Diseño e Implementación de un Mecanismo Seguro de Configuración de Servicios de Red

Alejandro Rosúa Parra y Jordi Ortiz Murillo

Supervisor: Rafael Marín López

Murcia, 15 de Febrero de 2008

Índice general

Prólogo

Con la evolución de las tecnologías se incrementan las expectativas que la sociedad tiene de ellas. En particular, las tecnologías de comunicaciones han sufrido importantes cambios en los últimos años. Estos cambios han permitido el desarrollo de múltiples tecnologías inalámbricas, siendo la telefonía móvil una de las más popular. Pero es en la transmisión de datos a través de IP en la que se están centrando todos los esfuerzos, intentando equiparar las redes de datos a la telefonía.

Para poder equiparar las capacidades de la telefonía móvil en las conexiones de datos, es necesario que un usuario posea conexión, independientemente de su localización y su operador, e incluso si se encuentra en movimiento.

Uno de los principales problemas para ofrecer estos servicios es el proceso de conexión de los usuarios a la red. Para poder proveer de acceso a la red a un usuario, es necesario que ambos se identifiquen de manera unívoca, es decir, todo usuario debe ser autenticado. Para ello se despliegan extensas infra-estructuras que permiten realizar este proceso de forma dinámica y se desarrollan protocolos para simplificar el proceso.

Superado el primer escollo que supone la autenticación, será necesario proveer al usuario de información para configurar su acceso. Este suministro tampoco es trivial y además suele ser dependiente de la tecnología empleada. Esta dependencia se puede tornar problemática sobre todo para entornos de movilidad. A la unión del proceso de autenticación con la recuperación de esta información y suministro de material criptográfico se le denomina proceso de bootstrapping.

Se pretende introducir esta problemática con el objetivo de desarrollar más en profundidad una solución al problema del bootstrapping, independiente del medio y del servicio a iniciar. En este documento se propone una aproximación basada en la extensión del protocolo de autenticación EAP, introduciendo finalmente una implementación funcional que muestra las capacidades de bootstrapping del método. Si bien no es el objetivo final de este documento, se introducirá el problema que suponen los procesos de reautenticación y cómo la solución propuesta puede servir para mitigar sus efectos

sobre la movilidad en las conexiones de datos.

Capítulo 1

Introducción

Con el lanzamiento de los primeros satélites de comunicaciones a principios de los 60 la tecnología inalámbrica dio un gran paso hacia el futuro. Dos décadas después, la primera versión de teléfonos móviles analógicos produjo la primera aproximación popular a las comunicaciones inalámbricas. Esta primera versión denominada popularmente 1G (primera generación) apareció en los años 80, y se caracterizaba por su localidad y por la ausencia de ningún mecanismo de seguridad para proteger las conversaciones. Por localidad se entiende aquí la incapacidad de estas redes de ofrecer servicios a lo ancho y largo del planeta, un servicio que a día de hoy cualquier usuario considera básico.

Rápidamente estas primeras redes propiciaron la aparición de las siguientes revisiones. La denominada segunda generación, abanderada por su versión europea GSM aparecida en 1990, dio paso no sólo a la seguridad y digitalización de las comunicaciones móviles sino también al transporte de datos propiamente dicho en este tipo de redes. Las capacidades de transferencia de datos de esta tecnología, irrisorias en la actualidad, propiciaron la aparición de un nuevo tipo de usuario que deseaba poder realizar gestiones a través de su aparato móvil. Una revisión posterior de GSM apareció para solventar la falta de capacidad de transferencia de datos. GPRS irrumpió ampliando las capacidades de datos de GSM y optimizando el uso de la red con este fin, ofreciendo capacidades de algunos cientos de kilobits por segundo. Esta generación fue la encargada de la introducción del concepto de roaming, dotando al usuario por primera vez de una verdadera capacidad de movilidad. El término roaming o itinerancia en las comunicaciones inalámbricas define la extensión de la conectividad del servicio contratado a una localización distinta de la localización donde dicho servicio fue registrado. En otras palabras, roaming implica la obtención de un servicio en un dominio distinto al que cubre el proveedor que nos ofrece el servicio. Roaming GSM se define como la

capacidad por parte de un cliente de un celular telefónico de realizar y recibir llamadas de voz, enviar y recibir datos o acceder a otros servicios, incluidos servicios de datos locales, cuando se encuentra fuera del área de cobertura de su red local, usando una red visitada.

La llegada de la tercera generación supone una gran ruptura con respecto a la anterior en cuanto al uso de los aparatos como dispositivos para la transferencia de datos. La aparición de tecnología UMTS (*Universal Mobile Telecommunications System*) como tecnología 'universal' de acceso comienza a ofrecer unas capacidades de transmisión de datos más que notorias del orden del Mbps(Megabit per second). Esto ha propiciado una popularización de uso de las capacidades de transmisión de datos por parte de los usuarios y ha abierto un nuevo campo en el desarrollo de servicios basados en estas capacidades. En general las tecnologías inalámbricas de acceso a la red cableada han mejorado notoriamente. Esta evolución junto con la aparición de la tecnología WLAN (Wireless LAN) ofrecen a día de hoy capacidades de conexión inalámbricas de bajo coste. No es raro encontrarnos un medio compartido inalámbrico para acceder a la red dentro de un domicilio, una oficina e incluso parques y calles de las principales ciudades del mundo. Existen de hecho iniciativas para ofrecer conectividad total WLAN en ciudades completas.

En definitiva, la proliferación de diferentes tecnologías inalámbricas permite a las personas la conectividad a los distintos servicios de red provistos por las mismas, eligiendo los dispositivos que mejor se acoplan a sus necesidades en un determinado instante. De este modo, uniendo tecnologías de acceso cableado con extensión inalámbrica y telefonía móvil se logra proveer al usuario de un servicio continuo (sin interrupciones) de red. Esta capacidad queda definida con el término 'Always Best Connected' (en adelante ABC).

Un escenario de tipo ABC, donde un usuario puede seleccionar el mejor acceso a la red a través de diferentes tecnologías de acceso en cualquier momento, implica una gran complejidad. En términos de relaciones comerciales entre operadores y proveedores de servicio son necesarios acuerdos en la forma de manejar las suscripciones a través de múltiples dominios

En relación con el anterior término, ha surgido una cuarta generación con la intención de proveer al usuario de la capacidad de movimiento no sólo entre la misma tecnología (intra-tecnología) sino entre diferentes tecnologías (inter-tecnología). Esto quiere decir que un usuario podrá realizar cambios desde un punto de conexión a la red a otro (*handoff*) no sólo dentro de la red celular, sino también con cualquier otro punto de otra tecnología inalámbrica para la que se tenga soporte en el dispositivo del usuario, como por ejemplo WLAN. Para ello se han llegado a grandes acuerdos entre las compañías encargadas de proveer estos servicios a lo largo del globo. Un factor clave

Figura 1.1: Escenario básico Handover.

en este concepto es la ausencia de pérdida de datos durante este proceso de handoff inter-tecnológico y lo que es más importante, la necesidad de un proceso de acceso a la red independiente del medio que se emplee. En la figura 1.1 podemos observar un escenario genérico de handover que involucra movimientos entre diferentes tecnologías, redes y dominios administrativos. De esta forma, el proceso de handoff se puede ver como el paso de una zona de cobertura a otra, la transferencia del servicio de una estación base a otra que puede implicar además un proceso de registro en la red en cada cambio.

En particular, para realizar esta operación de registro y poder controlar y proveer servicios a sus usuarios, las operadoras despliegan costosas infraestructuras AAA(Authentication, Authorization and Accounting). Las tecnologías AAA proporcionan al operador un método para autenticar, autorizar el acceso a servicios y monitorizar a sus usuarios.

Ya en sus inicios la incipiente ARPANET vio la necesidad de controlar el acceso a las computadoras que ofrecían la posibilidad de realizar una conexión no permanente conocidas como TAC(Terminal Access Controller). En aquellos entonces el protocolo TATACS vió la luz como RFC[1] para proveer

control de acceso vía telnet. Con la aparición del protocolo PPP(Point-to Point Protocol) empleado para la mayor parte de las conexiones realizadas sobre líneas RTB(Red Telefónica Básica) apareció el método de autenticación PAP(Password Authentication Protocol) que no tardó en ser sustituido por CHAP(Challenge-handshake Authentication Protocol). No obstante, al día de hoy, ya existen otras tecnologías como IEEE 802.11i [2] o protocolos como PANA(Protocol for Carrying Authentication for Network Access) [3], que pueden ser empleadas para el proceso de autenticación entre el usuario y el equipamiento del proveedor.

1.1. La Importancia de las Infraestructuras AAA

Las arquitecturas AAA son la base del acceso seguro a la red, cada vez más codiciado tanto por parte de operadores como consumidores. Al operador le interesa proteger su red de accesos no autorizados y el usuario, por su parte, se encuentra interesado en la privacidad y fiabilidad de sus comunicaciones. Las arquitecturas AAA ofrecen tres servicios básicos:

- Autenticación → Permite identificar a un usuario de manera unívoca, con lo que se evitan situaciones de suplantación de identidad por parte de usuarios ilícitos y se asegura el acceso a los usuarios permitidos.

- Autorización → Provee de control de acceso a los recursos de la red. Para ello, primero es necesario identificar al usuario a través de un proceso de autenticación y, dependiendo del perfil que cumpla, se le autoriza el acceso para realizar la acción deseada.

- Auditoría → Permite obtener información acerca del acceso al servicio. Además de la creación de diarios de acceso, este servicio permite otros muchos procesos interesantes como puede ser la tarificación.

Además de estos tres servicios, las arquitecturas AAA permiten gestionar y asistir otros servicios de seguridad como son *confidencialidad* e *integridad*. Decimos que una comunicación es confidencial cuando nadie que la vea/escuche pueda extraer información real de ella. Por su parte, es íntegra si nadie puede modificarla entre emisor y receptor sin que estos se percaten. Autenticación, autorización, confidencialidad e integridad están directamente relacionados teniendo como punta del iceberg al proceso de autenticación.

Para autorizar un usuario previamente debe haberse autenticado y es este proceso el que desencadena la generación de material criptográfico que permitirá los mecanismos de confidencialidad e integridad.

Entre los protocolos que pueden emplearse para el diseño de arquitecturas AAA destacan RADIUS y Diameter. RADIUS(Remote Authentication Dial In User Service)[4] surge de la necesidad de centralizar el control de los usuarios en una única base de datos, en lugar de tener este proceso distribuido en los puntos de conexión a la red. Nacido para ofrecer un mecanismo de autenticación entre el punto de acceso y un servidor de autenticación, está ampliamente extendido y forma un estándar para el proceso de autenticación. Entre los problemas que presenta RADIUS destaca que no está diseñado para entornos multidominio, además de su incapacidad de soportar más de 255 solicitudes simultaneas. Diameter[5], por su parte, está destinado a ser el sucesor del longevo RADIUS. Introduce múltiples mejoras como son por ejemplo el cambio de uso de transporte no orientado a la conexión, UDP, por transporte orientado a la conexión y sin pérdidas, TCP, solucionando problemas como los ya mencionados.

Gracias al despliegue de las infraestructuras AAA construidas a través del uso de los protocolos AAA, se consigue dotar de seguridad y capacidad de trabajo multidominio a los procesos de autenticación. Estas nuevas características pueden introducir grandes lapsos en el proceso de autenticación, que se tornan un problema en entornos de movilidad. Durante el movimiento del terminal en estos entornos, se puede llegar a perder la conectividad con un proceso de autenticación lento, ya que debe realizarse un proceso completo de nuevo cada vez que el nodo móvil realiza un handoff y se une a un nuevo punto de conexión a la red, por lo que este proceso puede llegar a tardar varios segundos, pudiéndose dar interrupciones en el servicio.

1.2. Un Mecanismo Flexible de Autenticación

Resumiendo lo expuesto hasta el momento, nos encontramos con múltiples tecnologías de acceso en contacto con el usuario, un punto de acceso a la red que se encarga de facilitar la autenticación contra una autoridad de autenticación, normalmente reflejada en un servidor AAA, y con un protocolo AAA específico que debe ser implementado por el punto de acceso. Con cada nuevo mecanismo de autenticación se hace imprescindible la modificación de los elementos de la red, tanto del cliente como del operador. En general, sería deseable tener un único protocolo para el proceso de autenticación de usuarios, independiente de la tecnología de acceso o del protocolo empleado entre servidor AAA y punto de acceso. Por otra parte, sería interesante la posibili-

dad de extensión de este protocolo a lo largo de su uso. Esta problemática se une a la ya mencionada de las interrupciones de servicio debidas al proceso de re-autenticación y distribución de claves en el servicio de movilidad.

Bajo esta filosofía nace el protocolo EAP (Extensible Authentication Protocol)[6]. Este protocolo permite realizar un proceso de autenticación completo pero sin especificar de antemano el algoritmo de autenticación a utilizar. Cada uno de los posibles algoritmos de autenticación a emplear se conoce como método EAP. Estos no están predefinidos, de hecho EAP contempla la creación de nuevos métodos/mecanismos de autenticación. Normalmente EAP se encapsula directamente sobre las capas de nivel de enlace como PPP o IEEE 802, sin requerir el uso de IP, aunque a día de hoy existen ciertos protocolos que funcionan sobre el nivel IP, como PANA[3] o IKEv2[7], empleados para ofrecer control de acceso a la red y que transportan EAP.

El protocolo EAP define tres entidades implicadas en el proceso de autenticación tal como podemos ver en la figura 1.2. El cliente o *EAP Peer* que solicita la autenticación. El servidor o *EAP Server*, en el extremo opuesto de la comunicación, es el encargado de decidir tras el proceso de autenticación si el peer es autenticado positiva o negativamente. Entre ambos se encuentra el autenticador o *EAP Authenticator*, que realiza las funciones de agente de reenvío para el proceso de autenticación. Por ejemplo, en un escenario de acceso a un servicio de red inalámbrico, podemos encontrar una tecnología de acceso determinada en la red del operador, un nodo que desea conectarse al servicio de red y que realizará las funciones de peer y un punto de acceso perteneciente al operador que realizará las funciones de autenticador. En la parte concerniente al operador, encontraremos el punto de acceso conectado a la red, que realiza las funciones de autenticador, y el servidor AAA que ejerce las funciones de servidor EAP. Notar que entre autenticador y servidor AAA, pueden interactuar arquitecturas AAA pertenecientes a otros dominios administrativos.

Este escenario muestra las posibilidades de EAP creando ese nivel de abstracción para el nivel de autenticación con respecto al sistema elegido para su transporte. Además, como podremos imaginar no es común tener un único servidor AAA, incluso es posible que no haya una única tecnología de acceso al operador. También es normal encontrarnos con múltiples puntos de acceso al servicio de red que actuarán como autenticador para el proceso de autenticación.

Figura 1.2: Componentes de EAP.

1.3. Los Métodos de Autenticación

Dentro de los métodos de EAP, se puede distinguir entre aquellos que sólo realizan funciones de autenticación y aquellos que tras realizar estas funciones exportan material criptográfico. En este último grupo nos encontramos con métodos que permiten la encapsulación en su interior de otros métodos EAP con el fin de extender la funcionalidad de estos. Estos métodos son los denominados tunelados. Un caso práctico del uso de estos métodos lo podemos encontrar, por ejemplo, en escenarios en los que se requiera una autenticación de terminal previa a la autenticación del usuario.

Muchos son los protocolos que ofrecen soporte directo a la autenticación mediante EAP e incluso delegan el proceso completo a este. Las ventajas que ofrece EAP son múltiples. Además de la capacidad de abstracción que ofrece este protocolo a los procesos que lo utilizan, EAP evita la necesidad de diseñar un mecanismo específico de autenticación para el protocolo dotando su diseño de sencillez y limpieza. Por otra parte EAP posee sus métodos EAP como mecanismos de extensión, permitiendo al protocolo distintos comportamientos transparentes a las capas inferiores. Como valor añadido muchos de los métodos EAP son capaces de generar y derivar material criptográfico durante el proceso de autenticación. Con este material criptográfico es posible securizar un canal para la transmisión de datos de forma confidencial e integra en la red de acceso, entre el peer y el autenticador. El proceso de generación y distribución de claves se encuentra descrito en el *EAP Key Management Framework* (EAP-KMF)[8] y hace uso de criptografía simétrica. En concreto el esquema define la generación y distribución de una clave principal MSK(Master Session Key) la cual es generada o derivada en peer y servidor y es enviada por este último al autenticador. Adicionalmente, se genera otra clave extendida EMSK(Extended Master Session Key), de la que se puede derivar material criptográfico adicional para usos más específicos. En particular, para la comunicación entre el autenticador y el peer, de la MSK se derivan claves TSK(Transient Session Key) que son las que finalmente se emplean para securizar el medio de acceso.

Además de la capacidad de evolucionar de forma conjunta con los meca-

nismos de autenticación gracias a su extensibilidad mediante sus métodos, EAP tiene como característica nada despreciable su facilidad de integración en arquitecturas AAA. Diameter y RADIUS en particular definen mecanismos concretos para el empleo de EAP como mecanismo para la autenticación.

Por otra parte, los modelos actuales de gestión de claves y autenticación EAP no resuelven el problema del impacto sobre el mantenimiento sin cortes del servicio, producido por el proceso de handoff en el caso de redes inalámbricas móviles. Cuando un peer llega a un nuevo autenticador debe ejecutar un método EAP completo, independientemente de si ya había sido autenticado previamente y posee material criptográfico vigente. Esta ejecución completa del método provoca varios intercambios entre el peer EAP y el servidor EAP que se encuentra en el dominio home del peer. Si este dominio no está cerca del dominio actual donde el peer está recibiendo el servicio, estos intercambios conllevan una tardanza nada despreciable, con el consiguiente impacto considerable en la comunicación mantenida por el peer.

1.4. El Problema del Suministro de Datos de Autorización

Hasta este momento hemos hablado del proceso de autenticación como parte indispensable del proceso de conexión a la red. Independientemente de la información necesaria para llevar a cabo ese proceso, otros parámetros relativos a la conexión pueden ser necesarios para que el usuario pueda completar el acceso a la red y obtener de forma adecuada un determinado servicio, habilitado justo después de que el proceso de autenticación finalice. Al proceso conjunto por el cual el usuario se identifica y obtiene la información necesaria, así como material criptográfico para acceder al servicio de forma segura, se le ha denominado proceso de *bootstrapping*.

De este forma, es posible que en determinados escenarios un proceso de autenticación no sea suficiente para proveer al usuario de acceso a la red, sino más bien es un paso inicial que desencadena el proceso completo de bootstrapping. Tras esta autenticación, puede ser necesaria una distribución de claves fiable y segura que permita posteriormente la provisión de información de conexión particular al cliente del servicio.

En general, en el proceso de bootstrapping participan tres entidades:

- Cliente de Bootstrapping (*cliente*) → El cliente o usuario que desea tener acceso al servicio de red.
- Servidor de Bootstrapping (*agente*) → Está encargado de la autorización y control de acceso al servicio de red.

- Objetivo de Bootstrapping (*destino*) → En este caso es la entidad encargada de proveer el servicio.

La forma de interactuar de las tres entidades es la que sigue: El cliente desea acceder al destino, para ello comienza un proceso de autenticación contra el agente. El agente comprueba la credencial del cliente y decide sobre la petición de este. Si la decisión es positiva se suministra al cliente información sobre cómo debe acceder al destino. En este momento el proceso de bootstrapping ha sido completado y el cliente puede acceder al destino empleando el mecanismo necesario. En algunos casos el propio destino puede hacer de pasarela para llegar al agente de autenticación.

Visto desde un punto de vista más práctico y equiparándolo con el esquema que se propuso anteriormente, el cliente estaría localizado en el nodo que desea tener acceso a la red, el servidor AAA sería el encargado de autenticar y autorizar al cliente y probablemente de proveerle información de la conexión. Por su parte, la entidad que suministra el servicio (objetivo) será el punto de conexión a la red. Como se ha comentado, es posible que éste sea el encargado de contactar con el servidor para verificar el acceso al servicio que solicita el cliente.

Dado este esquema general, es posible utilizar diferentes protocolos para comunicar las entidades explicadas durante un proceso de bootstrapping. La naturaleza de estos protocolos va a depender del servicio particular al que se está intentando acceder. En este proyecto en particular, nuestro mayor interés se centra en el suministro de información relacionada con el servicio de acceso a la red. En este sentido, teniendo en cuenta que EAP es un protocolo de autenticación utilizado para controlar el acceso a ese servicio de red; que además, para determinados métodos, tiene la capacidad de generar material criptográfico en el servidor, cliente; y su independencia del protocolo o tecnología de transporte, hacen de él un serio aspirante para los procesos de bootstrapping. Sin embargo, como se ha remarcado en múltiples ocasiones, para que un proceso se pueda denominar de bootstrapping, es necesario proveer de información de autorización de forma segura al cliente. En general, el diseño inicial de EAP no consideró esta opción, centrándose únicamente en el aspecto de la autenticación. No obstante y dada su capacidad de extensión mediante métodos, se han diseñado mecanismos de bootstrapping basados en EAP para dotar al protocolo de estas capacidades de transporte de información no relacionada con la autenticación. Concretamente, estos mecanismos emplean un método EAP tunelado de tal manera que tras una autenticación exitosa inicial del método, se crea un túnel seguro que permite encapsular nuevos métodos EAP e intercambiar información de autorización relacionada con un servicio entre el peer y el servidor. De hecho, existen algu-

nas propuestas y desarrollos en esta dirección. Algunos ejemplos los podemos encontrar en [9] que propone el uso de un método EAP tunelado, sin especificar cuál, para enviar información entre el peer y el servidor, codificándola en un formato TLV(Tipo Longitud Valor). También encontramos un ejemplo interesante en el proyecto DAIDALOS [10] que hace uso del método PEAPv2 para transportar datos de autorización expresados en SAML(Security Assertion Markup Language) relacionados con un servicio.

Sin embargo, el empleo de métodos EAP tunelados actuales para el proceso de bootstrapping muestra ciertas limitaciones y problemas que deben tenerse en cuenta previamente a su implantación. Particularmente, cliente y servidor se ven obligados a un proceso concreto de autenticación antes de crear el túnel seguro que permita transportar los datos de bootstrapping. Así la mayoría obliga a una autenticación basada en TLS (con certificados de servidor) antes de la construcción del túnel (p.ej. [11],[12]). Una vez que el túnel TLS se ha formado, entonces comienza otra fase donde se autentica al cliente a través de otro método EAP ejecutado sobre el túnel, que además se utiliza para enviar información de bootstrapping posteriormente.

Se dan también casos donde el operador o el cliente requieren métodos EAP que no poseen soporte para tunelado, un claro ejemplo lo tenemos con EAP-TLS, ampliamente empleado. En estos casos, ni el cliente ni el operador pueden obtener las ventajas de realizar un bootstrapping a través de EAP. Por otra parte, si se toma la decisión de extender este método para el soporte del tunelado, ese esfuerzo será sólo aplicado al método en cuestión y cualquier otro método que se desee emplear necesitará de una implementación específica, multiplicándose el trabajo necesario por el número de métodos no tunelados que se deseen emplear. Es decir, no existe una metodología común a la hora de diseñar un mecanismo de tunelado y cada método ha diseñado su propia solución. Así pues sería deseable que esta metodología estuviese desvinculada del método transportado cosa que no es cierta en los métodos tunelados actuales.

Finalmente, se ha detectado que en la implementación (incluso en algunos caso en el propio diseño) de ciertos métodos no se exportan valores representativos para ciertos parámetros definidos en EAP[8]. Por ejemplo, en ocasiones únicamente la MSK es exportada en el servidor y peer, pero la EMSK no es exportada o tan siquiera definida. En ciertos escenarios, sin embargo, estos parámetros son de vital importancia. Uno de ellos precisamente es aquel que suministra un servicio de acceso rápido a la red[13].

1.5. Objetivos del Proyecto

Hemos visto hasta ahora que mientras que EAP es una opción muy a tener en cuenta a la hora de implementar procesos de bootstrapping, adolece de ciertas características para enviar información de autorización. En parte, esto es debido a que su planteamiento inicial de uso fue como protocolo de autenticación. Por tanto, sería interesante realizar una ampliación de sus capacidades hacia la filosofía de soporte del proceso de bootstrapping.

Para ampliar EAP existen dos alternativas, la primera es ampliar el protocolo definiendo nuevos códigos EAP para nuevos tipos de mensaje. Esta opción no es muy deseable debido a que rompemos con el estándar obligando a realizar importantes modificaciones en el equipamiento tanto de operador como cliente. La segunda opción sería el empleo de un método EAP tunelado. Esta última aproximación resulta muy interesante ya que permite total transparencia al nivel EAP y tan sólo es necesaria la incorporación a las instalaciones actuales de EAP de un nuevo método, acción que no producirá un gran impacto ya que es uno de los mecanismos de extensión que el propio estándar establece como válido.

Sin embargo, hemos destacado ya que los métodos EAP tunelados para operaciones de bootstrapping muestran ciertas desventajas. Para ello, este proyecto implementa una solución basada en un nuevo método EAP denominado **EAP-EXT**, que soluciona los problemas antes expuestos. De manera muy esquemática EAP-EXT es un método tunelado capaz de transportar cualquier método de autenticación ya implantado, separando por tanto el proceso de autenticación del proceso del establecimiento del túnel. Es más, aprovecha el material criptográfico generado por los métodos ejecutados durante la fase de autenticación para securizar el intercambio posterior de datos de bootstrapping. Además el mismo material criptográfico es empleado para derivar y exportar las claves indicadas por el EAP KMF.

