Universidad de Murcia

Facultad de Informática

DISEÑO DE UN SISTEMA DE CONTROL DE ACCESO EN REDES HETEROGÉNEAS CON PRIVACIDAD BASADO EN KERBEROS

Tesis Doctoral

Presentada por:
Fernando Pereñíguez García

Supervisada por:
Dr. Antonio Fernando Gómez Skarmeta

Murcia, Marzo de 2011

D. Antonio Fernando Gómez Skarmeta, Catedrático de Universidad del Área de Ingeniería Telemática y presidente de la Comisión Académica del Programa de Postgrado de Tecnologías de la Información y Telemática Avanzadas de la Universidad de Murcia, INFORMA:

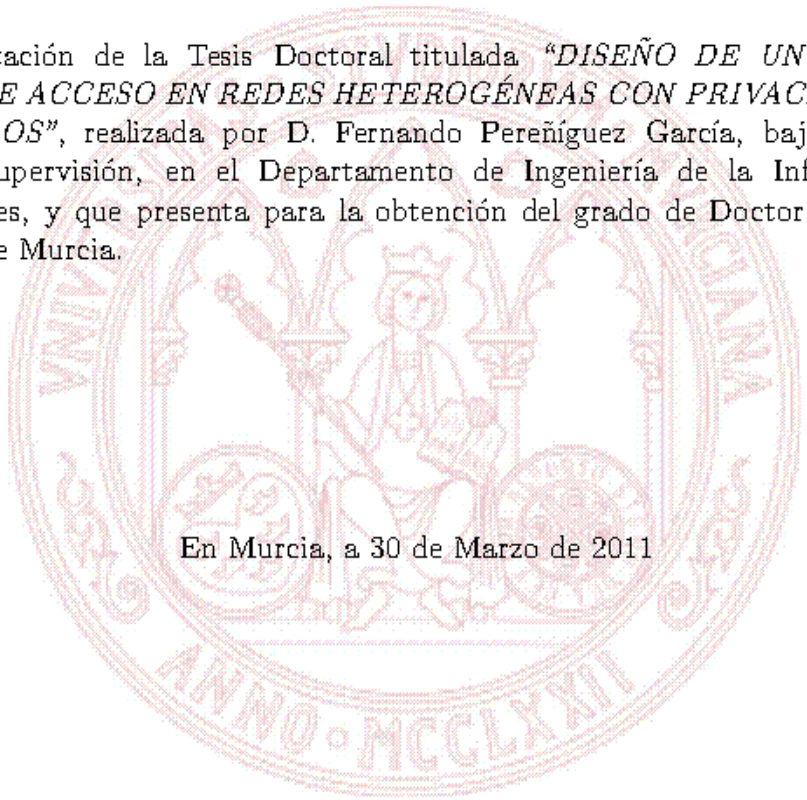
Que la Tesis Doctoral titulada *"DISEÑO DE UN SISTEMA DE CONTROL DE ACCESO EN REDES HETEROGÉNEAS CON PRIVACIDAD BASADO EN KERBEROS"*, ha sido realizada por D. Fernando Pereñíguez García, bajo la inmediata dirección y supervisión de D. Rafael Marín López y D. Antonio Fernando Gómez Skarmeta, y que la Comisión Académica ha dado su conformidad para que sea presentada ante la Comisión General de Doctorado.

En Murcia, a 22 de Marzo de 2011

D. Antonio Fernando Gómez Skarmeta

D. Antonio Fernando Gómez Skarmeta, Catedrático de Universidad del Área de Ingeniería Telemática en el Departamento de Ingeniería de la Información y las Comunicaciones de la Universidad de Murcia, AUTORIZA:

La presentación de la Tesis Doctoral titulada *"DISEÑO DE UN SISTEMA DE CONTROL DE ACCESO EN REDES HETEROGÉNEAS CON PRIVACIDAD BASADO EN KERBEROS"*, realizada por D. Fernando Pereñíguez García, bajo mi inmediata dirección y supervisión, en el Departamento de Ingeniería de la Información y las Comunicaciones, y que presenta para la obtención del grado de Doctor Europeo por la Universidad de Murcia.



En Murcia, a 30 de Marzo de 2011

D. Antonio Fernando Gómez Skarmeta

Resumen

En los últimos años, los sistemas de comunicaciones inalámbricos han adquirido una gran relevancia debido a la proliferación de un variado número de tecnologías de transmisión inalámbrica, que emplean el aire como medio de propagación. Adicionalmente, los usuarios han mostrado un gran interés en el uso de comunicaciones inalámbricas pues éstas ofrecen la posibilidad de intercambiar información mientras el usuario cambia de punto de acceso a la red. Este creciente interés ha provocado la aparición de dispositivos móviles tales como *smart phones*, *tablet PCs* o *netbooks* que, equipados con múltiples interfaces de conexión, permiten a usuarios móviles acceder a servicios de red e intercambiar información con otros usuarios en cualquier lugar y momento. Para soportar esta experiencia que permite al usuario estar siempre conectado, las redes de comunicaciones están adoptando un esquema basado en el protocolo *Internet Protocol* (IP) donde un núcleo de red basado en la tecnología IP actúa como punto de interconexión de un conjunto de redes de acceso basadas en diferentes tecnologías inalámbricas. Este futuro escenario, denominado como *Next Generation Networks* (NGNs), habilita la convergencia de redes de acceso heterogéneas con el objetivo de combinar todas las ventajas ofrecidas por cada tecnología de acceso.

Uno de los retos más importantes en las redes de próxima generación es el mecanismo a través del cuál las diferentes tecnologías se complementarán entre sí para conseguir un acceso a la red constante y fiable. De hecho, para la provisión de servicios multimedia que requieren un alto nivel de calidad en las comunicaciones, es necesario alcanzar movimientos suaves y transparentes (*seamless*) que reduzcan el número de paquetes perdidos cuando el usuario móvil cambia su punto de conexión a la red durante el denominado *handoff*. Sin embargo, durante el handoff, la conexión a la red podría ser interrumpida por diferentes motivos, lo que resulta en una pérdida de paquetes transmitidos y/o recibidos que, sin lugar a dudas, afecta a las comunicaciones activas. Por este motivo, con el fin de alcanzar movimientos rápidos sin interrupciones y mejorar la calidad del servicio al que accede un usuario móvil, un objetivo esencial reside en la reducción del número de paquetes perdidos durante el proceso de handoff.

El handoff requiere la ejecución de diversas tareas que afectan negativamente al tiempo que dura el proceso. Obviamente, mediante la reducción del tiempo necesario para completar cada una de estas tareas, el tiempo total de interrupción provocado por el handoff disminuirá y, por lo tanto, el número de paquetes perdidos. En particular, en el contexto de esta tesis doctoral, prestaremos atención al *control de acceso a la red*, el cuál es uno de los procesos ejecutados durante el handoff que más afecta al tiempo de recuperación

de la conexión. El control de acceso es demandado por los operadores de red con el objetivo de restringir el acceso a la red sólo a usuarios autenticados. Uno de los protocolos estandarizados más utilizados para el proceso de autenticación es el denominado *Extensible Authentication Protocol* (EAP). De hecho, motivado por las interesantes cualidades que posee el protocolo, EAP está adoptando una posición relevante como solución de control de acceso en las redes heterogéneas de próxima generación. En primer lugar, EAP ofrece un framework flexible que permite la ejecución de múltiples mecanismos de autenticación denominados *métodos EAP*. Estos métodos EAP son ejecutados entre un usuario móvil (*peer EAP*) y un servidor de autenticación (*servidor EAP*), a través de una entidad intermedia (*autenticador EAP*) que actúa como *Network Access Server* (NAS) controlando el acceso a la red. En segundo lugar, debido a que EAP es independiente de la tecnología subyacente empleada en la red de acceso, el protocolo es capaz de operar en un entorno inalámbrico heterogéneo. En tercer lugar, EAP puede ser fácilmente integrado con las infraestructuras de *Autenticación, Autorización y Accounting* (AAA) existentes, las cuáles han sido ampliamente desplegadas por los operadores de red para controlar a sus suscriptores, asistiendo no sólo los procesos de autenticación sino también de autorización.

Sin embargo, el proceso de autenticación EAP ha mostrado serias deficiencias cuando es aplicado en escenarios móviles. Por un lado, una autenticación EAP típica requiere un conjunto de mensajes considerable entre el peer EAP y el servidor EAP. Por ejemplo, un método EAP ampliamente usado como es EAP-TLS requiere, en el mejor de los casos, hasta ocho mensajes para autenticar al usuario. Por otro lado, la conversación EAP tiene lugar entre el peer EAP y el servidor EAP localizado en el dominio origen (*home*) del usuario donde éste está subscrito. En particular, en los escenarios de *roaming*, el servidor EAP podría encontrarse lejos del usuario móvil (peer EAP), lo que incrementa la latencia introducida por cada intercambio de mensajes. Estos inconvenientes adquieren mayor relevancia si tenemos en cuenta que una autenticación EAP debe ser ejecutada cada vez que el usuario realiza un handoff y que, hasta que no sea completada, el usuario no recuperará el acceso al servicio de red. Por lo tanto, este problema afecta a la comunicaciones en curso debido a que la autenticación EAP podría introducir una latencia prohibitiva (en ciertos casos hasta segundos) que se traduciría en una pérdida sustancial de paquetes y, en consecuencia, una degradación en la calidad del servicio prestado al usuario móvil.

Con el objetivo de disminuir el tiempo de autenticación EAP, es necesario definir un proceso de *re-autenticación rápida* para reducir tanto el tiempo dedicado a la transmisión de mensajes sobre la red como el número de mensajes necesarios para autenticar el usuario. El primer objetivo es logrado mediante la habilitación de un servidor de re-autenticación local, localizado cerca del usuario móvil, encargado de re-autenticar al usuario y así evitar contactar el servidor EAP home. Para satisfacer el segundo objetivo, los investigadores han acordado que un proceso rápido y seguro de *distribución de claves* es la forma correcta de re-autenticar al usuario con un número reducido de mensajes. Además, considerando que una autenticación EAP puede generar material criptográfico válido durante un periodo de tiempo, al proceso de re-autenticación rápida debería consistir de dos fases. Inicialmente, durante la *fase de bootstrapping*, el usuario móvil ejecutaría una autenticación EAP tradicional con el servidor EAP home. Esta autenticación inicial genera

material criptográfico que es empleado para habilitar un proceso eficiente de distribución de claves en una posterior *fase de re-autenticación rápida*. En concreto, para llevar a cabo esta distribución de claves de forma secura, se recomienda el uso de un modelo de distribución de tres partes.

Además de la necesidad de optimizar el control de acceso a la red basado en EAP, la protección de la *privacidad del usuario* es otro de los retos que deben ser afrontados por la próxima generación de redes heterogéneas. La privacidad es el derecho que tienen los usuarios de controlar quién conoce determinados aspectos sobre ellos, sus comunicaciones, sus actividades y, en última instancia, información sobre ellos mismos. Por lo tanto, la protección de la privacidad del usuario puede considerarse como el medio a través del cuál los usuarios controlan el acceso a su información privada. Debido a la naturaleza de las redes inalámbricas, un usuario malicioso podría capturar mensajes de cualquier comunicación activa que ocurra bajo su área de cobertura y tener acceso a información relacionada con distintos usuarios. Por lo tanto, dada esta situación, la privacidad es un aspecto relevante tanto para las aplicaciones como para los usuarios móviles en la próxima generación de redes heterogéneas. De hecho, la protección de la privacidad del usuario es un requisito esencial en dichas redes debido a que, sin mecanismos de habiliten una protección de la privacidad, los usuarios móviles podrían ser rastreados y su actividad revelada. En particular, el control de acceso a la red es un proceso donde la protección de la privacidad del usuario es de gran importancia. Durante la autenticación EAP, entre otra información, el usuario debe proporcionar su *identidad* con el objetivo de ser autenticado. Mediante el análisis de esta información, una atacante podría, por ejemplo, fácilmente tracear la localización de usuario móvil. Por este motivo, durante el control de acceso a la red, además de preservar el *anonimato* del usuario, el problema de la *trazabilidad* de un usuario anónimo también debe ser cuidadosamente estudiado.

En resumen, un importante reto debe ser afrontado en los futuros sistemas de control de acceso basados en EAP para redes NGN: **la definición de un proceso de distribución de claves rápido y seguro, que habilite un proceso de re-autenticación rápida a la vez que un acceso autenticado anónimo y que no se pueda trazar**. Precisamente, ésta es la problemática abordada en este tesis doctoral mediante el desarrollo de un sistema de control de acceso novel que ofrezca un conjunto de características que, hasta la fecha, no han confluido en una misma solución de re-autenticación rápida: *(1)* aplicable a las futuras redes NGN basadas en EAP; *(2)* reducción de la latencia introducida por el proceso de autenticación en entornos móviles, con independencia del tipo de handoff realizado por el usuario; *(3)* que el proceso cumpla fuertes requisitos de seguridad; *(4)* fácil despliegue en redes existentes; *(5)* compatibilidad con las actuales tecnologías estandarizadas; y *(6)* soporte de protección de privacidad del usuario.

La primera contribución de este tesis doctoral desarrolla una arquitectura genérica de transporte para re-autenticación rápida con el objetivo de reducir el tiempo dedicado al control de acceso a la red en las futuras redes NGN basadas en EAP. La solución es adaptada especialmente para asistir un acceso a la red rápido basado en un proceso de distribución de claves para re-autenticar al usuario. De forma más específica, la arquitectura define una nuevo método EAP denominado *EAP Fast Re-authentication*

Method (EAP-FRM) que trabaja en modo de configuración *standalone*, donde el servidor EAP es implementado por el autenticador EAP. Sin embargo, cuando es necesario, el método EAP-FRM puede contactar un servidor de autenticación externo usando un protocolo AAA como RADIUS o Diameter. EAP-FRM es una solución genérica capaz de transportar cualquier protocolo de distribución de claves que, en el contexto de EAP-FRM, es denominado *Fast Re-authentication Protocol* (FRP). Se asume que el FRP distribuye una clave secreta entre el peer y el autenticador. Esta clave es usada por el método EAP-FRM para generar material criptográfico como, por ejemplo, la *Master Session Key* (MSK) usada para proteger en enlace inalámbrico a través de una asociación de seguridad. EAP-FRM es una solución flexible que no compromete el proceso de re-autenticación rápida cuya eficiencia depende del FRP usado. Además, dado que se emplea el mecanismo de extensibilidad disponible en EAP (definición de nuevos métodos EAP), la arquitectura EAP-FRM no requiere modificación alguna sobre el protocolo EAP o las tecnologías inalámbricas existentes.

La segunda contribución de esta tesis doctoral integra EAP-FRM con un protocolo de distribución de claves específico. En particular, se ha escogido *Kerberos* como protocolo de distribución de claves de tres partes para definir una proceso de re-autenticación rápido y seguro. Además, Kerberos es un protocolo bien conocido que ha sido usado extensivamente para controlar el acceso a los recursos de red. Siguiendo estrictamente la especificación estandarizada del protocolo, Kerberos es integrado con EAP-FRM para controlar el acceso al servicio de red. En particular, el autenticador EAP actúa como *servicio de aplicación* ofreciendo acceso al servicio de red. Comparado con las soluciones existentes, la *arquitectura EAP-FRM Kerberizada* ofrece numerosas e importantes ventajas. En primer lugar, la solución es capaz de alcanzar un proceso de re-autenticación consistente en sólo tres mensajes intercambios entre peer EAP y autenticador EAP, por lo que no se requiere la participación de un servidor de autenticación externo. En segundo lugar, a diferencia de otras soluciones proactivas como pre-distribución de claves o pre-autenticación, las solución propuesta define un modo de operación proactiva que no requiere una pre-reserva de recursos en autenticadores candidatos. En tercer lugar, mediante el uso de Kerberos para realizar la distribución de claves, nos apoyamos en la madurez de este protocolo bien conocido y verificado. De este modo, evitamos la definición de un nuevo protocolo, lo cuál es un proceso complejo y propenso a errores (*error-prone*). Finalmente, gracias al modo de operación *cross-realm* de Kerberos, la solución soporta con gran flexibilidad entornos multi-dominio donde no se exige la existencia de una relación de confianza directa entre los dominios visitado y home.

