en el nivel inmediatamente inferior del directorio raíz:

- **Análisis&Diseño**, en este directorio se recopilan las imágenes originales correspondientes a los diagramas, incluidos en la memoria, concernientes al análisis y el diseño de la aplicación implementada, incluyendo los diagramas de casos de uso, de clases y de secuencia.
- **Código**, este directorio contiene el código fuente de la aplicación desarrollada repartido en dos subdirectorios denominados **cliente** y **servidor**, respectivamente, que como sus nombres indican albergan el código fuente de los programas cliente y servidor, los cuales permiten su ejecución de forma independiente y se comunican a través de un socket TCP. Sin embargo, el contenido del directorio *scapy* y del fichero *iwlibs.py* corresponde a librerías que han sido implementadas por terceros, si bien fueron convenientemente modificadas por el autor de este trabajo previamente a su integración con esta aplicación.
- **Ejecutables**, en último lugar, este directorio almacena el código objeto intermedio, esto es, código que un intérprete externo puede ejecutar dentro de su entorno de ejecución particular, correspondiente al código fuente que está presente en el directorio *Código*. Para concretar más, el subdirectorio **client** contiene un conjunto de clases empaquetadas en un archivo *jar* que pueden ser ejecutadas por un intérprete de Java y que implementan el programa cliente, incluyendo la interfaz gráfica de usuario. Cuando es accesible a través de la línea de órdenes un intérprete de Java, el programa cliente puede ser lanzado mediante el script *runclient.bat* (en *Windows*) o el script *runclient.sh* (en *Linux* u otros sistemas derivados de *Unix*). Antes de esta acción, debe ejecutarse el servidor, cuyo código objeto se encuentra en el directorio llamado **server**, para lo cual puede utilizarse el script *runserver.sh*, siempre que exista algún intérprete de Python accesible desde la línea de órdenes.

- **NOTA:** En el fichero **leeme.html**, contenido en el subdirectorio **Ejecutables**, se expone una breve introducción sobre el objetivo, el funcionamiento, la instalación y los requisitos de los sistemas en los que puede ejecutar la aplicación desarrollada, entre otras cuestiones. En ese mismo directorio, otro fichero denominado **licencia.html** contiene una traducción al español de la licencia de uso y de distribución de la aplicación, esto es, la licencia *GPL ver. 2*, la cual es compatible con los términos de las licencias de las aplicaciones, o las librerías de clases o funciones, de terceras partes que se han utilizado para la implementación de la aplicación.

Apéndice B: Glosario

En este apéndice se ha compilado un glosario que incluye las definiciones de los acrónimos más importantes citados en esta memoria, así como los significados de los términos a los que se refieren estos acrónimos, resumidos mediante una breve descripción:

- **AAA** *Authentication, Authorization and Accounting*, generalmente se refiere a una arquitectura de seguridad para sistemas distribuidos que establecen medidas de seguridad para controlar la autenticación, la autorización y la auditoría de los usuarios. También puede hacer referencia a un servidor que implementa ciertos mecanismos de seguridad vinculados a estas medidas.
- **AAA** *Authentication, Authorization and Accounting*, también puede calificar a una clave que se deriva de la MSK, generada a partir de la autenticación EAP, aunque en la práctica muchos métodos EAP no diferencian entre las claves MSK y AAA. A su vez, la clave AAA permite derivar una clave temporal de sesión de algún modo que depende de la suite de cifrado usada.
- **AAD** *Additional Authentication Data*, se refiere a los datos de una MPDU protegida mediante CCMP, caracterizados porque su autenticidad y su integridad están resguardadas por dicho protocolo aunque no su privacidad. Estos datos incluyen todas las direcciones MAC de la MPDU y parte de los campos Frame Control, Sequence Control y también del campo QoS Control, si estuviera presente.
- **ACK** *Acknowledgment*, es un tipo de trama de control mediante la cual una estación confirma que ha recibido una trama de datos, de gestión o incluso una trama de tipo PS-Poll (en el caso de un AP), para lo cual envía un ejemplar de una trama de este tipo dirigida hacia el emisor de la trama recibida.
- **A-MPDU** *Aggregate MAC Protocol Data Unit*, estructura que contiene múltiples MPDUs destinadas a una misma dirección MAC (la misma RA) y transportadas por una única PPDU, aunque su tamaño no puede exceder de 64 KB y cada trama puede necesitar tanto su confirmación (probablemente realizada mediante una trama de confirmaciones en bloque) como su cifrado de forma individual.
- **A-MSDU** *Aggregate MAC Service Data Unit*, estructura que contiene múltiples MSDUs transportadas mediante una única MPDU no fragmentada, siempre que su longitud no exceda de 7935 bytes y que las direcciones MAC de cada MSDU (DA y SA) contenida en la MPDU se correspondan con las mismas direcciones de la MPDU (RA y TA, respectivamente).
- **AP** *Access Point*, un punto de acceso es un nodo (esto es, una estación) distinguido de una red 802.11 que opera en modo Infraestructura. A través del punto de acceso, las restantes estaciones pueden acceder al sistema de distribución de dicha red, aunque también puede desempeñar otras funciones.

- **AS** *Authentication Server*, en la arquitectura definida en el estándar IEEE 802.1X para el control de acceso basado en Puertos, un servidor de autenticación es la entidad encargada de la autenticación de los Clientes, esto es, los dispositivos que intentan acceder a una red, y de conceder o denegar el acceso a tales Clientes en función de las credenciales que presentan.