De forma más concreta, este trabajo describe e implementa un método tunelado que:

- Permite la secuenciación de varios métodos EAP en su interior, siendo capaz de emplear cualquier método interno independientemente de si este cumple con las especificaciones del EAP KMF.

- Se ejecuta en dos fases. Una primera fase de autenticación durante la cual se ejecutan los distintos métodos internos especificados y una segunda fase de autorización en la cual de forma segura se permite el intercambio de información de esta índole entre Servidor EAP y Peer.

- Funciones de bootstrapping para cualquier servicio de red, en particular

el servicio de acceso a la red, debido a su capacidad de transporte de información.

- Soporta la negociación de capacidades de forma segura, permitiendo flexibilidad en los mecanismos de negociación de claves y algoritmos de cifrado.
- Implementación del EAP KMF, dando este soporte extra a los métodos que se transporten.
- Un mecanismo de re-autenticación rápida aprovechando el material criptográfico generado durante una autenticación EAP inicial.

En lo que resta de este documento se dará una visión extendida de lo que supone el problema del bootstrapping. En el siguiente capítulo, realizaremos una descripción de las tecnologías relacionadas con este proceso ya introducidas en este capítulo. En el capítulo 3, pasaremos a mostrar el diseño de la solución propuesta al problema del bootstrapping EAP-EXT. El capítulo 4, introducirá los conceptos empleados para llevar a cabo la implementación realizada, además se hará un pequeño análisis de los resultados obtenidos en las pruebas. Para finalizar, se ofrecerá al lector un capítulo de conclusiones y vías futuras para la extensión de este documento o la implementación asociada.

Capítulo 2

Análisis de Tecnologías

En el capítulo anterior se ha introducido al lector en el problema que supone el bootstrapping en entornos móviles. Con este propósito se introdujeron conceptos como arquitectura AAA o EAP entre otros, que necesitan ser expandidos para llegar a entender la solución que en este documento se propone.

En este capítulo realizaremos una introducción avanzada a las arquitecturas AAA e introduciremos el modelo EAP para autenticación, que servirá de base a nuestra propuesta. Para finalizar se describirán algunos protocolos capaces de emplear EAP, con el fin de introducir al usuario a todos los posibles elementos de un sistema de autenticación, sistema que nos servirá para dar solución al problema del bootstrapping.

2.1. Tecnología AAA (Autenticación, Autorización y Auditoría)

Las tecnologías AAA tienen como principal función dar soporte para un acceso seguro a los servicios de red. Las infraestructuras AAA proporcionan servicios de *Autenticación, Autorización y Auditoría*. Mediante estos procesos de autenticación, autorización y auditoría se controla y gestiona el acceso de los usuarios a los recursos ofrecidos, proporcionando un nivel de seguridad adecuado. Vamos a pasar a explicar estos tres conceptos con el fin de tener una mejor visión global de las tecnologías AAA:

- *Autenticación.* En la seguridad de ordenadores, la autenticación es el proceso de intento de verificar la identidad digital del remitente de una comunicación, como una petición para conectarse. En un web de confianza, 'autenticación' es un modo de asegurar que los usuarios son

quién ellos dicen ser, que el usuario que intenta realizar funciones en un sistema es de hecho el usuario que tiene la autorización para realizar dichas funciones. Los métodos de autenticación se suelen dividir en tres grandes categorías[14], en función de lo que utilizan para la verificación de identidad:

a) algo que el usuario sabe

b) algo que éste posee

c) una característica física del usuario o un acto involuntario del mismo. A esto se le conoce como autenticación biométrica.

Es fácil ver ejemplos de cada uno de estos tipos de autenticación: un password (Unix) o passphrase (PGP) es algo que el usuario conoce y el resto de personas no, una tarjeta de identidad es algo que el usuario lleva consigo, la huella dactilar es una característica física del usuario, y un acto involuntario podría considerarse que se produce al firmar (al rubricar la firma no se piensa en el diseño de cada trazo individualmente). Por supuesto, un sistema de autenticación puede (y debe, para incrementar su fiabilidad) combinar mecanismos de diferente tipo, como en el caso de una tarjeta de crédito junto al PIN a la hora de utilizar un cajero automático.

- *Autorización.* Definición granular de permisos de acceso concedidos a un determinado usuario, dispositivo o sistema sobre uno o varios servicios. Es el proceso de verificación de que un usuario puede o no acceder a un servicio o recurso de la red. La autorización es el proceso de verificación por el cual una persona conocida tiene la autoridad para realizar una cierta operación. La autenticación, por lo tanto, debe preceder la autorización. La autorización determina la naturaleza del servicio que se concede a un usuario.

- *Auditoría.* Registro de la actividad de los usuarios: tiempo de conexión, servicios utilizados, datos accedidos, etc. Los datos recogidos se utilizan para obtener estadísticas, planificar ampliaciones de capacidad, facturación de servicios, auditoría y asignación de costes, entre otros fines. La información típica que se recopila en el proceso de auditoría suele ser la identidad del usuario, la naturaleza del servicio prestado, cuando se inició el servicio, y cuando terminó.

Las tecnologías AAA comprenden un conjunto de herramientas, procedimientos y protocolos que garantizan un tratamiento coherente de las tareas

de autenticación, autorización y auditoría o registro de actividad de las entidades que tienen acceso a un sistema de información. Para el desarrollo de tareas que define AAA se definen una serie de protocolos y servicios así como un esquema general. En los siguientes puntos se tratarán estos temas con más detalle.

2.1.1. Arquitectura general AAA

El objetivo de la Arquitectura general AAA es controlar y proporcionar autenticación, autorización y auditoria a sistemas y entornos sujetos a políticas establecidas por los administradores y usuarios del sistema[15]. Los siguientes términos especifican alguno de los aspectos más representativos de la arquitectura general AAA:

Dominio AAA. Un único dominio administrativo que contiene una colección de servicios ofrecidos por una misma entidad organizativa. Cada servicio AAA pertenece a un dominio AAA pero un dominio AAA puede ofrecer múltiples servicios.

Servidor AAA / Servicio AAA. Un servicio AAA es aquello que lleva a cabo una o más de una de las funciones de autenticación, autorización y auditoría. Un servidor AAA como puede ser un servidor DIAMETER es un ejemplo de servicio AAA. La definición de un servicio AAA es más amplia que la de un servidor AAA, por ejemplo, un servicio AAA puede incluir políticas.

Application Specific Module (ASM). Es un módulo que gestiona los recursos y permite políticas para el equipo de servicio. Este componente ayuda al servidor AAA a gestionar las solicitudes recibidas.

Política. Las políticas son un conjunto de reglas para administrar, gestionar y controlar el acceso a recursos de red. En la arquitectura AAA se evalúan políticas de estado y se aplican políticas de acción en función de la condición evaluada.

Request o solicitud. Es cualquier tipo de mensaje que solicita o requiere un servicio.

Usuario. Es el sujeto que realiza la solicitud para acceder a un recurso o servicio ofrecido por el operador de red.

Equipo de Servicio. Dispositivo que pertenece al proveedor de servicio y que suministra dicho servicio. Es un término utilizado para el equipo (aunque no limitado al mismo) como pueden ser smart switches, routers, firewalls, bandwidth brokers, equipos de acceso a la red, etc.

Proveedor de Servicio. Autoriza el acceso al servicio. Si pertenece a un dominio visitado debe tener acuerdos con el dominio home del que proviene el usuario.

Modelo Servidor AAA Genérico

Una vez que hemos explicado algunos de los términos más relevantes descriptivos de la tecnología AAA pasamos a explicar un modelo de servidor AAA genérico[16].

Hemos visto que un servidor AAA es capaz de realizar las funciones de autenticación de usuarios, manejo de las solicitudes de autorización y recogida de los datos de auditoría. Para realizar estas funciones, un servidor AAA interactúa con diversos módulos de los que está compuesto el propio servidor. En un primer paso en el funcionamiento de un servidor AAA se procede a la autenticación del usuario. Tras la llegada de una solicitud al servidor y tras haber autenticado al usuario autor de esa solicitud, se pasa a la autorización de la misma. Un servidor AAA posee reglas para inspeccionar una solicitud y llegar a una decision de autorización. Como hemos visto antes el ASM puede ayudar a tomar la decisión de autorización.

Para fines de auditoría, el servidor AAA debe tener algun tipo de base de datos para almacenar la información acerca de las solicitudes así como una cronología de las mismas y de los acontecimientos que ocurren en el servidor. Esta base de datos puede utilizarse para las autorizaciones pero también puede usarse para crear normas, por ejemplo, permitir una solicitud si antes fue registrado algún otro evento concreto.

Es necesario otra base de datos que contenga los recursos y servicios sobre los cuales se toman las decisiones de autorización. Esta base de datos podría requerir ser accedida por otros servidores AAA para construir peticiones de autorización más complejas.

También hay que disponer de un mecanismo de envío de mensajes entre los servidores AAA, junto con un protocolo que permita la comunicación entre diferentes servidores, debido, especialmente, a los escenarios multidominio, donde la organización que suministra el servicio está en diferente dominio al servidor que autentica y autoriza.

Con la ayuda de estos módulos el servidor AAA será capaz de tramitar las solicitudes AAA. El servidor inspeccionará el contenido de la solicitud,

determinará el tipo de la solicitud, recuperará las reglas de políticas del repositorio, realizará diversas funciones y finalmente escogerá una de las siguientes opciones para terminar el proceso de cada componente de la solicitud:

a) deja que el componente sea evaluado por un ASM.

b) consulta las políticas y el registro de eventos para autorizar o no la solicitud.

c) remite la solicitud a otro servidor AAA para que este la evalúe.

Resumiendo, un servidor se encarga de recibir la solicitud y determinar si se trata de una autenticación o una autorización. Tras esto selecciona una política del repositorio y realiza alguna acción según la política seleccionada enviado esta a un modulo ASM específico para procesarla. Si es necesario reenvía la solicitud a otro servidor AAA para que la trate.

El reenvío de la solicitud a otro servidor AAA para que sea este el que se encargue de procesarla se da sobretodo en escenarios multi-dominio. Como hicimos mención anteriormente, un escenario **multi-dominio** es aquel en el que el dominio que se encarga de la autenticación y de la autorización del usuario no es el mismo que el dominio que suministra el servicio. Por tanto hay más de un dominio involucrado a la hora de resolver la solicitud del usuario. Debe existir un acuerdo o contrato entre ambos dominios para proveer al usuario del servicio solicitado. Este acuerdo podría ser un servicio de roaming o servicios distribuidos.

El escenario multi-dominio es el más común. Por otra parte tenemos el caso de un único dominio donde el dominio home actúa como proveedor de servicio. El servidor AAA home y el proveedor de servicio pueden verse como una única entidad. Usaremos este caso más simple para explicar los modelos de autorización que veremos en la sección siguiente.

2.1.2. Modelos de autorización

Existen tres modelos distintos que describen tres secuencias de acceso para que el usuario obtenga un servicio ofrecido por la infraestructura AAA[17]:

Modelo *agent*. En este modelo el servidor AAA proveedor de servicio funciona como un agente entre el usuario y el propio servicio. El servidor AAA recibe la solicitud del usuario y, tras autenticarlo y autorizarlo, reenvía la solicitud junto con información derivada de la aplicación de políticas de usuario y del propio servicio al equipo de servicio. El equipo de servicio lleva a cabo la petición del usuario y responde al

Figura 2.1: Escenario Multidominio

servidor AAA informándole de que la solicitud fue atendida. Finalmente el servidor AAA informa al usuario de si el servicio se llevó a cabo satisfactoriamente o no. Podemos ver este modelo representado en la figura 2.2.a)

Modelo *pull.* Este modelo se suele usar en las propuestas de IP móvil y en algunas propuestas de QoS. El usuario envía su solicitud directamente al equipo de servicio. Este a su vez actuando como cliente la reenvía al servidor AAA. El servidor AAA la evalúa y retorna una respuesta apropiada al equipo de servicio. Si la respuesta es positiva el equipo de servicio suministra el servicio solicitado al usuario. Un esquema de este modelo está representado en la figura 2.2.b)

Modelo *push.* En este modelo el usuario debe conseguir una credencial temporal del servidor AAA que indique que el usuario puede acceder al servicio. Para ello hace la solicitud al servidor AAA que le responde con el correspondiente ticket. El usuario se dirige luego al equipo de servicio con la solicitud del servicio y el ticket conseguido anteriormente. El equipo de servicio usa el ticket para verificar que la solicitud esta aprobada y suministra el servicio. El ticket solicitado al servidor

AAA tiene una limitación de tiempo. Puede contener además alguna información sobre la localización del servicio y podría ser usado en más de una solicitud. El protocolo de autenticación Kerberos es un claro ejemplo del modelo push. Podemos ver este modelo en la figura 2.2.c)

Figura 2.2: Modelos de autorización.

Los tres modelos son posibles en escenarios multi-dominio con acuerdos de roaming.

2.1.3. Protocolos AAA

Necesitamos los protocolos AAA para la comunicación entre las distintas entidades que conforman la infraestructura AAA. Como podemos ver en [18] estos protocolos deben cumplir una serie de requisitos, los más significativos son:

- Requisitos de transporte.
  - El diseño del protocolo AAA debe ser independiente del transporte.
  - Debe permitir escalabilidad, tiene que permitir soportar millones de usuarios y millones de solicitudes simultáneas.
  - Soporte para múltiples servidores AAA y recuperación ante fallos. Para soportar altas cargas es necesario balanceadores de carga entre multiples servidores AAA y recuperación de fallos.

    - Soporte multi-dominio.

- Requisitos de representación. Se recomienda el uso de una representación Atributo-Valor para los datos AAA, lo que se denomina un AVP (Attribute-Value Pair), soportando al menos los tipos simples de atributos (entero, real, string...) y dando soporte para estructuras de atributos mas complejas. Además debe ser extensible para soportar nuevos atributos necesarios para nuevos servicios.

- Requisitos de seguridad.

    - El protocolo debe soportar autenticación mutua entre el cliente y el servidor AAA.
    - Como mínimo el protocolo AAA debe soportar el uso de un secreto compartido entre cada pareja usuario servidor AAA para verificar mutuamente sus identidades.
    - El protocolo debe también soportar verificación mutua de la identidad utilizando una infraestructura de clave pública que soporte expiración y revocación de claves.
    - Soporte para encriptación de atributos.

Entre los protocolos AAA destacan RADIUS, Diameter, TACACS y TACACS+. Realmente el único que cumple todos los requisitos anteriores es Diameter. De los mencionados, TACACS[1] es el protocolo que menos de estos requisitos cumple. Tiene demasiadas carencias y además es un protocolo propietario. Gran parte de los entornos AAA estan implementados utilizando RADIUS[4] a pesar de sus problemas de extensibilidad y seguridad entre otros. Diameter es el protocolo que más nos interesa pues se trata de una evolución de RADIUS que resuelve los problemas asociados a ese protocolo. Debido a la importancia de estos dos últimos protocolos, nos centraremos en explicarlos a continuación.

RADIUS

El protocolo RADIUS[4] (acrónimo en inglés de Remote Authentication Dial-In User Server), como su nombre indica, fue propuesto inicialmente para gestionar los accesos remotos Dial-In (llamada telefónica). Este protocolo de autorización y autenticación actualmente es usado para aplicaciones de acceso a la red y movilidad IP.

RADIUS trabaja sobre UDP, concretamente utiliza el puerto 1813 para sus conexiones. El trabajar con UDP conlleva bastantes desventajas. La principal de esas desventajas es el no tener un control de pérdidas de paquetes,

con lo que un cliente que realiza una solicitud no tiene certeza de que esta haya llegado hasta el servidor RADIUS, o de si su solicitud fue descartada por el servidor.

RADIUS opera con un modelo cliente servidor, siendo el NAS (servidor de acceso a la red) el cliente y como servidor tenemos un servidor RADIUS. Cuando se realiza la conexión el NAS envía la solicitud al servidor. Esta solicitud lleva información para primero autenticar y segundo autorizar (siempre que la autenticación haya sido exitosa) de manera correcta al cliente. Estos procesos de autenticación y autorización siempre van ligados y por ese orden en un servidor RADIUS. El servidor RADIUS comprueba que la solicitud es correcta utilizando esquemas de autenticación como PAP, CHAP o EAP. Si es aceptado, el servidor autorizará el acceso al sistema del ISP y le asigna los recursos de red como una dirección IP, etc.

Una de las principales características del protocolo RADIUS y razones de su extensión es la facilidad a la hora de facturar al usuario por parte de las operadoras. Esto es factible debido a que RADIUS tiene capacidad para el manejo de sesiones, por lo que sabe cuando comienza y acaba una sesión. Estos datos son útiles para creación de estadísticas, facturación, etc.

Como mencionamos antes, RADIUS no cumple todos los requisitos de un buen protocolo AAA. Entre sus limitaciones se encuentra el no poder atender millones de solicitudes simultaneas, tiene una limitación de 255 solicitudes pendientes a la vez. Otra limitación se da a la hora de soportar nuevos servicios. Para el soporte de nuevos servicios muchas veces es necesario modificar el protocolo. Por otra parte RADIUS es extensible pudiéndose implementar nuevos dialectos para adaptarlo a unas necesidades concretas. Siguiendo con el repaso a los requisitos que debe cumplir un protocolo AAA, RADIUS tiene una limitación en la longitud de los valores de los atributos. El limite de esta longitud en RADIUS esta fijada a 255 bytes, por lo que se puede quedar corto en algunos casos. Por último, y para finalizar este tema, cabe mencionar que un servidor RADIUS puede funcionar como proxy. Los servidores Proxy RADIUS se utilizan para una administración centralizada y pueden reescribir paquetes RADIUS al vuelo. Que se puedan reescribir los paquetes tiene la ventaja de poder hacer conversiones entre distintas modificaciones del protocolo para su comunicación, pero puede ser un grave problema de seguridad.

Diameter

Debido a estos problemas del protocolo RADIUS surgió una evolución llamada Diameter[5]. El protocolo está destinado a proporcionar un marco de autenticación, autorización y auditoría para aplicaciones de acceso a la red o movilidad IP. Diameter proporciona un método de actualización del

protocolo flexible, al contrario de lo que ocurre con RADIUS.

Así pues, el principal objetivo del protocolo Diameter es el de proporcionar un protocolo base extensible que procure servicios de autenticación, autorización y auditoría a nuevas tecnologías de acceso. Se considera extensible ya que sin necesidad de modificar directamente el protocolo base se le pueden añadir extensiones que implementen nuevas características. A estas extensiones se les llama aplicaciones Diameter. Una aplicación Diameter puede extender el protocolo base diameter mediante la creación o modificación de atributos, usando nuevas aplicaciones de autenticación, autorización o auditoría, etc.

El protocolo base Diameter proporciona las siguientes facilidades:

- Entrega de AVPs
- Negociación de capacidades
- Notificación de errores
- extensibilidad y adicción de nuevos AVPs.
- Servicios básicos necesarios para las aplicaciones tales como el manejo de sesiones de usuario.

El protocolo Diameter maneja los datos mediante pares atributo-valor llamados AVP. Diameter dispone de AVPs propios para el protocolo base además de AVPs que pueden ser definidos por las aplicaciones Diameter. Los AVPs son usados por el protocolo base para transportar información de autenticación y autorización del usuario, intercambiar información de los recursos y para soportar características de redirección y proxy de mensajes Diameter en una jerarquía de servidores.

Diameter es un protocolo peer-to-peer. Cualquier nodo puede iniciar una solicitud. En este sentido un cliente diameter es un disposivo al borde de la red que realiza el control de acceso, como por ejemplo un NAS (network access server). El cliente Diameter genera las solicitudes de autenticación, autorización y auditoría para el usuario. Un agente Diameter es un nodo que no autentica o autoriza mensajes a nivel local, sino que se valen de proxys y redirecciones para tal efecto. Un nodo Diameter actúa como agente para algunas solicitudes y como servidor para otras.

Para finalizar vamos a ver un resumen de las características generales del protocolo Diameter:

- Diameter soporta tolerancia a fallos, ya que cada mensaje de solicitud debe ser contestado con un mensaje de respuesta. A su vez tiene notificación de errores

- Diameter utiliza TCP o SCTP como protocolo de nivel de transporte. A diferencia de RADIUS, Diameter utiliza un protocolo de transporte fiable.
- Diameter está diseñado con un modelo peer-to-peer que permite que tanto servidores como clientes puedan iniciar el envío de mensajes.
- Diameter tiene descubrimiento dinámico de peers (usando DNS SRV y NAPTR). A su vez también puede descubrir dinámicamente rutas.
- Diameter tiene soporte Multi-dominio, a diferencia de su antecesor RADIUS.
- El protocolo Diameter es escalable y extensible.
- Diameter está orientado a sesiones, es decir, las transacciones se organizan por sesiones de usuarios.
- El protocolo Diameter usa seguridad a nivel de transporte ya que obliga a soportar IPSec y permite TLS.
- Diameter tiene compatibilidad transicional con RADIUS. Se puede trabajar con ambos protocolos en la misma red mediante el uso de una aplicación que permite el soporte de RADIUS en redes que trabajan con Diameter, facilitando la transición de un protocolo a otro.
- El protocolo tiene un espacio de direcciones mayor para AVPs e identificadores (32 bits en lugar de los 8 de RADIUS).
- Diameter admite ACKs en el nivel de aplicación, definiendo métodos de fallo y máquinas de estado

2.2. El protocolo EAP

EAP [6] (Extensible Authentication Protocol) es un protocolo que permite la ejecución de distintos mecanismos de autenticación de manera extensible. A estos mecanismos se les denomina métodos EAP. EAP de forma habitual corre directamente sobre la capa de enlace, con protocolos como PPP o redes IEEE 802.11, sin la necesidad de IP. Una de las ventajas de la arquitectura EAP es su flexibilidad. EAP se usa para seleccionar un mecanismo específico de autenticación. En vez de actualizar continuamente el autenticador con cada nuevo método de autenticación EAP permite un servidor de autenticación (p.ej. un servidor AAA) que implementa métodos de autenticación, al que

acude el autenticador para realizar el proceso. Los métodos EAP se ejecutan entre un peer EAP y un servidor EAP, a través de un autenticador EAP que simplemente se encarga del reenvío de paquetes.

EAP fue diseñado para la autenticación de acceso a la red, donde la capa de conectividad IP puede no estar disponible. EAP provee solo del soporte necesario para realizar el transporte de protocolos de autenticación de forma fiable. Una de las consecuencias de esto es que solo admite tener un paquete en circulación, no maneja múltiples paquetes simultáneamente. Además el protocolo tiene soporte para la eliminación de duplicados y para retransmisión, aunque depende de las capas inferiores para tener garantías de orden. La fragmentación no está soportada por el protocolo, sin embargo los métodos EAP individuales si pueden soportarla.

Otra característica a tener en cuenta es que la autenticación EAP la inicia el autenticador. Como la mayoría de protocolos son iniciados por el cliente es necesario un método para estos protocolos que añada a EAP al menos un intercambio [1].

Como hemos visto, la principal función de EAP es la de proveer mecanismos de autenticación. Este protocolo es interesante ya que se puede integrar en infraestructuras AAA para llevar a cabo los procesos de autenticación que se dan en esta. A continuación vamos a estudiarlo con más detalle.

2.2.1. Modelo EAP

Conceptualmente, una implementación EAP contendrá los siguientes componentes:

- EAP Lower Layer. Esta capa se corresponde con el nivel de transporte y se encarga del envio y recepción de mensajes entre el peer y el autenticador. Algunos lower layer posibles pueden ser PPP, las IEEE 802, IKEv2, etc.

- Nivel EAP o EAP layer. EAP layer recibe y transmite paquetes EAP a través del lower layer. Implementa la detección de paquetes duplicados y la retransmisión. También envía y recibe mensajes desde el nivel de peer y autenticador.

- Nivel EAP peer y autenticador. Según el código del paquete EAP, el EAP layer demultiplexa los paquetes que llegan al nivel nivel EAP peer y autenticador. Una implementación EAP en un host puede soportar una funcionalidad de peer, de auntenticador o ambas. En el caso de que presente ambas estará presente este nivel.

[1] un mensaje de ida y otro de vuelta

- EAP method layers. Como vimos antes, los métodos de EAP implementan algoritmos de autenticación concretos y reciben y transmiten mensajes a través de los niveles peer y autenticador. Estos métodos son responsables de la posible fragmentación.

Figura 2.3: Pila de componentes EAP.

2.2.2. Configuraciones en EAP

En el protocolo EAP hay tres entidades básicas que son el peer, el autenticador y el servidor EAP. El peer se encuentra dentro del dispositivo móvil y el autenticador se encontrará en el NAS. Para el servidor tenemos dos opciones que dan lugar a las dos configuraciones siguientes:

Configuración standalone. El servidor se encuentra junto al autenticador en el NAS.

Configuración pass-through El servidor EAP se encuentra en un servidor AAA como módulo ASM de este. Para la autenticación del peer el autenticador, situado en el NAS, reenvía los paquetes EAP al servidor AAA.