La contribución final de esta tesis doctoral se centra en la protección de la privacidad del usuario durante el proceso de re-autenticación rápida basada en Kerberos. En general, los protocolos de distribución de claves, y en particular Kerberos, indican explícitamente en sus mensajes la identidad de las entidades que están autorizadas a usar la clave distribuida. En nuestro contexto, donde Kerberos es aplicado al control de acceso a la red, ello significa que la identidad del usuario queda expuesta durante el proceso de re-autenticación rápida. Con el fin de evitar los riesgos que esto introduce en la privacidad del usuario, se desarrolla una arquitectura novel denominada *Privacy Kerberos* (PrivaKERB) que preserva

la privacidad del usuario durante su actividad basada en Kerberos. PrivaKERB es un framework flexible que ofrece, por medio de tres niveles de privacidad, anonimato, no trazabilidad en los accesos a servicios y no trazabilidad en los intercambios de mensajes. Las mejoras definidas para Kerberos emplean los mecanismos de extensibilidad disponibles en el protocolo, por lo que la especificación estandarizada de Kerberos se mantiene intacta. Además, aquellas operaciones que requieren cierta asociación con el usuario específico (por ejemplo, la facturación de servicios consumidos) son soportadas por PrivaKERB. Todas estas características son ofrecidas introduciendo una sobrecarga casi despreciable sobre el protocolo base, lo que nos permite afirmar que el proceso de re-autenticación no es perjudicado. En particular, el framework de privacidad es compatible con EAP y puede ser fácilmente integrado con la autenticación EAP y la arquitectura de re-autenticación rápida basada en EAP-FRM y Kerberos.

En definitiva, el sistema de control de acceso propuesto en esta tesis doctoral satisface todos los requisitos mencionados anteriormente: *soporte de cualquier tipo de handoff* (gracias al uso de EAP, el cuál es independiente la tecnología subyacente), *proceso seguro* (gracias al uso de Kerberos, el cuál es un protocolo de distribución de claves basado en un modelo de tres partes, bien conocido y testeado), *bajo impacto de despliegue* y *no modificación de estándares existentes* (debido a que nuestra propuesta no requiere modificaciones en protocolos estandarizados y emplea los mecanismos de extensibilidad disponibles en EAP, protocolos AAA y Kerberos) y *soporte de privacidad* (mediante el uso de las extensiones de privacidad definidas en PrivaKERB).

Del trabajo desarrollado en esta tesis doctoral, diversas líneas de investigación afloran como trabajo futuro. Por un lado, podemos identificar desarrollos específicos cuyo fin es mejorar la arquitectura de re-autenticación rápida desarrollada. Por ejemplo, mientras que una mejora interesante para el método EAP-FRM sería el soporte de fragmentación, sería beneficioso permitir a la solución de re-autenticación rápida basada en EAP-FRM y Kerberos, interactuar con una infraestructura de autorización basada en tecnologías tales como SAML o XACML. De forma similar, la protección de privacidad proporcionada por PrivaKERB podría ser mejorada mediante, por ejemplo, la inclusión de soporte de privacidad para los servicios. De este modo, la identidad del servicio accedido (autenticador) no es revelada a entidades no autorizadas. Por otro lado, nuestras contribuciones sirven como referencia para futuras investigaciones. Por ejemplo, el establecimiento rápido de sesiones autenticadas es actualmente un problema de gran importancia en áreas como *redes vehiculares* (VANETs), *entornos federados* y dentro de grupos de estandarización tales como *IEEE 802.21*. Consideramos que los conceptos presentados en esta tesis doctoral constituyen un excelente punto de partida hacia la búsqueda de soluciones que consideren las particularidades de cada área. Del mismo modo, la protección efectiva de la privacidad del usuario durante el acceso a la red constituye un importante paso para alcanzar una solución integral de privacidad a todos los niveles de red (*cross-layer*).

Agradecimientos

He de reconocer que la emoción me invade al recordar todos los momentos y personas que han estado a mi lado durante el desarrollo de esta tesis. Sin duda, gracias a su apoyo hoy puedo estar escribiendo estas líneas. Espero que mi propuesta de agradecimientos sea capaz de reflejar esta realidad. No obstante, de antemano, me gustaría pedir disculpas a aquellas personas que, por mi despiste, deberían estar mencionadas aquí.

En primer lugar, no puedo sino dar las gracias a mis padres y mi hermana. Desde que tengo uso de razón, siempre me han ofrecido su apoyo incondicional, han estado a mi lado en los momentos difíciles y me han apoyado en todas mis decisiones. Por ello, no puedo más que estarles agradecido de por vida.

Otro pilar importantísimo ha sido *mi Almu*. La paciencia que ha demostrado tener esta mujer es infinita pues, ya no sólo me aguanta día a día, sino que ha soportado mis malos humores y agobios durante estos años de trabajo. Gracias por tu cariño y por hacerme tan feliz.

También quisiera dar las gracias a todas las personas que, de un modo u otro, me han acompañado a lo largo de este proceso. A mis compañeros de despacho, tanto a los que ya no están (Pepe Juárez, Alberto Caballero) como a los que les queda poco para doctorarse (Emilio, Juanjo, Maite y Alberto), por sus consejos y el buen clima de trabajo. Al gran Antonio Ruiz, quien me ha demostrado lo que es trabajar duro. Al increíble Fernando Bernal, cuya ayuda ha sido decisiva en el proceso de implementación y desarrollo de prototipos en esta tesis. A la gente de *Dibulibu*, en especial Alex, Pedro J. y Juan Antonio, gracias a los que me fueron más fáciles aquellos primeros meses de trabajo tras acabar la carrera. Y como no, mis colegas José Luis y Raúl, que aunque no me ven mucho el pelo, se acuerdan de mí continuamente.

Tampoco me puedo olvidar del estupendo trato que recibí en Samos. *Many thanks to the friends I met in Karlovassi for the good moments. A special mention deserves professor Georgios Kambourakis for his kind attention during my stay. The work developed in this small island of the Aegean Sea has been of vital importance on the completion of this thesis.*

Así mismo, quisiera expresar mi agradecimiento a mi director de tesis, Antonio. Él me abrió las puertas al mundo de la investigación y me brindó la oportunidad de trabajar junto a él en uno de los mejores grupos de investigación de esta Universidad.

En último lugar, aunque no por ello menos importante, quisiera agradecer a la **Fundación Séneca** (Agencia Regional de Ciencia y Tecnología) su vital apoyo económico mediante una *Beca-Contrato Predoctoral de Formación del Personal Investigador* (**FPI**).

University of Murcia

Faculty of Computer Science

DESIGN OF AN ACCESS CONTROL SYSTEM FOR HETEROGENEOUS NETWORKS BASED ON KERBEROS

PhD Thesis

Author:
Fernando Pereñíguez García

Thesis Advisors:
Dr. Antonio Fernando Gómez Skarmeta

Murcia, March 2011

Abstract

In recent years, wireless telecommunications systems have been prevalently motivated by the proliferation of a wide variety of wireless technologies, which use the air as a propagation medium. Additionally, users have been greatly attracted for wireless-based communications since they offer an improved user experience where information can be exchanged while changing the point of connection to the network. This increasing interest has led to the appearance of mobile devices such as smart phones, tablet PCs or netbooks which, equipped with multiple interfaces, allow *mobile users* to access network services and exchange information anywhere and at any time. To support this *always-connected* experience, communications networks are moving towards an *all-IP* scheme where an IP-based network core will act as connection point for a set of accessible networks based on different wireless technologies. This future scenario, referred to as the *Next Generation of Heterogeneous Networks* (NGNs), enables the convergence of different heterogeneous wireless access networks that combine all the advantages offered by each wireless access technology per se.

One of the most challenging aspects in NGNs is the mechanism through which the different technologies will complement each other to achieve a constant access to networks. Indeed, for the provision of high-quality multimedia services in NGNs, it is necessary to achieve smooth and seamless movements each time the mobile user changes its connection point to the network during the so-called *handoff*. Nevertheless, during the handoff, the connection to the network may for various reasons be interrupted, which causes a packet loss that finally impacts on the on-going communications. For this reason, in order to obtain seamless and faster movements between access networks and improve the quality of the service perceived by the user, an important objective rests on the reduction of the number of lost packets during the handoff process.

The handoff process requires the execution of several tasks that negatively affect the handoff latency. Obviously, by reducing the time required to complete each task, the total disruption time introduced by the handoff process will decrease and, consequently, so will the number of lost packets. In particular, in the context of this PhD thesis, we pay attention to the *network access control* process which has been demonstrated to be one of the most important factors that negatively affects the handoff latency. This process is demanded by network operators in order to restrict access to the network only to authenticated users. A common way of performing this process in wireless networks has been guaranteed by the deployment of the so-called *Extensible Authentication Protocol*

(EAP). In fact, EAP is acquiring an important position as the possible access control solution in future NGNs thanks to important features offered by the protocol. First, instead of proposing a specific authentication process, EAP offers a flexible framework that allows the definition of multiple authentication mechanisms (called *EAP methods*) which are executed between the mobile user (*EAP peer*) and the authentication server (*EAP server*). The process is performed through an intermediary entity (*EAP authenticator*) which acts as *Network Access Server* (NAS) controlling the access to the network. Second, EAP is independent of the underlying wireless access technology, and thus is able to operate in a heterogeneous wireless environment. Third, EAP allows an easy integration with existing *AAA infrastructures*, which are widely deployed by network operators and institutions to assist not only authentication but also authorization processes.

Nevertheless, the EAP authentication process has shown certain inefficiency in mobile scenarios. On the one hand, a typical EAP authentication involves considerable signalling between the EAP peer and the EAP server. For example, a widely used EAP method such as EAP-TLS requires up to eight messages to authenticate the user in the best case. On the other hand, the EAP conversation takes place between the EAP peer and the EAP server located in the peer's *home domain*, where the peer is subscribed. Specifically, in roaming scenarios, the EAP server may be far from the mobile user (EAP peer) and, therefore, the latency introduced per each exchange increases. These drawbacks acquire more importance considering that an EAP authentication must be performed each time a mobile user performs a handoff and must be successfully completed before gaining access to the network. Therefore, this problem can affect on-going communications since the latency introduced by the EAP authentication during the handoff process may provoke a substantial packet loss, resulting in a degradation in the service quality perceived by the user.

In order to decrease the EAP authentication time, it is necessary to define a fast re-authentication process to reduce both the time devoted to transmission over the network and the number of messages necessary to re-authenticate the user. The first objective is accomplished by enabling a *local re-authentication server*, placed near the mobile user and in charge of re-authentication, thus avoiding contacting the home EAP server to re-authenticate the user. To satisfy the second objective, researchers agree that a fast, secure key distribution process is the correct way to re-authenticate the user in a reduced number of messages. Furthermore, considering that a successful EAP authentication generates key material valid for a certain period of time, the fast re-authentication process should consist of two phases. Initially, during the *bootstrapping phase*, the mobile user performs a full EAP authentication with the home EAP server. This initial authentication generates a cryptographic material that is used to enable an efficient key distribution process in a posterior *fast re-authentication phase*. In particular, the use of a three-party key distribution model is recommended to carry out this key distribution in a secure manner.

Apart from the need to optimize the EAP-based network access control process, preserving the *privacy* of the mobile user is another challenging issue in NGNs. Privacy is the users' right to control who knows certain aspects about them, their communications,

their activities and, ultimately, the information about themselves. Privacy protection can thus be viewed as the means for users to control access to their private information. Nevertheless, wireless networks introduce new risks that could affect the privacy of the user. Due to the wireless networks' nature, a malicious user can eavesdrop on or capture messages from any active communication that takes place under its coverage area and gain access to the private information transmitted by such user. Therefore, given the openness of wireless networks, privacy is a serious concern for both emerging applications and mobile users in next generation wireless systems. In fact, the protection of user's privacy is important for NGNs, since without privacy-preserving mechanisms in place, mobile users can be easily tracked and profiled. In particular, the network access control is a process where the protection of the user's privacy is of great importance. During the EAP authentication process, the user is expected to provide its identity in order to be authenticated. By analyzing this information, an eavesdropper can, for example, easily track the location of a specific mobile user. For this reason, during the network access control, in addition to user anonymity, user untraceability must be assured so that eavesdroppers are unable to obtain anonymous profiles.

In summary, an important issue needs to be addressed in future EAP-based NGN access control systems: **the definition of a fast and secure three-party key distribution enabling an efficient re-authentication process while providing an authenticated anonymous and untraceable access**. This is precisely the problematic addressed in this PhD thesis by developing a novel access control system that offers a set of features not covered so far by a single fast re-authentication solution: *(1)* applicable for EAP-based NGNs; *(2)* reduction of the authentication latency in mobile environments irrespective of the type of handoff performed by the user; *(3)* provision of strong security properties; *(4)* easy deployment in current networks; *(5)* compatibility with current standardized technologies; and *(6)* user privacy support.

The first contribution of this PhD thesis develops a generic transport-based architecture aimed at reducing the time spent on providing network access in EAP-based mobile networks. The solution has been specially adapted to assist a fast network access based on a secure key distribution process to re-authenticate the user. More precisely, the architecture defines a new EAP method called *EAP Fast Re-authentication Method* (EAP-FRM) which works on a *standalone* configuration mode where the EAP server is also implemented by the EAP authenticator. Nevertheless, when required, the EAP-FRM method can contact a backend authentication server by using an AAA protocol such as RADIUS or Diameter. EAP-FRM is a generic solution able to transport any key distribution protocol which, in the context of EAP-FRM, is referred to as *Fast Re-Authentication Protocol* (FRP). The FRP is expected to distribute a shared secret key to both the peer and the authenticator. This key is used by EAP-FRM to generate keying material like, for example, the *Master Session Key* (MSK) used to protect the wireless link through a security association. EAP-FRM is a flexible solution that does not compromise the fast re-authentication process whose efficiency depends on the specific FRP in use. Furthermore, given that we have used the extensibility mechanism available at EAP which is the definition of new EAP methods, the EAP-FRM architecture does not impose any modification on either to EAP or existing

wireless technologies.

The second contribution of this PhD thesis integrates EAP-FRM with a specific key distribution protocol. In particular, we have chosen Kerberos as the secure three-party key distribution protocol required to provide a secure fast re-authentication process. Indeed, Kerberos is a well-known secure three-party key distribution protocol which has been extensively used to control access to network resources. Strictly following the standard protocol specification, Kerberos is integrated with EAP-FRM to control access to the network service where the EAP authenticator acts as Kerberos application server offering access to such service. Compared with existing proposals, the resulting *Kerberized EAP-FRM architecture* offers important advantages. First, the solution is able to achieve a fast re-authentication process consisting of only three messages between the peer and the authenticator that does not require communication with an external authentication server. Second, unlike other proactive techniques such as key pre-distribution or pre-authentication, the proposed solution defines a proactive operation mode that does not require any pre-reservation of resources in candidate authenticators. Third, by using Kerberos as key distribution protocol, we avoid the definition of a new key distribution protocol, which is a complex and error-prone process, and we rely on a mature and well-known secure three-party key distribution protocol. Finally, thanks to the Kerberos *cross-realm* operation, the solution handles with multi-domain scenarios in a flexible manner since it is not required the existence of a direct trust relationship between the visited and the home domains.