- **ATIM** *Announcement Traffic Indication Message*, es un tipo de trama de gestión usada en un IBSS, mediante la cual una estación notifica a otra estación, cuando se trata de una trama unicast, o a un grupo de estaciones, cuando se trata de una trama de multicast o de broadcast, que está almacenando una o más MSDUs destinadas a dicha estación o a dichas estaciones que operan en el modo de ahorro de energía.

- **BSS** *Basic Service Set*, es un conjunto de estaciones 802.11 que comparten la misma función de coordinación, por ejemplo, un punto de acceso y todas las estaciones asociadas a éste, o bien todas las estaciones que pertenecen a una red 802.11 que opera en modo Ad-Hoc.

- **BSSID** *BSS Identification*, identificador de 48 bits, similar a una dirección MAC con el formato IEEE 802, que identifica de forma única a un BSS. En el caso de un BSS que opera en modo Infraestructura, el BSSID coincide con la dirección MAC del AP perteneciente a dicho BSS, mientras que en un IBSS se construye a partir de un número aleatorio de 46 bits.

- **CCK** *Complementary Code Keying*, técnica de modulación basada en la codificación mediante espectro expandido, que usa un código de Walsh-Hadamard de 8 chips, y también en la modulación digital mediante DQPSK, de manera que permite alcanzar velocidades de transmisión de 5.5 o de 11 Mbps, en función del código de expansión específico empleado.

- **CCM** *Counter mode with Cipher-block chaining Message authentication code*, modo de cifrado para algoritmos de cifrado en bloque mediante clave simétrica que recurre a su vez a otros dos modos de cifrado en bloque diferentes para proteger los datos, como son: el modo Contador, para proteger la privacidad de los datos, y el modo CBC-MAC, para proteger la integridad y la autenticidad del origen de los datos.

- **CCMP** *Counter mode with CBC-MAC Protocol*, protocolo de seguridad introducido en la enmienda 802.11i, cuya implementación es obligatoria para todas las estaciones compatibles con las RSNs, y que emplea el algoritmo de cifrado AES en modo CCM con una clave de 128 bits para proveer la seguridad requerida por los datos que están vinculados a una asociación de seguridad basada en este protocolo.

- **CSMA/CA** *Carrier Sense Multiple Access with Collision Avoidance*, técnica para arbitrar el acceso a un medio compartido, mediante la cual cada usuario comprueba que el medio no está ocupado antes de transmitir. En otro caso, debe esperar un tiempo aleatorio dado por una función de retroceso exponencial binario. También se evita el acceso simultáneo mediante cierto mecanismo de reserva del medio.

- **CSMA/CD** *Carrier Sense Multiple Access with Collision Detection*, técnica para arbitrar el acceso a un medio compartido, mediante la cual cada usuario comprueba que el medio no está siendo utilizado por otro usuario, antes del inicio y también durante el transcurso de una transmisión. En otro caso, antes de transmitir debe esperar un tiempo aleatorio dado por una función de retroceso exponencial binario.

- **CTS** *Clear To Send*, cierto tipo de trama de control gracias a la cual una estación acepta una solicitud de reserva del medio inalámbrico a través de una trama de tipo RTS, realizada por otra estación que desea enviar una trama, o varios fragmentos de una trama, a la primera estación, esto es, la estación receptora de la trama de RTS.
- **DCF** *Distributed Coordination Function*, función que facilita la coordinación para el acceso al medio inalámbrico de las estaciones pertenecientes a un mismo BSS y que está basada en el método de acceso al medio CSMA/CA. Cualquier estación que cumpla con las exigencias del estándar 802.11 debe implementar esta función.
- **DIFS** *DCF Inter-Frame Space*, tiempo mínimo que debe esperar una estación que opera bajo la DCF, a causa del mecanismo de control de acceso al medio, para iniciar una secuencia de intercambio de tramas (que consta de al menos una trama).
- **DS** *Distribution System*, el sistema de distribución de una red 802.11 posibilita la interconexión de los distintos BSSs que integran un ESS y, posiblemente también, la conexión de estos componentes con otras LANs diferentes.
- **DSSS** *Direct Sequence Spread Spectrum*, se trata de una técnica de codificación de canal mediante espectro expandido basada en la utilización de un código de expansión seudoaleatorio, p. ej. en el estándar IEEE 802.11 se emplea un código de Barker de 11 chips, que hace que el espectro de la señal se asemeje al ruido.
- **DBPSK** *Differential Binary Phase Shift Keying*, modulación basada en el desplazamiento diferencial de fase binario, esto es, una técnica de modulación digital en la que cada bit está representado por una diferencia de fase de 0^{o} o bien de 180^{o} entre dos elementos consecutivos de la señal.
- **DQPSK** *Differential Quadrature Phase Shift Keying*, modulación mediante desplazamiento diferencial de fase en cuadratura, esto es, una técnica de modulación digital en la que cada bit está representado por una diferencia de fase de 0^{o}, 90^{o}, 180^{o} o 270^{o} entre dos elementos consecutivos de la señal.
- **EAP** *Extensible Authentication Protocol*, es un protocolo que facilita la autenticación de usuarios o dispositivos gracias a la definición de un marco de trabajo para la autenticación, que puede ser extendido mediante la adición de nuevos métodos o procedimientos de autenticación. También especifica el formato genérico de los paquetes y los mecanismos para la retransmisión y la detección de duplicados.
- **EAPOL** *EAP Over Lan*, formato de paquetes de red definido en el estándar IEEE 802.1X para transportar los paquetes EAP intercambiados entre un Cliente (Supplicant) y un Autenticador, incluyendo los datos transmitidos por ambos tipos de entidades con el propósito de generar material de claves.
- **EIFS** *Extended Inter-Frame Space*, tiempo mínimo que debe esperar una estación que opera bajo la DCF para iniciar la transmisión de una trama 802.11, impuesto por el mecanismo de control de acceso al medio, tras la detección de alguna trama errónea, esto es, una trama incompleta o cuyo campo FCS no es válido.
- **ERP-PBCC** *Extended Rate Phy-Packet Binary Convolutional Code*, variante de la modulación PBCC, redefinida en la enmienda 802.11g, y que permite alcanzar velocidades de transmisión de 22 y 33 Mbps, gracias a un código convolucional de tasa 2/3, a la modulación 8-PSK y a las velocidades de modulación de 11 y de 16.5 MBaudios.