La configuración pass-through es la más usada ya que ofrece un sistema de acceso y autenticación a la red flexible y escalable. Esto es debido a que se vale de la infraestructura AAA para administrar a sus usuarios y autenticadores. Una configuración pass-through permite centralizar las operaciones de autenticación en una entidad de alto nivel como un servidor AAA. Con una configuración standalone con varios autenticadores y usuarios, cada autenticador tiene que tener las claves y credenciales de cada usuario con los consiguientes problemas de escalabilidad que surgen al aumentar el número de usuarios y autenticadores.

Con una configuración pass-trough, y teniendo en cuenta las características de EAP, el modelo de intercambio de mensajes sería el que se describe a continuación. Tenemos la fase inicial de descubrimiento. Como dijimos antes, la autenticación la inicializaba el autenticador. En esta fase el autenticador pide al peer que se identifique. Normalmente el autenticador no posee la información acerca de contra quién autenticar al peer, por lo que inicia un intercambio de mensajes con el peer solicitando esa información. Si dispone de esos datos la fase del intercambio de la identidad no es necesaria, es por ello que esta fase es opcional. En la siguiente fase se realiza la autenticación del peer. Esta fase se realiza entre el peer y el servidor, estando el autenticador de intermediario tal como vimos en la configuración pass-trough. Entre el peer y el autenticador se usa un protocolo EAP lower layer para transportar los paquetes EAP. Por otra parte, entre el autenticador y el servidor EAP se usara un protocolo AAA como puede ser RADIUS o Diameter. En una siguiente fase se transportará material criptográfico entre el servidor y el autenticador. Este material criptográfico sirve para establecer una asociación de seguridad entre el peer y el autenticador EAP. Dicha asociación de seguridad se crea en una siguiente fase entre peer y autenticador mediante la ejecución de un protocolo de asociación de seguridad para proteger el tráfico entre ambos.

2.2.3. Claves en EAP

Además de proporcionar autenticación, ciertos métodos EAP también son capaces de exportar material criptográfico en forma de claves. Estas claves serán las mismas tanto en el peer EAP como en el servidor EAP si la ejecución del método de autentación finalizó correctamente. Las claves se utilizarán para establecer una asociación de seguridad entre el peer y el autenticador EAP. Después de la autenticación EAP se genera el material criptográfico compuesto de cuatro claves genéricas. Nos disponemos a describir estas claves según podemos ver en [8], ya que nos serán utiles a lo largo de este documento:

Figura 2.4: Intercambios en EAP.

Master Session Key (MSK) Se emplea como clave principal para establecer una asociación de seguridad entre el peer y el autenticador. Tras la autenticación EAP es generada tanto por el peer como por el servidor EAP. Este último se encarga de enviarsela al autenticador EAP.

Extended Master Session Key (EMSK) Es el material criptográfico adicional. Se comparte entre el peer y el servidor que pueden usarla para derivar nuevas claves. Esta clave no se debe proveer a ninguna otra entidad externa.

Transient EAP Keys (TEKs) Son claves de sesión que se utilizan para proteger el intercambio de mensajes EAP. Estas claves son internas y no se exportan a ninguna entidad. Se descartan cuando acaba el proceso de autenticación.

Transient Session Keys (TSKs) Se crean tras ejecutar el protocolo de asociación de seguridad entre el peer y el autenticador usando la MSK, y se usan para proteger el tráfico de datos entre ambos.

EAP Key Management Framework

Además de exportar material criptográfico, un método EAP puede también exportar parámetros asociados al proceso de autenticación como el peer autenticado, el server autenticado o un identificador único de sesión de autenticación EAP. También puede importar y exportar parámetros de la capa de transporte (EAP lower layer) conocidos como 'channel bindings'. Vamos a introducir la jerarquía de claves de EAP y cómo estas son usadas y transportadas.

En entornos en los que esté contemplada la derivación de claves el proceso de autenticación tiene lugar normalmente en tres fases:

- Discovery
- Autenticación
  - Autenticación EAP
  - Transporte de claves sobre AAA (opcional)
- Protocolo de Asociación Seguro
  - Asociación segura unicast
  - Asociación segura multicast (opcional)

De estas fases, la de discovery, el transporte de las claves sobre AAA y el protocolo de asociación seguro completo, son manejadas fuera de EAP. En particular, la fase de discovery y el protocolo de asociación seguro completo son manejadas por el nivel de transporte, de el transporte de la clave como ya se ha indicado se encarga la arquitectura AAA subyacente.

En la fase de discovery los peers encuentran a los autenticadores y descubren sus capacidades. Un peer puede encontrar un autenticador con el cual desea realizar una asociación de seguridad, proveyendo acceso a una red o tras un bridge. Esta fase se puede iniciar de manera automática o manual, dependiendo de la capa de nivel de transporte sobre la que EAP se esté ejecutando.

La fase de autenticación puede comenzar conforme el peer y el autenticador se hayan descubierto mutuamente. Esta fase, si ocurre, siempre incluye una autenticación EAP, donde el método EAP elegido puede soportar derivación de claves. En la fase de autenticación EAP, ambos, peer y server pueden derivar material criptográfico EAP.

En escenarios que incluyan un servidor de autenticación como entidad externa al autenticador, es necesario un paso adicional para transportar la

clave desde este al autenticador. Para no incumplir el principio de 'independencia del modo' de EAP, cuando un servidor de autenticación externo está presente, todo el material criptográfico necesario para el nivel de transporte será transportado desde el servidor EAP al autenticador. Puesto que en las implementaciones existentes la derivación de la TSK y las técnicas de transporte dependen únicamente de la MSK, esta es la única clave replicada en la fase de transporte de la clave sobre AAA.

Con el concepto *'independencia del modo'* reintroducimos los modos en que puede trabajar un autenticador. EAP normalmente es empleado en situaciones en que los peer desean tener acceso a la red vía varios autenticadores. Como comentamos anteriormente, estos autenticadores pueden trabajar en modo *standalone* o en modo *passthrough*.

Es fundamental para EAP que en la capa de métodos EAP, la conversación entre el peer y el servidor EAP no se vea afectada tanto si el autenticador trabaja en un modo u otro. Bajo el punto de vista de EAP, la función principal del protocolo AAA es mantener el principio de 'independencia del modo', permitiendo al peer abstraerse sobre si el autenticador trabaja en modo standalone o passthrough.

La consecución de una autenticación EAP y la derivación de material criptográfico por un peer y un servidor EAP no implica que el peer se consigne a la red asociada con el servidor EAP. Por el contrario, esta consigna se realiza como parte del *'protocolo de asociación seguro'*. El protocolo de asociación seguro es ejecutado entre el peer y el autenticador con el fin de crear una asociación de seguridad unicast y multicast entre el peer y el autenticador.

El método de derivación de claves de EAP tiene como raíz una credencial de larga duración utilizada por el método EAP seleccionado. Si la autenticación está basada en una clave pre-compartida , las partes almacenarán esta clave y el método a emplear. El servidor EAP almacenará además el Peer-Id así como información adicional. Esta información normalmente se emplea fuera de EAP para determinar si se debe garantizar el acceso al servicio (autorización). El peer por su parte, almacena la información necesaria para elegir qué secreto emplear para qué servicio.

Si la autenticación está basada en probar la posesión de una clave privada correspondiente a la clave pública de un certificado, las partes almacenarán el método EAP y las entidades de confianza empleadas para validar los certificados. El servidor EAP también almacena el Peer-Id, mientras que el peer almacena qué certificado emplear para qué servicio. Basándose en la credencial de larga duración los métodos derivan dos tipos de material criptográfico EAP:

1. Material criptográfico calculado localmente por el método y no expor-

tado, como las Transient EAP Keys(TEKs).

2. Material criptográfico exportado por los métodos EAP: Master Session Key (MSK), Extended Master Session Key(EMSK) y el vector de inicialización (IV).

Tal y como se indica en EAP[6]: *'Para proveer material criptográfico que pueda ser usado en una suite de cifrado especificada con posterioridad, un método EAP que soporte la derivación de claves tiene que exportar una Master Session Key (MSK) y una Extended Master Session Key(EMSK) de 64 bytes como mínimo cada una.'*

Los métodos EAP también pueden exportar el IV, si bien su uso está desfasado. Ya hemos dicho que la EMSK no puede ser provista a una entidad externa al peer o server EAP. A su vez, tampoco se puede proveer ninguna cantidad de esta EMSK a una entidad externa al peer o al server EAP mediante la cual la EMSK pueda ser calculada sin que suponga romper ninguna de las reglas fundamentales de la criptografía, como por ejemplo invertir una función de resumen.

Los métodos EAP que soportan derivación de claves y autenticación mutua deberían exportar un identificador de sesión EAP específico del método conocido como **Session-Id**, así como uno o más identificadores para el peer específicos del método **Peer-Id** y también para el servidor **Server-Id**. Los métodos EAP podrían importar/exportar parámetros de channel binding. Los Peer-Id y Server-Id identifican las identidades involucradas en generar el material criptográfico.

Peer-Id Si un método EAP que genere claves, autentica una o más identidades de peer específicas del método, esas identidades son exportadas por el método como Peer-Id. Es posible exportar más de un Peer-Id por un método EAP. No todos los métodos EAP proveen identidad de peer específica del método, en estos métodos se define como una cadena vacía. En los métodos EAP que deriven claves y no provean Peer-Id, el servidor EAP no autenticará la identidad del peer EAP con el cual se derivó el material criptográfico.

Server-Id Si un método EAP que genere claves autentica una o más identidades de servidor específicas del método, estas son exportadas por el método como Server-Id. Es posible exportar más de un Server-Id por un método EAP. No todos los métodos EAP proveen un identificador de servidor específico del método. En los métodos en que no esté definido, el Server-Id será una cadena vacía. Si el método EAP no genera material criptográfico, el Server-Id tiene que ser una cadena vacía. Cuando

un método EAP que derive claves no exporte Server-Id, el peer EAP no autenticará la identidad de el servidor EAP con el cual se derivó el material criptográfico.

Session-Id El Session-Id unicamente identifica una sesión EAP entre un peer (identificado por un Peer-Id) y un servidor (identificado por un Server-Id) EAP. En los casos en que no se empleen códigos de tipo EAP extendidos (es decir, su código EAP sea distinto de 254), el Session-Id es la concatenación de un byte correspondiente al código EAP y un identificador temporal único obtenido del método(conocido como Method-Id). Donde se empleen códigos EAP extendidos, el Session-Id consiste en los códigos EAP extendidos (incluyendo los campos Tipo, Vendor-Id y Vendor-Type definidos en el propio EAP[6]) concatenados con un identificador único temporal obtenido del método (Method-Id). Este Method-Id es construido por lo general con los contadores usados en los intercambios del método EAP. La inclusión de los códigos de tipo o de los códigos de tipo extendidos en el Session-Id aseguran que cada método EAP posee un espacio de Session-Id distinto. Puesto que una sesión EAP no está relacionada con un único autenticador o a puertos específicos del peer y autenticador, el puerto o identidad del autenticador no son incluidos en el Session-Id.

2.2.4. Ventajas y desventajas de EAP

Para concluir con el protocolo EAP vamos a ver una serie de ventajas y desventajas que presenta el mismo. Entre las ventajas de EAP tenemos:

- El protocolo EAP puede soportar múltiples mecanismos de autenticación sin tener que pre-negociar uno en particular.

- Los dispositivos NAS no tiene porque conocer cada método y actuarán como una entidad de paso según hemos visto en el comportamiento pass-through. Un autenticador puede autenticar peers locales y a su vez actuar como pass-through para peer de redes no locales al mismo.

- La separación del autenticador del servidor que realmente provee el servicio de autenticación[2] simplifica la gestión de credenciales y la toma de políticas de decisión.

[2]backend authentication server

Problema de la Re-autenticación Rápida

Merece la pena destacar que, entre las desventajas, los modelos actuales de autenticación y criptografía basados en EAP son desgraciadamente ineficientes en casos en que el peer EAP sea un dispositivo móvil. Cuando un peer llega a un nuevo autenticador, las restricciones de seguridad probablemente requerirán al peer ejecutar un método EAP completo, independientemente de si ha sido ya autenticado en la red y posee por tanto material criptográfico vigente. La ejecución de un método EAP completo puede implicar muchos round-trips entre el servidor y el peer EAP, retrasando en el tiempo el objetivo final del proceso que es refrescar la autenticación y las claves producidas en ésta.

Sería deseable proporcionar un protocolo de re-autenticación independiente del método ejecutado y a su vez eficiente. Para ello, el material criptográfico de la anterior ejecución podría ser empleado para proporcionar una re-autenticación. A su vez, también sería interesante disponer de un servidor local de baja latencia al peer que facilite la re-autenticación.

La re-autenticación de tipo HOKEY hace referencia al mecanismo de re-autenticación rápida basado en la administración de claves durante el proceso de handover[19].

Entre sus características se encuentran las deseadas anteriormente, que se pueden resumir en:

- Reutilizar el material criptográfico de la anterior autenticación para la próxima.

- Proporcionar un mecanismo de traspaso o derivación de claves en un servidor de autenticación local mucho más cercano en cuanto a latencia de las comunicaciones.

EAP-ER

La propuesta dentro del HOKEY WG para suministrar un proceso de reautenticación rápida se denomina EAP-ER (EAP Extensions for Efficient Re-authentication) y se basa en un modelo de dos partes de distribución de claves. EAP-ER describe una serie de extensiones para EAP que permiten la re-autenticación de forma eficiente para un peer que previamente ha llevado a cabo una autenticación completa EAP. Además mantiene la validez del material criptográfico que se derivó durante la autenticación inicial. Estas extensiones incluyen modificaciones del protocolo como la inclusión de dos nuevos mensajes o la creación de una nueva jerarquía de claves.

El intercambio para una re-autenticación EAP basada en EAP-ER necesita de una completa autenticación EAP previa, de la que se deriva la clave MSK por parte del servidor AAA y del cliente EAP. Como ya vimos, esta clave se transporta desde el servidor al autenticador EAP. En EAP-ER, de esta clave, tanto el autenticador como el peer derivan una nueva clave denominada re-authentication Root Key (rRK) que es la base de la jerarquía de claves EAP-ER. De la rRK a su vez se obtiene la clave rIK (re-authentication Integrity Key). Estas claves no se revelan a otras entidades.

Figura 2.5: EAP-ER.

Cuando el peer se mueve a un autenticador que soporta EAP-ER, manda un mensaje inicial de re-autenticación que se protege mediante la clave rIK y que contiene un peer-id, un NAI (Network Access Identifier) opcional y un número de secuencia. Cuando el servidor recibe el mensaje comprueba la validez del mismo y si las credenciales son válidas genera una clave de re-autenticación MSK (rMSK) de la rRK y responde con un mensaje EAP Finish Re-auth que transporta esa rMSK al autenticador. De esta forma, cuando el peer recibe un mensaje de inicialización de re-autentición (Initiate RE-auth message) lo comprueba, verifica la rMSK localmente y lleva a cabo

el establecimiento de una asociación de seguridad con el autenticador si todo es correcto.

La solución EAP-ER es capaz de efectuar un proceso de re-autenticación con un único roundtrip entre el autenticador y el servidor, lo que tiene un gran impacto en cuanto a la reducción de las latencias por handover. Por otra parte, esta solución tiene como inconveniente los cambios que hay que realizar sobre el protocolo EAP original. EAP-ER requiere efectuar cambios en las maquinas de estados EAP de todas las entidades, en particular en los autenticadores que deben ser capaces de manejar los nuevos mensajes.

Channel Binding

Existe un problema de seguridad relacionado con el uso de EAP. En particular, es posible que un autenticador EAP comprometido o mal implementado envíe información incorrecta o maliciosa al peer o al servidor pudiéndose hacerse pasar por otro autenticador. El peer normalmente no verifica la identidad del autenticador, por lo que esto puede desembocar en una vulnerabilidad de seguridad.

En [20] se puede ver como un autenticador puede descubrir a otro que se esté haciendo pasar por un peer. Por otra parte es posible que un autenticador pass-through actue como un cliente AAA, que provea información correcta al servidor AAA mientras envía información errónea o malintencionada al peer AAA, valiéndose del protocolo lower-layer.

Para este secuestro de identidad el autenticador se vale de los parámetros del nivel de transporte (EAP Lower Layer) que en EAP pueden ser importados, y que son conocidos por 'channel binding parameters'. Un ejemplo de estos parámetros para redes 802.11 empleando como mecanismo de AAA RADIUS podría ser:

- Called-Station-Id [4][21]
- Calling-Station-Id [4][21]
- NAS-Identifier [4]
- NAS-IP-Address[4]
- NAS-IPv6-Address[22]

Para protegerse de esta vulnerabilidad, los métodos EAP pueden desarrollar mecanismos para proteger estos parámetros. Channel binding es el proceso por el cual los parámetros de nivel de transporte son verificados por consistencia entre el peer EAP y el servidor EAP.

En cualquier solución con parámetros 'channel binding' el servidor debe ser capaz de autenticar la información de servicio provista por el autenticador.Cuando las dos partes intercambian parámetros de 'channel binding', se logra verificar que peer y servidor poseen la misma información de servicio provista por el autenticador. Que las dos partes posean la misma información no implica que esta sea correcta, un autenticador podría estar proveyendo la misma información falsa a ambos. Para evitar que esto ocurra, se configura fuera de linea la información relativa al autenticador en los servidores de autenticación, en particular la identidad del mismo, evitándose problemas de suplantación. Por norma general, en un servidor EAP se configurará la información relativa a los autenticadores de su dominio, esto provoca en escenarios de roaming la imposibilidad de autenticar la información suministrada por el autenticador a servidor y peer si no existe una relación de confianza directa entre los proveedores o la necesidad de que esta autenticación la realice un tercero con relación de confianza con el dominio perteneciente a ambos.

EAP no define el concepto de identificador de servicio o sus parámetros. Existen varias opciones para intercambiar parámetros de channel binding como son las propuestas de Ohba[23] y Arkko[24]. La primera evita la transmisión de parámetros entre peer y servidor permitiendo la comprobación de los mismos mediante el mecanismo de derivación de claves, lo que implica un método capaz de generar material criptográfico. La propuesta de Arkko se basa en el transporte de los parámetros de channel binding sobre el canal seguro generado por un método EAP capaz de transportar datos.

2.3. Algunos EAP Lower-Layer

Como vimos antes, un EAP lower-layer se encarga de transportar los paquetes entre el peer y el autenticador. Los EAP lower-layer permiten al peer llevar a cabo el proceso de autenticación basado en EAP con el autenticador. Dependiendo de la tecnología empleada a nivel de enlace, existen diferentes tipos. Por otra parte, existen a su vez protocolos independientes del nivel de enlace, ya que operan por encima del nivel de red, que son capaces de transportar una autenticación EAP entre un peer y un autenticador. En este apartado vamos a ver algunos de los diferentes EAP lower-layer existentes.

Para empezar hay que comentar los requisitos que debe cumplir un EAP lower-layer. EAP asume las siguientes características de un lower-layer:

- Transporte no fiable. EAP define su propio comportamiento de retransmisión de paquetes por lo que no asume que el nivel de transporte es fiable.

- Detección de errores. Métodos como CRC, checksum, MIC son necesarios para un lower layer de EAP. Sin este requisito un error no detectado podría colarse en las cabeceras de los paquetes EAP con los consecuentes fallos de autenticación.

- Seguridad. En cuanto a seguridad EAP no requiere que la lower layer posea servicios de seguridad. Sin embargo, cuando estos servicios están disponibles, los métodos EAP que soportan derivación de claves pueden usarlos para proveer de material criptográfico.

- Minimo MTU[3]. EAP es capaz de funcionar en capas inferiores que proporcionan un tamaño de MTU EAP de 1020 octetos o mayor. Por otro lado EAP no soporta descubrimiento de MTU ni fragmentación y reensamblado de mensajes.

- Duplicado de paquetes. Aunque es recomendable que el lower layer controle los paquetes duplicados esto no es un requisito.

- Orden garantizado. Que los paquetes lleguen en orden será una de las tareas que debe cumplir el lower layer, ya que originalmente EAP fue definido para correr sobre PPP y este posee dicha característica.

2.3.1. PPP

El protocolo PPP[25] (Point to Point Protocol) esta diseñado para transportar datagramas entre dos peer sobre enlaces simples. Estos enlaces proveen operaciones simultaneas y bi-direccionales full-duplex y asume que los paquetes serán entregados en orden. El protocolo PPP tiene tres componentes principales:

1. Mecanismo de encapsulación.- consigue la multiplexación de diferentes protocolos de red sobre el mismo enlace simultáneamente.

2. Protocolo de control de enlace (LCP). Con el fin de ser lo suficientemente versátil para ser portable a una amplia variedad de entornos, PPP proporciona un Link Control Protocol (LCP). El LCP es usado para acordar automáticamente las opciones de formato de encapsulación, gestionar los tamaños de paquete, detectar bucles, manejo de errores comunes y terminar el enlace.

[3]Maximum Transfer Unit - Unidad máxima de transferencia

3. Protocolos de control de red (NCP).- Estos protocolos (Network Control Protocols) se encargan de gestionar las necesidades requeridas por los protocolos de la capa de red.

Como hemos visto, EAP en un principio fue definido para usarse con PPP como lower layer. El protocolo PPP tiene una fase de autenticación. Esta fase se ejecuta justo después de la ejecución del protocolo de control de enlace (LCP). En esta fase de autenticación podemos usar un proceso de autenticación EAP para crear una asociación de seguridad entre los dos extremos.

2.3.2. IEEE 802.1X

La IEEE 802.1X[26] es una norma de la IEEE que define un control de admisión a la red basada en puertos. Llamamos puerto a un punto de conexión al servicio de red en una red de área local[4]. La norma IEEE 802.1X permite la autenticación de dispositivos conectados a un puerto LAN, estableciendo una conexión punto a punto o previniendo el acceso por ese puerto si falla la autenticación. 802.1X está disponible en ciertos conmutadores de red y puede configurarse para autenticar nodos que están equipados con software suplicante (STA). Esto elimina el acceso no autorizado a la red al nivel de la capa de enlace de datos.

Esta norma se usa en redes 802.11 para controlar el acceso de sus estaciones (STA) a los puntos de acceso (AP). En redes 802.11 se definen dos tipos de puerto:

- Puerto no controlado. Permite tráfico con restricciones, permite el procesamiento de un conjunto de frames 802.11.
- Puerto controlado. Permite tráfico sin restricciones siempre que esté habilitado. Puede estar en estado autorizado o no autorizado. En estado no autorizado no se puede enviar ningún tipo de tráfico, mientras que un puerto controlado en estado autorizado permite que el STA pueda intercambiar tráfico de datos en la red.

Para el manejo del puerto controlado es necesario un proceso de autenticación. Este proceso se basa en el protocolo de autenticación extensible (EAP). El autenticador lleva a cabo la autenticación EAP con un servidor EAP por el puerto no controlado. Si el resultado de esta autenticación es satisfactorio su puerto controlado pasa al estado autorizado. De igual forma cuando se le comunica a la estación STA el éxito de la autenticación está también pasa su puerto controlado a estado autorizado.

[4]LAN

2.3.3. IEEE 802.11i

IEEE 802.11i[2] fue creado para suministrar seguridad a redes inalámbricas. El estándar 802.11i elimina muchas de las debilidades de sus predecesores tanto en lo que autenticación de usuarios como a robustez de los métodos de encriptación se refiere. Lo consigue en el primer caso gracias a su capacidad para trabajar en colaboración con 802.1X, y en el segundo, mediante la incorporación de encriptación Advanced Encryption Standard (AES). Aparte de incrementar de manera más que significativa la seguridad de los entornos WLAN, también reduce considerablemente la complejidad y el tiempo de handoff de los usuarios de un punto de acceso a otro a través de un mecanismo de pre-autenticación a nivel de enlace.

En el tema de autenticación de usuarios las tecnológias inalámbricas anteriores a 802.11i ofrecian una autenticación basada en el protocolo WEP (Wired Equivalent Privacy). Este protocolo se demostró que era vulnerable ya que fue comprometido. Combinando WEP con el protocolo de autenticación 802.1X obtenemos una mejora ya que el cliente WEP esta obligado a solicitar acceso a la red utilizando EAP (Extensible Authentication Protocol), tal como establece 802.1X. 802.11i hereda de 802.1X los conceptos de puerto controlado y puerto no controlado.

Básicamente, 802.11i incrementa la seguridad WLAN utilizando algoritmos de encriptación y técnicas basadas en claves más avanzadas. Cuando una estación inalámbrica solicita abrir una sesión con el punto de acceso, entre ambos extremos se establece una clave denominada Pairwise Master Key (PMK). Para ello se utiliza típicamente el estándar LAN y WLAN 802.1X, que permite al responsable de seguridad aplicar un método de autenticación tan potente como desee, desde las simples combinaciones usuario-contraseña hasta certificados digitales. Se trata de un mecanismo de autenticación de usuario basado en un servidor de autenticación (puede ser RADIUS o Diameter) y en el protocolo Extensible Authentication Protocol (EAP). Tras finalizar el proceso de autenticación EAP con éxito, el servidor de autenticación retorna la PMK al punto de acceso, y, entonces, éste y la estación intercambian una secuencia de cuatro mensajes, denominada 4-way hadshake[5].