The final contribution of this PhD thesis focuses on protecting user privacy during the proposed Kerberos-based fast re-authentication process. In general, key distribution protocols, and Kerberos in particular, explicitly indicate in their messages the identities of the entities which are authorized to use the distributed key. When applied to enable fast network access, it means that the user's identity is exposed during the fast re-authentication process. In order to avoid the privacy risks that this situation may provoke, we design a novel privacy architecture called *Privacy Kerberos* (*PrivaKERB*) that preserves the privacy of the user during its Kerberos-based activity. PrivaKERB is a flexible framework offering anonymity, service untraceability and exchange untraceability through three different levels of privacy. The privacy enhancements to Kerberos are defined by using the extensibility mechanisms available at the protocol, thus not violating the standard Kerberos specification. Furthermore, operations that require some association with the specific user (e.g., charging) are supported within PrivaKERB. All these features are achieved with an almost negligible overhead to the standard protocol, and therefore do not jeopardize the fast re-authentication process. PrivaKERB is a general purpose solution not only applicable to our fast re-authentication solution but also to any system based on Kerberos. In particular, the privacy framework is fully compatible with EAP and can easily be integrated with EAP authentication and Kerberized EAP-FRM fast re-authentication architecture.

As a result, the proposed access control system accomplishes all the requirements previously mentioned: *support of any type of handoff* (thanks to the use of EAP, which is independent of the underlying technology), *strong security* (thanks to the use of Kerberos,

which is a well-known and tested three-party key distribution protocol), *low deployment impact* and *avoid standard modifications* (since our proposal does not require modifications in current standardized protocols and employs the extensibility mechanisms available in EAP, AAA protocols and Kerberos) and *privacy support* (by using the PrivaKERB privacy extensions).

The work developed within this PhD thesis gives a rise to other future research activities. On the one hand, we can identify enhancements to the privacy-enhanced Kerberized EAP-FRM architecture itself. For example, while an interesting improvement to the EAP-FRM method relies on fragmentation support, it would be beneficial to allow the Kerberized EAP-FRM fast re-authentication solution to interact with advanced authorization infrastructures based on well-known technologies such as SAML and XACML. Similarly, the privacy protection provided by PrivaKERB can also be improved, for example, by considering the identity protection of services so that the identity of an accessed service (authenticator) is not revealed to unauthorized parties. On the other hand, our contributions serve as reference guidelines for future researches. For example, the fast establishment of authenticated sessions is currently a problem of great importance in other areas such as *Vehicular Networks* (VANETs), *Federated Networks* and within the IEEE 802.21 standardization group. We consider that the concepts presented in this PhD constitute an excellent starting point for finding solutions that consider the specific particularities of each area. Similarly, the effective privacy protection during the network access control is an important step towards the definition of an integral cross-layer privacy solution.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of this introductory chapter is to establish the foundations and problematic which have guided the research process in the development of this PhD thesis. Initially, the chapter stresses the importance of *Next Generation Networks* (NGNs) given the rapid growth in the number of wireless applications, services and mobile devices, together with the increasing interest of mobile users in enjoying the *always-connected experience*. Although NGN networks offer a wide set of business opportunities to both users and networks operators, several challenges must be addressed in order to achieve a fully operational solution. In particular, one of the most important aspects is the assurance of service continuity through the definition of the concept of *seamless* handoff. This concept implies the reduction of the so-called handoff latency to avoid long disruptions in the on-going communications when a mobile user moves from one point of attachment to the network to another. In this regard, one aspect that negatively affects handoff latency is the time devoted to network access control. As will be discussed, this problem has not been ignored, since researchers are seeking mechanisms enabling fast network access. Nevertheless, current proposals neglect some aspects such as minimizing deployment impact, avoiding modification of current standards or providing privacy to the user.

The remainder of the chapter presents the essential objectives, as well as the main contributions of the (PhD) thesis. Additionally, the organization and distribution of the document are described in this introductory chapter by presenting the results obtained through some publications derived from the research work.

1.1 Next Generation of Heterogeneous Wireless Networks

Throughout to history, different communication methods have been developed in order to allow people interact. *Telecommunication systems* especially have attracted special interest since they allow the transmission of messages over long distances. The first breakthrough into modern electrical telecommunications came with the development of the telegraph in 1832. With this invention, Samuel F.B. Morse demonstrated the feasibility of sending

information via electromagnetic signals and opened the way to other popular systems like the telephone or the teletype. Nevertheless, these communication methods require the use of conducting metal wires and their use is limited to a certain perimeter. These problems were fixed by Marconi when, in the first decade of the 20th century, he performed the first transatlantic communication without cables, using the air as propagation medium. It is considered a historic milestone that enabled the development of the so-called *wireless telecommunications*.

Nowadays, telecommunication systems have a significant social, cultural and economic impact on modern society. We are immersed in a world of globalized information primarily supported by telematic communications performed over computer networks deployed around the world. Particularly interesting has been the evolution of wireless technologies, which have experienced an awesome evolution in the last decade. Users have shown increasing interest in wireless-based communications since they offer an improved user experience where information can be exchanged while changing the point of connection to the network. For this reason, an enormous variety of mobile devices like smart phones, laptops, tablet PCs and netbooks have recently appeared [1]. These new kinds of devices take advantage of the benefits offered by the various wireless access technologies and offer the *mobile users* the possibility of accessing networks services and exchanging information regardless their location.

Depending on the provided range of coverage and the areas of application, the wide variety of wireless technologies can be classified [2, 3] into four different groups (see Fig. 1.1): *Wireless Personal Area Networks* (WPANs), *Wireless Local Area Networks* (WLANs), *Wireless Metropolitan Area Networks* (WMANs), and *Wireless Wide Area Networks* (WWANs).

WPAN describes an application of wireless technology that is intended to address usage scenarios that are inherently personal in nature. The emphasis is on instant connectivity between devices that manage personal data or which facilitate data sharing between small groups of individuals. One example is synchronizing data between a PDA and a desktop computer. Another is spontaneous sharing of a document between two or more individuals. The nature of these types of data sharing scenarios is that they are ad-hoc and often spontaneous. Wireless communication adds value for these types of usage models by reducing complexity (e.g. it eliminates the need for cables). *Bluetooth* [4], *RFID* [5] and ZigBee [6] are some examples of commonly used wireless technologies to establish WPAN networks.

Conversely, WLAN technologies provide access to corporate network resources, shared applications or multimedia services (without jeopardizing user's mobility) within a defined region like an office building or campus. The most famous WLAN technology is *IEEE 802.11* [7], which is commercially named *WiFi*.

Unlike WLANs, WMANs provide coverage to a metropolitan area such as a city and the surrounding suburbs. Compared with WLANs, WMANs enable telecommunication operators to offer network connectivity over a wide area with a considerably lower deployment cost. Currently, the best-known wireless metropolitan area network is IEEE 802.16 [8], deployed in the WiMAX networks [9], which is an emerging technology.

Whereas WMAN addresses connectivity within a defined region like a city, WWAN addresses the need to stay connected while travelling outside this boundary. WWANs have been implemented by cellular technologies which enable wireless computer connectivity either via a cable to a cellular telephone or through PC Card cellular modems. Nowadays, GPRS [10], UMTS [11] and LTE [12] are the most widely deployed technologies by cellular operators.

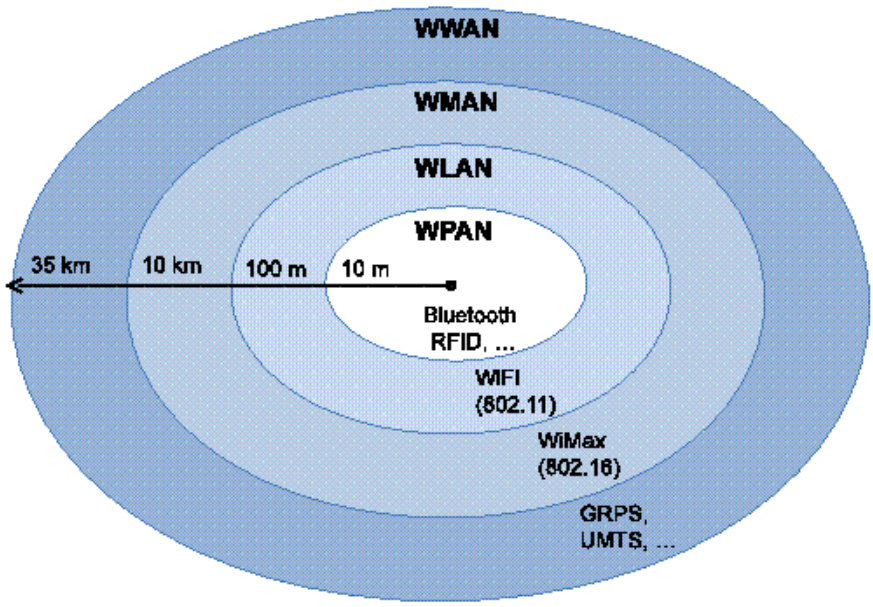


Figure 1.1: Wireless Technologies Classification

With the rapid growth in the number of wireless applications, services and devices, the use of a single wireless technology from the set of available technologies will not be a common case to deliver high speed data rate and to provide good quality of service (QoS) to mobile users in NGNs. In fact, the next generation of wireless systems are being devised with the vision of heterogeneity in which a mobile user/device will be able to connect multiple wireless networks seamlessly and simultaneously. For example, WiFi-based WLANs and WiMAX-based WMANs are already being integrated with UMTS-based WWANs and wireline networks to provide seamless broadband connectivity to mobile users in a transparent fashion. This new scenario, which represents the so-called *Next Generation of Heterogeneous Networks* (NGNs), enables the convergence of different heterogeneous wireless access networks in order to combine all the advantages offered by each link-layer technology.

In moving towards the establishment of a wireless-based universal access system, the use of the *Internet Protocol* (IP) has been considered as the tool to homogenize the communications performed over the different (wired and wireless) link-layer technologies. Therefore, users can communicate and access network services through IP without worrying about the underlying technology. This architecture, commonly referred as *all-IP* network configuration, envisages the deployment of an IP-based network core providing high

capabilities (e.g., bandwidth) and a set of access networks based on different wireless technologies (such as WiFi, WiMAX, GPRS or UMTS) which allow the provision of high-quality multimedia services such as Voice over IP (VoIP) [13].

The heterogeneity and complexity of NGNs bring about a number of challenges that must be tackled in order to achieve a fully operational environment. One of the most challenging aspects in NGNs is the mechanism through which the different technologies will complement each other to achieve constant and reliable network access. This continuity must be assured even when the user changes between different networks and administrative domains. To accomplish this, it is necessary to minimize the disruption time required to recover network connectivity when the user changes the connection point to the network (called *point of attachment*) during the so-called *handoff process*. Obviously, by reducing the number of lost packets during the handoff process, we obtain seamless and faster movements between access networks and improve the quality of the service perceived by the user.

Thus, to achieve mobility without interruptions, it is crucial to reduce the time required to complete the handoff. As described in [14], the handoff process requires the execution of several tasks that negatively affect the handoff latency. In particular, the authentication and key distribution processes have been proven to be one of the most critical components since they require considerable time [15 19]. The implantation of these processes during the *network access control* demanded by network operators is destined to ensure that only allowed users can access the network resources in a secure manner. Thus, while necessary, these security services must be carefully taken into account, since they may significantly affect the achievement of seamless mobility in NGNs.

At the same time, preserving the privacy of the user is another relevant issue that is attracting a good deal of attention from the research community. This problem has acquired enormous importance in NGNs since, due to the nature of wireless networks, an eavesdropper is able to trace the network activity of any user within its coverage area. Despite privacy being a broad concept that affects aspects such as sensitive information (*content privacy*) or the user's location (*location privacy*) transmitted over the network, the protection of identification-related data (*identification privacy*) has been revealed as one of the most critical [20]. If an eavesdropper can observe the identifier associated to the entity that initiates a certain transaction, it is relatively easy to determine the actions performed by a specific user and, ultimately, to monitor the user's activity.

In particular, this situation happens during the network access control. As previously mentioned, each time a user connects to a network, an authentication process is required in order to verify that the user is authorized to access the network. During the authentication, the user is expected to provide its identity to the authentication server. Thus, in order to avoid the user's activity monitoring, the authentication mechanism is required to provide an adequate level of privacy to protect the user's identity.

In this sense, this PhD thesis focuses on reducing the impact that a network access control involves during access to the network while preserving user privacy. In particular, we propose a novel approach for reducing the time devoted to completing the authentication process during the handoff and one which allows the user to remain anonymous and

untraceable throughout.

1.2 Mobility Scenarios in NGNs

In a typical NGN scenario users are expected to be potentially mobile. Equipped with wireless-based multi-interface lightweight devices like smartphones or tablet PCs, users will go about their daily life (which implies to perform movements and changes of location) while demanding access to network services such as VoIP or video streaming. In the *Deliverable D111 "Consolidated Scenario Description"* of the *IST DAIDALOS* project [21], *Mr. Bart M. Watson* perfectly illustrates how NGNs will change a user's life:

"Bart is having morning coffee and getting dressed while watching his personalized newscast on screens around the house - his new service follows him into every room that he enters - when a call from his boss Hector is signalled. Bart walks to the living room, as this is where external video calls are received by default and accepts Hector's call, who is urging him to come to the office prior to the briefing. He jumps up and enters his car. The vehicle automatically activates voice call. Also, the TV program he was watching is transferred but on hold during the voice call. He can resume watching it once he has finished the call. His boss informs him that he needs to pick up customer Rosalyn Royce at the airport."

As we can see, the concept of *mobility* demands session continuity when the user is moving across different networks. In other words, active communications need to be maintained without disruption (or limited breakdown) when the user performs a handoff and changes its point of attachment. As outlined in section 1.1, this aspect is of vital importance in the context of NGNs to allow the user to roam seamlessly between different networks without experiencing temporal interruption or significant delays in active communications. Seamless mobility is an essential property that must be achieved in the different types of handoff in which the mobile user may be involved. In particular, from the classification proposed by Nasser et al. [14] and later refined by Marin [22], it is especially important to consider the following types depend on several factors:

- *Wireless technologies.* Heterogeneity is the base on which NGNs are built. Users may have at their disposal different wireless networks, each based on a different transmission technology. Depending on the technology employed before and after the handoff, we can distinguish between:

  Intra-technology handoff. The handoff process performed by the mobile user takes place between point of attachments that support the same network technology. For example, the transition from an IEEE 802.11b access point to a geographically neighboring IEEE 802.11b access point is considered as a horizontal handoff process.

  Inter-technology handoff. In contrast, the handoff process of the mobile user is performed between points of attachment that support different network

technologies. For example, the transition from an IEEE 802.11b access point to an IEEE 802.16 base station is considered a vertical handoff process.

- *Administrative domains.* An administrative domain is a group of end systems and networks operated by a single organization or administrative authority [23]. Administrative domains are an important concept of NGNs since each wireless access network can be controlled by a different administrative authority. Consequently, the classification of handoffs in terms of administrative domains is a crucial issue. Since they usually provoke a longer handoff process, (in this PhD thesis) we try here to reduce this time during the network access control operations.