- **ESS** *Extended Service Set*, un conjunto de BSSs y, posiblemente también, una o más LANs de cualquier tipo, que aparentan constituir un único BSS, desde el punto de vista de la capa LLC de una estación asociada a cualquiera de estos BSSs.
- **ESSID** Véase *SSID*.
- **FCS** *Frame Check Sequence*, último campo de una trama 802.11, el cual contiene un código o suma de comprobación de 32 bits que se calcula sobre todos los campos de la trama, excepto sobre el propio campo FCS, mediante el algoritmo CRC-32 y que permite la detección de errores durante la recepción de la trama.
- **FHSS** *Frequency Hop Spread Spectrum*, técnica de codificación de canal mediante espectro expandido basada en alternar la transmisión por canales con diferentes frecuencias, p. ej. en el estándar 802.11 se emplean no más de 80 canales con un ancho de banda de 1 MHz cada uno.
- **GFSK** *Gaussian Frequency Shift Keying*, es una técnica de modulación digital mediante desplazamiento en frecuencia que se caracteriza por que la señal en banda base que codifica los datos binarios, antes de ser modulada, se procesa por medio de un filtro gaussiano, reduciendo así el espectro de la señal modulada.
- **GMK** *Group Master Key*, clave maestra o material de claves que típicamente participa en la generación de la GTK, desempeñando el papel de clave de una función hash que se aplica sobre una cadena de caracteres predefinida, la dirección MAC del Autenticador y un número aleatorio o seudoaleatorio, entre otros datos.
- **GTK** *Group Transient Key*, conjunto de claves temporales utilizadas para la protección del tráfico de broadcast/ multicast, que tiene una longitud de 40 o 104, 128 o bien 256 bits, en función de que la suite de cifrado empleada para el objetivo anterior sea WEP, CCMP o TKIP, respectivamente. En realidad, este conjunto contiene una clave: la TK, que en ocasiones se divide en varias claves con distintas funciones.
- **HR-DSSS** *High Rate-Direct Sequence Spread Spectrum*, especificaciones del nivel físico que se introdujeron en la enmienda 802.11b y que permiten transmitir a la velocidad de 5.5 o de 11 Mbps mediante las técnicas de codificación CCK o PBCC y las modulaciones (D)BPSK o (D)QPSK, empleando para esto el mismo esquema de canales que la capa basada en DSSS.
- **IBSS** *Independent Basic Service Set*, término con el que se designa a una red 802.11 que opera en modo Ad hoc, esto es, un BSS que constituye una red independiente y autocontenida, por lo que no dispone de sistema de distribución.
- **ISM** *Industrial, Scientific and Medical* (*ICM*, en español), se refiere a un conjunto de bandas de radio-frecuencia reservadas internacionalmente para su uso industrial, científico o médico, por lo que generalmente no se requiere la obtención de una licencia para la transmisión a través de tales bandas.

- **ICV** *Integrity Check Value*, código o secuencia de 4 bytes de longitud que se emplea para comprobar la integridad de la carga de datos de una trama 802.11 protegida mediante el protocolo de seguridad WEP. Este código es generado mediante una suma de comprobación basada en una variante del algoritmo CRC-32, de forma similar a como se genera el valor del campo FCS de la misma trama.

- **IE** *Information Element*, campo presente en muchas tramas de gestión, en las cuales pueden existir una o varias instancias de este campo que contienen información de diversa índole sobre un BSS. El formato de un IE define dos subcampos de 1 byte de longitud que determinan el tipo y la longitud del IE, y un subcampo de longitud variable con la información específica del IE.

- **IEEE** *Institute of Electrical and Electronics Engineers*, organización de profesionales que promueve la innovación y la excelencia tecnológica, y fomenta el desarrollo de estándares en diversos ámbitos industriales, por ejemplo, en el que concierne a a las redes de computadores, mediante la familia de estándares 802.

- **IETF** *Internet Engineering Task Force*, organización internacional abierta y sin ánimo de lucro que propone, estudia, aprueba y publica estándares relacionados con la arquitectura, la regulación, el desarrollo y el adecuado funcionamiento de Internet.

- **ISO** *International Organization for Standardization*, organización internacional que fomenta el desarrollo de normas y estándares internacionales para la fabricación y el comercio en una amplia variedad de ramas industriales y que está compuesta por los organismos de estandarización nacionales de la mayoría de países.

- **ITU** *International Telecomunnication Union*, organismo especializado de las Naciones Unidas que se encarga de elaborar estándares y regular las telecomunicaciones, entre otras acciones de ámbito internacional, incluyendo la coordinación para el uso del espectro radioeléctrico global y de las órbitas de los satélites.