Como puede verse en [27], durante el proceso 4-way hadshake, se utilizan la PMK y diversos valores generados aleatoriamente tanto desde la estación como desde el punto de acceso, renovándose varias veces durante la sesión para securizar el proceso de pacto de una nueva clave, denominada Pairwise Transient Key (PTK). Ésta se compone a su vez de tres subclaves: una para firmar los cuatro mensajes que intervendrán en el proceso, otra para asegurar

[5]saludo a 4 vias

los paquetes de datos transmitidos entre los dos dispositivos implicados y una tercera para encriptar la llamada 'clave de grupo', que será enviada desde el punto de acceso a la estación y que permitirá a aquél difundir tráfico multicast a todos los clientes a él asociados, sin tener que enviar a cada uno de ellos un mismo paquete encriptado de forma diferente. A lo largo del proceso, estación y punto de acceso negocian también el tipo de encriptación que utilizarán para cada conexión. De esta negociación resultarán dos claves. Una de ellas es la clave de grupo ya mencionada; la otra, denominada clave pairwise, se utilizará para las transmisiones de datos en modo unicast que sólo afectan al punto de acceso. Si el 4-way-handsake finaliza con éxito, el punto de acceso y la estación STA quedan autenticados y con claves para proteger el enlace inalambrico entre ambos, quedando el puerto controlado de cada uno en estado autorizado.

2.3.4. IKEv2

El protocolo IKE (Internet Key Exchange Protocol) es un protocolo que se usa para autenticación y para establecer y mantener asociaciones de seguridad IPsec. IPsec (IP Security) provee de autenticidad e integridad a los datagramas IP mediante el uso de cabeceras AH (Authentication Header), o bien integridad, autenticidad y confidencialidad usado cabeceras ESP (Encapsulating Security Payload). Estos servicios se proveen a través del establecimiento de un estado compartido que define los algoritmos criptográficos usados, algunas claves necesarias para dichos algoritmos, etc. Para establecer este estado compartido IPsec se vale del protocolo IKE.

El protocolo IKE se encuentra en su segunda versión. Esta versión surgió por las deficiencias encontradas en la versión antigua IKEv1. IKEv2[7] no puede interoperar con la versión 1, aunque ambas versiones se pueden ejecutar sin ambigüedad sobre el mismo puerto UDP ya que las cabeceras tienen un formato bastante común.

IKE lleva a cabo una autenticación mutua entre las dos partes implicadas en el proceso y establece una asociación de seguridad IKE, que incluye material de secreto compartido que puede ser usado para establecer de forma eficiente SAs para el uso de AH y ESP. IPsec tiene dos modos de funcionamiento:

- Modo transporte. En este modo sólo la carga útil del paquete IP se cifra y/o se autentica. El enrutamiento permanece intacto ya que la cabecera del paquete IP no sufre modificación alguna.
- Modo tunel. En el modo túnel, todo el paquete IP (datos más cabeceras del mensaje) es cifrado y/o autenticado. Debe ser entonces encapsulado

en un nuevo paquete IP para que funcione el enrutamiento. El modo túnel se utiliza para comunicaciones red a red (túneles seguros entre routers, p.e. para VPNs) o comunicaciones ordenador a red u ordenador a ordenador sobre Internet.

Todas las comunicaciones IKE consisten en parejas de mensajes, una solicitud y una respuesta. A estas parejas se las denomina intercambio. La entidad que manda las solicitudes tiene la responsabilidad de asegurar la fiabilidad. Si la respuesta no se recibe dentro de un intervalo de tiempo, la entidad que mandó la solicitud la reenviará o abandonará la conexión.

El primer intercambio del protocolo se denomina 'IKE SA INIT' que sirve para establecer una IKE SA. El intercambio 'IKE SA INIT' negocia parámetros de seguridad para la IKE SA, se intercambian *nonces* y se envían valores *Diffie-Hellman*.

El segundo tipo de intercambio que se da tras el establecimiento de la IKE SA es el 'IKE AUTH'. Este intercambio transmite las identidades, autentica el primer intercambio y levanta una SA para la primera 'CHILD SA' (asociación de seguridad IPsec). Es aquí donde se negociará el uso de cabeceras AH y ESP y se escogerá entre modo tunel o modo transporte. También es aqui donde IKEv2 proporciona la posibilidad de usar EAP como mecanismo de autenticación de las entidades participantes. IKEv2 puede transportar EAP con propósitos de autenticación, aunque impone restricciones. Solo permite autenticar mediante EAP a la entidad que inicia las solicitudes o iniciador, para autenticar a la entidad que recibe la solicitud IKEv2 utiliza otros mecanismos.

Normalmente hay un único intercambio 'IKE SA INIT' y un único intercambio 'IKE AUTH' que hacen un total de 4 mensajes, para establecer la asociación de seguridad IKE así como la primera 'CHILD SA'. Hay casos en que puede haber más de uno de estos intercambios, pero siempre se deber completar un intercambio 'IKE SA INIT' antes de cualquier otro tipo de intercambio.

Tras completar un intercambio 'IKE AUTH' es posible que no sean necesarios nuevos intercambios, pero también pueden crearse nuevas SA mediante nuevos intercambios denominados intercambio 'CREATE CHILD SA' e intercambio 'INFORMATIONAL'. El intercambio 'CREATE CHILD SA' sirve para crear nuevas CHILD SA. Se envía protegido por la IKE SA y puede iniciarse por cualquier extremo. Por último, el intercambio 'INFORMATIONAL' se usa para eliminar una asociación de seguridad. También es usado para informar de errores y para comprobar si la otra parte sigue funcionando correctamente.

2.3.5. PANA

El protocolo PANA[3] (Protocol for Carrying Authentication for Network Access) permite a los clientes autenticarse a través de EAP en la red de acceso usando protocolo que trabaja por encima del nivel IP. Como tal, PANA permite transportar cualquier método EAP por encima de IP. PANA es un protocolo basado en UDP. Tiene su propio mecanismo de retransmisión de mensajes para proporcionar fiabilidad a los intercambios.

El protocolo PANA funciona entre un cliente denominado PaC (PANA Client) y un servidor llamado PAA (PANA Agent). Los mensajes del protocolo lo conforman una serie de solicitudes y respuestas. Cada mensaje puede transportar cero o más AVP (Attribute-Value Pairs) dentro de su carga útil. La principal carga útil de PANA es la que se encarga de llevar a cabo la autenticación EAP. PANA ayuda al PaC y al PAA a establecer una sesión EAP. El protocolo permite interactuar con una infraestructura AAA subyacente que verifica las credenciales presentadas por el usuario a través de EAP.

Los mensajes PANA se envían entre el PaC y el PAA como parte de una sesión PANA. A continuación vamos a ver las distintas fases que componen una sesión PANA:

- Fase de autenticación y autorización. Con esta fase se inicia una nueva sesión PANA y se ejecuta EAP entre el PaC y el PAA. La sesión puede ser inicializada por cualquiera de las dos entidades. La carga útil de EAP (que transporta un método EAP en su interior) es usada para la autenticación. Es el PAA quien comunica el resultado de la autenticación y autorización al PaC concluyendo esta fase.

- Fase de Acceso. Tras la finalización de forma exitosa de la fase anterior el dispositivo de acceso permite el acceso a la red y el tráfico IP a través de las EPs[6]

- Fase de re-autenticación. Durante la fase de acceso el PAA o el PaC pueden iniciar una re-autenticación para actualizar el tiempo de vida de la sesión PANA antes de que esta expire. Esto se hace en la fase de re-autenticación. Al finalizar esta fase se vuelve a la fase de acceso si la re-autenticación ha sido satisfactoria.

- Fase de terminación. El PaC o el PAA pueden decidir terminar con la sesión y por tanto con el acceso al servicio en cualquier momento.

[6]Enforcement Point - es la entidad que finalmente aplica las decisiones de autenticación y autorización.

Para ello se manda un mensaje de desconexión. Si este mensaje no es enviado la sesión finalizará cuando concluya el tiempo de sesión.

Por último cabe destacar que la protección de mensajes contra modificación entre PaC y PAA es posible gracias a la exportación de claves de EAP junto con un método EAP específico. Estas claves se usan para crear una asociación de seguridad PANA.

Figura 2.6: Componentes de PANA.

2.4. Métodos EAP

Los métodos EAP se encargan de implementar algoritmos de autenticación. Como vimos en la introducción, EAP es una opción interesante a la hora de implementar procesos de bootstrapping. La alternativa más recomendable para esa implementación es a través de los métodos EAP, ya que permiten la transparencia al nivel EAP y no modificamos el estándar EAP para ampliar esta funcionalidad. En esta sección vamos a describir algunos de los métodos EAP existentes así como sus principales características para poder tener una visión más amplia de las mejoras introducidas por nuestro método EAP-EXT.

En la sección de claves en EAP se comentó que, además de proporcionar autenticación, existen métodos EAP capaces de exportar material criptográfico en forma de claves. Dentro de los métodos EAP podemos hacer la siguiente clasificación:

- Métodos que no exportan material criptográfico dotando solo de autenticación.

- Métodos que después del proceso de autenticación exportan material criptográfico. Entre estos últimos tenemos:
  - Métodos tunelados que permiten encapsular otros métodos EAP así como otro tipo de información dentro de su campo de datos.
  - Métodos no tunelados que no permiten dicha encapsulación.

Visto esto vamos a explicar algunos métodos EAP:

EAP-MD5 [6] El mecanismo de acción de EAP-MD5 consiste en recoger un username y un password del usuario, encriptarlos usando un algoritmo hash MD5 y enviarlo al servidor AAA. Es un método bastante simple y tiene algunos defectos. No posee un mecanismo de cambio de claves en el tiempo, no puede cumplir el requisito sobre autenticación simétrica entre el cliente y el punto de acceso y viceversa lo que le hace susceptible a ataques man-in-the-middle. Debido a estos factores de vulnerabilidad no debería ser considerado como un método seguro EAP.

EAP-TLS (Transport Level Security) es un método EAP basado en el RFC2716 [28] que utiliza para el procedimiento de autenticación un certificado de clave pública. Proporciona un medio de autenticación mutua entre el cliente y el autenticador, así como entre el autenticador y el cliente. Como pre-requisito necesita que cada entidad se encuentre autenticada, incluyendo el cliente y el autenticador y que posean un certificado de clave pública firmado por la misma autoridad de certificación. Esta solución tiene fuertes propiedades de autenticación y es muy segura, pero requiere de una infraestructura de clave pública para su uso. EAP-TLS proporciona autenticación mutua, intercambio y establecimiento de claves, apoyo a la fragmentación y reensamblaje, y reconexión rápida.

EAP-TTLS o TLS tunelado[29], es una extensión de EAP-TLS. En este método de EAP, se establece un túnel seguro entre el servidor y el cliente utilizando un algoritmo de clave pública y los certificados emitidos por una autoridad certificadora de confianza mutua. Una vez que este túnel está establecido, otro método de autenticación se emplea y la transacción se realiza a través del túnel seguro. Debido a este túnel el método de autenticación interno puede ser menos seguro, como por ejemplo EAP-MD5. EAP-TTLS provee de beneficios de autenticación mutua, instrumentos para una negociación cifrada segura, capacidad para usar certificados y password, y mantenimiento de la identidad del usuario en privado, ya que cualquier contraseña de autenticación se producirá dentro del túnel.

EAP PEAP [30] Este método tiene un comportamiento parecido a EAP-TTLS. Crea una sesión TLS para transportar otro método de autenticación. Una vez creado el túnel puede utilizarse otro método menos seguro dentro del mismo. La diferencia con EAP-TTLS consiste en que PEAP proporciona un medio de autenticación del autenticador al cliente, pero no en la otra dirección. Esto reduce complejidad y costes. PEAP incluye autenticación y encriptación de mensajes, intercambio seguro de claves, soporte para fragmentación y reemsamblado y reconexión rápida.

EAP-SIM [31] se basa en la idea de utilizar EAP para Subscriber Identity Modules los cuales proveen el método actual estándar de autenticación utilizado por muchos proveedores de equipos celulares. Este método EAP utiliza como credencial una tarjeta inteligente como dispositivo de almacenamiento. Los clientes usan estas SIMs para suministrar sus credenciales en los procesos de autenticación. El método EAP con tarjetas SIMs en redes inalámbricas es parecido a las técnicas de autenticación usadas en las redes GSM. Este método tiene la ventaja de usar un dispositivo físico como credencial en lugar de una clave o password, pero no es independiente de la sesión ya que esa credencial debe usarse para todas las sesiones del usuario o dispositivo.

EAP-AKA [32] es una pequeña variación de EAP-SIM. Se desarrolla para dispositivos 3GPP y reemplaza a las tarjetas SIM usadas en redes GSM por USIMs (user service identity modules) utilizadas en redes UMTS. EAP-AKA presenta un nivel de seguridad mayor debido al uso de claves permanentes para la autenticación mutua.

En la siguiente tabla podemos observar con más claridad las principales características de los métodos EAP descritos anteriormente.

Requisito	**EAP-MD5**	**EAP-TLS**	**EAP-TTLS**	**EAP-PEAP**	**EAP-AKA**
Tunelado	No	No	Si	Si	No
Genera material criptográfico	No	No requerido	Si	Si	Si
Autenticación Mutua	No	Si	Si	No	Si
Autoprotección	Si	Si	Si	Si	Si
Protección Man-in-the-middle	No	Si	Si	Si	Si
Negociación protegida de la suite criptográfica	No	No requerido	Si	Si	Si
Oculta identidad del usuario	No	No	Si	Si	opcional
Reconexión Rápida	No	Si	Si	Si	opcional

Figura 2.7: Comparativa Métodos EAP

Capítulo 3

Diseño de un método EAP para el bootstrapping EAP-EXT

Hasta el momento, hemos intentado introducir al lector en el problema que puede suponer en entornos de movilidad los procesos de autenticación, en particular, los procesos de autenticación previos a un mecanismo de bootstrapping. Se han introducido una serie de tecnologías que de forma directa o indirecta, están relacionadas con el proceso de bootstrapping a la hora de autenticar y autorizar el acceso a un servicio. Así nos encontramos con las tecnologías inalámbricas que necesitan de mecanismos de bootstrapping implementados a través de arquitecturas AAA y EAP.

Procedemos a reintroducir al lector en el problema que puede suponer el problema del bootstrapping en entornos de movilidad inter-dominio e inter-tecnología con la intención de justificar el desarrollo de una solución que proveerá la abstracción necesaria de los niveles inferiores para crear un proceso de autenticación exportable, dinámico y extensible. Esta solución está llamada a ser un mecanismo de bootstrapping genérico que permitirá entre otros logros la llegada de la cuarta generación.

3.1. El problema del bootstrapping

Las operadoras despliegan grandes y costosas infraestructuras de acceso y de control para poder proveer a sus clientes los servicios contratados, entre ellos nos encontramos de forma destacada, por su naturaleza de servicio básico del operador, con el servicio de acceso a la red.

En general, todo acceso a la red viene precedido de un proceso de autenticación mediante el cual, red y usuario, se identifican de manera unívoca. Muchas son las razones para llevar a cabo un proceso de identificación de

cliente, sin embargo, destaca de entre todas la necesidad del operador de controlar y proteger el acceso a los servicios que suministra. Puesto que, en general estos servicios suponen un coste asociado, el operador se ve en la necesidad de tarificar el uso de la red y evitar accesos no permitidos, y de ahí la necesidad de desplegar infraestructuras que permitan procesos de autenticación y autorización de forma flexible.

No obstante, el acceso a cualquier servicio en general y a la red en particular, obliga al cliente al conocimiento de cierta información relacionada con el mismo, que será la que le permita además configurar su acceso. Esta información pueden ser tan simple como su propia dirección de red o tan compleja como parámetros de calidad de servicio o de enrutamiento, entre otros. De forma general, se podría tener esta información preconfigurada en el equipo de cliente, sin embargo cuando el número de clientes aumenta su gestión se hace difícil. Por tanto, sería deseable tener un mecanismo de autenticación escalable y dinámico, que además fuese capaz de suministrar la información necesaria para el acceso al cliente de forma segura. Afortunadamente, gracias a las arquitecturas AAA desplegadas en la actualidad, los operadores son poseedores de estos mecanismos dinámicos tan anhelados de autenticación y provisión de información. A todo proceso que involucra la autenticación de un usuario y el suministro de información y de material criptográfico para acceso a un servicio se le denomina normalmente *bootstrapping*.

En todo proceso de bootstrapping la primera tarea a realizar es un proceso de autenticación durante el cual se asegura que ambas partes, red y usuario, son quien dicen realmente ser. Una característica deseable de este proceso es la posible generación de material criptográfico que pueda ser empleado para securizar toda comunicación posterior entre cliente y operador.

Una vez identificado el usuario, es necesario pasar a una segunda fase de autorización, donde se verificará de qué forma ese usuario tiene permitido el acceso a la red. En muchos casos esta decisión será simplemente tomada en base a si el usuario se ha identificado de forma satisfactoria o no. Otros casos requerirán la comprobación de políticas definidas por la operadora de la red accedida o por la operadora de origen en escenarios multidominio. Si se autoriza el acceso al usuario se puede dar paso al proceso de auditoría (accounting) mediante el cual se monitoriza el uso del servicio por parte del usuario, esta monitorización puede ser empleada para tareas simplemente de control o para facturación.

La dirección a seguir por parte de las operadoras es la de proveer un mecanismo dinámico y seguro que permita suministrar al usuario la información necesaria para que este pueda acceder de forma adecuada al servicio de red. Este método de acceso está basado en la posesión por parte del usuario de una credencial con un periodo de validez elevado que será la que permita

realizar el proceso de autenticación inicial. Durante este proceso se genera una relación de confianza entre el cliente y el servidor de autenticación, que actúa de tercera parte confiable. Mediante esta relación se consigue crear una asociación de seguridad entre el punto de conexión a la red y cliente.

En segundo lugar y gracias a esta asociación de seguridad, se provee al cliente de forma segura la información relativa al acceso como podrían ser parámetros de calidad de servicio, configuración del nivel de red, etcétera, lo que por supuesto implica que el usuario tiene autorizado el acceso.

Para este proceso de autenticación existen múltiples opciones. El protocolo EAP en particular, posee ciertas características que le convierten en una posibilidad muy atractiva. A su capacidad de abstracción frente al lower layer o el protocolo AAA empleado, hay que añadir la capacidad de algunos de sus métodos de generar material criptográfico. Por ejemplo, soluciones como la descrita en [9], aprovechan, de forma adicional, la capacidad de algunos métodos EAP como PEAPv2 [11] o EAP-FAST [12] de transportar información genérica junto con la información de autenticación, para realizar un mecanismo de bootstrapping específico de un servicio de movilidad basado en el protocolo Mobile IPv6 (MIPv6). En concreto, gracias al mecanismo de bootstrapping se le suministra al nodo móvil información acerca de una entidad en la red que será la encargada de gestionar la movilidad del usuario (denominada *home agent*). De esta forma, se evita la engorrosa configuración estática de los dispositivos con la dirección IP específica de esta entidad, a parte de que permite a la operadora seleccionar un *home agent* concreto dependiendo del perfil del usuario y de la situación del servicio de movilidad en un determinado momento.

La realidad es que puesto que EAP define un mecanismo de extensión propio mediante sus métodos, es posible la implantación y creación de nuevo métodos EAP de autenticación que incluyan la capacidad de bootstrapping sin necesidad de modificar los estándares ya establecidos y desplegados en equipamiento existente. Además son muchas las tecnologías empleadas para el acceso a la red que permiten el uso de EAP como mecanismo de autenticación, como por ejemplo PPP, 802.1x, 802.11i, IKEv2, etcétera. Además los dos protocolos AAA más relevantes, RADIUS y Diameter, definen mecanismos específicos para el transporte de EAP([20], [33]).

Gracias a estas propiedades, EAP se convierte en firme candidato a la hora de desarrollar mecanismos de bootstrapping de servicios relacionados con el acceso a la red. Para evitar la modificación de los estándares existentes, se ha tomado la decisión de desarrollar un método EAP que permita ejercer de mecanismo de bootstrapping de diferentes servicios de red. La motivación principal radica en el hecho de que, aunque existen métodos tunelados que ya realizan ciertas operaciones de bootstrapping su proceso no está completa-

mente refinado, obligando a realizar autenticaciones de un determinado tipo o proponiendo mecanismos de tunelado diferentes en cada método. Nuestra propuesta en cambio separa claramente el proceso de autenticación del de tunelado, haciendo uso de otros métodos ya existentes y capaces de generar material criptográfico para la fase de autenticación. Tras esta fase y aprovechando las claves exportadas por los métodos, nuestra propuesta extiende la funcionalidad de EAP para permitir el transporte de datos referentes al servicio desde el servidor EAP hasta el peer. Este método EAP además permite el secuenciamiento de varios métodos EAP de forma segura, esta característica resulta interesante en escenarios en los que es necesario ejecutar varios mecanismos de autenticación relacionados antes de obtener el acceso a la red. Sirva como ejemplo un acceso con autenticación de terminal previa a la autenticación del usuario. Como añadido, este método ofrece la opción de un mecanismo de re-autenticación rápida para reducir los efectos negativos que infieren los procesos de autenticación en entornos de movilidad debido a los retardos incorporados, tal y como describíamos en la sección 3.7.

3.2. Descripción de EAP-EXT

EAP-EXT es un método EAP tunelado diseñado para extender las capacidades nativas de autenticación de EAP, dotando a este de la capacidad añadida de servir de proceso de bootstrapping. EAP-EXT utiliza otros métodos EAP para realizar todo el proceso de autenticación, obteniendo de estos la información criptográfica necesaria para securizar el transporte. EAP-EXT permite el trasiego de información necesario entre el servidor de autenticación y el peer para la configuración del acceso a un servicio de forma segura. El transporte se realiza mediante el encapsulamiento de la información en TLVs. Además y de forma añadida, EAP-EXT permite suplir ciertas carencias de los métodos ejecutados y transportados por EAP-EXT, a la hora de cumplir con las recomendaciones definidas en el EAP-KMF[8]. En particular, es capaz de aprovechar la clave MSK exportada por métodos ejecutados dentro de EAP-EXT (*métodos internos*) para proteger sus propios paquetes y para derivar no sólo una nueva clave MSK sino también una nueva clave EMSK. De esta forma, aunque el método EAP interno no genere una EMSK (como pasa normalmente) al final del proceso con EAP-EXT sí que se dispondrá de esa clave para otras aplicaciones (p.ej. [34]).

Como aportación adicional, y teniendo en cuenta las diferentes claves gestionadas por EAP-EXT, se presenta además un mecanismo de re-autenticación rápida que permite llevar a cabo la operación del método en un número reducido de intercambios y sin necesidad de ejecutar ningún otro método.

3.2.1. Intercambios en EAP-EXT

Antes del comienzo de EAP-EXT se produce un intercambio EAP-Request /Response Identity iniciado por el autenticador. En base a la identidad contenida en el mensaje EAP-Response, el servidor selecciona el método EAP-EXT y el método interno que se utilzará para la autenticación. A continuación, envía el primer mensaje EAP-EXT al peer.

Los métodos internos son transportados en Method TLVs. Por su parte, los AUTH TLV y ENC TLV proveen de seguridad al intercambio de información, los primeros como mecanismo de autenticación de las partes y los segundos como mecanismo de privacidad. Mientras que ningún método interno genere material criptográfico EAP, ningún AUTH TLV es incluido, por tanto las capacidades intercambiadas no están protegidas. Sin embargo, tras la generación de material criptográfico EAP por un método interno, los intercambios EAP-EXT tienen que ir protegidos con un AUTH TLV. Los AUTH TLVs en mensajes EAP-EXT tienen que ser calculados empleando la EAP-EXT-AUTH-KEY generada a partir del material criptográfico EAP del último método interno que realizó una autenticación satisfactoria. Esto implica que si hay múltiples métodos internos que son ejecutados secuencialmente dentro de EAP-EXT, una nueva EAP-EXT-AUTH-KEY es generada cada vez que un método interno en la secuencia genere material criptográfico, de esto se puede deducir que todo método EAP interno a emplear por EAP-EXT tiene que ser capaz de generar material criptográfico. Si por el contrario sólo hay un método EAP interno, intercambios EAP-EXT adicionales con un AUTH TLV necesitan ser realizados después de la finalización del método EAP interno y antes de enviar un mensaje EAP-Success o un EAP-Failure. Con este último intercambio se confirman los datos negociados por ambas partes durante la comunicación sin posibilidad de ser modificados por un tercero.

Al final de una ejecución EAP-EXT satisfactoria, material criptográfico EAP es derivado de la MSK generada por el último método EAP interno satisfactorio y es exportado por el EAP-EXT. La función pseudo aleatoria usada para derivar el material criptográfico EAP y las USRKs (Usage Specific Root Keys)[34] podrían ser negociadas dentro de EAP-EXT usando PRF TLVs. Un bit (F-bit) es usado para indicar el final del intercambio EAP-EXT. La Figura 3.1 muestra un ejemplo de secuencia de mensajes EAP-EXT con un único método EAP interno y con negociación de la función de derivación de claves PRF, se puede observar claramente como los dos últimos intercambios de EAP-EXT no transportan ningún Method-TLV, estos dos son los intercambios adicionales que se comentaron con anterioridad.