  Intra-domain handoff. In this kind of handoff, the mobile user roams between points of attachment managed by the same administrative domain.

  Inter-domain handoff. In this case, the mobile user roams between points of attachment controlled by different administrative domains.

- *Networks.* It is important to highlight that a handoff process may not imply a change of access network. Considering this, another criterion to classify the handoff focuses on the networks involved in the process.

  Intra-network handoff. The handoff happens between points of attachment that are deployed within the same network. That is, after the handoff is successfully completed, the user remains within the same IP network segment.

  Inter-network handoff. On the contrary, this kind of handoff supposes that the user roams between points of attachment located in different networks. In other words, once the handoff is finalized, the user changes to another IP network segment.

- *Security.* This type of handoff depends on the type of security required once the authentication process is completed with the new point of attachment. Within the network access service, two options can be considered: link-layer security and network-layer security. While the former strongly depends on the technology employed at link-layer, the latter is independent of the underlying technology. During a handoff, the user can experiment a change in the required security conditions, which lead us to distinguish between:

  Intra-security handoff. In this kind of handoff, the user changes between points of attachment which require the same security once the authentication process is successfully completed. For example, roaming between IEEE 802.11i [7] access points.

  Inter-security handoff. Conversely, in this kind of handoff the user changes to a point of attachment requiring different security from the previous one. For example, an inter-security handoff takes place when the user transits from an IEEE 802.11 [7] access point to an access router requiring the use of IPSec [24].

In general, mechanisms capable of providing seamless mobility in NGNs should be able to operate regardless of the type of handoff. This is specifically applicable to any fast re-authentication solution in NGNs. In this sense, these scenarios will be used as references in this PhD thesis when designing an optimization of the network access control process aimed at reducing the number a lost packets during the handoff.

1.3 The Network Access Control in NGNs

Network access control is a topic in which network operators have shown important interest. In fact, the deployment of wireless network technologies in public places bears the danger of unauthorized users gaining access to network services. For this reason, it is extremely important to restrict the access to the network only to authenticated and authorized users. Secure user authentication and authorization, which are essential processes for a reliable access control mechanism, are vital for wireless networks.

In NGNs, network access control systems are expected to integrate with the so-called *Authentication, Authorization and Accounting* (AAA) infrastructures [25]. This kind of infrastructure has been widely deployed by network operators and institutions nowadays [26]. Furthermore, AAA infrastructures are independent of the particular technology employed by the user to access the network service. In fact, this property is a key point that has motivated its adoption in future NGNs, where mobile users are expected to move across heterogeneous wireless access networks [27].

As the name suggests, AAA infrastructures provide three basic and essential network access control related services: authentication, authorization and accounting. These three important blocks are used by network operators and users in the construction of a network architecture that helps to protect against fraud, attacks or inappropriate resource utilization. These services, which are provided by the so-called *AAA servers*, are logically organized and each one has a specific purpose:

- *Authentication.* This service provides a means of identifying a user that requires access to some service (e.g., network access). Authentication logically precedes authorization. During the authentication process, users provide a set of credentials (e.g., password or certificates) in order to verify they are who they claim to be. If the credentials are correctly verified by the AAA server, the user is granted access to the network. If the credentials cannot be verified, authentication fails and network access is denied.

- *Authorization.* Authorization typically follows the authentication and entails the process of determining whether the client is allowed to perform and request certain tasks or operations. For example, after logging into a system, the user may try to issue commands. The authorization process determines whether the user has the authority to issue such commands. Authorization is the process of enforcing policies, determining what types or qualities of activities, resources, or services a user is permitted. Usually, authorization occurs within the (context of) authentication.

Once you have authenticated a user, they may be authorized for different types of access or activity.

- *Accounting.* The third component in the AAA framework is accounting, which measures the resources a user consumes during network access. This can include the amount of time a service is used or the amount of data a user has sent and/or received during a session. Accounting is carried out by gathering session statistics and usage information, and is used for billing, trend analysis, resource utilization and capacity planning activities.

To assist and provide these services, the AAA architecture defines a generic framework where a user interacts with an AAA server in order to secure access to a network service. The AAA specification [25] provides a guidance to design such architectures and gives some recommendations about how the AAA architecture can interact with other network management entities. In particular, these recommendations are concreted in [28] where it is described which specific requirements must be satisfied by an AAA protocol destined to be used in this kind of scenario. In any case, the AAA specification does not impose any specific protocol to perform each security service.

The heterogeneous nature of NGNs mandates that the authentication and key distribution processes will be performed across different technologies and in different kinds of handoffs, as explained in sections 1.1 and 1.3. This situation is motivating researchers to define a unique network access control solution independent of the underlying wireless technology. In fact, nowadays each wireless technology defines its own mechanisms to perform the authentication and key distribution processes. As in the adoption of IP as tool to homogenize the communications performed over different technologies in NGNs, it seems reasonable to adopt a universal network access control mechanism that is agnostic to the particular technology.

In this sense, we can find various alternatives to implement the authentication process during network access service by using IP-based protocols. Typically, some protocols used during the handoff like Mobile IP [29], SIP [30], IKEv2 [31] or PANA [32], provide some authentication mechanism to authenticate the user before enabling the network access service. These kind of protocols can transport authentication information from the mobile user to some entity in the network in charge of verifying the credentials presented by the user. This verification process can be performed locally or by interacting with a backend authentication server (e.g., an AAA server) in charge of the authentication and authorization operations for the network access service.

Since all these protocols operate over the IP layer, they are agnostic to the specific link-layer technology used to access the network and, therefore, can be considered as candidate solutions to define a universal network access control mechanism for NGNs. Nevertheless, these kind of authentication solutions are designed to work with the presence of IP network connectivity between the mobile user and the entity of the network that authenticates the user. However, in certain circumstances, the authentication process may need to work without the presence of IP network connectivity like, for example, in those

situations where the user must be authenticated at link-layer before gaining access to the network access service. This happens, for example, in WiFi networks.

To overcome this problem, the standardization body *Internet Engineering Task Force* (IETF) has designed the *Extensible Authentication Protocol* [33] (EAP). EAP exhibits some interesting features that clearly favour its adoption in NGNs. On the one hand, the protocol has been conceived to be independent of the underlying technology used to transport the protocol messages. This independence is of vital importance for NGNs since it allows the authentication and key distribution process to be done through different link-layer technologies or other IP-based protocols used in network access control such as PANA or IKEv2. On the other hand, EAP allows an easy integration in already deployed AAA infrastructures. Thanks to this feature, network operators that have deployed an AAA infrastructure in their domains, can easily deploy EAP as a solution to control the authentication process.

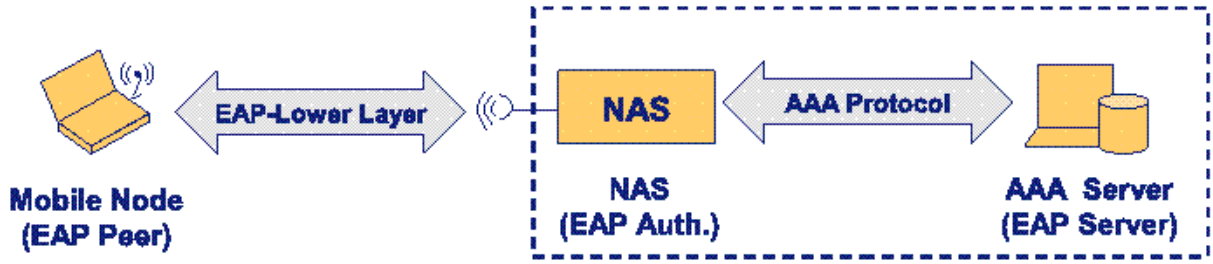


Figure 1.2: The EAP/AAA authentication model

Moreover, instead of proposing a specific authentication mechanism, EAP defines a flexible framework that permits different types of authentication mechanisms through the so-called *EAP methods*. EAP methods constitute the mechanism to extend the functionality of EAP without modifying the protocol itself. Specifically, EAP methods are performed between an *EAP peer* and an *EAP server* through an *EAP authenticator* which merely sends authentication information back and forth between the EAP peer and the EAP server. Whereas the EAP peer is co-located with the mobile node, the EAP authenticator is commonly placed on the so-called *Network Access Server* (NAS), which is in charge of interfacing with the peer to carry out the authentication (e.g., an access point or an access router). The EAP server can be placed with the authenticator (*standalone mode*) or more typically placed on an AAA server (*pass-through mode*). An *EAP lower-layer* is used to transported the EAP packets between the EAP peer and the EAP authenticator and an AAA protocol is used for the same purpose between the EAP authenticator and the EAP server in the pass-through mode (see Fig. 1.2). On the one hand, existing EAP lower-layers are IEEE 802.1X [34], IEEE 802.11 [7], IEEE 802.16e [35], PANA [32] and IKEv2 [31]. On the other, RADIUS [36] and Diameter [37] are the most well-known AAA protocols.

Apart from this flexibility, some EAP methods not only provide authentication, but are also able to generate key material for posterior use once the EAP authentication has been successfully completed. In particular, some of this cryptographic material like the *Master Session Key* (MSK) and *Extended Master Session Key* (EMSK) [38] can be used to protect the wireless link between the mobile user and the authenticator. To do this, it is first necessary to perform a *key distribution process* in order to provide some cryptographic material to the peer and authenticator which allows the execution of a *security association protocol* (e.g., 4-way handshake in IEEE 802.11 [7] at link-layer or IKEv2 [31] at IP-layer) that will protect the wireless communication.

All these properties make EAP a promising authentication framework for heterogeneous networks. In fact, different standardization organizations have used EAP when developing wireless technologies and protocols for network access control. For example, in WiFi-based WLAN networks, the IEEE 802.11 [7] specification defines an authentication system that uses EAP as mechanism to verify the user's identity. Similarly, for WMAN networks using the WiMAX technology, the IEEE 802.16e [35] specification relies on EAP to develop an integrated authentication mechanism. Also the use of EAP at application layer has been considered [39] to define a federated authentication and authorization architecture to control the access to network services (web and non-web services) by using a single user credential. Even more, the *Third Generation Partnership Project* (3GPP) has also considered the use of EAP to develop systems that allow the interworking between cellular technologies and wireless broadband technologies. In particular, an extensive research work can be found [40 45] on interworking of 2G/3G technologies with WLAN networks. There are basically two kinds of architectures for 3G-WLAN interworking: *tightly coupled* and *loosely coupled*. In a tightly coupled architecture, the 3G networks consider WLANs as their access networks, while WLANs are deployed as separate access networks in a loosely coupled architecture. The 3GPP standardisation organization has been developing a 3G-WLAN interworking architecture [46] based on a loosely coupled architecture. However there is another effort to standardize for a tightly coupled architecture such as the *Unlicensed Mobile Access* [47] (UMA) technology. Regardless of the type of architecture, EAP is used to authenticate the user through the WLAN with the same credentials used in the 3G network. This authentication process can be performed thanks to EAP methods like EAP-SIM [48] and EAP-AKA [49] which are based on the authentication mechanisms adopted by 2nd and 3rd generation of mobile networks, respectively. Indeed, the EAP-AKA method has also been proposed by recent works to design interworking architectures not only for 3G-WiMAX [50, 51] but also between new 3GPP radio technology (LTE) and non-3GPP access networks [52].

1.4 The Problems of EAP in Mobile Environments

As we can observe, EAP is being widely used as an authentication protocol due to its flexibility, wireless technology independence and integration with AAA infrastructures. However, EAP has shown some drawbacks when mobility is taken into consideration. The

reason why the EAP authentication process is not so optimized for mobile scenarios is due to two main motives. First, it involves several round-trips for it be completed. Depending on the EAP method in use [53], this number can vary. For example, one of the most common methods, EAP-TLS [54], involves up to eight messages between peer and server to complete, in the best case. Secondly, each round-trip is performed with the EAP server placed on the EAP peer's home domain, where the peer is subscribed. Especially in roaming scenarios, the EAP server may be far from the mobile user (EAP peer) and, therefore, the latency introduced per each exchange increases. These issues are raised when an EAP peer moves from one authenticator to another (*inter-authenticator handoff*). In this case, the peer needs to perform an EAP authentication with the EAP server, through the new EAP authenticator. Therefore, every time the EAP peer moves to a new EAP authenticator, it may suffer from high handoff latency during EAP authentication.

This problem can affect the on-going communications since the latency introduced by the EAP authentication during the handoff process may provoke a substantial packet loss, resulting in a degradation in the service quality perceived by the user. In this sense, the performance requirements of a real-time application will vary according to the type of application and its characteristics such as delay and packet-loss tolerance. The ITU-T G.114 [55] recommendation indicates, for Voice over IP applications, an end-to-end delay of 150 ms as the upper limit and rates 400 ms as a generally unacceptable delay. Similarly, a streaming application has tolerable packet-error rates ranging from 0.1 to 0.00001 with a transfer delay of less than 300 ms. As has been proved in [15], a full EAP authentication[1] based on a typical EAP method such as EAP-TLS [54] can provoke an unacceptable handoff interruption of about 600 milliseconds (or even in some cases several seconds) for these kind of applications.

1.4.1 Overview of Existing Solutions

In the literature we can find a wide set of proposals that have sought to reduce the latency introduced by the EAP authentication during the network access control process. According to the strategy with which each proposal has been designed, the different solutions can be classified in five main groups:

- *Context Transfer*. Fast re-authentication solutions based on this mechanism [18,56 59] transfer the security context established with the current authenticator to a new authenticator where the user expects to move in the near future. When the user moves to the new authenticator, it can reuse the same context (e.g., cryptographic keys and associated lifetimes) established with the previous authenticator. Therefore, the user does not need to be authenticated and a security association between user and authenticator to protect the wireless link can be established using the transferred cryptographic material. Depending on when the transference is performed, we can distinguish between *reactive* and *proactive* schemes. While reactive solutions perform

[1] Note that the term *full* is used in comparison with *reduced* to denote that in the execution of an EAP method there is no optimization to reduce the number of exchanges during the EAP authentication.

the context transfer after the handoff, proactive solutions anticipate the process and perform it before the movement.

- *Pre-authentication.* Pre-authentication proposes a scheme where the peer starts an authentication process with a candidate authenticator through the current authenticator *before* it performs the handoff. Once the authentication is successfully completed, the peer roams to the candidate authenticator, where the key material will be already present due to the previous pre-authentication process. Therefore, the time devoted to the authentication is decoupled from the handoff process, allowing the former not to affect the latter. Similarly to the previous schemes, the network access control only consists in establishing a security association with the new authenticator. So far, solutions propose to perform the pre-authentication process either at link [60] or network [15, 17, 61] layer.

- *Key Pre-distribution.* This approach [62 64] adopts a proactive strategy to reduce the EAP authentication time. More precisely, this technique proposes the pre-installation of a key in those authenticators where the peer might move in the near future. After that, through a key pre-installation process, these authenticators where the user can potentially associate to, are provided with a key. In this way, when the peer roams to a new authenticator, the cryptographic material which allows the establishment of the security association is already present, thus avoiding contacting an authentication server.

- *Use of a local server.* Under current EAP authentication model [33], any authentication/re-authentication process requires a full EAP conversation with the EAP server located in the user's home domain. This introduces considerable latency, especially when the mobile user is moving across a foreign domain far from the home domain. To mitigate this problem, a group of solutions [64 67] define the existence of a local server for fast re-authentication operations. This local server is typically located in the domain the user is visiting (*visited domain*) and is in charge of the re-authentication process.