- **IV** *Initialization Vector*, en el contexto de los protocolos de seguridad WEP y TKIP, un vector de inicialización es una secuencia de datos que se utiliza para componer o para calcular una clave de cifrado. En el caso de los anteriores protocolos, el IV permite construir una clave distinta para el cifrado de cada MPDU.

- **KCK** *Key Confirmation Key*, clave de 128 bits perteneciente a la PTK y que se utiliza para verificar la autenticidad/integridad de los paquetes de tipo EAPOL-Key que transportan ciertos mensajes del 4-Way Handshake o del Group Key Handshake, mediante la función hash basada en clave HMAC-MD5 o HMAC-SHA1-128.

- **KEK** *Key Encryption Key*, clave de 128 bits, perteneciente a la PTK, que se emplea para cifrar el campo Key Data de ciertos paquetes de tipo EAPOL-Key que transportan una GTK y que están involucrados en una instancia del 4-Way Handshake o bien del Group Key Handshake. El algoritmo utilizado para esta operación de cifrado puede ser ARC4 o bien AES Key Wrap.

- **LDPC** *Low Density Parity Check Code*, se trata de una clase de códigos correctores de errores lineales que operan sobre bloques y que se caracterizan principalmente porque las filas y las columnas de sus correspondientes matrices de comprobación de paridad son dispersas.

- **MAC** *Medium Access Control*, es la subcapa inferior de las dos subcapas en las que típicamente se divide la capa de enlace. Entre las funciones desempeñadas por esta subcapa se encuentran el control de acceso y el direccionamiento de los nodos que comparten el medio de transmisión, el empaquetamiento y la fragmentación de MSDUs mediante MPDUs y el control de errores de estas últimas.

- **MAC** *Message Authentication Code*, también puede referirse a un código o secuencia de datos utilizada para autenticar un mensaje. Típicamente, este código es generado por una función hash que recibe como entrada una clave secreta y un mensaje de longitud arbitraria cuya integridad y autenticidad se pretende proteger.
- **MIB** *Management Information Base*, base de datos virtual con estructura jerárquica que contiene información relacionada con la gestión de las entidades de protocolos implementadas por los computadores o los dispositivos que forman parte de una red de comunicaciones. Esta base de datos se compone de objetos gestionados con diferentes nombres, identificadores, tipos de datos y niveles de acceso.
- **MIC** *Message Integrity Code*, definición similar a la acepción *Message Authentication Code* del acrónimo MAC.
- **MIMO** *Multiple Input Multiple Output*, es un conjunto de técnicas de procesamiento de la señal en las cuales se utilizan diferentes antenas y transceptores para transmitir distintas señales simultáneamente y para la recepción de diferentes codificaciones o combinaciones de las señales transmitidas. Entre estas técnicas se incluyen la precodificación, la multiplexación espacial y la diversidad de código.
- **MLME** *MAC subLayer Management Entity*, entidad de gestión de la subcapa MAC, esto es, la entidad que facilita el acceso y mantiene los atributos de la MIB asociados a la subcapa MAC. En una estación 802.11, esta entidad ofrece ciertas primitivas involucradas en la implementación de algunos servicios y funciones relacionados con la gestión de un BSS o la interacción entre sus estaciones.
- **MMPDU** *MAC Management Protocol Data Unit*, unidad de datos intercambiada entre dos entidades simétricas (peer) de la subcapa MAC, como parte de alguna interacción relacionada con el protocolo de gestión de la subcapa MAC.
- **MPDU** *MAC Protocol Data Unit*, unidad de datos intercambiada entre dos entidades que implementan la subcapa MAC de dos nodos o estaciones de una red, recurriendo a los servicios de la capa del nivel físico para este fin. Generalmente, las MPDUs se corresponden con las tramas intercambiadas a nivel del enlace de datos.
- **MSDU** *MAC Service Data Unit*, unidad de datos entregada por una entidad de la subcapa LLC a la subcapa MAC para su envío a otra entidad simétrica presente en otra estación. Generalmente, esta MSDU constituirá la carga de datos de una MPDU, aunque también puede ser fragmentada por la subcapa MAC para su transporte en fragmentos más pequeños mediante múltiples MPDUs.
- **MSK** *Master Session Key*, se trata de una clave o bien de material de claves generado por algunos métodos EAP, una vez que la autenticación ha concluido con éxito, de alguna forma específica establecida por el método EAP concreto. Esta clave o material de claves debe tener una longitud de 64 octetos, como mínimo, según se

especifica en el RFC 3748.

- **NAV** *Network Allocation Vector*, una especie de temporizador interno mantenido por cada estación que se actualiza por medio de la detección de cierta clase de tramas y que, además, se emplea para inhibir la transmisión de tramas por dicha estación cuando el valor del contador vinculado al citado temporizador es mayor que cero, independientemente del resultado de la detección de la portadora física.

- **NIST** *National Institute of Standards and Technology*, agencia federal de los E.E.U.U. que carece de poder regulador pero promueve la innovación y la competitividad industrial a través del fomento de métricas científicas, estándares y tecnologías, con el objetivo de favorecer la estabilidad económica, estimular el comercio y mejorar la calidad de vida.

- **OFDM** *Orthogonal Frequency-Division Multiplexing*, técnica que permite la codificación de datos binarios mediante múltiples señales, llamadas subportadoras, las cuales se multiplexan mediante FDM. En las capas del nivel físico especificadas en las enmiendas 802.11a/g se emplean 52 subportadoras moduladas mediante BPSK, QPSK, 16-QAM o 64-QAM, aunque cuatro subportadoras no codifican datos.