Dentro de una conversación EAP tradicional, el peer y el autenticador

puede utilizar un único método de autenticación, tras el cual el autenticador debe enviar una respuesta que confirme o deniegue la solicitud de autenticación (success o failure). No obstante resulta interesante la posibilidad de realizar múltiples procesos de autenticación relacionados. Este mecanismo por ejemplo es el empleado para un entorno en el que se requiera una autenticación del terminal seguida por una autenticación del usuario, el terminal podría emplear EAP-SIM [31] o EAP-AKA [32] mientras que el usuario podría emplear por ejemplo EAP-TLS. Existen proyectos como DAIDALOS[10] que procuran mediante autenticación de dispositivo y de usuario, ofrecer a este último la posibilidad de múltiples roles a la vez que proporcionarle anonimato entre otros servicios.

Aunque por su diseño EAP no permite la ejecución de múltiples métodos en un mismo proceso de autenticación, si está contemplado en éste la ejecución de métodos de los denominados tunelados, los cuales permiten la ejecución de un proceso completo de otro método en su interior. EAP-EXT extiende la funcionalidad de EAP permitiendo la ejecución de múltiples métodos EAP internos de forma segura gracias a la relación criptográfica generada entre los métodos internos y EAP-EXT. La Figura 3.2 muestra un ejemplo de secuencia de mensajes EAP-EXT con múltiples métodos EAP internos, observar como en este caso no es necesario realizar intercambios adicionales sin transporte de método alguno.

3.2.2. Channel Binding

Como mencionábamos al final de la sección 2.2.4, los métodos EAP pueden importar y exportar parámetros del nivel de transporte (EAP Lower layer) conocidos como 'channel binding parameters' o simplemente 'channel bindings'. Estos parámetros contienen información relacionada con las entidades participantes en un acceso al servicio de red determinado. Gracias al uso o intercambio de los mismos entre el peer y el servidor, estas entidades pueden comprobar que la entidad intermedia, es decir el autenticador, no intentó engañar a ninguna de las partes con ciertos parámetros

Existen al día de hoy dos alternativas de verificar la consistencia de estos datos entre peer y autenticador. La primera [23] relaciona los parámetros enviados por el autenticador al peer y al servidor, con el proceso de derivación de claves. Si los parámetros enviados difieren, las claves generadas en peer y servidor serán diferentes. De esta forma, la clave MSK enviada al autenticador y derivada por el servidor, será diferente a la que posee el cliente, con lo que el proceso de creación de asociación de seguridad entre peer y autenticador fallará.

Por el contrario, la segunda propuesta [24] solo señala la importancia del

Figura 3.1: EAP-EXT ejecutando un único método interno

Figura 3.2: EAP-EXT ejecutando dos métodos internos

envío de esta información de channel binding a través de un túnel seguro entre el cliente y el servidor. En particular a través de este túnel, el cliente y el servidor intercambian la información que recibieron del autenticador para comprobar si coinciden. Por tanto para poder aplicar esta extensión es necesario que el método sea capaz de transportar datos de forma protegida con una clave conocida sólo por los extremos de la comunicación EAP y no por el autenticador. Las claves TEK[1] pueden ser empleadas con este fin.

A través de EAP-EXT, es posible para el peer y el servidor negociar el uso de cualquiera de estas dos propuestas. Adicionalmente, gracias al túnel seguro definido es posible intercambiar, de manera protegida, entre peer y servidor la información de channel binding necesaria. En concreto, mediante el intercambio de Channel-Binding-Mechanism TLVs entre peer y servidor se seleccionará el mecanismo de channel binding a emplear en caso que el C-bit indique que ambos soportan la capacidad de channel binding.

3.3. Manejo de errores

Un error podría ocurrir por múltiples razones, por ejemplo, debido a un fallo en la autenticación del método EAP interno o un TLV EAP-EXT mal formado, desconocido o ausente. Un error podría ser detectado igualmente por el peer o por el servidor. Un mensaje EAP-EXT que causa un error es referenciado como un mensaje erróneo. Los mensajes EAP-EXT con el E-bit activado son usados para indicación de errores. Estos mensajes son referidos como indicaciones de error. Una indicación de error tiene que contener un AUTH TLV, y no debería contener otros TLVs. Cualquier mensaje erróneo (incluyendo una indicación de error errónea) sin un AUTH TLV válido, tiene que ser descartada sin realizar otra acción.

Para un Request erróneo con un AUTH TLV válido, el peer envía un Response de indicación de error. Para un Response erróneo con un AUTH TLV válido, el servidor envía un Request de indicación de error que es respondido por el peer con un Reponse de indicación de error. El servidor devuelve un mensaje EAP-Failure en respuesta a un Response de indicación de error con un AUTH TLV válido.

[1] Transient EAP Keys

3.4. Claves

3.4.1. Esquema de Derivación

En el documento *EMSK Root Key Derivation*[34] se define un esquema general que permite la derivación de una jerarquía de claves utilizando como clave raíz la EMSK. Para realizar el proceso de derivación se emplea una función de derivación de claves KDF. La KDF emplea la EMSK, una etiqueta de clave, datos opcionales y una longitud de clave deseada para la derivación de la clave resultante. La KDF debe producir una misma salida para una misma entrada.

$clave = KDF(EMSK, etiquetadeclave, datosopcionales, longitud)$

La etiqueta de clave está destinada a identificar el tipo de clave. La KDF debe ser capaz de tratar hasta 2048 bytes de datos opcionales. Una implementación de la KDF debe ser capaz de producir como mínimo 2048 bytes de salida, en cualquier caso, es recomendable que las claves raíz tengan como mínimo 64 bytes de longitud.

En particular, la KDF se define como $KDF = PRF(key, data)$, la PRF por defecto ha sido extraída de la función PRF expandida (PRF+) de IKEv2[7] que está basada en el empleo de HMAC-SHA-256, habiendo sido elegida por su simplicidad y eficiencia. La función PRF+ se define como sigue:

$prf + (K, S) = T_1|T_2|T_3|T_4|...|T_n$ donde:

$T_1 = prf(K, S|0x01)$
$T_2 = prf(K, T_1|S|0x02)$
$T_3 = prf(K, T_2|S|0x03)$
$T_4 = prf(K, T_3|S|0x04)$

Este proceso se repite hasta lograr la longitud de clave deseada. S se refiere aquí a la etiqueta de clave, los datos opcionales y la longitud. Puesto que PRF+ está definida únicamente para 255 iteraciones podría producir hasta 8160 bytes de longitud de clave.

3.4.2. Claves derivadas en EAP-EXT

Siguiendo el esquema de la sección anterior, EAP-EXT deriva un conjunto de claves necesarias para su operación. Teniendo en cuenta que EAP-EXT transporta métodos EAP de autenticación que exportan claves, y que algunos de ellos sólo exportan la clave MSK definida en el EAP-KMF[8], se emplea como clave raíz para generar toda la jerarquía, precisamente la clave MSK que generó el último método interno que realizó un proceso de autenticación satisfactorio. Denominemos a esta clave MSK_i. Utilizando esta clave MSK_i se deriva una jerarquía tal y como muestra la figura 3.3, y que procedemos a

explicar a continuación.

Figura 3.3: Jerarquía de claves EAP-EXT

MSK y EMSK exportadas por EAP-EXT

Dos claves de 64 bytes MSK y EMSK son exportadas desde EAP-EXT siguiendo las recomendaciones del EAP-KMF. Como explicábamos, son derivadas de la MSK_i exportada por el último método EAP interno finalizado con éxito, usando la KDF definida en [34] de la siguiente manera.

(MSK, EMSK) = KDF(MSK_i, 'EAP-EXT-EAP-Keying-Material', 128)

En realidad, la función KDF genera 128 bytes, siendo los primeros 64 bytes la MSK y los 64 subsiguientes la EMSK.

EAP-EXT-AUTH-KEY

Para dotar de integridad y autenticación a los mensajes intercambiados en EAP-EXT, se deriva una clave EAP-EXT-AUTH-KEY utilizada en el cálculo de los AUTH TLVs. EAP-EXT-AUTH-KEY es usada dentro de EAP-EXT sólo y nunca es exportada. Por tanto, se trata de una clave TEK tal y como definíamos en la sección 2.2.3. La EAP-EXT-AUTH-KEY también es derivada de la MSK generada por el último método EAP interno satisfactorio (MSK_i), usando la prf+ definida en [7] de la siguiente manera.

EAP-EXT-AUTH-KEY = prf+(MSK_i, 'EAP-EXT-Authentication-Key',length);

El algoritmo de hash por defecto para prf+ es PRF_HMAC_SHA2_256. El campo longitud dependerá del algoritmo de integridad seleccionado por el servidor durante el intercambio EAP-EXT. Cuando HMAC-SHA2-256 es usado como algoritmo de integridad, la longitud de la EAP-EXT-AUTH-KEY es 32 bytes.

EAP-EXT-ENC-KEY

Para dotar de confidencialidad a ciertos datos, EAP-EXT-ENC-KEY es usada para cifrar el contenido de los Encrypted TLVs. Se deriva de la MSK_i, la MSK generada por el último método EAP interno satisfactorio, usando la KDF definida en [34] de la siguiente manera.

EAP-EXT-ENC-KEY = prf+(MSK_i, 'EAP-EXT-Encryption-Key',length);

La PRF usada para generar EAP-EXT-ENC-KEY es determinada por el algoritmo de integridad seleccionado por el servidor EAP durante el intercambio EAP-EXT. El algoritmo hash por defecto para prf+ es PRF_HMAC_SHA2_256. El campo longitud dependerá del algoritmo de integridad seleccionado por el servidor durante el intercambio EAP-EXT. Por ejemplo, cuando AES-CBC-128 es usado, la longitud de la EAP-EXT-ENC-KEY es de 16 bytes.

3.5. Formato del Mensaje EAP-EXT

El formato de mensaje incluyendo los campos comunes de la cabecera EAP definidos en [6] es mostrados a continuación.

Figura 3.4: Estructura de mensaje EAP-EXT

En la parte de la cabecera específica para EAP-EXT se incluyen una serie de bits que permiten tanto a cliente como servidor averiguar cierta información durante la ejecución del protocolo. En concreto tenemos:

- F → Este bit tiene que ser activado para indicar que este es el último mensaje EAP-EXT del emisor. En cualquier otro caso, este bit no tiene que ser activado.
- E → Este bit es activado cuando el mensaje es una indicación de error. Cuando este bit es activado, el F-bit tiene que ser activado también. Acudir a la sección 3.3 para una descripción detallada sobre indicación de errores en EAP-EXT.

- Version → Este campo de 8 bits indica la versión del método EAP-EXT. Esta versión inicial del método define la versión 0.
- Reserved → Este campo de 6 bits es reservado para futuras extensiones. Este campo tiene que ser puesto a cero por el emisor y el receptor tiene que ignorar este campo.
- Capacidades → Este campo contiene capacidades EAP extendidas. El campo de capacidades tiene el siguiente formato.

Figura 3.5: EAP-EXT Capabilities

Cada bit del campo de capacidades posee un significado particular. La semántica de cada bit es la siguiente.

- R → Este bit es activado para indicar que el emisor soporta un mecanismo de re-autentiación rápida que describiremos en la sección 3.7.
- C → Este bit es activado para indicar que el emisor soporta el mecanismo de channel binding para la MSK. Cuando este bit es activado en un mensaje Request/EXT, un Channel-Binding-Mechanism TLV tiene que ser incluido también para indicar el mecanismo de channel binding soportado por el servidor. Si el peer soporta y desea permitir uno de los mecanismos de channel binding soportados por el servidor, envía un mensaje Response/EXT con este bit activado y un Channel-Binding-Mechanism TLV conteniendo el mecanismo de channel binding seleccionado. Si este bit es activado en el intercambio final de Request/EXT y Response/EXT con negociación satisfactoria de el mecanismo de channel binding y el método EAP-EXT finaliza su ejecución satisfactoria, el peer y el server tienen que activar el mecanismo de channel binding negociado.

Otros bits son reservados para uso futuro, y tienen que ser inicializados a cero por el emisor y ignorados por el receptor.

3.6. EAP-EXT TLVs

Un mensaje EAP-EXT puede contener cero, uno o más TLVs (Tipo-Longitud-Valor). El formato de TLV se puede observar en la Figura 3.6.

Figura 3.6: EAP-EXT TLV

- Tipo → Este campo indica el tipo del TLV.
- Longitud → Este campo indica la longitud de este TLV en bytes, incluyendo los campos tipo y longitud del propio TLV.
- Valor → Este campo contiene datos específicos para el tipo de TLV.

En la tabla 3.1 tenemos definidos los tipos de tlv para EAP-EXT.

Nombre	**Propósito**
Method TLV	Se emplea para transportar los mensajes generados por los métodos EAP internos de EAP-EXT. Contiene la carga útil de un mensaje EAP desde el campo tipo del método, sin incluir este último.
Auth TLV	Contiene datos de integridad usados para protección de los mensajes EAP-EXT frente a modificaciones maliciosas de terceros. Para calcularlo se hace uso de la EAP-EXT-AUTH-KEY. El campo valor del TLV es calculado sobre el mensaje EAP completo incluyendo este campo inicializado a 0. Cuando la comprobación de integridad falla, el mensaje tiene que ser descartado silenciosamente. El algoritmo de integridad por defecto es HMAC-SHA-256 [35].
Peer-ID TLV	Contiene el identificador único de cliente, Peer-Id[8], que describimos en la sección 2.2.3 que puede ser empleado en algunos procesos de reautenticación HOKEY[36]. En general se exportará en el formato FQDN.
Server-ID TLV	Contiene el identificador único de servidor, Server-Id[8], que describimos en la sección 2.2.3 que puede ser empleado en algunos procesos de reautenticación HOKEY[36]. En general se exportará en el formato FQDN.

Nombre	**Propósito**
PRF TLV	Se emplea durante la negociación de la función PRF a emplear. Contiene uno o más números PRF de 1 byte definidos en [34]. Cuando este TLV es transportado en un Request, indica el/los número(s) PRF soportado(s) por el servidor. Cuando este TLV es transportado en un mensaje Request/EXT, el correspondiente mensaje Response/EXT podría contener un número PRF exactamente soportado y seleccionado por el peer de entre los números PRF del mensaje Request/EXT. Si la negociación es fructuosa se emplea la PRF seleccionada, en cualquier otro caso, la PRF por defecto especificada en [34] es usado para la KDF.
Channel-Binding-Mechanism TLV	Contiene uno o más números de un byte referentes a mecanismos channel binding. Cuando este TLV es transportado en un mensaje Request/EXT, indica el/los número(s) de mecanismo de channel binding soportado(s) por el servidor. Cuando este TLV es transportado en un mensaje Request/EXT, el correspondiente mensaje Response/EXT podría contener este TLV. Un Channel-Binding-Mechanism TLV en un mensaje Response/EXT tiene que contener exactamente un número de mecanismo channel binding que es soportado y seleccionado por el peer de entre los números de mecanismo de channel binding del mensaje Request/EXT. Si el mecanismo de channel binding es negociado satisfactoriamente, el mecanismo de channel binding negociado es activado. Se soportan los mecanismos descritos en [23] y [24]. Cuando este TLV es transportado, se incluyen los datos de channel binding en un Channel-Binding-Data TLV.
Channel-Binding-Data TLV	Si en la negociación del mecanismo de channel binding se acordó el uso de la propuesta de [24], los parámetros de channel binding deben ser encapsulados en tlvs de este tipo en forma de cadena de bytes.

Nombre	Propósito
Encryption-Algorithm TLV	Permite la negociación de los algoritmos de encriptación empleados para cifrar los Encrypted TLVs. Cuando este TLV es transportado en un Request/EXT, indica los algoritmos criptográficos soportados por el servidor EAP. Cuando este TLV es transportado en un mensaje Request/EXT, el correspondiente mensaje Response/EXT podría contener este TLV. Un Encryption-Algorithm TLV en un mensaje Response/EXT tiene que contener exactamente un número de algoritmo de encriptación soportado y seleccionado por el peer de entre las opciones en el Encryption-Algorithm TLV contenido en el mensaje Request/EXT. Notar que un EAP Server podría obligar al EAP peer a usar el algoritmo de encriptación por defecto (AES-CBC-128). En tal caso, el servidor EAP no incluye el Encryption-Algorithm TLV en el mensaje Request/EXT y, de la misma forma, el peer EAP no lo incluye en el mensaje Response/EXT tampoco. Se definen inicialmente los siguientes algoritmos: AES-CBC-128, AES-CBC-256, 3DES, IDEA. Tener en cuenta que si los algoritmos no son negociados satisfactoriamente empleando Encryption-Algorithm TLV, el algoritmo de encriptación por defecto (AES-CBC-128) es usado en su lugar.
Integrity-Algorithm TLV	El servidor EAP puede seleccionar de entre diferentes algoritmos de integración y informar al peer EAP sobre esta selección a través del Integrity-Algorithm-TLV. Si el servidor EAP no incluye este TLV el valor por defecto es HMAC-SHA-256. Se definen los siguientes algoritmos: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256 (por defecto), HMAC-SHA-384, HMAC-SHA-512.
Encrypted TLV	Se emplea para ofrecer confidencialidad de los datos incluidos. Contiene uno o más TLVs cifrados con la EAP-EXT-ENC-KEY. La longitud de este campo depende del algoritmo de cifrado en uso. Incluye un campo 'IV' (Vector de inicialización) de bits aleatorios, el bit más significativo primero. La longitud de 'IV' depende de que algoritmo de encriptación se esté empleando. Hay que tener en cuenta que dependiendo del algoritmo de encriptación y la longitud del texto plano, podrían ser añadidos datos de relleno a los datos antes de la operación de cifrado.
Sequence TLV	Se emplea para mantener un número de secuencia en los intercambios EAP-EXT, evitando un ataque de reenvío. El número de secuencia comienza en 0 y llegará hasta $2^{32} - 1$, volviendo a cero en el siguiente envío. Este TLV es especialmente relevante para la solción de reautenticación rápida presentada en la sección 3.7.

Nombre	Propósito
Nonce TLV	Este TLV transporta un número pseudo-aleatorio de longitud variable, su utilidad se verá en la siguiente sección.

Tabla 3.1: Tabla de TLVs

3.7. Mecanismo de Re-autenticación Rápida en EAP-EXT

Algunos métodos especifican el uso de claves y un estado anteriores en las entidades involucradas en el proceso de autenticación con el fin de permitir una re-autenticación en un número menor de intercambios la próxima vez que sea necesario otra autenticación. Sin embargo, no todos los métodos proveen la posibilidad de un mecanismo de re-autenticación rápida con un número menor de intercambios. En general no es deseable tener que realizar un proceso de autenticación completo cada vez que un terminal actualmente autenticado y en posesión de material criptográfico válido, tenga que reautenticarse, bien por vencimiento de los tiempos de espera, por petición de la red o simplemente por movilidad del usuario de un autenticador a otro.

La latencia de la red entre el peer y el servidor EAP puede llegar a ser significativa pudiendo convertirse en una fuente de retardos problemática durante la re-autenticación en escenarios de roaming o en redes saturadas. Una forma de reducir esta latencia es reducir el número de intercambios necesarios para completarla.

Gracias a la jerarquía de claves mostrada en la sección 3.4.2 y las características propias del método propuesto, es posible la definición de un proceso de re-autenticación rápida a nivel de EAP-EXT que permita rebajar los tiempos de re-autenticación mediante la reducción del número de intercambios. Para desarrollar esta solución, poniendo como requisito haber completado una fase de autenticación previa, se procede a eliminar el proceso completo de autenticación anterior a la fase de binding en las siguientes reautenticaciones, siempre que el material criptográfico generado por EAP-EXT para uso interno (EAP-EXT-AUTH-KEY y EAP-EXT-ENC-KEY) sea todavía válido.

El proceso diseñado se encuentra descrito en la figura 3.7. Como es habitual el autenticador comienza la autenticación con el envío del mensaje EAP Request/Identity al cual el peer responde con su identidad al servidor a través del mensaje EAP Response/Identity. Cuando el servidor comprueba la identidad puede determinar si posee una clave EAP-EXT-AUTH-KEY de una ejecución anterior. Si es así puede comenzar el mecanismo de re-

autenticación rápida basada en EAP-EXT. Para ello el servidor inicia directamente la fase de binding del método EAP-EXT enviando un mensaje EAP-EXT/Request con el flag F activado indicando que se trata de la fase final del método. Este mensaje lleva un AUTH TLV creado a través de la clave EAP-EXT-AUTH-KEY que cliente y servidor poseen de una ejecución anterior.

Figura 3.7: Esquema de reautenticación rápida EAP-EXT

Cuando la solicitud anterior llega al peer, este, viendo el bit F activo concluye que comparte material criptográfico válido con el servidor AAA, comprueba el TLV AUTH y si la comprobación es satisfactoria responde con un Response con flag F activo, el número de secuencia, un número aleatorio y el TLV AUTH generado con la EAP-EXT-AUTH-KEY. Los números aleatorios comentados tanto para el Request como para el Response se envían dentro de un TLV Nonce. Si todo ocurrió de forma satisfactoria, el servidor EAP comprobará el TLV Auth con la misma conclusión que obtuvo el peer y generará una nueva MSK haciendo uso de los números aleatorios intercambiados, enviándola al autenticador junto con el EAP-Success.

Como se puede observar, los mensajes EAP-EXT deben incluir ahora un número de secuencia para evitar ataques de reenvío, donde un atacante podría reenviar un mensaje EAP-EXT de re-autenticación rápida de una ejecución anterior y por tanto obsoleto. Para evitar que algunas de las partes involucradas en el proceso de re-autenticación (peer y servidor) acepten mensajes antiguos, se incluye este número de secuencia. El número inicial, puede ser acordado durante la ejecución de EAP-EXT completa y preliminar al uso de este mecanismo de re-autenticación diseñado. Para ello se puede definir un nuevo TLV denominado Secuencia que contiene un número de secuencia iniciado aleatoriamente y que es protegido por el túnel constituido en la ejecución inicial de EAP-EXT.

Además se incluye un TLV nonce en el mensaje EAP-EXT Request que incluye un número pseudo-aleatorio generado por el servidor y otro TLV nonce en el mensaje EAP-EXT Response pero generado por el cliente. Esta parte de números N_{peer} y $N_{servidor}$ son empleados para nuevas claves MSK y EMSK a ser exportadas por el método EAP-EXT. En concreto se emplea el mismo formato de derivación descrito en la sección 3.4.2 pero incluyendo en el proceso de derivación, a través del campo de *datos opcionales*, los dos número aleatorios intercambiados.

3.7.1. Selección del servidor AAA correcto

Como decíamos, el proceso de re-autenticación rápida comienza con el intercambio de mensajes EAP Request y EAP Response Identity. El cliente debe indicar una identidad concreta que tiene un formato de *Network Access Identifier* (NAI) [37]. Este formato especifica una identidad de la forma *usuario@dominio*. Gracias a la parte dedicada al *dominio* la infraestructura AAA enruta los mensajes EAP al dominio de destino. Sin embargo, a nosotros nos interesa que ese mensaje llegue precisamente al servidor AAA que posee las claves EAP-EXT-AUTH-KEY y EAP-EXT-ENC-KEY. Ciertamente si el EAP Response/Identity llega a otro servidor , como éste no tiene ningún estado acerca del usuario empezará una autenticación EAP-EXT completa involucrando múltiples intercambios.

Por ello, hace falta especificar un dominio muy concreto que permita enrutar este mensaje al servidor AAA concreto. Basándonos en la solución aportada en [38], es posible considerar que cada servidor AAA pertenece a un dominio que sólo contiene a esa entidad. Por ejemplo el servidor AAA *aaaeap* del dominio *um.es* se encuentra en el subdominio *aaaeap.um.es*. Por tanto el cliente utilizando la identidad *usuario@aaaeap.um.es* referencia de forma unívoca qué servidor mantiene un estado criptográfico con él. Un ejemplo de uso de esta propuesta podemos verlo en la figura **??**

3.8. Consideraciones de seguridad

A la debilidad que supone un intercambio de capacidades inicial sin securizar, hay que añadir las debilidades propias de los métodos tunelados. EAP-EXT soluciona estas debilidades haciendo uso del material criptográfico derivado del exportado por sus métodos internos. En particular será la relación creada en dicho material la que evite los posibles ataques a este método.

3.8.1. Necesidad de la fase de confirmación

Puesto que EAP-EXT no dispone de material criptográfico hasta que un método interno no exporte al menos una clave, los intercambios realizados van desprotegidos. Puesto que estos intercambios iniciales puede llevar cierta información relevante, es posible que una entidad intermedia modifique algunos de estos valores sin que sea detectado ni por el peer ni por el servidor. Por tanto, se hace necesario confirmar los valores negociados durante esa fase sin protección, con un intercambio adicional de mensajes protegidos tras la creación del material criptográfico EAP. Esto es lo que se corresponde con la llamada fase de confirmación o binding de la propuesta.

3.8.2. Ataques en métodos tunelados

Ha sido reconocido [6] que existe una vulnerabilidad potencial a un ataque tipo *man-in-the-middle* cuando un método EAP se encapsula dentro de otro método EAP, como ocurre en el caso de aquellos tunelados. La figura 3.8 muestra el tipo de ataque mencionado.