- *Modifications to EAP.* Other attempts to achieve a lightweight authentication process have proposed the modification of the EAP protocol itself. Existing solutions like [16, 68] try to optimize the protocol itself in order to achieve a fast EAP authentication process, not only by reducing the number of required message exchanges between the EAP peer and server but also taking advantage of the local server feature. To achieve this objective, extensive modifications to the EAP packet format and EAP state machines are required.

Ideally, the set of solutions presented previously, should be able to reduce the authentication latency in mobile environments, irrespective of the type of handoff performed by the user. This is an important requisite since, if a fast re-authentication mechanism is only able to handle certain types of handoffs, the seamless mobility requisite is only accomplished in a small number of possible scenarios.

Unfortunately, this general requisite is not typically satisfied by existing solutions. In particular, context transfer solutions have shown that, in practice, the transference of cryptographic material is feasible between authenticators using the same technology and managed by the same operator. The last restriction causes inter-network handoffs to be supported only when these networks are managed by the same operator. Nevertheless, this requirement may not be accomplished in inter-domain handoffs since, by definition, this kind of handoff takes place between authenticators belonging to different domains not administered by the same network operator. Additionally, context transfer solutions are typically dependent on the underlying technology, with 802.11 networks being the most widely studied [56, 57, 59] wireless scenario. As a consequence, inter-technology handoffs are not well supported. Finally, the context transfer mechanism has an important security vulnerability known as the *domino effect* [69] according to which, if one authenticator is compromised, the rest of authenticators visited by the user are also affected.

Regarding pre-authentication solutions, there are proposals which operate at link-layer and, therefore, are dependent on the specific underlying technology. Moreover, they only work between authenticators located within the same network. Therefore, only intra-technology, intra-network, intra-domain and intra-security handoffs are supported. On the other hand, network layer pre-authentication mechanisms are applicable to the different types of handoffs. The only exception is found in inter-domain handoffs. Although from a technical point of view, network-layer solutions can afford these kinds of handoffs, in practice, network operators are reluctant to allow foreign users to reserve some resources before the user establishes connectivity with the domain. As with link-layer solutions, network-layer mechanisms are mostly compatible with standardized wireless technologies since they propose to perform a regular EAP authentication before the handoff by using the corresponding standard EAP lower-layer. Nevertheless, a critical aspect in both types of pre-authentication techniques is the precise selection of the authenticator with which to perform a pre-authentication process and the pre-reservation of resources in the network operator side.

Alternatively, key pre-distribution solutions require the participation of an auxiliary third party in charge of pre-installing keys in candidates authenticators where the user can potentially move in the near future. Like with pre-authentication, a key aspect relies on the correct selection of those authenticators where cryptographic material must be distributed. To avoid unnecessary use of resources, an effective selection mechanism is required that identifies the set of authenticators where the peer, with a high probability, will move in the short term. Generally, these selection algorithms need information about both the user's movements and the network topology. While in intra-domain scenarios this information can be supposed to be available, in inter-domain scenarios this information is not typically available since operators are usually reluctant to share network topological information with other organizations. The earliest solutions [62, 70] proposing the use of key pre-distribution have been specifically designed for 802.11 networks. Nevertheless, recent solutions coined within standardization organizations [63, 64] have been designed on the basis of EAP and are technology independent. Even so, the key pre-distribution implies some interface in the NASes which allows the third-party to *push* a key. However, nowadays typical NASes

do not provide these kind of interfaces, which should be standardized.

A key aspect of solutions enabling a local server in charge of re-authentication tasks relies on the establishment of a trust relationship between the user and the local server placed in the visited domain. Although this technique supports most of the handoff types, inter-domain handoffs are supported depending on whether this relationship can be established before the handoff [64] or not [66]. Moreover, existing solutions follow a two-party key distribution model to distribute a key from the home to the local AAA server, which is known to be inappropriate from a security standpoint [71]. Deployment impact of these solutions is not envisaged as highly difficult since they generally employ available AAA protocol extensions to distribute a key to the local re-authentication server.

A final group of solutions has made efforts in modifying the EAP protocol itself [68] to enable a fast re-authentication process. Thus, since they are defined at the EAP level, these solutions are able to support every type of handoff. In general, these proposals achieve a fast re-authentication process by distributing keys which are derived from a previous full EAP execution [65]. This key distribution process will provide keys to both the mobile user and the authenticator from a trusted server and will avoid running a lengthy full EAP authentication. Additionally, these proposals also take advantage of a local server optimization. Nevertheless, these mechanisms have been typically criticized because they modify the EAP protocol specification and require considerable modifications to the existing wireless technologies and protocols which are using EAP nowadays. This creates an important deployment barrier for these types of solutions.

1.4.2 Design Goals

In order to decrease the EAP authentication time, it would be beneficial to reduce the number of messages involved during the EAP authentication and to reduce the processing time devoted to each message. Additionally, it may be useful to enable a *local re-authentication server* placed near the mobile user. This server would be in charge of re-authentication in the domain the peer is moving towards, thus enabling a re-authentication process without contacting the home server. Moreover, an ideal solution would not require changing any existing standard or wireless technology.

Taking into account that the execution of an initial EAP authentication generates key material valid for a certain period of time, it has been recognized [65] that the execution of an initial and single process involving a full EAP authentication can generate cryptographic material that can be used to enable more efficient accesses in future handoffs, without requiring the execution of full EAP authentications. This initial step, commonly referred to as *bootstrapping phase*, precedes the *fast re-authentication phase* where, through a secure key distribution process: (a) it is verified that the mobile user was successfully authenticated and, therefore, owns valid cryptographic material to access the network; and (b) it generates valid keys which are distributed to both the peer and the authenticator, through a key distribution protocol. These keys are useful in establishing a security association in the wireless access network. In other words, the deployment of a secure key distribution protocol which uses initial cryptographic material derived from an initial

full EAP authentication is a requirement for fast network access [65].

On the basis of this approximation, the designed fast re-authentication solution must accomplish the following requirements and aims [65]:

(D1) *Low latency operation.* The fast re-authentication mechanism must reduce the authentication time executed during the network access control process compared with a traditional full EAP authentication. Furthermore, the achievement of a reduced handoff latency must not affect the security of the authentication process.

(D2) *EAP lower-layer independence.* Any keying hierarchy and protocol defined must be independent of the lower-layer protocol used to transport EAP packets between the peer and the authenticator. In other words, the fast re-authentication solution must be able to operate over heterogeneous technologies, which is the expected scenario in NGNs. Nevertheless is allowed to, in certain circumstances, the fast re-authentication mechanism could require some assistance from the lower layer protocol.

(D3) *Compatibility with existing EAP methods.* The adoption of a fast re-authentication solution must not require modifications to existing EAP methods. In the same manner, additional requirements must not be imposed on future EAP methods. Nevertheless, the fast re-authentication solution can enforce the employment of EAP methods following the *EAP Key Management Framework* [38].

(D4) *AAA protocol compatibility and keying.* Any modification to the EAP protocol itself or the key distribution scheme defined by EAP, must be compatible with currently deployed AAA protocols. Extensions to both RADIUS and Diameter to support these EAP modifications are acceptable. However, the fast re-authentication solution must satisfy the requirements for the key management in AAA environments [38,69].

(D5) *Compatibility with other optimizations.* The fast re-authentication solution must be compatible with other optimizations destined to reduce the handoff latency already defined by other standards like, for example, the pre-authentication defined in IEEE 802.11 [7].

(D6) *Backward compatibility.* The system should be designed in such a manner that a user not supporting fast re-authentication should still function in a network supporting fast re-authentication. Similarly, a peer supporting fast re-authentication should still operate in a network not supporting the fast re-authentication optimization.

(D7) *Low deployment impact.* In order to support the aforementioned design goals, a fast re-authentication solution may require modifications in EAP peers, authenticators and servers. Nevertheless, in order to favour the protocol deployment, the required changes must be minimized (ideally, they should be avoided) in current standardized protocols and technologies.

(D8) *Support of different types of handoffs.* The fast re-authentication mechanism must be able to operate in any kind of handoff regardless of whether it implies a change of technology (intra/inter-technology), network (intra/inter-network), administrative domain (intra/inter-domain) or type of security required by the authenticator (intra/inter-security).

1.4.3 Security Goals

While EAP authentication follows a two-party model [38] which only considers the EAP peer and the EAP server, a key distribution process involves three parties: a *peer* requesting a *key distribution server* for the distribution of a key to an *authenticator*. The two-party model is valid for mutual authentication but is inappropriate for key distribution between three parties, especially in mobility scenarios [71]. In fact, the problems of using a two-party model for key distribution have already been analyzed in [71] and the related security threats in [72]. Therefore, from the key distribution standpoint, a three-party approach provides strong security properties [71].

In addition to the application of a three-party key distribution model, a secure key distribution process must accomplish the following security goals [69]:

(S1) *Authentication.* This requirement mandates that a management and key distribution mechanism must be designed to allow all parties involved in the protocol execution to authenticate every entity with which it is communicating. That is, it must be feasible to gain assurance that the identity of the another entity is as declared, thereby preventing impersonation. To carry out the authentication process, it is necessary to define the so-called *security associations* between the involved entities.

(S2) *Authorization.* During the network access control process, the user is not only authenticated but also authorized to access the network service. The authorization decision is taken by the AAA server and the result is communicated to the authenticator. The fast re-authentication solution proposed must not hinder the authorization process performed once the user is successfully authenticated.

(S3) *Key context.* This requirement establishes that any key must have a well-defined scope and must be used in a specific context for an intended used (e.g., cipher data, sign,...). During the time a key is valid, all the entities that are authorized to have access to the key must share the same key context. In this sense, keys should be uniquely named so that they can be identified and managed effectively. Additionally, it must be taken into account that the existence of a hierarchical key structure imposes some additional restrictions. For example, the lifetime of lower-level keys must not exceed the lifetime of higher-level keys.

(S4) *Key freshness.* A key is fresh (from the viewpoint of one party) if it can be guaranteed to be recent and not an old key being reused for malicious actions by either an attacker

or unauthorized party [73]. Mechanisms for refreshing keys must be provided within the re-authentication solution.

(S5) *Domino effect.* In network security, the compromise of keys in a specific level must not result in compromise of other keys at the same level or higher levels that were used to derive the lower-level keys. Assuming that each authenticator is distributed a key to carry out the fast re-authentication process, a key management solution respecting this property will be resilient against the *domino effect* [69] attack, so the compromise of one authenticator must not reveal keys in another authenticators.

(S6) *Transport aspects.* The solution developed must be independent of any underlying transport protocol. Depending on the physical architecture and the functionality of the involved entities, there may be a need for multiple protocols to perform the transport of keying material between entities involved in the fast re-authentication architecture. As far as possible, protocols already designed and used should be used to address the cryptographic material distribution. For example, while AAA protocols can be considered for this purpose between the EAP authenticator and server, the EAP protocol can be used between EAP peer and server.

Unlike other solutions, like [16] and [68], which are based on the design of a new key distribution protocol to carry out the fast re-authentication phase, we have opted for a well-known standardized secure three-party key distribution protocol: Kerberos [74]. The reason, already stated in [75], is motivated because the design of any cryptographic protocol and, in particular, a security key distribution protocol is usually complex and error-prone. In this sense, Kerberos is a widely deployed solution used to control the access to network resources such as printers or web servers. In particular, as will be described in chapter 4, we propose the application of Kerberos to control the access to the network service offered by authenticators.

The strong security provided by the Kerberos protocol, which has been perfectly demonstrated by both its extensive deployment and security studies [76 78], is one the main reasons we have selected this protocol. In particular, as we briefly analyze below, Kerberos is able to accomplish the security requirements previously mentioned.

Basically, Kerberos is a secure three-party key distribution establishment protocol used by a client and an application server (also referred to as *service*) to establish a *session key* generated by a trusted third-party, named *Key Distribution Center* (KDC). Based on pre-established trust relationships between client $\Leftrightarrow$ KDC and service $\Leftrightarrow$ KDC, the protocol defines an authenticated key distribution process [76] where the user must prove identity only once by using long-term credentials such as a password or a certificate (*Req. S1*). While Kerberos does not, by itself, manage authorization, the protocol has been conceived taking into account that once the user is successfully authenticated, an authorization process is necessary to determine under which conditions access is granted to a service. For this reason, the Kerberos protocol can be easily integrated with authorization methods (*Req. S2*). Indeed, Kerberos messages contain a specific field devoted to transporting authorization data between the participant entities.

The KDC performs the key distribution process to the client and service by different procedures. On the one hand, the distribution to the client takes place using a dynamic trust relationship established between client and KDC after an initial authentication. On the other, keys are distributed to services by means of the so-called *tickets*. A ticket is a piece of information, consumed by services, encrypted and integrity protected using a key shared only by the service and the KDC. In any case, Kerberos defines a precise context for the distributed keys (*Req. S3*) indicating, for example, validity time and entities to which the key is distributed. Additionally, Kerberos provides a mechanism to assure the freshness of the distributed key (*Req. S4*). While the freshness of the key distributed to the client is verified through pseudo-random numbers (*nonces*), timestamps are used within the ticket processed by services.

Since the keys distributed by the KDC are cryptographically independent, we can affirm that Kerberos is not vulnerable to the domino effect (*Req. S5*). Nevertheless, Kerberos has also proved to be resilient against other attacks such as impersonation or replay attacks by inductive reasoning [77] or using formal verification methods [78]. Finally, Kerberos also satisfies *Req. S6* since it is independent of the protocol used to transport the protocol messages. Although traditionally, Kerberos messages have been transported using TCP or UDP, other means can be used to carry the messages, as we will analyze.

1.5 Preserving User Privacy during the Network Access Control Process

Within NGNs, privacy is a key aspect [79,80] and represents a complex problem that can affect different network layers [81 86]. So, for example, a user can be traced by the use of the same IP or link-layer address when accessing a network service [81 83]. The same happens at application level [84,86], where many network applications need to take care in preserving user privacy through application-specific solutions. Therefore, the protection of user's privacy may become an aspect of vital importance for NGNs since, without privacy-preserving mechanisms in place, the user can be easily tracked and profiled in the mid or long term.

In general, privacy affects aspects such as location and identification [20]. While location privacy requires that the location of a mobile user is untraceable to unauthorized parties (including the network), identification privacy mandates users anonymity, except, maybe, for authorized parties. As we can see, these aspects are interrelated. If the user's identity is unknown, then location data is useless. At the same time, both types of privacy strongly depend on the authentication process, where user's identity must be exchanged and verified. If the authentication mechanism does not have an adequate level of privacy to protect identification related data, the location can be revealed to unauthorized third parties.

Thus, preserving user privacy during the network access control is a problem of great importance in the context of NGNs. Taking into account the particularities of the wireless

environments, the following privacy risks can be identified:

- *Activity monitoring.* The use of wireless interfaces allows a malicious entity to eavesdrop or capture all the packets sent and received by a user present within the attacker's coverage area. The analysis of this information gives access to valuable information about the user's activity (e.g., accessed services) or even predicts their location [87]. If the user's identity is not revealed, attackers cannot link this sensitive information with the specific user.

- *Movement tracking.* Mobile users are expected to change their location frequently and move across different domains. If the user's identity is exposed during the network access control, the location of the mobile user can be easily traced by unauthorized parties.

- *User profiling.* When mobile users visit domains others than the home domain, special care must be taken when revealing sensitive information about the user. Since these domains will not always be known or trusted beforehand and they manage active communications of the user, the user can be profiled in order to find out preferences or behaviour patterns. This aspect becomes more important taking into account that different domains can share the information they have collected in order to determine a more precise profile. Therefore, the user's identity must be inaccessible not only to eavesdroppers but also to the visited network.