- **OSI** *Open Systems Interconnection*, modelo de referencia elaborado por la ISO para esbozar la arquitectura de interconexión de los sistemas de comunicaciones. Este modelo consta de siete capas organizadas de forma jerárquica, de manera que cada capa solo puede solicitar los servicios de la capa inmediatamente inferior.

- **PAE** *Port Access Entity*, una entidad de acceso a Puerto controla el estado de un Puerto 802.1X, posibilitando el intercambio de tráfico de autenticación con una entidad similar. Típicamente, en este intercambio una PAE, que se denomina Autenticador, restringe el acceso a un Puerto, mientras que la otra PAE intenta ser autenticada por la anterior para conseguir el acceso y se denomina Cliente (Supplicant).

- **PBCC** *Packet Binary Convolutional Coding*, es una técnica de modulación basada en la codificación de canal mediante un código convolucional con una tasa de datos de 1/2 y las modulaciones BPSK o QPSK, gracias a las cuales puede alcanzar una velocidad de transmisión de 5.5 o de 11 Mbps, respectivamente.

- **PC** *Point Coordinator*, entidad que se encarga de conceder el turno de transmisión a las estaciones que operan mediante la PCF durante el período libre de contención. En un BSS, cuyas estaciones están coordinadas por la misma instancia de la PCF, este rol es desempeñado por el AP.

- **PCF** *Point Coordination Function*, función utilizada para coordinar el acceso al medio inalámbrico de algunas o todas las estaciones de un BSS que funciona en modo Infraestructura y que se caracteriza por la presencia de una entidad, llamada PC, encargada de otorgar el derecho a transmitir a tales estaciones. La implementación de la PCF es opcional, pero de realizarse se requiere su coexistencia con la DCF.

- **PIFS** *PCF Inter-Frame Space*, intervalo de tiempo mínimo, que debe esperar el PC de un BSS que funciona bajo la PCF, para anunciar el comienzo de un período sin

contención mediante una trama de Beacon o para retomar el control del medio tras el envío de una trama que requiere confirmación, sin que dicha confirmación haya sido recibida después de un intervalo SIFS.

- **PKI** *Public Key Infraestructure*, se trata de una infraestructura, incluyendo sistemas, aplicaciones, personas, políticas y procedimientos, dedicada a la creación, gestión, distribución, uso y revocación de certificados digitales. El objetivo fundamental de la citada infraestructura es la vinculación de las identidades de un conjunto de usuarios u otras entidades a las claves públicas que les han sido asignadas.

- **PLCP** *Physical Layer Convergence Protocol*, subcapa superior de las dos subcapas en las cuales se suele dividir la capa física, siendo su propósito fundamental mantener la independencia de la subcapa MAC con respecto al medio físico de transmisión. Para esto desempeña funciones tales como el entramado y la sincronización en la transmisión de MPDUs y la detección del estado del medio.

- **PMD** *Physical Medium Dependent*, subcapa inferior, de las dos subcapas en las cuales el IEEE suele dividir una capa física, que actúa como interfaz directa con el medio de transmisión. Su función principal es la transmisión/recepción de los bits que contiene una PPDU mediante su modulación/demodulación mediante una señal apropiada para su propagación por un medio de transmisión definido.

- **PMK** *Pairwise Master Key*, clave maestra de 256 bits que se deriva a partir de la clave MSK, mediante la autenticación 802.1X, en cuyo caso es válida durante la sesión en la que tiene vigencia dicha autenticación, o bien coincide con un clave estática configurada mediante un método "fuera de banda", llamada PSK. En cualquier caso, la PMK se utiliza para la generación de la PTK.

- **PN** *Packet Number*, número de secuencia de 48 bits incluido en la cabecera CCMP de una MPDU protegida mediante CCMP, empleado para actualizar el contador de repetición apropiado del receptor de la MPDU, cuando sea necesario, previniendo así los ataques de repetición de MPDUs. También se utiliza en la construcción del nonce requerido para el cifrado y la verificación de la integridad de dicha MPDU.

- **PPDU** *PLCP Protocol Data Unit*, unidad de datos intercambiada entre dos entidades simétricas que implementan la subcapa PLCP de la capa física. En el caso de las redes 802.11, este tipo de PDU contiene la MPDU, posiblemente tras la aplicación de alguna codificación de canal y siendo precedida, en general, por un preámbulo y una cabecera que indica la longitud y la modulación de la MPDU.

- **PSK** *Pre-Shared Key*, clave maestra de 256 bits que puede ser configurada en dos o más estaciones de un BSS para intercambiar datos bajo la protección de WPA o WPA2. La configuración de la misma instancia de la PSK en dos estaciones actúa como una forma de autenticación implícita entre tales estaciones. Además, la PSK desempeña el rol de la PMK en las estaciones que usan este tipo de autenticación.

- **PTK** *Pairwise Transient Key*, colección de claves temporales con una longitud de 384 o 512 bits, dependiendo de que la suite de cifrado unicast usada sea CCMP o TKIP, respectivamente. Esta colección contiene las claves llamadas KCK, KEK y TK y se deriva a partir de la PMK mediante el 4-Way Handshake, por lo que es válida

durante la sesión en la que está vigente dicha instancia del 4-Way Handshake.