En él, cuando el secuenciamiento de métodos de autenticación EAP está permitido, podría ocurrir que ni el peer ni el servidor tengan la seguridad de que la misma entidad ha participado en la ejecución de todos ellos. Por ejemplo puede ocurrir que un autenticador inicie un método tunelado con un servidor donde sólo se autentica a este último (p.ej. PEAP). Una vez finalizado el túnel seguro (donde sólo el servidor está autenticado), el servidor solicita del atacante que se autentique con otro método transportado dentro del túnel. El atacante entonces induce a un peer que tiene credenciales válidas para ejecutar ese método interno, a iniciar con el atacante (actuando ahora de autenticador) dicho método. El atacante entonces recoge los paquetes enviados por el peer y los encapsula a través del túnel ya establecido entre el atacante y el servidor. El método interno finaliza con éxito y el servidor termina el proceso enviando un EAP success y dando acceso al servicio, incluso cuando el atacante no fue el que realmente se autenticó sino otro peer. La situación es todavía peor porque en ciertos casos, el material criptográfico para crear el túnel es el que se utiliza para acceder a otros servicios. Desafortunadamente, ese material ya está disponible al atacante que inició el túnel y que obtuvo sin necesidad de ninguna autenticación previa (recordemos que sólo hay autenticación de servidor)

Para evitar este tipo de ataque es necesario relacionar de alguna forma los métodos EAP ejecutados en el interior del método y el método EAP externo de transporte. Para alcanzar este objetivo, EAP-EXT emplea el material criptográfico del método interno para crear una clave que proteja los datos

del externo. De esta forma se crea una relación criptográfica entre el método interno y el método externo. De hecho, aunque un atacante pueda inducir a otro peer a ejecutar el método interno, no dispondrá de la clave exportada por dicho método y por tanto no podrá generar el AUTH TLV para el método externo, que en nuestro caso es EAP-EXT.

Figura 3.8: Ataque de un autenticador malicioso con método tunelado

3.9. Conclusiones

Existía la necesidad de un mecanismo de bootstrapping genérico, dinámico y extensible. El diseño mostrado cumple con estas tres características gracias a ser un método EAP debido a que no necesita modificar los estándares existentes relacionados con esta tecnología. Además y como se indicó en la sección anterior, se han solucionado los posibles problemas de seguridad inherentes a su condición de método tunelado. Al día de hoy existen otras alternativas de boostraping basadas en métodos EAP tunelados, pero que han mostrado ciertas carencias como se ha explicado al inicio de este capítulo. Recordemos por ejemplo que obligan al establecimiento del túnel utilizando un mecanismo de autenticación específico.

El mecanismo diseñado permite el bootstrapping empleando para el proceso de autenticación cualquiera de los métodos EAP existentes capaces de

generar material criptográfico. Además, este paso de autenticación se realiza siempre antes del establecimiento de un túnel seguro para el intercambio de información de bootstrapping. Por otra parte, el método sigue fielmente las indicaciones del EAP-KMF[8] en cuanto a la exportación de material criptográfico y parámetros realcionados. Por ejemplo, se exporta la clave EMSK, muy útil en iniciativas como las propuestas dentro del grupo HOKEY.

De forma complementaria, se ofrece un mecanismo de reautenticación rápida que aprovecha el material criptográfico generado por la autenticación previa para generar nuevo material criptográfico válido sin necesidad de ejecutar ningún método en su interior. La unión de este mecanismo con el mecanismo de HOKEY reducen el tiempo necesario para la reautenticación de forma drástica, mitigando los problemas existentes durante el handoff en entornos multidominio.

Todos estos características se han logrado cumpliendo una de las principales premisas, la no modificación del estándar EAP, factor vital para la futura aceptación de esta solución.

Capítulo 4

Implementación

4.1. Análisis del entorno de desarrollo OpenDiameter

OpenDiameter es la implementación de código abierto del protocolo Diameter basado en el RFC 3588 diseñado por el AAA Working Group del IETF (Internet Engineering Task Force). Su última versión es la 1.0.7-i, si bien para este proyecto se empleó la versión 1.0.7-h debido a que era la versión vigente cuando se comenzó a implementar la solución.

De entre los dos grandes protocolos AAA, Radius y Diameter, creimos conveniente dar soporte a nuestro método, al más novedoso de ellos, Diameter. Diameter está llamado a ser el futuro de las arquitecturas AAA , mientras que Radius es un protocolo parcheado y en declive. Además OpenDiameter es un framework completamente orientado a objetos que emplea C++ y multiplataforma, frente a FreeRadius que emplea programación estructurada en C.

El código fuente del software OpenDiameter está disponible bajo una combinación de licencias públicas 'LGNU' y 'GNU'. La implementación del protocolo base como librería de C++ está disponible con soporte para los sistemas Linux, FreeBSD y Microsoft Windows. Sobre la librería ACE recae proveer la abstracción a nivel de sistema para los entornos soportados.

4.1.1. La librería ACE

El 'Adaptative Comunication Environment(ACE)' es un toolkit orientado a objetos que implementa patrones de diseño fundamentales para software de comunicaciones. ACE provee de un conjunto rico de wrappers c++ reutilizables y entornos de trabajo que implementan software común de comuni-

caciones a lo largo de múltiples sistemas operativos. Una única estructura común de clases fuente es empleada para todas estas plataformas, por lo que es relativamente sencillo portar ACE a otras nuevas.

Los componentes de comunicación software provistos por ACE incluyen tratamiento de eventos, inicialización de servicios, comunicación interprocesos, etcétera. Existen versiones de ACE para c++ y Java.

ACE está destinado a desarrolladores de servicios de comunicación y aplicaciones de alto rendimiento y tiempo real para UNIX, POSIX y WIN32. ACE simplifica el desarrollo de aplicaciones y servicios de red orientados a objetos que empleen comunicación interprocesos, manejo de eventos, linkado dinámico y concurrencia. ACE automatiza la configuración y reconfiguración del sistema enlazando los servicios con aplicaciones en tiempo real y ejecutando dichos servicios en uno o más procesos e hilos.

ACE está siendo empleado actualmente en proyectos y productos comerciales de docenas de compañías incluyendo Ericsson, Siemens o Motorola, entre otras.

4.1.2. EAP y PANA

Una implementación de EAP se encuentra disponible para el proceso de autenticación. También se provee una implementación de PANA para la red de acceso como lower layer. La pila de protocolos de EAP y PANA se provee en forma de librería para su posible exportación. Los APIs provistos por esas librerías pueden ser empleadas para integrar estas pilas de protocolos en las aplicaciones apropiadas. Ambas librerías implementadas bajo la plataforma OpenDiameter han sido diseñadas para que puedan interaccionar sin problemas. Tienen pequeños comentarios por el código y son simples de integrar. Actualmente la pila de EAP soporta los siguientes métodos:

- EAP-MD5
- EAP-Archie
- EAP-GPSK
- **EAP-TLS**

A los que se sumará en breve EAP-EXT. Hay que tener en cuenta que el objetivo de OpenDiameter no es dar soporte a EAP y PANA directamente, este soporte es el resultado de la necesidad que existe en muchos escenarios de integrar Diameter con estos dos protocolos. EAP sobre PANA puede ser empleado en la red de acceso de cliente mientras que Diameter se emplea como soporte AAA.

4.1.3. Librerías contenidas en la última revisión

- libdiamparser → Librería de análisis sintáctico de mensajes Diameter, con soporte para diccionario en XML y capacidad de análisis de tipos de AVP definidos por el usuario.
- libdiameter → Núcleo Diameter con soporte para accounting.
- libdiametereap → Librería de aplicación Diameter EAP.
- libdiameternasreq → Librería de aplicación Diameter NASREQ.
- libeap → Librería EAP.
- libpana → Librería PANA con soporte IPv4 e IPv6 para Linux, FreeBSD y Microsoft Windows.

4.1.4. Especificaciones soportadas

- Diameter Base Protocol ← RFC3588
- EAP ← RFC3748
- Diameter EAP ← RFC4072
- EAP State Machine ← RFC4137
- EAP-Archie [39]
- EAP-TLS ← RFC2716
- PANA protocol [3]

4.2. Integración de EAP-EXT en OpenDiameter

4.2.1. Familiarización con el entorno

Toda la implementación de la librería EAP de OpenDiameter parte de una clase base llamada StateMachine que, como su nombre indica, hace referencia al concepto de máquina de estados. Estas máquinas de estados aceptan eventos y se ejecutan cada una en un hilo independiente.

En el caso particular de EAP, tendremos siempre en cada entidad una MethodStateMachine y una SwitchStateMachine. Para entender la explicación posible asemejaremos las clases que intervienen a una 'pila de protocolos' que ofrecen servicios unos a otros. La MethodStateMachine es la encargada de realizar todo el trabajo referente al algoritmo de autenticación, mientras que la SwitchStateMachine es la encargada del nivel EAP y de comunicarse con las capas inferiores dando soporte a la MethodStateMachine. Esta estructura se puede observar en la Figura 4.1.

Figura 4.1: EAP

Envío y recepción de mensajes

Vamos a centrarnos en el envío y recepción de mensajes abstrayéndonos de conceptos de sincronización y otras curiosidades propias del framework. Los mensajes se encapsulan en objetos de tipo AAAMessageBlock utilizados por todo el framework para la representación de mensajes de red. Para enviar un mensaje desde una MethodStateMachine se prepara este y se deposita en el objeto correspondiente a su SwitchStateMachine. Seguidamente se le envía una notificación de evento de tipo ValidReq o ValidResp dependiendo de si nos encontramos en el autenticador o en el peer. La MethodStateMachine es la encargada de reservar el tamaño completo del mensaje EAP incluyendo la cabecera. Antes de notificar el envío a su correspondiente SwitchStateMachine se debe formatear la parte de cabecera correspondiente al método, si

es que este tuviese zona de cabecera propia. En el caso de EAP-EXT esta parte serían la versión, los flags y las capabilities.

El proceso de recepción sigue el procedimiento inverso al anteriormente mostrado. Cuando la SwitchStateMachine recibe un mensaje de la capa inferior recibe a su vez un evento EvRxMsg. Este evento lanza el proceso de análisis del paquete. En este punto se comprueba que el tipo y el método transportado son correctos para los métodos ejecutándose sobre esa SwitchStateMachine. Esta se lo proporciona a la MethodStateMachine y le envía un evento IntegrityCheck hacia arriba que activa la MethodStateMachine la cual debe encargarse de la carga útil del mensaje EAP. Dentro de esta carga útil se encuentra la parte específica de la cabecera.

Todos estos eventos que hemos dicho que se envían entre máquinas de estados generan transiciones en la máquina. Estas transiciones se traducen en acciones que son las encargadas de proporcionar el efecto deseado. Podemos tener varios tipos de MethodStateMachine, implementando distintos métodos EAP, que deben ser instaladas para que la SwitchStateMachine las instancie llegado su momento. No todas las MethodStateMachine implementadas en el framework tienen que estar instaladas, como es lógico se instalarán solamente las correspondientes a los métodos que se vayan a emplear en el proceso de autenticación.

4.2.2. Encapsulando otro método

El primer objetivo de EAP-EXT es encapsular otro método EAP de forma transparente tanto para el método como para la capa inferior. Vamos a explicar el proceso seguido para encapsular EAP-TLS, que es ampliamente utilizado en despliegues en entornos de producción.

Lo primero que tenemos que hacer es instalar la TLSMethodStateMachine. Hay que tener en cuenta que todo este proceso se deberá hacer por duplicado pues las PeerMethodStateMachine y las AuthMethodStateMachine son diferentes. En el caso del autenticador además existe una clase de SwitchStateMachine para cada una de las formas de funcionamiento del autenticador, passthrough, backend,... que tendremos que implementar. Para esta demo sólo se ha implementado la opción StandAlone. Así tenemos que hasta este punto tenemos que instalar cuatro MethodStateMachine:

- EapPeerEXTStateMachine → MethodStateMachine correspondiente al Peer EAP-EXT
- EapAuthEXTStateMachine → MethodStateMachine correspondiente al autenticador EAP-EXT

- InnerEapPeerTlsStateMachine → MethodStateMachine correspondiente al Peer EAP-TLS encapsulado
- InnerEapAuthTlsStateMachine → MethodStateMachine correspondiente al autenticador EAP-TLS encapsulado

Necesitamos dos máquinas de tipo SwitchStateMachine para ese nivel EAP antes comentado. Estas dos serán las encargadas de dar soporte a EAP-EXT:

- EapStandAloneAuthSwitchStateMachine → SwitchStateMachine para el autenticador EAP-EXT
- EapPeerSwitchStateMachine → SwitchStateMachine para el Peer EAP-EXT

El siguiente paso es lanzar el método EAP-EXT. Recordar que hasta ahora se ha hablado de **instalar** MethodStateMachine pero en ningún momento se han instanciado. De este instanciamiento se ocupan las SwitchStateMachine. Al lanzarse el entorno comienza la ejecución de las SwitchStateMachine que se encargan de instanciar sus MethodStateMachine correspondientes. Hasta el momento tenemos dos SwitchStateMachine correspondientes a EAP-EXT, por tanto será este método el instanciado mientras que **EAP-TLS queda instalado sin instanciar**. Al lanzarse EAP-EXT debe encargarse de instanciar la InnerMethodStateMachine correspondiente, en este caso EAP-TLS. Como ya hemos dicho la instanciación de los métodos la realizan las SwitchStateMachine. Llegados a este punto tenemos dos claras opciones, herencia o clientela. Se ha optado por la opción de clientela por las siguientes motivaciones. EAP-EXT es un método más, que si bien se puede ver conceptualmente como una capa intermedia virtual entre los métodos transportados y la capa inferior. Realmente esto no es del todo cierto, pero puede servir para entender mejor la relación entre EAP-EXT y el resto de métodos, ya que un servidor de autenticación que no seleccione EAP-EXT como mecanismo de autenticación puede emplear otro método haciendo que esa capa intermedia virtual desaparezca. La estructura de capas la podemos observar en la Figura 4.2

Además, viendo la estructura de Figura 4.3, vemos que EapMethodStateMachine y EapSwitchStateMachine, son completamente distintas. EapSwitchStateMachine tiene parte de orientación hacia 'tarea' al heredar de EaJobMultiplexor. Esta herencia múltiple puede crear problemas en la relación de Eap[Auth/Peer]EXTStateMachine con su correspondiente SwitchStateMachine, pues esta última espera una relación de clientela con una MethodStateMachine y nunca con una SwitchStateMachine. Nos decidimos por tanto por la relación de clientela. Para ello necesitaremos dos nuevas SwitchStateMachine que den soporte a nuestras dos InnerEap[Auth/Peer]TlsStateMachine,

Figura 4.2: Arquitectura de capas de EAP incluyendo a EAP-EXT

Figura 4.3: EAP-EXT Framework

teniendo claro que estas dos nuevas SwitchStateMachine no deben entrar en contacto directo nunca con el nivel EAP. De esta forma se crea una SwitchStateMachine interna en cada MethodStateMachine de EAP-EXT como se puede observar en la Figura 4.3, una para peer y otra para autenticador.

- InnerPeerSwitchStateMachine → SwitchStateMachine encargada de dar soporte al método interno en la parte del peer.
- InnerStandAloneAuthSwitchStateMachine → SwitchStateMachine encargada de dar soporte al método interno en la parte del autenticador.

Ahora crearemos dos nuevas MethodStateMachine internas que hereden de las dos MethodStateMachine que se instalaron en el inicio de esta explicación. Esto es así ya que ambas son abstractas y para poder ser instanciadas es necesario redefinir algunos de sus métodos, por tanto estas dos definiciones podrían ser innecesarias en el caso en que el método se defina como una clase final no virtual. Estas dos últimas MethodStateMachine serán las aportadas a las dos InnerSwitchStateMachine para ser instanciadas (de ahora en adelante InnerSwitchStateMachine se referirá tanto a la InnerStandAloneAuthSwitchStateMachine como a la InnerPeerSwitchStateMachine e InnerMethodStateMachine hará referencia tanto a la InnerEapPeerTlsStateMachine como a la InnerEapAuthTlsStateMachine, por su parte haremos referencia a EapPeerEXTStateMachine y EapAuthEXTStateMachine como MethodStateMachine). En general y como se ha querido mostrar en la Figura 4.3 con el color verde, podremos crear una InnerEapPeerXXXStateMachine y una InnerEapAuthXXXStateMachine para cualquiera que fuese un método XXX deseado, sin olvidar nunca que debe ser un método capaz de derivar material criptográfico.

Figura 4.4: Relación de EAP-EXT con su InnerMethod

En la Figura 4.4 podemos ver la relación descrita entre EAP-EXT y el método encapsulado. Se puede observar como la InnerSwitchStateMachine tiene una relación bidireccional con la MethodStateMachine. Esto es así porque si bien es la MethodStateMachine la encargada de instanciar a la InnerSwitchStateMachine, es esta última la recibe como parámetro una instancia de la MethodStateMachine. La razón de esta decisión es la necesidad de que la InnerSwitchStateMachine haga de puente entre la MethodStateMachine y la InnerMethodStateMachine.

Como ya se ha comentado el objetivo es que la MethodStateMachine se comporte de forma transparente a ojos de la InnerMethodStateMachine. Para ello es necesario que la InnerMethodStateMachine pueda enviar eventos hacia su InnerSwitchStateMachine, depositar mensajes y recibirlos. Esta InnerSwitchStateMachine no es necesaria para otra función que para dar transparencia al proceso. Es por esto por lo que hemos comentado que realiza funciones de puente, reenvía y recibe eventos y mensajes hacia/desde la MethodStateMachine. Por tanto podemos decir que la InnerSwitchStateMachine queda en pausa y es la InnerMethodStateMachine la que se relaciona con la MethodStateMachine.

Como podemos observar en la Figura 4.5, como capa intermedia nuestra MethodStateMachine tiene dos capas directamente relacionadas con ella. Por un lado tiene la InnerSwitchStateMachine y por otro su propia SwitchStateMachine.

Figura 4.5: Modelo conceptual

Llegados a este punto podemos comenzar a hablar de los eventos inter

máquina de estados que van a ser de relevancia para conseguir encapsular el método. Durante el envío, cualquier MethodstateMachine deposita su mensaje en su SwitchStateMachine y le envía el evento EvSgValidResp o EvSgValidReq dependiendo de si nos encontramos en el autenticador o en el peer. Durante la recepción, la MethodStateMachine espera que su SwitchStateMachine haya depositado el mensaje y recibir un EvSgIntegrityCheck de esta última. Estos son pues los eventos que nos interesan en esa relación observada en Figura 4.5 entre nuestra MethodStateMachine y nuestra InnerMethodStateMachine. Por tanto son estos eventos los que debemos reenviar hacia y desde la MethodStateMachine a través de la InnerSwitchStateMachine. Para ello se debe modificar en esta última los métodos Notify y Event, para que al ser invocados realicen una llamada a Notify o Event de la MethodStateMachine respectivamente. Además será necesario proporcionar métodos de acceso a las zonas de la InnerSwitchStateMachine donde son depositados los mensajes desde/hacia la InnerMethodStateMachine. Con esto habremos logrado superar los escollos para realizar el tunelado del método interno de manera transparente a este.

Código fuente

A continuación se van a mostrar algunas partes del código de vital importancia para realizar la funcionalidad requerida.

El siguiente código pertenece a la InnerSwitchStateMachine y es empleado para reenviar los eventos y recibir los mensajes. El que sigue es un ejemplo de cómo redefinir el Event para que realice la funcionalidad requerida.

```
void InnerPeerSwitchStateMachine::Event(AAA_Event event)
//Event from inner-SSM to EXT-MSM
  {
   switch(event)
    {
    case EapPeerSwitchStateMachine::EvSgValidReq:
    if(Decision() == EapPeerSwitchStateMachine::UNCOND_SUCC)
    {
      MethodStateMachine().IsDone() = true;
      extmsm->Event(EapPeerEXTStateMachine::EvSgInnerEndMethod);
    }
    else
    extmsm->Event(EapPeerEXTStateMachine::EvSgInnerSend);
  break;
    }
  }
```

La variable extmsm (ext method state machine) es la referencia cruzada entre EAP-EXT y la InnerSwitchStateMachine, está definido de la siguiente manera:

```
void setOuterPeerEXTStateMachine(EapPeerEXTStateMachine &arg)
  {
   extmsm = &arg;
  }
private:
  EapPeerEXTStateMachine *extmsm;
```

En cuanto al envio y recepción de mensajes, a parte del evento necesario para notificar de la llegada o envío, tan sólo es necesario ofrecer métodos de acceso a la zona de memoria destinadas a realizar la función de buffer. Estos métodos, SetRxMessage(msg) y msg *GetTxMessage(), ya se encuentran implementados y son empleados por el nivel inferior a toda SwitchStateMachine para realizar el envío y la recepción. Podemos emplearlos sin ningún problema.

Otro detalle importante de la implementación es la necesidad de invocar Eventos de la SwitchStateMachine para arrancarla y llevarla a un estado consistente, para ello se han definido los métodos OldEvent que invocan la implementación original del método Event, redefinida anteriormente. Como ya hemos comentado, todo cambio realizado para los Event debe también ser realizado para los Notify.

```
void OldEvent(AAA_Event event)
//Inherited from EapPeerSwitchStateMachine
  {
   EapPeerSwitchStateMachine::Event(event);
  }
```

Gracias a esto se consigue arrancar la InnerSwitchStateMachine y dejarla en un estado estable para ofrecer esa transparencia a la InnerMethodStateMachine que ya se describió. Veamos el ejemplo de como el autenticador EAP-EXT durante su arranque, lanza la InnerStandAloneAuthSwitchStateMachine

```
 void Start() throw(AAA\_Error)
  {
    EapStateMachine<EapAuthEXTStateMachine>::Start();
    ///Launch inner method
    eap->setOuterAuthEXTStateMachine(*this);
    eap->Start(); ///starts TLSMethod
```

```
    ////////////////////////////
    eap->OldEvent(EapAuthSwitchStateMachine::EvSgRestart);
    eap->OldEvent(EapAuthSwitchStateMachine::EvSgPortEnabled);
    eap->OldEvent(0);
    ////////////////////////////
  }
```

Por último y debido a que el parser de algunos métodos implementados y en particular EAP-TLS emplean el campo length de la cabecera EAP para conocer la longitud del mensaje, en la recepción de un mensaje es necesario introducir una cabecera EAP dummy para introducir este campo.

```
/* ************************************************************
***            PassThrough Inner Package Formating          ****
************************************************************ */
AAAMessageBlock * packetEXTtoInner(EapExtTLV *tlvtls)
{
AAAMessageBlock *pasadatos =
                  AAAMessageBlock::Acquire(tlvtls->getLength()-4+5);
//-4 tlv header, +5 EAPHeader
bzero(pasadatos->base(),(tlvtls->getLength()-4+5));
*((ACE_UINT16*)&pasadatos->base()[2]) =
                  htons((tlvtls->getLength()-4+5));
//Introduce length in dummy EAP Packet
     memcpy(&pasadatos->base()[5],
                  tlvtls->getValue(),
                  tlvtls->getLength()-4);
pasadatos->rd_ptr(5); //Go through the EAP Header
return pasadatos;
}
```

4.2.3. Fase de binding

Puesto que se ha implementado únicamente la opción de tener un método interno en ejecución, será necesario un intercambio más para alcanzar nuestro objetivo. Para ello, tan sólo es necesario enviar dos mensajes, uno desde el autenticador y otro desde el peer empleando el mecanismo antes narrado.

No obstante esta fase implica que hayamos obtenido la MSK del método interno y hayamos derivado las claves con la función prf+. Gracias a la clase PRF_Plus importada del proyecto OpenIKEv2[40], haremos la derivación de claves. Tras esto, se calcula el resumen de cada uno de los mensajes adjuntándole un Auth TLV inicializado a cero.

Código fuente

Con la función KeyDerive podemos derivar la MSK, EMSK, EAP-EXT-AUTH-KEY y EAP-EXT-ENC-KEY. Por su parte AuthenticateMsg inicializa los últimos 32 bytes de un mensaje a 0, realiza el cálculo de su resumen para una clave dada y lo introduce en las posiciones inicializadas con anterioridad. Notar que para simplificar la implementación se ha supuesto que el Auth TLV siempre estará en último lugar.

```
static void hexdump(std::string title,const char *data, int size);

static std::string *KeyDerive(std::string key,std::string s, short length)
{
char *output = new char[length];
PrfPlus prf_plus(PRF_HMAC_SHA256);
//Block output size PRF_HMAC_SHA256 is 32bytes
short iterations = length/32;
prf_plus.PRF_plus(iterations,key.c_str(),key.size(), s.c_str(), s.size(), output);
std::string *newkey = new std::string((const char *)output,length);
return newkey;
}

static AAAMessageBlock *AuthenticateMsg(std::string key, AAAMessageBlock *msg)
{
hexdump("Auth Message received",msg->base(),msg->size());
bzero((msg->base()+msg->size() - 32),32); //zero 32 ending bytes
AAAMessageBlock *result = AAAMessageBlock::Acquire(msg->size());
memcpy(result->base(),msg->base(),(msg->size() -32));
unsigned char *hmac_result = new unsigned char[32];  //32bytes -> sha256
bzero(hmac_result,32);
unsigned int size = 32;
HMAC(EVP_sha256(),key.c_str(),key.size(),
                  (unsigned char *)msg->base(), msg->size(),
                  hmac_result,&size);
hexdump("hmac_result",(const char *)hmac_result,32);
memcpy((result->base()+result->size() - 32),hmac_result,32);
//Copy hmac_result in last 32 bytes from result
hexdump("Auth Message",result->base(),result->size());
return result;
}
```

A continuación podemos ver un ejemplo de derivación de la MSK/EMSK.

```
std::string *MSKEMSK = KeyDerive(tls->MSK(),"EAP-EXT-EAP-Keying-Material", 12
```

A continuación vemos un ejemplo de como generar el mensaje autenticado correspondiente al binding.

```
char *datosnull = new char[32]; // 32 -> HMAC256
bzero(datosnull,32);
EapExtTLV *tlvAUTH = new EapExtTLV(AUTH_TLV,32,datosnull);
AAAMessageBlock *AUTHENC = tlvAUTH->parseTLVtoRaw();
msg = AAAMessageBlock::Acquire(8 + AUTHENC->size());
ACE_OS::memset(msg->base(),0, 8 + AUTHENC->size());
AAAMessageBlock *authmsg = AuthenticateMsg(msm.AUTHK(),AUTHENC);
```

4.2.4. Máquinas de estados EAP-EXT

Autenticador

Comenzaremos por el autenticador ya que es este el que inicia el proceso de authenticación que se ve reflejado en la Figura 4.6.