Therefore, an important issue needs to be addressed in future NGN access control systems: the provision of an *authenticated anonymous access*. Taking into account that authentication is a process where users need to provide their identity, a privacy-enhancing mechanism must be defined in order to allow users to get authenticated but avoiding the disclosure of identification-related data to unauthorized parties.

1.5.1 Related Work

The user privacy protection during the authentication process is a problem that has not been completely ignored by researchers. As surveyed in [81], a wide set of authentication protocols for wireless communication networks have been proposed where *user anonymity* [80] is integrated in the protocol design. For example, references [88] and [89] define two different authentication schemes where user anonymity is preserved by using a pseudonym that does not change. This problem is solved by [90] and [91], which incorporate a pseudonym renewal feature to allow the mobile user to remain untraceable. Nevertheless, all these authentication protocols do not consider any lightweight procedure to avoid contacting the home domain in each execution. This drawback is fixed in [92], which defines a re-authentication process while maintaining user anonymity. However, none of these solutions have been designed to be deployed in EAP networks and, therefore, they do not consider the privacy aspects related with EAP authentication or, in particular, any EAP-based fast re-authentication mechanism. As described in section 1.3, EAP is

acquiring an important position to become the standard authentication solution to be adopted by future wireless networks. For this reason, it seems reasonable to work on a privacy-enhanced authentication solution that can be applied to EAP-based networks.

It is worth noting that, in the context of EAP, we can find some research works addressing the issue of user privacy. Indeed, there exists EAP methods that define mechanisms to provide anonymity during the authentication process. For example, the last revision of EAP-TLS [54] has a special privacy extension that allows the peer's certificate to be sent within a TLS session providing confidentiality. Conversely, EAP-SIM [48] (based on the 2nd generation mobile network standard) and EAP-AKA [49] propose a solution where the user uses different types of pseudonyms that are frequently renewed. However, despite defining some procedure to minimize the signalling, they suffer from their inefficiency to provide a fast re-authentication process.

1.5.2 Privacy Goals

The intention of this PhD thesis is to develop an EAP-based access control system for NGNs not only able to provide a fast network access operation but which also respects user privacy. As described in section 1.5.1, to the best of our knowledge, this is an aspect that has not been covered by previous privacy proposals in the field of EAP authentication and access control.

Certainly, we are facing an interesting challenge, since the authentication process requires the user to provide some kind of *identifier* (e.g., a username) that allows it to be distinguished from other users. This characteristic also occurs in existing key distribution protocols [93], whose messages typically include the identity of the entities involved in the protocol execution in order to define a proper key context. In particular, the standard Kerberos protocol [74] (which, as described in section 1.4.3, is used here to carry out the key distribution) suffers from the lack of a mechanism to preserve user privacy.

Indeed, Kerberos identifies the different participant entities through the so-called Kerberos *principal identifiers*, which have the form of "principal_name@realm_name". The *principal_name* part unequivocally identifies an individual user in the administrative realm specified in the *realm_name* part. However, these principal identifiers are transmitted in cleartext in Kerberos. Exposure of this sensitive information to unauthorized third parties allows for several malicious acts that clearly violate the user's private sphere. The most obvious is that an eavesdropper is able to obtain access to the user's real identity and observe which services are being accessed, so violating the principle of *user anonymity* [80].

It turns out that, even in the hypothetical case that the client's real identity may remain unknown, an eavesdropper may obtain some general information about behavioural patterns of network accesses of specific anonymous users. In the particular case of Kerberos, the reason is that a user typically uses the same ticket to initiate multiple key distribution processes. As a consequence, an eavesdropper can determine that the same (anonymous) client is involved in different executions of the protocol merely by tracing the use of the same ticket and, therefore, *user untraceability* is not accomplished [80].

At the same time, it is reasonable to mandate that the privacy-enhancing mechanism

applied to the key distribution protocol must not affect the fast re-authentication process. To achieve this, two different objectives need to be accomplished. On the one hand, the privacy extensions enabling user anonymity and untraceability must be adapted to the EAP authentication model described in [33]. On the other, the privacy mechanism must not impact on the fast re-authentication process by introducing an additional latency that makes it unaffordable in NGN environments.

In summary, an adequate mechanism providing user privacy during the network access authentication should be designed that accomplishes the following requirements:

(P1) *User anonymity.* The privacy mechanism must allow users to remain anonymous during the authentication process not only to eavesdroppers but also to the visited network. Only trusted entities in the home network can have access to the user's real identity.

(P2) *User untraceability.* The privacy mechanism must make it impossible for eavesdroppers to trace the network access activity of a certain user. Therefore, the different networks visited by the users cannot be inferred by analyzing the network access authentication processes performed by users.

(P3) *Applicable during an EAP-based authentication.* The privacy mechanism must be able to provide user anonymity and untraceability during a network access control based on EAP as long as it respects both the EAP protocol specification described in [33] and the EAP KMF guidelines defined in [38].

(P4) *No jeopardizing of the fast re-authentication process.* The privacy mechanism must not impact on the fast re-authentication process. In other words, the additional latency required to execute the privacy-related operations must be affordable to achieve a lightweight authentication procedure.

1.6 Objectives and Main Thesis Contributions

Despite the important position of EAP as framework to carry out the authentication and key distribution processes in NGNs, it unfortunately (as described in section 1.4) shows certain deficiencies when applied in mobile environments. In general, the main reason is that a full EAP authentication is a time-consuming process that can provoke a considerable packet loss during the handoff, thus affecting on-going communications. To solve this problem, researchers agree [65] that a proper solution must enable a fast and secure three-party key distribution process. Such key distribution process provides specific keys, which may be generated from the key material exported during a previous EAP execution, to both the mobile user and the authenticator from a trusted server without running lengthy full EAP authentications.

As outlined in section 1.5, user privacy protection is another challenging issue associated with NGNs. In particular, the authentication carried out during the network access

represents a critical process since user's identity is exposed to unauthorized parties. For this reason, the definition of an anonymous authenticated access to the network is considered a key aspect in defining an integral privacy solution in NGNs. Unfortunately, this aspect has not been covered so far in the field of EAP-based fast re-authentication.

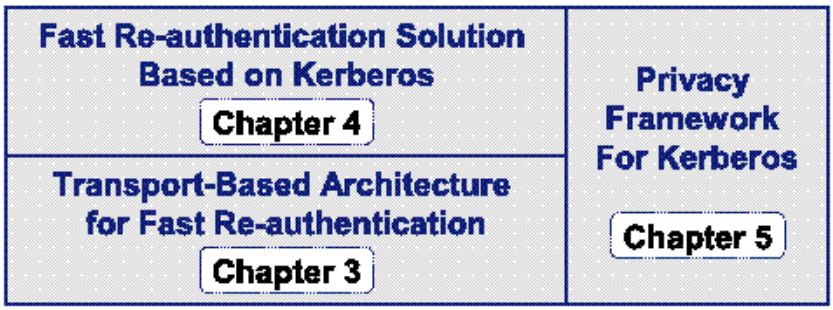


Figure 1.3: Overview of the Contributions Presented in this PhD Thesis

On the basis of this problematic, the objective of this PhD thesis is to define an efficient access control system satisfying that: *(1)* it is applicable for EAP-based NGNs; *(2)* it reduces the authentication latency in mobile environments irrespective of the type of handoff performed by the user; *(3)* it provides strong security properties; *(4)* it is easy to deploy in current networks; *(5)* it is compatible with current standardized technologies; and *(6)* it offers user privacy support. As outlined in table 1.1[2], a solution accomplishing all these requirements represents a novel contribution with respect to the existing fast re-authentication proposals. In the context of this work, the problem has been solved in a modular way, identifying different objectives that must be accomplished for the definition of an integral solution. As depicted in Fig. 1.3, each objective has led to a contribution in this PhD thesis, which is summarized below:

1. **Definition of a transport-based architecture for fast re-authentication**. This architecture is based on the definition of a new EAP method called *EAP Fast Re-authentication Method* (EAP-FRM) which works on standalone configuration mode. The architecture neither requires any change to the EAP specification nor modifications to existing lower-layers. Furthermore, EAP-FRM is a generic transport able to convey any key distribution protocol without compromising the fast re-authentication operation.

2. **Definition of a fast re-authentication solution based on Kerberos**. On the basis of EAP-FRM, a fast re-authentication solution is defined using Kerberos as secure three-party key distribution protocol. The adaptation of Kerberos to control the access of users to the network service is performed respecting the standardized

[2]This table is an extended version of that presented in [22]

	Context Transfer	Pre-authentication		Key Pre-distribution	Local Server	Modifications to EAP
		Link-Layer	Network Layer			
Intra-Technology	↑	↑	↑	↑	↑	↑
Inter-Technology	↘	↓	↑	↗	↑	↑
Intra-Network	↑	↑	↑	↑	↑	↑
Inter-Network	→	↓	↑	↑	↑	↑
Intra-Domain	↑	↑	↑	↑	↑	↑
Inter-Domain	↓	↓	↗	↘	↗	↗
Intra-Security	↑	↑	↑	↑	↑	↑
Inter-Security	↓	↓	↑	↗	↑	↑
Strong Security	→	→	→	→	→	↗
Low Deployment Impact	↘	↗	↘	→	↗	↓
Avoid Standards Modification	↘	↓	↗	↓	↗	↓
Privacy Support	↓	↓	↓	↓	↓	↓

** Arrows represent the capacity to accomplish a requirement: ↑ = high, ↗ = moderate, → = middle, ↘ = low, ↓ = null

Table 1.1: Overview of the Different Fast Re-authentication Proposals

Kerberos operation, and no modifications to the protocol itself are required. Furthermore, the Kerberized solution is able to achieve a fast re-authentication process consisting of only three messages between the peer and authenticator. It does not require communication with an external authentication server and preserves the resources reserved by network operators to assist the fast re-authentication process.

3. **Definition of a privacy framework for Kerberos**. This final objective addresses the need to protect the user privacy during the Kerberos-based fast re-authentication process. More precisely, a privacy framework for Kerberos, called *PrivaKERB*, is designed in order to preserve user anonymity and untraceability during the fast network access. By employing the extensibility mechanisms available in Kerberos, PrivaKERB is a lightweight solution imposing a negligible overhead. Furthermore, this framework is a general-purpose solution not only applicable to our fast re-authentication architecture, but also to any system using Kerberos.

1.6.1 Definition of a Transport-Based Architecture for Fast Re-Authentication

As regards the first objective, *definition of a transport-based architecture for fast re-authentication*, this thesis provides:

- *Analysis of the requirements to be accomplished by a generic fast re-authentication architecture for EAP networks*. The objective of this analysis is to identify which specific properties must be satisfied by a generic framework destined to facilitate the re-authentication process in EAP-based wireless networks.

- *Architecture proposal for EAP fast re-authentication*. Once the requirements to be accomplished have been identified, we define a generic architecture for fast re-authentication which is based on the definition of a new EAP method that operates in standalone mode called EAP-FRM. This makes the architecture fully compatible with current EAP implementations and deployed AAA infrastructures, thus favouring its adoption and deployment. Additionally, unlike current fast re-authentication solutions, the architecture is generic and does not impose either the use of a specific key distribution protocol to carry out the fast re-authentication process or any change to existing wireless technologies using EAP as authentication mechanism.

- *EAP-FRM format and AAA protocols extensions to implement the architecture*. The architecture is defined conceptually and the implementation framework is also deeply described. We provide the format of the EAP-FRM method messages and extensions for the most relevant AAA protocols (RADIUS and Diameter). This demonstrates that the use of the extensibility mechanisms offered by EAP and AAA protocols are sufficient to implement the fast re-authentication framework.

- *Definition of use cases.* To understand the operation of the re-authentication architecture and demonstrate its generality, we describe, as an example, some use cases where the fast re-authentication architecture is used in conjunction with different key distribution protocols.

- *Architecture evaluation.* The behaviour of the re-authentication architecture is tested through a proof-of-concept prototype. Results demonstrate that our approach is able to transport any key distribution protocol without excessively compromising the network access time.

1.6.2 Definition of a Fast Re-Authentication Solution Based on Kerberos

Once a generic fast re-authentication architecture for EAP-based wireless networks has been defined, the second objective deals with the *definition of a fast re-authentication solution based on Kerberos.* To carry out this objective, this thesis offers the following contributions:

- *Analysis of common deficiencies in existing fast re-authentication solutions based on a key distribution process.* This analysis tries to clarify the main disadvantages found in existing solutions dealing with fast network access through a key distribution process. In general, these solutions require both a resource pre-reservation on candidate authenticators and a communication with an external key distribution server located in the core network. Furthermore, they define new key distribution protocols which operators are reluctant to adopt, thus incrementing their deployability cost.

- *Selection of Kerberos as key distribution protocol to implement the fast re-authentication process.* Instead of defining a new fast re-authentication protocol, this PhD thesis has opt for the Kerberos protocol to develop a fast re-authentication protocol. As described in section 1.4.3, Kerberos is a well-known standardized protocol, whose security properties have been widely verified, and it avoids the various disadvantages presented by current key distribution solutions.

- *Definition of a Kerberized architecture for fast re-authentication.* On the basis of the EAP-FRM transport, a new fast re-authentication architecture is designed by adapting Kerberos to control the access to the network service. The different entities and communication interfaces are identified, as well as the different phases that integrate the fast re-authentication process. Additionally, further analysis of the proposed solution demonstrates the security of the process and its behaviour regarding authorization and accounting aspects.

- *Validation of the architecture.* The feasibility of the proposed architecture is verified through a wireless testbed prototype. Additionally, by using experimental results extracted from the testbed, more complex scenarios are simulated in order to

compare the proposed solution with other fast re-authentication proposals. Results demonstrate that the Kerberized architecture exhibits a lower authentication latency than other fast re-authentication schemes, without modifying any EAP-based wireless technology.

1.6.3 Definition of a Privacy Framework for Kerberos

Finally, the last objective of this PhD thesis, *definition of a privacy framework for Kerberos*, tries to enhance the key distribution process based on Kerberos by providing anonymity and untraceability to the user. The following actions are developed:

- *Analysis of the Kerberos protocol privacy issues.* The objective of this analysis is to identify the specific privacy risks that remain unsolved in the protocol. Specifically, attention is focused on sensitive information like the user identity which is transmitted within the protocol messages in an unprotected manner.

- *Definition of a privacy framework for Kerberos.* Once the main privacy vulnerabilities have been identified in Kerberos, a privacy framework, called *PrivaKERB*, is designed. The solution, which operates in both single-domain and multi-domain scenarios, offers three incrementally complemented opt-in privacy modes to cope with anonymity, service and exchange untraceability. By using the extensibility mechanisms available at Kerberos, PrivaKERB is a general-purpose solution not only applicable in our Kerberized fast re-authentication solution but also in any system using Kerberos.

- *Verification that the privacy enhancements do not affect the re-authentication procedure.* One important part of the work has focused on the demonstration that the privacy extensions to the basic Kerberos protocol are lightweight and do not impose a significant overhead that affects the authentication latency. To carry out this evaluation, an implementation prototype is used to test the solution in different scenarios.

- *Integration with the kerberized fast re-authentication solution.* The developed privacy framework is a general-purpose solution valid for any system based on Kerberos. In this final step, we demonstrate how the application of PrivaKERB to the proposed fast re-authentication solution based on Kerberos leads to an access control system that provides both fast network access and user privacy protection.