- **QAM** *Quadrature Amplitude Modulation*, se trata de una técnica de modulación digital por medio de una señal portadora que puede ser modulada tanto en fase como en amplitud, en función de los datos de entrada. En el caso de la modulación digital mediante 16-QAM y 64-QAM, estas técnicas emplean 16 y 64 símbolos distintos, respectivamente, cada uno de los cuales codifica 4 o 6 bits, según la técnica usada.

- **RADIUS** *Remote Authentication Dial-In User Service*, inicialmente fue un protocolo de red diseñado para ayudar a centralizar la autenticación de los usuarios conectados a un conjunto de servidores de acceso a la red, los cuales delegarían la autenticación en uno o más servidores RADIUS. Después se añadieron funciones complementarias a este protocolo para la autorización y la auditoría de los usuarios.

- **FC** *Request For Comments*, memoria u otra clase de documento técnico redactado o publicado por el IETF, que realiza una propuesta sobre algún método, técnica, protocolo o cualquier otra innovación cuyo propósito sea el desarrollo o el avance de Internet o de los sistemas conectados a Internet.

- **RIFS** *Reduced Inter-frame Space*, tiempo mínimo que una estación que implementa la enmienda 802.11n debe esperar, por causa de la función de acceso al medio, antes de transmitir una trama. El uso de este intervalo de espera reducido debe limitarse a ráfagas de tramas enviadas por el mismo transmisor hacia el mismo receptor inmediato (esto es, solo se aplica a tramas cuyas direcciones TA y RA no varían).

- **RSN** *Robust Security Network*, denota una red 802.11 en la que las únicas asociaciones de seguridad permitidas son RSNAs. Adicionalmente, este tipo de redes pueden ser distinguidas por la presencia de un campo, llamado RSN Information Element, en las tramas de Beacon, Probe Response, (Re-)Association Request y también en otras tramas intercambiadas durante una instancia del 4-Way Handshake.

- **RSNA** *Robust Security Network Association*, se trata de una asociación de seguridad, esto es, un conjunto de políticas, claves y otros datos compartidos por dos estaciones con el propósito de intercambiar información de forma segura, que está basada en los mecanismos de seguridad introducidos en la enmienda 802.11i, como son los protocolos TKIP o CCMP y la autenticación mediante PSK o el estándar 802.1X.

- **RTS** *Request To Send*, un tipo de trama de control que se utiliza para reservar el medio inalámbrico previamente a un intercambio de tramas iniciado por el emisor de una trama de este tipo. Entonces, si la reserva es aceptada por la estación de destino, tanto esta estación como la estación de origen podrán transmitir sin interferencia de las restantes estaciones dentro de su alcance.

- **SDM** *Spatial Diversity Multiplexing*, la multiplexación por diversidad espacial es una técnica MIMO que permite mejorar el rendimiento de las comunicaciones gracias a la transmisión de datos independientes mediante distintas señales emitidas por el mismo canal pero a través de distintas antenas, siendo recibidas y procesadas estas señales por medio de múltiples antenas y dispositivos de recepción.

- **SIFS** *Short Inter-Frame Space*, tiempo mínimo que una estación 802.11, conforme con el estándar original, debe esperar antes de transmitir una trama, a consecuencia del mecanismo de control de acceso al medio. También los sucesivos fragmentos de una ráfaga de tramas se transmiten después de un intervalo SIFS, así como ciertas

tramas, como las de ACK, CTS o muchas de las dirigidas hacia/desde el PC.

- **SME** *Station Management Entity*, entidad implementada por una estación 802.11, pero no vinculada a una capa OSI concreta sino que reside en un plano de gestión independiente interactuando con las entidades de gestión de varias capas. Según el estándar, interactúa con las entidades de gestión de las capas MAC y física para recopilar el estado o alterar el valor de varios parámetros de estas capas.

- **SSID** *Service Set Identifier*, identificador lógico, típicamente una cadena de caracteres codificada en ASCII que no excede de 32 bytes, que indica la identidad de una red 802.11, ya sea un IBSS o un ESS. En el caso de una red en modo Infraestructura, esto es: un ESS, también recibe el nombre de ESSID.

- **STA** *Station*, una estación 802.11 es un nodo de una red implementada conforme a este estándar, esto es, un dispositivo compatible con las especificaciones del estándar IEEE 802.11 e integrado en una red de esta clase.

- **STBC** *Space-Time Block Coding*, técnica MIMO basada en la diversidad de código que pretende mejorar la fiabilidad de la comunicación de datos y que esencialmente consiste en la transmisión redundante de datos gracias a su modulación mediante distintas señales que emplean diferentes codificaciones.

- **TK** *Temporal Key*, clave incluida en una PTK o una GTK y usada para el cifrado y la verificación de la autenticidad/integridad de los datos correspondientes al tráfico unicast o de broadcast/ multicast, respectivamente. Actualmente, el tamaño de esta clave puede ser de 128 o 256 bits, dependiendo de que la PTK o la GTK que la contiene esté vinculada a la suite de cifrado CCMP o TKIP, respectivamente.

- **TKIP** Temporal Key Integrity Protocol, protocolo de seguridad que introdujo WPA para superar las debilidades de WEP, manteniendo la compatibilidad con el hardware que soportaba éste. Después se incluyó en la enmienda 802.11i como una opción para las RSNs. TKIP protege la privacidad mediante una clave RC4 de 128 bits y la autenticidad mediante el algoritmo Michael y dos claves de 64 bits.

- **TLS** *Transport Layer Security*, protocolo de seguridad creado para reemplazar a SSL, el cual proporciona mecanismos para proteger la privacidad, mediante criptografía simétrica, y la integridad, mediante una función HMAC segura basada en clave, siendo aplicados sobre los protocolos del nivel de aplicación y operando sobre un protocolo de nivel de transporte fiable.