Figura 4.6: Máquina de estados EapAuthEXTStateMachine

1. La máquina se encuentra inicialmente en StInitialize.
2. Recibe un evento EvSgIntegrityCheck de su SwitchStateMachine, que indica que se inicia el proceso de autenticación, pasando a el esta-

do StWaitInner y lanzando la acción AcPruebaMaquina. Durante AcPruebaMáquina la EapAuthEXTStateMachine se encarga de inicializar, a través de la InnerStandAloneAuthStateMachine, a la InnerEapAuthTlsStateMachine. Finalmente se llega al estado StWaitInner que como su nombre indica queda a la espera de que EAP-TLS inicie el proceso de autenticación.

3. Cuando la InnerEapAuthTlsStateMachine realiza un envío de mensaje, envía hacia su InnerStandAloneAuthSwitchStateMachine el evento EvSgValidResp indicando que ha enviado un mensaje. Este evento es traducido por la InnerStandAloneAuthSwitchStateMachine a nuestra EapAuthEXTStateMachine como EvSgInnerSend. Es en este punto donde se ejecuta el cambio de estado de la EapAuthEXTStateMachine hacia el estado StWaitResponse y se ejecuta la acción AcPrepareRequest. En AcPrepareRequest se recoge el mensaje generado por la InnerEapAuthTlsStateMachine, se deposita en la InnerStandAloneAuthSwitchStateMachine y se encapsula para su envío hacia el peer. Es en este punto donde se debe instalar la cabecera EAP y EAP-EXT y darle los valores para su envío. Para ello se colocará el mensaje en la EapStandAloneAuthSwitchStateMachine y se le enviará un evento EvSgValidResp. Observese que este proceso es idéntico al realizado por la InnerEapAuthTlsStateMachine.

4. Cuando la respuesta del peer llegue al autenticador la EapStandAloneSwitchStateMachine depositará el mensaje y enviará el evento EvSgIntegrityCheck, que provoca la ejecución de AcRecibir. EapAuthEXTStateMachine, recoge el mensaje, comprueba su cabecera, la elimina e introduce una cabecera estándar para EAP-TLS. Esto es así porque para la comprobación de la integridad del mensaje en la implementación de EAP-TLS se empleó el tamaño provisto por la cabecera EAP como tamaño del mensaje, este escollo es por tanto fácilmente subsanable para una posible implementación final. Se deposita el mensaje en la InnerStandAloneAuthSwitchStateMachine y se envía un mensaje de tipo IntegrityCheck a la InnerEapAuthTlsStateMachine para que lo recupere.

5. Los pasos 3 y 4 se repetirán hasta que la InnerEapAuthTlsStateMachine termine el proceso de autenticación y genere la MSK. Para detectar si la InnerEapAuthTlsStateMachine ha finalizado o no se hace uso de un método en la InnerStandAloneAuthSwitchStateMachine que comprobará que esta se encuentra en estado Success y que su InnerEapAuthTlsStateMachine correspondiente ha terminado. En este mo-

mento termina la función de tunelado de EAP-EXT con la ejecución de AcInnerEnd. Ahora es el momento de recuperar la MSK y derivar las claves propias. Por medio del método MSK de la InnerEapAuthTlsStateMachine podremos recuperar la MSK generada por EAP-TLS. Se pasa al proceso de derivación de claves. Primero se deriva una clave de 128bytes que pasará a ser dividida en dos subclaves de 64 bytes (EMSK y AMSK), acto seguido se deriva la EAP-EXT-Authentication-Key de 32bytes para la firma de los AUTH-TLV. En esta misma acción se prepara el mensaje de Request autenticado con un AUTH-TLV firmado por la clave recién generada y se pasa a el estado StAuthEXT.

6. Al igual que ocurrió en el paso 4, al recibir el Response de parte del Peer nuestra EapStandAloneSwitchStateMachine nos notificará con un evento de tipo EvSgIntegrityCheck, generandose el cambio de estado hacia StAuthExtResponse y lanzandose la acción AcRecibirExtResponse. Durante esta última se comprobará la firma del AUTH-TLV y se pasará a esperar y se notificará el Success a si misma.

7. La recepción del EvSgSuccess realizará la transición hacia el estado StSuccess ejecutando la acción AcNotifySuccess, que mediante la notificación de un EvSgValidResp a la EapStandAloneAuthStateMachine de un mensaje vacío y el status de finalizada indicará a esta última que tiene que enviar el mensaje EAP-Success vacío.

Peer

El caso del Peer es muy similar y a continuación se explicarán las diferencias con respecto a la parte del autenticador. Como es lógico todas las máquinas de estados que han intervenido en el proceso detallado anteriormente son ahora sus homónimas en el Peer:

- EapAuthEXTStateMachine → EapPeerEXTStateMachine
- EapAuthStandAloneSwitchStateMachine → EapPeerSwitchStateMachine
- InnerStandAloneAuthSwitchStateMachine → InnerPeerSwitchStateMachine
- InnerEapAuthTlsStateMachine → InnerEapPeerTlsStateMachine

También hay que tener en cuenta que, puesto que es el autenticador el que arranca el proceso de autenticación, el peer debe ser arrancado después de

Figura 4.7: Máquina de estados EapPeerEXTStateMachine

este punto. Por tanto en el paso 1 de la máquina de estados del peer en lugar de pasar a esperar a la InnerPeerTlsStateMachine, pasaremos a recuperar el mensaje de la EapPeerSwitchStateMachine que habrá llegado desde el autenticador y se lo proporcionaremos a la InnerEapPeerTlsStateMachine por medio de su InnerPeerSwitchStateMachine. Es ahora cuando la máquina de estados del Peer pasa a esperar a su InnerMethodStateMachine. En cuanto a la última transacción de la máquina de estados con la ejecución de AcSendExtResponse, no solo se realiza la acción de enviar el Response al autenticador, sino que también se notifica a la EapPeerSwitchStateMachine la finalización del proceso de autenticación, delegando a esta la recepción del EAP-Success enviado en última instancia por el autenticador.

4.3. Despliegue de EAP-EXT en un escenario de acceso a la red

La implementación realizada para este proyecto es ejecutable, habiendo sido probada en un entorno típico de autenticación con 3 máquinas virtuales tomadas del testbed del proyecto Daidalos2[10].

La comunicación de las máquinas virtuales se ha realizado mediante interfaces tunctl virtuales con un uml-bridge, lo que ha permitido en una única captura tener todas las tramas del proceso. En cuanto a los tiempos comentar que las ejecuciones se han realizado vía conexiones ssh. Debido a problemas

Característica	Valor
Procesador	Intel Core Duo 1600Mhz
Memoria	2GB
Sistema Operativo	Ubuntu Linux 7.04 (Linux version 2.6.20-15-generic)

Figura 4.8: Host anfitrión

Característica	Valor
Procesador	1600Mhz
Memoria	256MB
Sistema Operativo	Ubuntu Linux 6.06LTS (Linux version 2.6.15-26-i386)

Figura 4.9: Máquinas virtuales

con el refresco de pantalla que se han filtrado para mostrar en este documento, se aprecian grandes lapsos de tiempo entre transferencia.

Para estas pruebas se ha optado por un escenario típico donde el servidor de autenticación y el autenticador se comunican sobre Diameter y Autenticador y Peer se comunican vía PANA. La ejecución se detiene en el momento en que se ha autenticado al peer con el mensaje de EAP-Success y la generación o transporte de las claves por parte de cada una de las partes. Este escenario y sus intercambios se puede observar en la Figura 4.10.

Para poder integrar EAP-EXT en el escenario, son necesarias dos modificaciones relevantes. La primera es la necesidad de instalar las method state machines tanto de EAP-EXT como de EAP-TLS en las aplicaciones ejecutando EAP. De manera adicional es necesario configurar tanto al servidor EAP como al peer para que instancien EAP-EXT para el usuario virtual empleado para la prueba. El autenticador encargado de hacer la conexión Diameter-PANA no es necesario modificarlo como ya se indicó en los capítulos anteriores. Como ya se ha indicado en esta sección, EAP-TLS es instanciado por la MethodStateMachine de EAP-EXT, pero para ello es necesario que se encuentre instalado. Esto nos ofrece la posibilidad además de ejecutar EAP-TLS en el mismo escenario con la misma configuración simplemente indicando a peer y server que ahora la autenticación se realizará con EAP-TLS.

4.4. Análisis

Como se puede apreciar tanto en las Figuras 4.6 y 4.7 y como ya se ha comentado, tenemos dos partes diferentes en las máquinas de estados. La primera parte es la referente al encapsulado de EAP-TLS con un bucle y obtención de la MSK para la derivación de las claves. La segunda es el inter-

Figura 4.10: Escenario de pruebas

cambio EAP-EXT autenticado. Además, ajeno a nuestra implementación y por medio de la herencia del framework general, se realizan los intercambios de identity y el EAP-Success. Este proceso se puede ver reflejado en la Figura 4.11 en un escenario típico con EAP-EXT encapsulando EAP-TLS.

4.4.1. EAP-TLS encapsulado

En la Figura 4.12 podemos ver un escenario EAP ejecutando EAP-TLS como método de autenticación. Todo este intercambio es el que se representó en la Figura 4.11 encapsulado en EAP-EXT hasta la generación de la MSK. El intercambio EAP-TLS mostrado en la Figura 4.12 es un intercambio conceptual sin fragmentación. EAP-TLS permite configurar un tamaño máximo de fragmento que en la captura que se va a analizar corresponde a un valor de 1000, ya que este intercambio se asemeja más a un intercambio real. A continuación procedemos a analizar una captura realizada con wireshark[41] correspondiente a un escenario típico con un servidor de autenticación en modo StandAlone, un autenticador trabajando en modo PassThrough y un

Figura 4.11: Escenario EAP-EXT básico

Figura 4.12: Escenario EAP-TLS básico

cliente. EAP es transportado entre el Servidor y el autenticador empleando Diameter[33] mientras que entre el autenticador y el cliente se ha optado por PANA[3]. Durante el análisis se va a comentar la parte de la trama correspondiente a EAP dejando de lado el hecho de que esté siendo transportada por Diameter o PANA.

Vamos a analizar un ejemplo de ambos mensajes, Request y Response, transportando sendos Method-TLV, el primer mensaje en ser analizado es el correspondiente al transporte del mensaje de EAP-TLS 'TLS-Start'. Como podemos observar en la Figura 4.15, en este caso EAP-TLS tan sólo transporta un byte de flags, que es el valor '20' subrayado. Podemos observar un aumento en EAP de 6 a 13 bytes, que son los correspondientes a la cabecera específica de EAP-EXT y a la cabecera de 4 bytes del TLV.

En respuesta a este paquete tenemos el TLS Client Hello, representado en la Figura 4.16. Como se puede observar en este caso la cantidad de datos transportada es mayor, no obstante la sobrecarga introducida por EAP-EXT es constante. En este caso EAP-EXT produce una carga de 114 bytes frente a los 107 de EAP-TLS, como en el caso anterior una sobrecarga de 7 bytes.

En ambos casos podemos observar como el valor correspondiente al campo tipo de la cabecera del tlv es 1.

Como se puede observar en la Figura 4.12 y se ratifica en la Figura 4.13, EAP-TLS realiza 5 intercambios con request y response. Para ver esto con mayor facilidad miraremos los bloques en azul celeste de la Figura 4.13 correspondientes a PANA, observando que se dividen en 7 grandes bloques. El primero de ellos, correspondiente al identity, lo obviaremos ya que es opcional, aunque la mayor parte de los despliegues actuales lo utilicen. El último por su parte corresponde a la transmisión de EAP-Success y tampoco es de mucho interés ya que no varía con respecto a EAP-EXT y el de EAP-TLS no es transportado por este.

Vamos a analizar la sobrecarga en bytes que ha supuesto hasta el momento EAP-EXT, teniendo en cuenta que nos encontramos ante el mejor caso, es decir, mínima sobrecarga. Por cada intercambio se introducen 7 bytes, 3 pertenecientes a Versión, bits FE, reservados y a las capabilities y los 4 restantes que pertenecen a la cabecera del TLV. La cabecera EAP no se incluye en este recuento debido a que la cabecera de EAP-TLS se elimina en el encapsulado. Tenemos 5 intercambios con Request y Response, lo que hace un total de 10 mensajes con una sobrecarga de 7 bytes cada uno, en total 70 bytes de sobrecarga para toda la transferencia EAP-TLS

Tras esta primera fase se habrá generado la MSK de EAP-TLS y este la habrá exportado a EAP-EXT de forma transparente. En estos momentos comienza la fase de transmisión de datos de EAP-EXT, donde se provocarán dos nuevos mensajes EAP-EXT Request y Response antes de llegar al Suc-

Figura 4.13: Captura de EAP-TLS en un escenario real

Figura 4.14: Captura de EAP-EXT en un escenario real

Figura 4.15: EAP-EXT Request con Method-TLV frente al mensaje EAP-TLS correspondiente

Figura 4.16: EAP-EXT Response con Method-TLV frente al mensaje EAP-TLS correspondiente

cess.

Podemos observar como el valor correspondiente al campo tipo de la cabecera del tlv ha cambiado a 2.

Estos dos mensajes no existían en EAP-TLS, por tanto su carga es completamente extra con respecto a EAP-TLS. Estos mensajes son de tamaño fijo para un algoritmo de firmado dado, en este caso se ha tomado el algoritmo por defecto. La sobrecarga producida por cada uno de ellos es de 44 bytes, 5 correspondientes a la cabecera EAP, 3 específicos de la cabecera EAP-EXT y 36 correspondientes al AUTH TLV. Si tenemos en cuenta que el response es de idéntico tamaño, tenemos que estos dos últimos intercambios producen una sobrecarga de 88 bytes que sumados a los 70 bytes de la fase 1 producen un total de 158 bytes de carga útil.

En total EAP-EXT produce una carga de 3859 bytes en el protocolo transportando EAP. Por su parte EAP-TLS produce un total de 3701 bytes. El aumento de carga es de un $4,26\,\%$. Esta visión es un tanto optimista debido a que EAP-EXT lleva su carga mínima, pero al mismo tiempo es muy pesimista ya que es un intercambio EAP-TLS ideal, sin fragmentación. Además, EAP necesita de un protocolo EAP-Lower Layer o AAA para su transmisión. Vamos a analizar los datos para PANA y Diameter.

- EAP-EXT sobre PANA. La primera fase de EAP-EXT involucrando los 10 intercambios ya mencionados provoca una carga incluyendo cabeceras PANA de 4308 bytes a los que tenemos que sumar 440 bytes debido a las 10 respuestas PANA-AUTH-ANSWER emitidas. Por su parte la segunda fase de EAP-EXT forma un total de 192 bytes debidos a sendos mensajes request y response de 96 bytes. En total EAP-EXT produce con PANA una carga de 4940 bytes. Por su parte EAP-TLS genera 4228 a los que hay que sumarles los 440 de los respectivos asentimientos, empleando por tanto 4668. El incremento en este caso de la carga es 5,82 %

- EAP-EXT sobre Diameter. En este caso los asentimientos son los propios de TCP y no se van a tener en cuenta. La primera fase ocupa 5604 bytes para EAP-EXT en 10 transacciones, lo que suponen 5524 bytes en EAP-TLS. La segunda parte emplea 444 bytes en 2 transacciones, así tenemos que EAP-EXT emplea un total de 6048 bytes frente a 5524 de EAP-TLS, lo que supone un aumento del 9,48 %

La razón de este aumento considerable viene dada por la complejidad de los mensajes Diameter frente a PANA. Diameter introduce información para el enrutado y de sesión, mientras que PANA sólo introduce un identificador

Figura 4.17: EAP-EXT Request con Auth-TLV

Figura 4.18: EAP-EXT Response con Auth-TLV

de sesión a parte de los datos EAP. Este es un cálculo muy optimista en cuanto a transacciones pero teniendo en cuenta el punto de vista de los datos es uno de los peores. Sólo podría ser peor sin fragmentación de EAP-TLS. Esto es así debido a que la segunda fase de EAP-EXT es fija siempre, es decir, para cada protocolo esa fase final va a producir siempre la misma sobrecarga, recordar que estamos hablando siempre de la implementación realizada para este proyecto con opciones mínimas. Así, cuanto más aumenten el número de transacciones de la fase 1 más aumentan los datos transportados y más asentimientos son los necesarios, aumentando la proporción. Veamos un supuesto aclaratorio. Supongamos que EAP-TLS produce 7000 bytes en 20 transacciones sobre Diameter. EAP-EXT en su primera fase habrá empleado por tanto 7140 bytes y 444 en la segunda, 7584 en total. La proporción en este caso se reduce a 8,3 %. En PANA este efecto se amplifica debido a los mensajes PANA-Auth-Answer generados con cada fragmentación. Ya se pudo ver en el análisis anterior de PANA como de 20 transacciones, tan solo 12 pertenecían a EAP, las 8 restantes no poseen carga útil para nuestros propósitos, siendo necesarias sólo para el nivel de transporte PANA.

De manera más esquematizada:

Método	**EAP**	**PANA+EAP**	**DIAMETER+EAP**
EAP-EXT	3859	4940	6048
EAP-TLS	3701	4228	5524

Figura 4.19: Tabla comparativa del coste en bytes

Hay que tener en cuenta que todo este análisis se ha realizado para el caso en que un único método interno sea ejecutado. Cuando varios métodos internos se ejecutan secuencialmente, como pudimos observar en la figura 3.2, en este caso el último intercambio correspondiente a la fase de binding, se lleva a cabo junto con el transporte de los dos últimos Method TLV del último método interno ejecutado. Esto es así, gracias a la EAP-EXT-AUTH-KEY generada por el método interno i-1, siendo i el número correspondiente al último método interno ejecutado. En este caso, la sobrecarga de EAP-EXT sobre cualquier otro método para la fase de binding es de 7 bytes más el tamaño del AUTH-TLV. Para el caso en que se emplee la PRF por defecto, el AUTH-TLV produce una sobrecarga de 44 bytes. Así cuando varios métodos EAP son ejecutados en su interior, la sobrecarga producida es de N*7+88, donde N es el número de mensajes generados por todos los métodos internos ejecutados.

Capítulo 5

Conclusiones y vías futuras

Las tecnologías móviles han evolucionado mejorando la experiencia del usuario en su acceso a los servicios de red. Las tecnologías celulares han permitido una movilidad global real, tanto de voz como de datos, siendo esta última característica la que más ha evolucionado. A su vez, se han conseguido grandes avances en las tecnologías inalámbricas de área local que las han situado como alternativas a la hasta ahora única opción de las redes cableadas. Próximamente se augura la expansión de tecnologías en actual desarrollo que permitirán aumentar de forma exponencial, no sólo la capacidad sino su radio de cobertura.

Así pues, los nuevos dispositivos permiten a un usuario la conexión a través de múltiples interfaces de distinta tecnología, acercándonos cada día más al paradigma ABC (Always Best Connected). Independientemente de la tecnología empleada para el acceso a la red, es necesario poseer un mecanismo de autenticación que permita identificar a un usuario de manera unívoca. Este proceso debería ser capaz de generar material criptográfico que permita securizar las comunicaciones, de vital importancia particularmente en medios de acceso inalámbricos.

Puesto que muchos son los mecanismos de autenticación y las tecnologías existentes en la actualidad, la necesidad de un mecanismo extensible e independiente del medio de autenticación propició el diseño, desarrollo e implantación de un protocolo denominado *Extensible Authentication Protocol* (EAP). De forma complementaria al proceso de autenticación, se ha visto, durante el desarrollo de este proyecto fin de carrera, que es necesario configurar de forma dinámica los parámetros necesarios para acceder adecuadamente al servicio de red. De hecho, realizar esta función de forma estática es complicado e ineficiente, y las operadoras necesitan mecanismos que permitan configurar e intercambiar información relevante para el acceso al servicio de forma dinámica. A este proceso conjunto mediante el cual un usuario se au-

tentica e intercambia de forma protegida información relativa al servicio con la red, se le denomina *bootstrapping*.

En general, sería deseable tener un mecanismo de bootstrapping dinámico, seguro, rápido y de fácil implantación en los despliegues actuales e independiente del medio. Teniendo en cuenta que el protocolo EAP es empleado para el proceso de autenticación durante el acceso inicial al servicio de red y es capaz de generar material criptográfico, se ha pensado en el uso de este protocolo, no sólo para la autenticación inicial sino también para suministrar cierta información protegida de autorización. Con estas premisas se ha diseñado la solución propuesta basada en un método EAP tunelado llamado EAP-EXT capaz de encapsular diferentes métodos de autenticación.

EAP-EXT aprovecha la capacidad de extensión de EAP para implementar un mecanismo de bootstrapping genérico capaz de emplear cualquiera de los algoritmos/métodos de autenticación existentes, en particular se hará uso de aquellos capaces de generar material criptográfico. EAP-EXT además soluciona problemas de seguridad y compatibilidad de muchos de estos métodos. Además EAP-EXT define un mecanismo de bootstrapping independiente del servicio, lo que le hace candidato de ejercer de mecanismo de bootstrapping para otros servicios de red.

Como aportación adicional y teniendo en cuenta que, en general, una autenticación EAP involucra múltiples intercambios, se ha diseñado un mecanismo de reautenticación rápida que aprovechan el material criptográfico de una primera autenticación satisfactoria, para derivar nuevo material criptográfico y reducir el número de intercambios necesarios para completar una nueva autenticación.

Adicionalmente, en este proyecto se ha realizado una implementación del método EAP-EXT con el objetivo de comprobar su viabilidad. Se ha implementado una versión funcional capaz de realizar un proceso completo de autenticación de un método interno, en este caso EAP-TLS, muy empleado en despliegues existentes basados en EAP. También se ha implementado la segunda fase de intercambio de información securizada. Esta implementación, lejos de ser una versión final para ser desplegada en entornos de producción, es un paso inicial para un desarrollo y despliegue final de la solución. De hecho, se está planteando la migración de la implementación a otras plataformas como FreeRadius o WPA Supplicant para 802.11i, para su uso con punto de conexión inalámbricos reales, que permitirá valorar aun más el impacto que produce EAP-EXT en los despliegues actuales.

Se ha dejado también como trabajo futuro la implementación y puesta en marcha del mecanismo de reautenticación rápida, donde se planea realizar pruebas de rendimiento reales para compararlas con soluciones propuestas tales como EAP-ER. De esta forma se puede valorar el uso de EAP-EXT

como alternativa al proceso de reautenticación rápida. El motivo de esta evaluación radica que EAP-EXT no implica modificaciones del estándar EAP, mientras que por ejemplo EAP-ER sí. En caso de resultados comparables en el proceso de reautenticación rápida, EAP-EXT ofrece esa ventaja adicional.

Apéndice A

Tecnologías relacionadas

A.1. Tecnologías móviles

A.1.1. Concepto de red celular

En sus inicios el servicio de radio telefonía móvil era provisto por medio de transmisores/receptores de alta potencia. Un sistema típico soportaba unos 25 canales con un radio efectivo de unos 80 km. Una forma de incrementar la capacidad del sistema es usar sistemas de baja potencia con un radio más corto y usar un mayor número de transmisores/receptores.

La esencia de una red celular es el uso de múltiples transmisores de baja potencia. Debido al pequeño alcance de cada transmisor, un área puede ser dividida en celdas , cada una de ellas provista de una antena. Cada celda tiene asignado un rango de frecuencias diferentes a las de otra celda adyacente para evitar interferencias o diafonía[1].

La potencia de transmisión en cada estación base está muy controlada, suficiente para permitir la comunicación con los dispositivos en la frecuencia asignada dentro de la celda y menos de la necesaria para invadir otras celdas con frecuencias distintas. Con esto se permite reutilizar frecuencias en celdas no adyacentes de modo que se pueda emplear la misma frecuencia en múltiples conversaciones simultaneas.

Puede darse el caso en que una celda no posea suficientes frecuencias disponibles para manejar una nueva llamada debido a la alta afluencia de usuarios a esta. Existen diferentes aproximaciones para solucionar este problema: Añadir nuevos canales, tomar prestada una frecuencia vecina, división de la celda, creación de sectores dentro de la misma celda y el uso de microceldas.