1.7 Thesis Structure

This first, introductory chapter presents the problem dealt within the PhD thesis: the reduction of the time devoted to the network access control process in future EAP-based next generation networks in the issue of user privacy. Special effort has been devoted to

identifying the requirements that must be taken into account when designing an advanced access control system, in order to overcome the various disadvantages presented by existing solutions addressing this problem.

Chapter 2 describes the different technologies related to the access control problem itself and which will be used in later chapters when presenting the contributions of this research work. Initially, since AAA infrastructures constitute the basic framework that support the authentication and authorization operations during the controlled access to the network service, an extensive description of AAA components and the most relevant AAA protocols is provided. Similarly, a detailed analysis of the EAP protocol is performed, with special attention to the authentication model proposed by EAP and the exported parameters after a successful EAP authentication, such as keying material. In terms of the key distribution protocol used to achieve a fast network access, this thesis proposes the use of Kerberos, so we provide extensive details about the protocol. For this reason, the third part of chapter 2 describes basic concepts, entities, protocol operation and information transported within Kerberos messages. The chapter concludes with a detailed description of the different solutions proposed so far to reduce the authentication latency during the handoff. The fast network access solution developed in this PhD thesis will pay attention to avoiding vulnerabilities and deficiencies detected in existing fast re-authentication solutions.

In chapter 3 the first contribution of this PhD thesis is provided through the definition of a generic fast re-authentication architecture called EAP-FRM. The architecture, designed using the extensibility mechanisms available in EAP and existing AAA protocols, offers a generic transport to convey any key distribution protocol. The chapter describes the different phases which integrate an EAP-FRM fast re-authentication process and details the EAP-FRM method message format and required extensions to AAA protocols. Next, several use cases serve to demonstrate the capabilities of the solution and its ability to avoid modifications to existing protocols. Finally, we develop a prototype that is used to evaluate the performance of the solution.

By using EAP-FRM architecture as a reference, chapter 4 works on the definition of a fast re-authentication solution that uses Kerberos as secure three-party key distribution protocol. The chapter opens by identifying common deficiencies present in existing fast re-authentication solutions based on a key distribution process. Unlike previous solutions which typically define new key distribution protocols, the chapter proposes a novel framework where the well-known Kerberos protocol is used to implement a fast re-authentication process. Kerberos is a standardized authentication protocol widely used to control access to resources and whose security has been well verified. On the basis of Kerberos, we propose an efficient architecture for fast re-authentication which is able to operate in different modes of operation. The description not only provides information regarding the different entities and communication interfaces among them, but also regarding authorization and accounting aspects. Next, using information extracted from an implemented wireless testbed, the solution is evaluated based on both simulation and analytical model analysis.

Chapter 5 concludes the design of the access control system by defining some enhancements to Kerberos that enables the privacy of the user to be protected during

fast network access. The chapter starts by analyzing the specific privacy problems that arise in Kerberos, focusing specially on the sensitive information that is transmitted in cleartext. Next, we design a privacy framework for Kerberos, named *PrivaKERB*, which comprises a multimode solution able to offer anonymity, service access untraceability and exchange untraceability to Kerberos clients. Special interest is devoted to discussing the proposed solution, analyzing important aspects such as security and deployment aspects. Finally, employing a real implementation testbed, we measure the additional overhead required by the privacy extensions in comparison with standard Kerberos to demonstrate that the privacy solution is compatible with a fast network access operation.

To conclude, chapter 6 enumerates the conclusions extracted from the research work undertaken in this PhD thesis and provides some future directions that have been identified for each contribution presented.

1.8 Related Publications

The research work developed in this thesis has led to the publication of different works at both national and international conferences, and in research journals and books. The most relevant contributions are presented.

- *F. Pereniguez, R. Marin, and A.F.G. Skarmeta.* **Análisis de Propuestas de Re-autenticación Rápida en Entornos Móviles**. In JITEL 2008: VII Jornadas de Ingeniería Telemática, pages 233 - 240, Alcalá de Henares, Madrid, 2008.

  This conference paper [94] represents the first contact with the fast re-authentication problem in mobile networks by surveying the different solutions which try to reduce network access time.

- *F. Pereniguez.* **Pre-autenticación y Securización del Tráfico en Entornos de Movilidad sobre Redes 802.11**. Ed. Euroeditions, 1ª Edición, Mayo 2008.

  This work [95] studies the fast re-authentication problem in the specific context of IEEE 802.11 networks and proposes a pre-authentication mechanism using existing technologies.

- *R. Marin-Lopez, Y. Ohba, F. Pereniguez, and A.F. Gomez.* **Analysis of Handover Key Management schemes under IETF perspective**. Elsevier Computer Standards & Interfaces, 32(5-6):266 - 273, 2010. Special issue on Information and Communications Security, Privacy and Trust: Standards and Regulations.

  In this work [96] an extensive analysis of the *EAP Extensions for EAP Re-Authentication Protocol* (ERP) is provided, which is the solution proposed within the IETF to solve the problem of EAP in mobile environments.

- *R. Marin-Lopez, F. Pereniguez, Y. Ohba, F.Bernal, and A.F. Gomez-Skarmeta.* **A Transport-based Architecture for Fast Re-authentication in Wireless**

Networks. In SARNOFF'09: Proceedings of the 32nd international conference on Sarnoff symposium, pages 40 - 44, Piscataway, NJ, USA, 2009. IEEE Press.

In this conference paper [97] we present a generic architecture to enable fast re-authentication in EAP-based wireless networks.

- *F.Pereniguez-Garcia, R. Marin-Lopez, F. Bernal-Hidalgo, and A. Gomez-Skarmeta.* **Architecture for Fast EAP Re-authentication based on a new EAP method (EAP-FRM) working on standalone mode.** IETF Internet Draft, IETF draft-marin-eap-frm-fastreauth-03.txt, March 2011.

  The architecture for fast re-authentication described in the previous publication was presented to the IETF community through this Internet-Draft [98].

- *R. Marin Lopez, F. Pereniguez Garcia, F. Bernal Hidalgo, and Antonio F. Gomez Skarmeta.* **Procedimiento de Re-autenticación.** European Patent, Publication N° PCT/ES2010/070069, September 2010.

  The architecture for fast re-authentication developed during the last two contributions was considered as an invention whose property should be protected through a patent [99].

- *R. Marin-Lopez, F. Pereniguez, F. Bernal, and A.F. Gomez.* **Secure Three-party Key Distribution Protocol for Fast Network Access in EAP-based Wireless Networks.** Elsevier Computer Networks, 54:2651 - 2673, October 2010.

  This work [16] develops a new fast re-authentication protocol based on a three-party model whose security properties are verified through a formal verification tool.

- *R. Marin Lopez, F. Pereniguez Garcia, Y. Ohba, F. Bernal Hidalgo, and A.F. Gomez-Skarmeta.* **A Kerberized Architecture for Fast Re-authentication in Heterogeneous Wireless Networks.** Springer Mobile Networks and Applications, 15(3):392 - 412, 2010.

  Unlike the previous work, which defines a new protocol from scratch, this journal publication [100] defines a secure, fast re-authentication solution based on Kerberos which is a widely deployed standardized protocol.

- *F. Pereniguez, G. Kambourakis, R. Marin-Lopez, S. Gritzalis, and A.F. Gomez.* **Privacy-enhanced Fast Re-authentication for EAP-based Next Generation Network.** Elsevier Computer Communications, 33(14):1682 - 1694, 2010.

  This work [101], performed during an internship at the Laboratory of Information & Communication Systems Security (Samos, Greece), addresses the problem of preserving user privacy while reducing the authentication time during network access.

Chapter 2

Background and State of the Art

2.1 AAA Infrastructures: Authentication, Authorization and Accounting (AAA)

As described in section 1.3, network operators need to control their subscribers so that only authenticated and authorized ones can access to the network services. Typically, the correct support of a controlled access to the network service has been guaranteed by the deployment of the so-called *Authentication, Authorization and Accounting* (AAA) infrastructures [25]. AAA essentially defines a framework for coordinating these individual security services across multiple network technologies and platforms.

An overview of the different components is the best way to understand the services provided by the AAA framework.

- *Authentication.* This process involves validating a credential (e.g., password, digital certificate, biometrical information,...) in order to verify the user's identity. A successful authentication is required before granting the user access to the service.

- *Authorization.* Authorization is a process that determines what rights and under which conditions access to a service is granted to an authenticated user. For example, in the case of the network service, this can include providing a specific IP address or invoking a filter to determine which applications are supported. Authentication and authorization are usually performed together in AAA environments.

- *Accounting.* It provides a methodology for collecting information about the user resource consumption. For example, this information can include use time, amount of sent and received information, etc. The network operator will employ this information to capacity planning purposes or auditing tasks. Nevertheless, accounting information is mainly used for billing.

These concepts are so important that, standardization organizations such as the Internet Research Task Force (IRTF) and the Internet Engineering Task Force (IETF)

have developed several research studies related to the management of the authentication, authorization and accounting processes. Despite these organizations have focused on different AAA aspects, they have actively collaborated in the achievement of an effective and efficient AAA environment. More precisely, while the IRTF has worked in defining the elements and requirements of a general AAA infrastructure, the objective of the IETF has been to standardize the required protocols that allow the communication between the different entities that integrate an AAA infrastructure.

The following sections provide a detailed description for a general AAA scheme, involved entities and the most relevant AAA protocols.

2.1.1 Generic AAA Architecture

The general AAA scheme, as defined in [25], requires the participation of four different entities (see Fig. 2.2) that take part in the authentication, authorization and accounting processes:

- A *user* desiring to access a specific service offered by the network operator. This function can be also performed by a *network equipment* acting as application agent for the user (e.g., a NAS).
- A *domain* where the user is registered. This domain, typically referred to as *home domain*, is able to verify the user's identity based on some credentials. Optionally, the home domain not only authenticates but also provides authorization information to the user
- A *service provider* controlling the access to the offered services. The service provider can be implemented by the domain where the user is subscribed (known as *home domain*) or by a different domain. In the case the service provider is located outside the home domain, the access to the service is provided on condition that an agreement is established between the service provider and the home domain. These bilateral agreements, which may take the form of formal contracts or *Service Level Agreements* (SLAs), suppose the establishment of a trust relationship between the involved domains that will allow the service provider to authenticate and authorize foreign users coming from another administrative domains.
- A *service provider's service equipment* which will be typically located on a device that belongs to the service provider. In the case of network access service, an 802.11 access point acts as a service equipment.

Generic AAA Server

To carry out the authentication and authorization processes, the AAA architecture defines a *AAA server* located in both the service provider and the home domain. The AAA server

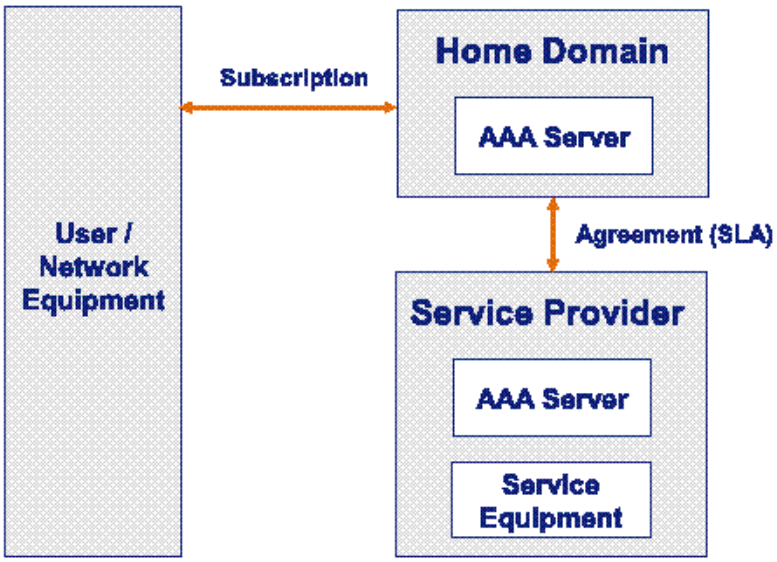


Figure 2.1: Generic AAA Architecture

is an entity which is capable of authenticating users, handling authorization requests and collecting authorization data. A general overview of a AAA server and the interaction with other network entities is shown in Fig. 2.2. As observed, the user interacts (typically through an intermediate network element such as an access point) with the AAA server, by sending AAA requests. These requests are associated with services or applications that delegate the authentication and authorization processes to the AAA server. Using some internal rules defined in the AAA server, the request is evaluated and a decision is made according to the authentication, authorization and accounting tasks related to the user. As observed in the figure, the appropriate policies that must be applied to the user are fetched from a *policy database*. Since rules and conditions can frequently vary, the existence of a repository allows incorporating new policies and information. A generic AAA server must be able to handle the particular situations where conditions associated with existing services can change. Similarly, a AAA server is expected to register in an *event repository* the different received requests and the operations performed.

The AAA server also communicates with an *Application-Specific Module* (ASM) which is able to manage an application and to configure services. Examples of ASMs are QoA managers, bandwidth brokers or Mobile IP agents [102]. By defining the concept of ASM, a generic interface can be defined between AAA servers and any type of management entity without getting into details of each application. Each service application has application-specific information (ASI) that only the ASM understands. The AAA server in charge of authorizing the request may not understand the details of each piece of information. Furthermore, the information may have or may not have an unique structure as well. Therefore, in general, the AAA server is not required to have application-specific knowledge and needs to refer to the ASM. All the AAA server needs to know is the location (ASM) to which it needs to send the application-specific data. As we will see later on,

AAA protocols define specific data units (called attributes) for each type of ASI. As long as the AAA can recognize the attribute type, it can simply tell what application it relates to. Typically, some sort of identifier for the ASM is included in the request to the AAA server, so that it can route the packet to the proper destination. For instance, an NAI may be used either to help the AAA server identify the ASM or to let the AAA server know that the ASM belongs to a completely different administrative domain.

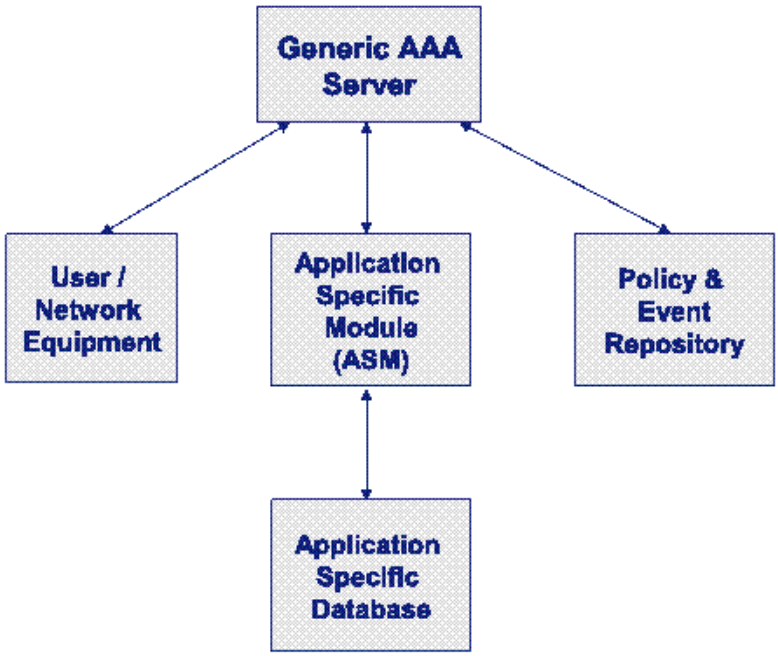


Figure 2.2: AAA Server Structure

Finally, regarding the protocols used for interaction between the different entities, it is important to mention that the protocol used between every pair of entities it is not assumed to be the same. For instance, the communication protocol used between the AAA server and the policy repository may be different from those used for interaction between the AAA server and the ASM. The same applies for the interaction between ASM and its database (if this data-base is not co-located with the management entity itself).