- **TSC** *TKIP Sequence Counter*, contador de secuencia de 48 bits que precede a la carga de datos de una MPDU protegida mediante TKIP, con el fin de evitar o paliar los ataques de repetición. Este contador se incrementa de forma monótona en cada envío de una trama de datos vinculada a una asociación de seguridad basada en TKIP y también participa en la generación de la clave RC4 para cada trama.

- **TSN** *Transition Security Network*, un tipo de red 802.11 que permite ambas clases de asociaciones de seguridad, esto es, tanto Pre-RSNAs, como RSNAs. Además este tipo de redes pueden identificarse porque anuncian el uso de WEP como suite de cifrado para grupo dentro de los RSN IEs presentes en las tramas de Beacon.

- **TXOP** *Transmission Opportunity*, intervalo de tiempo durante el cual, una estación que

implemente las técnicas para la provisión de calidad del servicio conforme a la enmienda 802.11e, obtiene el derecho a iniciar una secuencia de intercambio de tramas a través del medio inalámbrico.

- **TxBF** *Transmit Beamforming*, técnica MIMO basada en la precodificación, esto es, una técnica ejecutada solo por el transmisor y que consiste en la transmisión de datos idénticos mediante señales idénticas pero con cierto desfase y, posiblemente, con distinta potencia, para favorecer la interferencia constructiva de estas señales en la región ocupada por el receptor y para aumentar el nivel de la señal recibida.

- **WECA** *Wireless Ethernet Compability Alliance*, organización de empresas con interés en la tecnología Wi-Fi constituida en el año 1999 y que posteriormente originaría la Wi-Fi Alliance.

- **WDS** *Wireless Distribution System*, sistema que permite la interconexión de los puntos de acceso de una red 802.11, que opera en modo Infraestructura, mediante enlaces inalámbricos. Aunque este sistema puede reducir el rendimiento de la red cuando las estaciones y los APs usan las mismas bandas de comunicaciones, preserva la semántica de las tramas 802.11 cuando son intercambiadas entre estos APs.

- **WEP** *Wired Equivalent Privacy*, protocolo de seguridad introducido en el estándar IEEE 802.11 original, cuyo objetivo fundamental era dotar a las redes 802.11 de una seguridad similar a la de las redes cableadas, a pesar de que su implementación es opcional. Este protocolo protege la confidencialidad y la integridad de los datos mediante los algoritmos RC4 y CRC-32, respectivamente.

- **WFA** *Wi-Fi Alliance*, asociación de empresas encargada de promover las redes locales inalámbricas basadas en el estándar 802.11 y de certificar la interoperabilidad de los productos basados en esta tecnología a través de una serie de certificaciones avaladas mediante la marca comercial Wi-Fi.

- **WPA** *Wi-Fi Protected Access*, protocolo de seguridad especificado por la Wi-Fi Alliance como una solución interina a los fallos de seguridad de WEP, que está basado en la versión 3.0 del borrador de la enmienda 802.11i y que pretende mejorar la privacidad, la autenticidad y la integridad de WEP mediante el protocolo TKIP y la autenticación de WEP mediante el estándar IEEE 802.1X y el protocolo EAP.

- **WPA2** *Wi-Fi Protected Access 2*, protocolo de seguridad basado en la enmienda 802.11i y que cuenta con todas las características de seguridad de WPA. También soporta la protección de la privacidad/integridad/autenticidad de las tramas de datos gracias al protocolo CCMP, la aplicación de los mecanismos de seguridad introducidos por WPA a un IBSS y la preautenticación de las estaciones.

Apéndice C: Bibliografía

En este apéndice se enumeran los principales recursos bibliográficos que fueron descubiertos, examinados y posteriormente seleccionados como material introductorio, de consulta, de referencia o bien para analizar con mayor profundidad o, por el contrario, para contribuir a la síntesis de las cuestiones abordadas en este trabajo. A consecuencia de la abundancia y de la naturaleza tan dispar de la documentación recopilada, tales recursos bibliográficos se han clasificado en varías categorías, las cuales incluyen libros, especificaciones técnicas, trabajos de investigación, artículos de revistas y también páginas web de sitios especializados en materias afines. Finalmente, cabe aclarar que se han omitido las publicaciones a las cuales pertenecen los artículos científicos y otros documentos citados, debido a que se desconocía su fuente de publicación primaria y también que se ha alterado el orden convencional de los elementos de una referencia (p. ej. autores, título, publicación, fecha, etc) con el propósito de mejorar la legibilidad y de resaltar el tema abordado por el documento al que se hace referencia.

Libros, estándares y especificaciones técnicas

[AE 2004] *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*, W. A. Arbaugh, J. Edney. Addison-Wesley, 2004.

[BJR1999] *UML. El Lenguaje Unificado de Modelado*, G. Booch, I. Jacobson, J. Rumbaugh. Addison-Wesley, 1999.

[Gas2005] *802.11 Wireless Networks: The Definitive Guide, 2nd Edition*, M. S. Gast. O'Reilly Media, 2005.

[HD 2005] *Security in Wireless LANs and MANs*, T. Hardjono, L. R. Dondeti. Artech House, 2005.

[KR 2008] *Computer Networking: A Top-Down Approach*, *4ª edición*, J. F. Kurose, K. W. Ross. Addison-Wesley, 2008.