Este concepto de disposición celular define una acción básica para todo

[1] Crosstalk

dispositivo moviéndose dentro de la red, el proceso de **handoff**. Handoff es el proceso por el cual un dispositivo hace el cambio de una estación base a otra, en la mayoría de los casos esto supone un cambio de celda. El proceso de handoff puede ser manejado de diversas formas e involucra varios factores.

El handoff puede ser iniciado por la red, en cuyo caso la decisión la realiza la red de forma independiente teniendo en cuenta la potencia de las señales recibidas por los dispositivos. Alternativamente, existe otro esquema en el que el terminal informa a la red sobre las señales que este recibe, este esquema se denomina asistido por el dispositivo, si bien la decisión la toma la red. Finalmente existe otro modo que es controlado totalmente por el dispositivo, en este caso es el dispositivo con la información que posee el que decide hacia donde cambiar. El principal parámetro empleado para tomar la decisión de cambio suele ser la fuerza de la señal del dispositivo hacia la estación base, normalmente la estación base emplea una media móvil para calcular este valor,si bien muchos otros parámetros pueden ser utilizados.

A.1.2. Global System for Mobile Communications

Antes del desarrollo de GSM[2], los países del continente europeo usaban numerosas e incompatibles tecnologías de red celular de 1ª generación. GSM fué desarrollada para proveer a Europa de una nueva tecnología de 2ª generación común a todos los países, de tal manera que los mismos terminales de abonado pudiesen emplearse a lo largo del continente.

GSM se ha convertido en una tecnología muy satisfactoria y probablemente la más popular a nivel mundial, apareció por primera vez en Europa en 1990. Sistemas similares se han desarrollado en otros continentes.

El espectro reservado por GSM es de 25MHz para transmisión de base[3] y de móvil[4], aunque otros rangos han sido empleados fuera de Europa. Los usuarios acceden a la red usando una combinación de FDMA[5] y TDMA[6]. Se emplean frecuencias portadoras cada 200KHz lo que produce un total de 125 canales full-duplex. Los canales son modulados a una tasa de 270.833 Kbps. Como en la anterior generación analógica AMPS, hay dos tipos de canales, tráfico y control.

En GSM el proceso de Handoff para el cambio de celda está soportado siempre que no sea con otros sistemas, es decir no soporta mantener el servicio durante el roaming. GSM incorpora un servicio de paginación por el cual se

[2]Global System for Mobile Comunications
[3]935-960MHz
[4]890-915MHz
[5]Frequency Division Multiple Access
[6]Time Division Multiple Access

envía un mensaje broadcast en el que se indica a un determinado terminal que debe realizar una acción, por ejemplo que tiene una llamada entrante. Puede que el sistema no conozca exactamente en que celda se encuentra el terminal, en tal caso se paginan las celdas en las que se tiene constancia puede estar, en el peor de los casos todas. Para solucionar este problema se crearon location areas y location updating, en concepto se trata de introducir un indentificación de localización en cada celda, así durante el cambio de celda, el terminal detectará que este ha cambiado y se lo notificará a la estación base. Si el identificador es único para cada área, el sistema puede determinar la posición exacta del dispositivo, en caso contrario será una zona.

El IMSI[7] y la clave de autenticación se almacenan de forma permanente en la SIM[8], también se almacenan de forma semi-permanente el LAI[9] actual, la clave de cifrado y la lista de redes preferidas o prohibidas, como añadido tiene la capacidad de almacenar algunos datos de usuario como por ejemplo un listín telefónico. La SIM es un dispositivo móvil en forma de tarjeta inteligente. Esto permite al usuario cambiar el terminal sin necesidad de notificar a su operadora, de hecho, la mayor parte de los terminales disponibles no funcionan sin introducir una SIM válida excepto para llamadas de emergencia. La SIM está protegida por dos códigos, el PIN para su uso diario y el PUK para operaciones más avanzadas.

GSM adolecía de algunas limitaciones en cuanto a las conexiones de datos. Poseía una velocidad de transferencia de 9,6 Kbps velocidad insuficiente para la mayor parte de las aplicaciones. Además el tiempo de establecimiento de conexión a la red de 15 a 30 segundos era demasiado alto. Por otra parte y desde el punto de vista del usuario pagaba por tiempo conectado no por carga de datos transmitida, si a esto le sumamos la baja velocidad de transferencia, los costes se tornaban sobreelevados. También padecía problemas para mantener la conectividad en roaming.

A.1.3. General Packet Radio Service

Más conocido como GPRS o GSM-IP por su uso de la tecnología IP[10] es una optimización de GSM orientada a la transmisión de datos que emplea transmisión de paquetes en lugar de utilizar circuitos, más adecuados para voz. En GPRS se crean nuevos canales de radio para transportar datos con una reserva flexible. GPRS tiene la ventaja que cada canal puede ser compartido por múltiples usuarios mejorando así la eficiencia de utilización de los

[7] International Mobile Subscriber Identity
[8] Subscriber Identity Module
[9] Location Area Identity
[10] Internet Protocol

recursos de red. Los enlaces de subida y de bajada se reservan por separado. Los recursos radio se pueden compartir de forma dinámica entre voz y datos, es decir, un terminal puede transmitir voz y datos de forma simultanea. Se pueden emplear varias técnicas de codificación en el canal radio ofreciendo tasas entre 9 y 150 Kb/s.

La seguridad es similar a la de GSM pero emplea un algoritmo de cifrado distinto optimizado para datos. Estas capacidades de transmisión crean un nuevo modelo de negocio orientado a la comunicación móvil de datos y aparece un nuevo rol de usuario no existente hasta el momento que emplea el servicio de telefonía celular para la transmisión de datos.

Se introducen nuevos datos en la SIM, Routing Area y algunos parámetros de red. El terminal puede funcionar de tres maneras, comunicación de voz y datos simultanea sin interrupción, conmutación simultanea pero si llega una llamada la transmisión de datos se interrumpe hasta el final de esta y finalmente exclusividad, es decir, sólo se accede a un servicio en un instante concreto.

GPRS ha sido por tanto una evolución no traumática de GSM ya que no ha sido necesaria una gran inversión para su puesta en marcha por el aprovechamiento de parte de las infraestructuras de GSM. Esto tiene la siguiente consecuencia, GPRS parte de la misma cobertura que GSM.

En GPRS se define el concepto de Routing Areas, estas están formadas por grupos de celdas y a su vez forman Service Areas. No existe correspondencia directa con GSM pero una RA ha de ser un subconjunto de una y sólo una LA. El reducido tamaño de las RA permite una mayor optimización de los recursos radio.

A.1.4. Universal Mobile Telecommunications System

La 3ª generación, actualmente en proceso de implantación UMTS, se diseñó a partir de objetivos como tener calidad de voz equiparable a la de la red cableada, 144kbps para usuarios en movimiento a alta velocidad como por ejemplo en vehículos, 384kbps para usuarios con movimiento reducido como por ejemplo peatones, soporte para 2.048 Mbps para uso de oficinas,...

Desde 1992 hasta 1998 no apareció la idea de un estándar común de 3G. Hasta ese momento sólo hubo investigación y mejoras de 2G sin integración. En diciembre de 1998 se crea el 3GPP[11] como organización para coordinar la producción de estándares conjuntos entre diferentes foros de estandarización.

Inicialmente 3GPP se creó para generar especificaciones técnicas para un estándar común para redes 3G extendiendo el núcleo de red de las redes

[11] 3rd Generation Partnership Project

GSM, pero actualmente también se encarga de la estandarización de mejoras a tecnologías de GSM como GPRS o EDGE. Para asegurar que la visión americana se tenía en cuenta se creo el 3GPP-2 que toma como base el trabajo IS-95. IS-95 se refiere al sistema CDMA[12] empleado en Norte América.

La raiz de UMTS es una red ATM[13] de transmisión de paquetes. La localización en redes UMTS sigue la misma filosofía de uso de áreas que en GPRS, en UMTS se introduce el concepto de UTRAN Registration Area que engloba una o más celdas. Este concepto permite una mayor optimalidad en la localización de los dispositivos móviles para diferentes tipos de servicios.

A.1.5. Integración de 3G con IP

El empleo de IP como protocolo capaz de funcionar sobre cualquier tecnología de nivel de enlace le hace atractivo para la integración de redes heterogéneas. Esta cualidad le hace candidato para integrar redes de telefonía móvil como UMTS y redes cien por cien IP como WLAN. Esta propuesta es la que se ha venido denominando 4ª generación o 4G, donde un usuario podrá intercambiar sesiones de datos entre tecnologías de distinta naturaleza sin perder la conexión a la red, por ejemplo se podrá intercambiar una vídeo-conferencia en nuestro terminal móvil con vídeo-llamada UMTS y acceso 802.11 (WLAN) sin necesidad de reconectar.

Existen dos aproximaciones para lograr esta integración. Por un lado existe una definición por parte del 3GPP de lo que ellos han denominado Wireless IP, que define dos alternativas. El uso de Simple IP basado en el protocolo PPP[14] o MIP[15]. En estas propuestas se propone la utilización de un protocolo AAA denominado RADIUS[16] para labores de autenticación y autorización y DiffServ para ofrecer calidad de servicio. Esta propuesta intenta integrar los protocolos del IETF sin tener en cuenta la no optimalidad de ellos en estos procesos inalámbricos.

Por otra parte existe se tiene al diseño de redes all-IP, consistentes en el desarrollo de una red que se basa en IP como principal protocolo para el transporte y la conmutación. No existe ningún estándar pero se han realizado diversos estudios[42] al respecto. Se propone el uso de IP como transporte para datos de usuario y señalización. Para defender esta aproximación tenemos la capacidad de IP de ser eficaz para aplicaciones de tiempo real, el desacoplamiento que ofrece para las aplicaciones respecto al nivel de red o la

[12]Code Division Multiple Access
[13]Asynchronous Transfer Mode
[14]Point to Point Protocol
[15]Mobile IP
[16]Remote Authentication Dial in User Service

buena arquitectura de capas que presenta, entre otras. Además a todos estos motivos técnicos es conveniente sumar motivaciones puramente económicas, como el ahorro en el equipamiento del núcleo de la red al necesitar sólo soporte para IP y la posibilidad que ofrece de que aparezcan proveedores de servicios independientes de la conexión del usuario.

Todas estas propuestas poseen el problema del traspaso de contextos durante la movilidad. La simplicidad del protocolo Mobile IP adolece de problemas en entornos de micromovilidad muy frecuente, donde pueden surgir problemas dependiendo del tipo de sesión que esté establecida, pondremos un ejemplo bastante revelador al respecto, en un entorno celular, un terminal realiza cambios de celda muy frecuentes en los que es necesaria una actualización de la información del dispositivo, en este entorno de movilidad rápida con MIP puede adolecer de lentitud. Si sobre esta red celular estamos manteniendo una comunicación VoIP[17], podemos sufrir pérdidas de conexión ya que no es tan permisiva como una conversación de voz normal. El oído humano es adaptativo en cuanto a pérdidas de fragmento de información en la conversación, sin embargo muchos de los protocolos de computadoras como por ejemplo TCP[18] no permite esa desconexión temporal resultando en una caída de las sesiones.

A.2. Tecnologías inalámbricas de datos

A.2.1. WLAN

Ya por 1979 se realizaron las primeras aproximaciones hacia conexiones inalámbricas de datos por medio de infrarojos e incluso mediante walkie-talkies transportando señales de audio moduladas por modems.

Estas primeras aproximaciones no hicieron otra cosa que abrir los ojos a una nueva necesidad y deseo de la sociedad de poder intercambiar la información sin cables. No obstante, no fue hasta 1997 cuando comenzaron a aparecer los primeros aparatos implementando el estándar 802.11 del IEEE. Muchos son los nombres que se han empleado para esta especificación, Wireless ethernet, enfatizando su relación con la red local cableada tradicional, Wi-Fi(Wireless Fidelity) acuñado por la 'Ethernet Compability Program' para su programa de certificaciones de compatibilidad con el estándar o simplemente WLAN (Wireless LAN).

Este estándar se ha convertido en la más popular de las tecnologías para proveer de conectividad inalámbrica en el último tramo de la red, el corres-

[17]Voice over IP
[18]Transmission Control Protocol

pondiente al usuario. Inicialmente, el estándar definió dos medios 'físicos', infrarojos y ondas electromagnéticas(aunque el infrarojo sea también una onda electromagnética se distingue por pertenecer al espectro visible de la luz), el primero de ellos no tardó en pasar a un segundo plano debido a sus limitaciones en cuanto a despliegue. La capacidad de las ondas electromagnéticas de superar obstáculos como paredes o material de oficina le concedieron la hegemonía en estos despliegues.

Existen varias versiones del estándar con características tecnológicas, históricas y políticas distintas:

versión	**Año**	**Espectro**	**Velocidad**	**Cobertura**
802.11	1997	2,4Ghz	2Mbps	20-100 metros
802.11a	1999	5Ghz	54Mbps	35-120 metros
802.11b	1999	2,4Ghz	11Mbps	38-140 metros
802.11g	2003	2,4Ghz	54Mbps	38-140 metros
802.11n	Junio 2009(Est)	2,4Ghz	248Mbps	70-250 metros
802.11y	Junio 2008(Est)	3,7Ghz	54Mbps	50-5000 metros

Figura A.1: 802.11

Una curiosidad de estos datos es que la banda de 2,4Ghz es la ideal para calentar líquidos como el agua por lo que es la empleada por los microondas y exime del pago de licencias de uso.

Muchas son las características que empujan al consumidor al empleo de WLAN:

- Para ofrecer conectividad es necesario instalar estaciones base y antenas en ese lugar, pero una vez hecho esto, ofrecer servicios a un nuevo usuario es más un problema de autorización que de infraestructura.

- Permiten el despliegue de los famosos 'hot spots', ofreciendo a usuarios en itinerancia una forma de acceder a la red sencilla y rápida, sin ataduras físicas.

- En edificios de los declarados históricos, facilitan la tarea de desplegar la infraestructura de red a no ser necesario tocar ninguna estructura del edificio.

- Un efecto social generado por estas redes es la creación de redes comunitarias que ofrecen conexión interna de altas velocidades evitando así las limitaciones impuestas por la tecnología de acceso como el ADSL o simplemente por el operador.

No obstante, estas bondades no excluyen del uso de tecnologías cableadas, de hecho las hace aconsejable para equipamiento fijo, con necesidades mayores de ancho de banda o requisitos de seguridad altos.

Uno de los conceptos claves de estas tecnologías es el de 'scanning', o lo que viene siendo lo mismo, encontrar la red. En las redes cableadas esta acción es tan sencilla como buscar una roseta en la pared y enchufar el cable, pero en WLAN no tenemos referencias visuales, es necesario que los dispositivos sean capaces de encontrar sus puntos de conexión.

Existen dos tipos de escaneo de red, activo y pasivo. Activo es aquel en el que el dispositivo envía peticiones constantemente para encontrar una red mientras que pasivo es aquel en el que el dispositivo espera la llegada de información de la red para ahorrar energía, algo muy importante en dispositivos verdaderamente móviles y portables como PDA's o teléfonos celulares. Esta capacidad de encontrar nuevas redes define además una nueva necesidad, la necesidad de distinguir unas redes de otras, para el usuario estas redes se distinguen por medio del SSID el cual originalmente se ideo como un identificador de conjuntos de servicios, pero dada su representación cercana al usuario terminó convirtiéndose en el nombre de la red, existe en particular un SSID tipo broadcast que es el empleado durante el escaneo. También se definen conceptos como canal más dependientes del nivel físico asociado.[43]

A.2.2. 802.16 WIMAX

También conocido como Wireless MAN (Wireless Metropolitan Area Network) es la especificación para ofrecer acceso inalámbrico en el último kilómetro de la red, en inglés 'last mile', como alternativa a tecnologías cableadas como xDSL o Cable. Emplea la banda de los 10 a los 66 Ghz. En sus versiones iniciales sólo se ofreció con opción estática (802.16d no permite movilidad), despreocupándose por tanto de los procesos de handoff. Ofrece unas velocidades de conexión notables que se encuentran al igual que su alcance entre la alta capacidad y bajo alcance de las WLAN y la 'baja' capacidad y alto alcance de la 3ª generación de móviles. La especificación 802.16e o Mobile WIMAX es la ampliación de WIMAX para ofrecer movilidad. Comparado con WLAN, además de su lógico mayor radio de cobertura, WIMAX ofrece mejoras en cuanto al tratamiento de la calidad de servicio, WLAN en este aspecto sigue una filosofía heredada de las redes de area local cableadas en las que la calidad de servicio está basada en una etiqueta, mientras que WIMAX permite asegurar calidad de servicio para cada flujo de manera independiente.

Comparativamente:

Servicio	**Definición**	**Usos**
Servicio garantizado	Transporte de paquetes de tamaño fijo	Transporte de T1/E1
Servicio RT extendido	Paquetes de tamaño variable en tiempo real con base periódica	VoIP
Servicio RT	Flujos de datos en tiempo real en intervalos periódicos	Video MPEG
Servicio No RT	Permisividad de retardos en la transferencia pero con transferencia mínima	FTP con capacidad mínima garantizada
Best effort	Tráfico sin restricciones	HTTP

Figura A.2: WIMAX QOS

versión	**Velocidad**	**Cobertura**
802.11g	54Mbps	38-140 m
UMTS	128kbps - 28 Mbps	1-4 km
WIMAX	10-70 Mbps	10km

Figura A.3: Comparativa 802.16

Apéndice B

Siglas

Sigla	Expansión
GSM	Global System for Mobile communications
GPRS	General Packet Radio Service
UMTS	Universal Movile Telecommunications System
Mbps	Megabit per second
Kbps	Kilobit per second
WLAN	Wireless Local Area Network
AAA	Authentication Authorization and Accounting
TAC	Terminal Access Controler
TACACS	Terminal Acces Controler Access-Control System
PPP	Point to Point Protocol
RTB	Red telefónica básica
PAP	Password Authentication Protocol
CHAP	Challenge Handshake Authentication Protocol
RADIUS	Remote Authentication Dial In User Service
EAP	Extensible Authentication Protocol
EAP-KMF	EAP Key Management Framework
MSK	Master Session Key
EMSK	Extended Master Session Key
TSK	Transient Session Key
FDMA	Frecuency Division Multiple Access
TDMA	Time Division Multiple Access
IMSI	International Mobile Subscriber Identity
ATM	Asynchronous Transfer Mode
PANA	Protocol for carrying Authentication for Network Address
UDP	User Datagram Protocol

Sigla	**Expansión**
TCP	Transmission Control Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
TLV	Type Length Value
SAML	Security Assertion Markup Language
TLS	Transport Layer Security

Bibliografía

[1] B. A. Anderson. *TACACS User Identification Telnet Option.* IETF RFC 0927, June 1984.

[2] IEEE 802.11i Std., Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security, July 2005.

[3] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. *Protocol for Carrying Authentication for Network Access (PANA).* IETF Internet Draft, draft-ietf-pana-pana-18, Sept. 2007.

[4] C. Rigney, S. Willens, A. Rubens, and W. Simpson. *Remote Authentication Dial In User Service (RADIUS).* IETF RFC 2865, June 2000.

[5] P. Calhoun and J. Loughney. *Diameter Base Protocol.* IETF RFC 3588, Sept. 2003.

[6] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. *Extensible Authentication Protocol (EAP).* RFC3748, June 2004.

[7] C. Kauffman. *Internet Key Exchange (IKEv2) Protocol.* IETF RFC 4306, Dec. 2005.

[8] B. Aboba, D. Simon, and P. Eronen. *Extensible Authentication Protocol (EAP) Key Management Framework.* IETF Internet Draft, draft-ietf-eap-keying-22.txt, Nov. 2007.

[9] G. Giaretta, I. Guardini, E. Demaria, J. Bournelle, and M. Laurent-Maknavicius. *MIPv6 Authorization and Configuration based on EAP.* IETF Internet Draft, draft-giaretta-mip6-authorization-eap-04, Oct. 2006.

[10] *IST EU DAIDALOS Project.* http://www.ist-daidalos.org.

[11] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson. *Protected EAP Protocol (PEAP) Version 2.* IETF Internet Draft, draft-josefsson-pppext-eap-tls-eap-10, Oct. 2004.

[12] N. Cam-Winget. *Dynamic Provisioning using EAP-FAST.* IETF Internet Draft, draft-cam-winget-eap-fast-provisioning-02, March 2006.

[13] M. Nakhjiri and Y. Ohba. *Derivation, delivery and management of EAP based keys for handover and re-authentication.* IETF Internet Draft, draft-ietf-hokey-key-mgm-01, May 2008.

[14] D. W. DaviesandW. L. Price. Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer. John Wiley and Sons, 1984.

[15] J. Salowey, G. Sliepen, A. Taal, and D. Spence. *Policies in AAA.* IRTF Internet Draft, draft-irtf-aaaarch-aaa-pol-01, March. 2001.

[16] S.M.C.M. van Oudenaarde, L.H.M. Gommans, C.T.A.M. de Laat, and F. Dijkstraand A. Taal. *Prototype of a Generic AAA Server.* IRTF Internet Draft, draft-irtf-aaaarch-prototype-02, Sept. 2004.

[17] J. Vollbrecht et al. *AAA Authorization Framework.* IETF RFC 2904, Aug. 2000.

[18] M. Beadles and D. Mitton. *Criteria for Evaluating Network Access Server Protocols.* IETF RFC 3169, Sept. 2001.

[19] T. Clancy et al. *Handover Key Management and Re-authentication Problem Statement.* IETF Internet Draft, draft-ietf-hokey-reauth-ps-07, Nov. 2007.

[20] B. Aboba and P. Calhoun. *RADIUS support for EAP.* IETF RFC 3579, June 2003.

[21] P. Congdon, B. Aboba, A. Smith, G.Zorn, and J. Roese. *IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines.* RFC3580, September 2003.

[22] B. Aboba, G.Zorn, and D. Mitton. *RADIUS and IPv6.* RFC3162, August 2001.

[23] Y. Ohba, M. Parthasarathy, and M. Yanagiya. *Channel Binding Mechanism based on Parameter Binding in Key Derivation.* IETF Internet Draft, draft-ohba-eap-channel-binding, December 2006.

[24] J. Arkko and P. Eronnen. *Authenticated Service Information for the Extensible Authentication Protocol*. IETF Internet Draft, draft-arkko-eap-service-identity-auth, October 2005.

[25] W. Simpson. *The Point-to-Point Protocol (PPP)*. IETF RFC 1661, July 1994.

[26] IEEE 802.1X Std., Standards for Local and Metropolitan Area Networks: Port based Network Access Control, 2004. IEEE Standards for Information Technology.

[27] *Comunicaciones World*. http://www.idg.es/comunicaciones/index.asp.

[28] B. Aboba and D. Simon. *PPP EAP TLS Authentication Protocol*. IETF RFC 2716, Oct. 1999.

[29] P. Funk and S. Blake-Wilson. *EAP Tunneled TLS Authentication Protocol (EAP-TTLS)*. IETF Internet Draft, draft-ietf-pppext-eap-ttls-05, July 2004.

[30] A. Palekar, D. Simon, G. Zorn, and S. Josefsson. *Protected EAP Protocol (PEAP)*. IETF Internet Draft, draft-josefsson-pppext-eap-tls-eap-06, March 2003.

[31] H. Haverinen and J. Salowey. *Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)*. IETF RFC 4186, Jan. 2006.

[32] J. Arkko and H. Haverinen. *Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)*. IETF RFC 4187, Jan. 2006.

[33] P. Eronen, T. Hiller, and G. Zorn. *Diameter Extensible Authentication Protocol (EAP) Application*. IETF RFC 4072, Aug. 2005.

[34] J. Salowey, L. Dondeti, V. Narayanan, and M.Ñakhjiri. *Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)*. IETF Internet Draft, draft-ietf-hokey-emsk-hierarchy-02, Nov. 2007.

[35] J. Katoen. *NIST FIPS 180-2, Secure Hash Standard*, Aug. 2002. With Change Notice 1 dated Feb. 2004.

[36] V. Narayanan and L. Dondeti. *EAP Extensions for EAP Re-authentication Protocol (ERP).* IETF Internet Draft, draft-ietf-hokey-erx-08, Nov. 2007.

[37] B. Aboba, M. Beadles, J. Arkko, and P. Eronen. *The Network Access Identifier.* IETF RFC 2486, Dec. 2005.

[38] R. Marin, P. Garcia, and A. Gomez-Skarmeta. *Cryptographic Identity Based Solution for Fast Handover on EAP Wireless Networks.* In *The 9th International Conference on Mobile and Wireless Communications Networks, MWCN 2007*, pages 46–51, Cork, Ireland, Sept. 2007.

[39] J. Walker and R. Housley. *The EAP Archie Protocol.* draft-jwalker-eap-archie-01, June 2003.

[40] Openikev2 project. `http://openikev2.sourceforge.net/`.

[41] WIRESHARK. http://www.wireshark.org.

[42] 3gpp ip-based services. `http://www.3gpp.org/ftp/tsg\_sa/WG1\_Serv/TSGS1\_04-Quebec/Docs/S1-99411\_R2000IP-basedIP-basedservicesinRelease2000SiemensSP-99268.doc`.

[43] 802.11 wireless networks: The definitive guide.