[Sta2000] *Comunicaciones y redes de computadores*, *6ª edición*, W. Stallings. Prentice Hall, 2000.

[Tan2003] *Redes de computadoras, 4ª edición*, A. S. Tanenbaum. Pearson Education, 2003.

[802.11] *IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Computer Society, ediciones de 1999 y de 2007.

[802.1X] *IEEE 802.1X: Port-Based Network Access Control*, IEEE Computer Society, edición de 2004.

[RFC2898] *PKCS #5: Password-Based Cryptography Specification*, B. Kalisky, 2000.

[RFC3610] *Counter with CBC-MAC (CCM)*, D. Whiting, R. Housley, N. Fergurson, 2003.

[RFC3748] *Extensible Authentication Protocol (EAP)*, B. Aboba, L. Blank, J. Vollbrecht, J. Carlson, H. Levkoweltz, 2004.

Artículos (papers) y trabajos de investigación

[ANN2002] *Man-in-the-Middle in tunneled authentication protocols*, N. Asokan, V. Niemi, K. Nyberg.

[Arb2001] *An inductive chosen plaintext attack against WEP/WEP2*, W.A. Arbaugh.

[ASW2001] *Your 802.11 wireless network has no clothes*, W.A. Arbaugh, N. Shankar, Y. C. J. Wan.

[BGW2001] *Intercepting mobile communications: the insecurity of 802.11*, N. Borisov, I. Goldberg, D. Wagner.

[BHL2006] *The final nail in WEP's coffin*, A. Bittau, M. Handley, J. Lackey.

[Bec2010] *Enhanced TKIP Michael attacks*, M. Beck.

[Cha2006] *Break WEP faster with statistical analysis*, R. Chaabouni.

[FM 2000] *Attacks on additive encryption of redundant plaintext and implications on Internet security*, S. Fluhrer, D. A. McGrew.

[FMS2001] *Weakneses in the Key Scheduling Algorithm of RC4*, S. Fluhrer, I. Mantin, A. Shamir.

[FMS2002] *Attacks on RC4 and WEP*, S. Fluhrer, I. Mantin, A. Shamir.

[Hel1980] *A criptanalytic Time-Memory Trade-Off*, M. E. Hellman.

[Hul2002] *Practical exploitation of RC4 weaknesses in WEP enviroments*, D. Hulton.

[HM 2005] *Security analysis and improvements for IEEE 802.11i*, C. He, J. C. Mitchell.

[IRS2001] *Using the Fluhrer, Mantin, and Shamir attack to break WEP*, J. Ioannidis, A. D. Rubin, A. Stubblefield.

[IRS2004] *A key recovery attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)*, J. Ioannidis, A. D. Rubin, A. Stubblefield.

[JMU2005] *Vulnerabilities of IEEE 802.11i wireless LAN CCMP protocol*, M. Junaid, M. Mufti, M. Umar.

[MO 2009] *A practical message falsification attack on WPA*, M. Morii, T. Ohigashi.

[PTW2007] *Breaking 104 bit WEP in less than 60 seconds*, A. Pyshkin, E. Tews, R. P. Weinmann.

[LT 2010] *CCMP known plaintext attack*, L. Lueg, D. P. Tomcsanyi.

[Tew2007] *Attacks on the WEP protocol*, E. Tews.

[Tuo2003] *Overview, details and analysis of RADIUS protocol*, J. Tuomimäki.

[Woo2004] *A note on the fragility of the 'Michael' Message Integrity Code*, A. Wool.

Informes técnicos, artículos de revistas, de magacines o "white papers"

[Aki2005] *802.11i Authentication and Key Management (AKM)*, D. Akin. The CWNP Program (Cisco Systems).

[EFO2007] *Establishing wireless Robust Security Networks: a guide to IEEE 802.11i*, B. Eydt, S. Frankel, L. Owen, K. Scarfone. NIST S. P. 800-97.

[Dwo2004] *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, M. Dworkin. NIST S. P. 800-38C.

[Ins2001] *The Linux Socket Filter: Sniffing Bytes over the Network*, G. Insolvibile. Linux Journal.

[Ou2005] *Understanding the updated WPA and WPA2 standards*, G. Ou. ZDNet News & Blog.

[Wal2002] *802.11 Security Series - Part II: The Temporal Key Integrity Protocol*, J. R. Walker. Intel Corporation.

[Wal2002] *802.11 Security Series - Part III: AES-based Encapsulations of 802.11 Data*, J. R. Walker. Intel Corporation.

Páginas y sitios web

IEEE 802.11, The Working Group setting the standards for wireless LANs: http://grouper.ieee.org/groups/802/11.

Wi-Fi Alliance – Home page: http://www.wi-fi.org.

The official Linux Wireless wiki: http://wireless.kernel.org.

The Linux Wireless LAN how to, J. Tourrilhes: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Wireless.html.

802.11 – (Wi-Fi) wireless network security, R. Siles: http://www.raulsiles.com/resources/wifi.html.

The unofficial 802.11 security web page, B. Aboba: http://archive.today/ddV8.

Seguridad Wi-Fi – WEP, WPA y WPA2, G. Lehembre: http://www.hsc.fr/ressources/articles/hakin9_wifi/hakin9_wifi_ES.pdf.

Python programming languague – Official website: http://www.python.org.

Creating a GUI with JFC/SWING (the Java™ tutorials): http://download.oracle.com/javase/tutorial/uiswing.

Scapy: http://www.secdev.org/projects/scapy.

Tracking system for the software development of the project Scapy: http://bb.secdev.org/scapy/overview.