AAA Server Operation

Using the different components (ASMs, policy and event repository) previously described, the AAA server is able to handle requests received from the users. In general, a request received by a generic AAA server, follows this chain of actions [27]:

1. The user or the network equipment on its behalf sends a service request to the AAA server. In order to access the requested service, the user is required to present authentication and authorization credentials.

2. The AAA server verifies the authentication credentials for the user after consulting its user databases and possibly the policy repository. When authentication is successful, the AAA server inspects the contents of the authorization request and determines what kind of authorization is requested.

3. If needed, the AAA server retrieves policy rules from the repository and performs an authorization decision on each component (attribute) of the request according to one the following alternatives:

   (a) When the component is an ASI that must be processed by the ASM, the component is sent to the ASM to be evaluated.

   (b) Query the policy repository for an answer.

   (c) In a multiple administrative domain scenario, the component is forwarded to another AAA server.

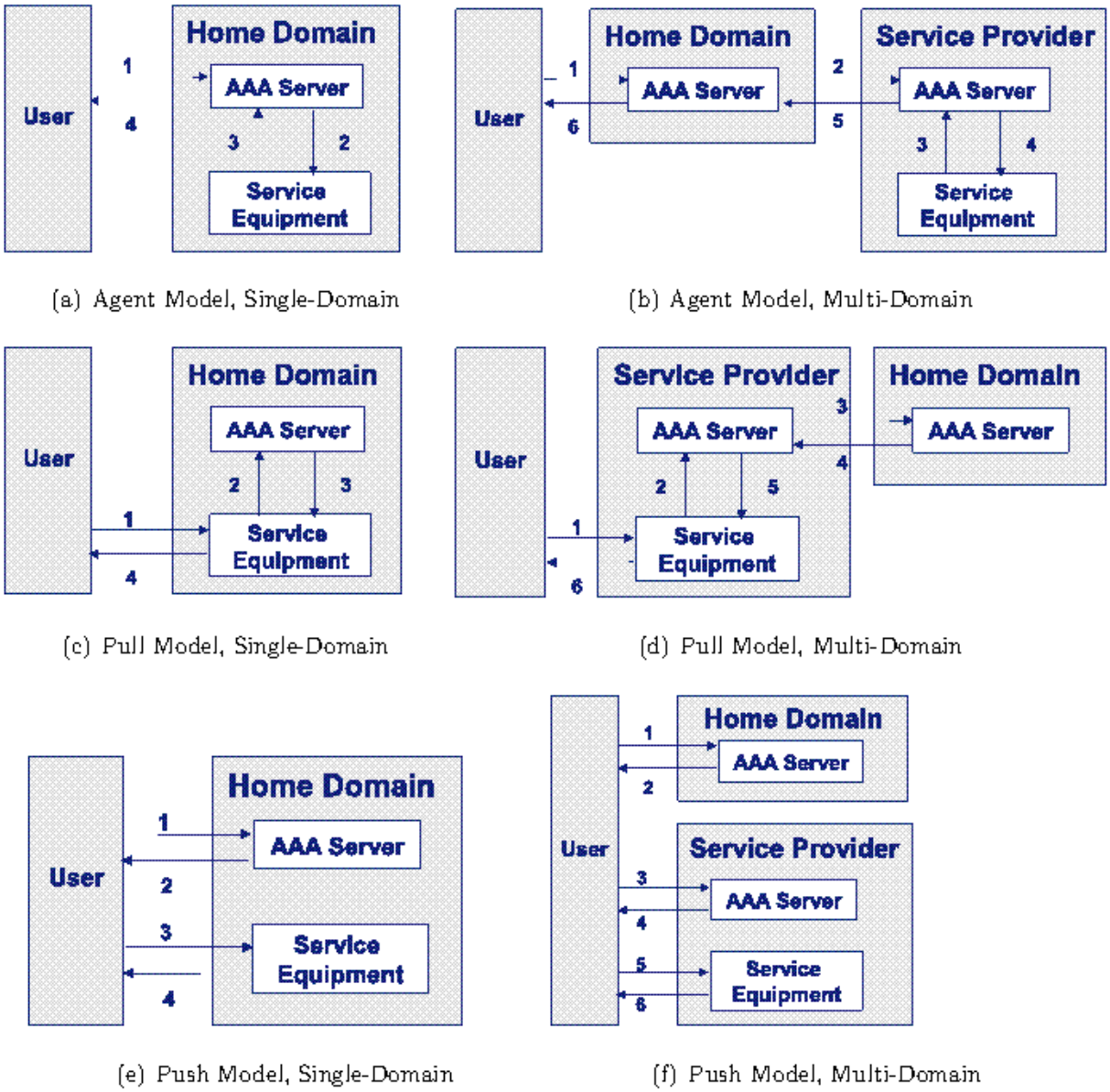
4. The AAA server informs the application-specific network equipment (service equipment) about the authorization result and possibly provides necessary information for allocating the resources to set up the service at the point it is being delivered to the user.

2.1.2 AAA Message Flow

Depending on whether the home domain and the service provider belong to the same or different organizations, two different scenarios can be distinguished. On the one hand, in the *single-domain scenario*, the home organization and the service provider are combined in a single entity. In other words, the home domain acts as a service provider as well, deploying a AAA server to control the access to a set of services. On the other hand, in the multi-domain scenario, the organization that authenticates and authorizes the user is different from the organization providing the service. As described in section 2.1.1, the user is granted access to the service thanks to the roaming agreements (SLAs) established between service provider and home domain. Typically, in a multi-domain scenario, a single service provider is involved in satisfying the user needs (*roaming*). Nevertheless, a more complex situation appears when the collaboration of different service providers (*distributed*) is required to provide a complete service to the user.

Authentication and authorization of the user to access a service is accomplished on the basis of the interaction between the user, the AAA server, and the service equipment. Regardless of whether the type of scenario (single-domain or multi-domain), three major models are recognized for the interaction among these entities, where each one leads to a different sequence of operations:

- *Agent Sequence*. This sequence is used for scenarios where the user only contacts the AAA infrastructure in order to solicit access to a specific service. In the single-realm case (Fig. 2.3(a)), the user sends a service request (*1*) to the AAA server. The



(a) Agent Model, Single-Domain

(b) Agent Model, Multi-Domain

(c) Pull Model, Single-Domain

(d) Pull Model, Multi-Domain

(e) Push Model, Single-Domain

(f) Push Model, Multi-Domain

Figure 2.3: AAA Interaction Models

AAA server authenticates and authorizes the user based on the information of the user and the conditions extracted from the policy repository. After that, the AAA server sends the authorization (and other configuration information) to the service equipment (*2*). The service equipment performs the necessary configurations in order to provide the service to the user and responds the AAA server (*3*) one the procedure has been completed. Finally, the AAA server replies to the user (*4*) that authorization is complete and the service is ready to attend its demands. As we can

see, in this model the AAA server acts as an agent for the user. In the multi-domain scenario (Fig. 2.3(b)) the user also sends a service request to the home AAA server (*1*). If the user is allowed to access the service, the request is forwarded (*2*) to the service provider's AAA server that processes the request (*3*, *4*) as described for the single-realm case. Finally, the confirmation to the user is sent from the service provider's AAA server to the user through the home AAA server (*5*, *6*).

- *Pull Sequence*. In this scenario, the user solicits access to a service by directly contacting the service equipment (*1*). On the reception of the solicitation, the service equipment forwards the request to local domain's AAA server (*2*). In a single-domain scenario (Fig. 2.3(c)), the service equipment receives the decision (*3*) taken by the service provider's AAA server and notifies the user accordingly (*4*). In a multi-domain scenario (Fig. 2.3(d)), the service provider's AAA server forwards the service request to the home AAA server (*3*) that authenticates the user and may provide some authorization to the service provider's AAA server (*4*). Similarly to the single-domain case, the decision is sent to the user through the service equipment (*5*, *6*). It is worth noting that this model is typically used in EAP-based network access authentication. It is important to note that the description of this model matches with Kerberos operation, as we will analyze later on.

- *Push Sequence*. This model is a token-based approach. In the single-domain scenario (Fig. 2.3(e)), the user firstly gets a token like a ticket or certificate from the service provider's AAA server (*1*, *2*). When the user requests a service from the service equipment, the user presents the ticket (*3*) in order to demonstrate that it has been previously authorized by the AAA server. After successful verification of the presented token, the service equipment either accepts or rejects the service request and notifies the user (*4*). Fig. 2.3(f) describes the process in a multi-domain scenario. In this case, the user firstly acquires a token from the home AAA server (*1*, *2*). This token is presented to the service provider's AAA server (*3*) which, based on the presented credentials and internal rules, decides if access to service is granted and under which conditions. If so, the user obtains a final token (*4*) that is used against the service equipment (*5*) to demonstrate that the AAA infrastructure has authorized access. The final decision taken by the service equipment is received by the user in a final message (*6*).

From the different operation modes previously described, the pull and push sequence models will be the most interesting ones in the context of this PhD thesis. In particular, the research work is developed for network access service where an entity called *Network Access Server* (NAS), acting as a service equipment, controls the access to the network service. By means of traffic filtering, the NAS allows that only authenticated and authorized users can send and receive information. Furthermore, the multi-realm scenario will be the most referenced since, in NGNs, mobile users are expected visit foreign domains.

2.1.3 Relevant AAA Protocols

To allow the communication between the different entities that integrate a AAA infrastructure, it is required the deployment of a *AAA protocol*. Specifically, the IETF has studied [28] which requirements must be accomplished by a AAA protocol when used for network access. These requirements are specially useful when evaluating existing AAA protocols or designing new ones. Between the different desirable properties, the most significant are:

- *Scalability.* The AAA protocol must be capable of working in a highly demanding environment with millions of users, thousands for simultaneous requests and tens of thousands devices and AAA servers.
- *Mutual authentication.* This requirement refers to the ability to support mutual authentication between the AAA client and server.
- *Secure communication.* The AAA protocol must allow communication to be secured. At the same time, the AAA protocol must allow an underlying security service (such as *IPSec*) to be used.
- *Reliable transport.* The AAA protocol must provide mechanisms that guarantee the delivery of the transmitted data. For example, retransmission techniques are desirable to avoid packet loss.
- *Auditability.* The AAA protocol should handle processes in such a manner that it is feasible to determine what actions have been performed by a specific user.
- *Extensibility.* The AAA protocol must be extensible in such a manner that new attributes can be defined to transport application-specific information.

The most relevant AAA protocols are RADIUS [36] and Diameter [37]. Actually, Diameter is the most complete AAA protocol accomplishing the requirements identified in [28]. Nevertheless, up to now, the most widely deployed AAA protocol in current AAA infrastructures is RADIUS. In the following, it is provided a brief overview of both.

RADIUS

RADIUS was originally designed to serve the purpose of allowing a NAS to forward a dial-up user's request and its credentials to a backend server. Nowadays, RADIUS is the most widespread AAA protocol used in wireless networks to carry out the authentication and authorization processes during network access control.

RADIUS is a client-server protocol where a NAS usually acts as a RADIUS client. During authentication procedures, the RADIUS client is responsible for passing user information in the form of requests to the RADIUS server and waits for a response from the server. Depending on the policy, the NAS may only need a successful authentication

or further authorization directives from the server to enable data traffic to the client. The RADIUS server, on the other hand, is responsible for processing requests, authenticating the users, and returning the information necessary for client configuration to deliver the service to the user.

The typical RADIUS conversation consists of the following messages:

- *Access-Request*. This message is sent from the RADIUS client (NAS) to the server to request authentication and authorization for a particular end user.
- *Access-Challenge*. This message, sent from the RADIUS server to the client, is used by the server to obtain more information from the NAS about the end user in order to make a decision about the requested service.
- *Access-Accept*. This message is sent from the RADIUS server to the NAS to indicate a successful completion of the request.
- *Access-Reject*. This message is sent by the server to indicate the rejection of a request.

Apart from these main messages, the RADIUS base specification [36] defines some others to transmit accounting information (*Accounting-Request*/*Accounting-Response*) or the status of the RADIUS entities (*Status-Client*/*Status-Server*). Typically, the main part of a RADIUS conversation consists of several Access-Request/Access-Challenge message exchange where the RADIUS client and server exchange information transported within RADIUS attributes. Depending on whether the client is successfully authenticated or not, the RADIUS server finalizes the communication with an *Access-Accept* or *Access-Reject*, respectively.

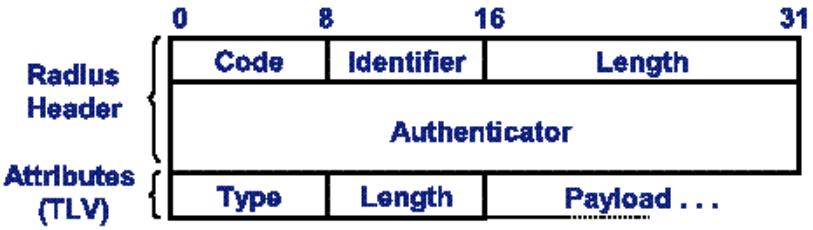


Figure 2.4: RADIUS Packet Format

As depicted in Fig. 2.4, the RADIUS packet format is very simple and consists of a header and a body. The header includes a *code* to identify the type of RADIUS packet (access request, response, etc.), an *identifier* that is used to match requests and responses, the *length* of the entire RADIUS packet and an *authenticator* destined to provide integrity protection to the RADIUS message. The body part of the RADIUS message carries information in the form of attributes. Each attribute can be a self-contained package including information of variable length and formed according to type, length, and

value (TLV) format. Initially, the RADIUS specification [36] defines some attributes like *User-name* to indicate the name of the user to be authenticated, *NAS-IP Address* to identify the IP address of the NAS which is requesting authentication of the user or *Service-Type* to specify the type of service being requested by the user. Nevertheless, attributes are the main vehicle to extend the RADIUS functionality. For example, Ref. [103] defines a new attribute type named *EAP-Message* destined to contain an EAP message.

Regarding the protocol used to transport RADIUS messages, protocol designers considered that UDP was the most appropriate one since TCP session establishment is a time-consuming process involving state machines. Nevertheless, the lack of a reliable transport causes serious problems to RADIUS. For example, clients are unable to distinguish when a request is received by the server or a communication problem has occurred and the RADIUS packet has not reached its destination. Similarly, a client cannot distinguish whether a server is down or discarding requests.

RADIUS security is based on the use of shared secrets between the RADIUS client and the server. In real deployments, this basic security mechanism has been known to cause several vulnerabilities:

- Shared secrets must be statically configured. No method for dynamic shared secret establishment is defined in the RADIUS protocol.
- Shared secret are determined according to the source IP address in the RADIUS packet. This introduces management problem when the client's IP address change.
- When using RADIUS proxies, the RADIUS client only shares a secret with the RADIUS server in the first hop and not with the ultimate RADIUS server. In other words, the trust relationship between the RADIUS client and the final RADIUS server is transitive rather than using a direct trust relationship. If a server in the chain is compromised, some security problems arise.
- RADIUS does not provide high transport protection. For example, an observer can examine the content of RADIUS messages and trace the content of a specific attribute.

To overcome these security weakness, it has been proposed the use of TLS [104] to provide a means to secure the RADIUS communication between client and server on the transport layer [105]. Nevertheless, the main research efforts have focused on the design of a new AAA protocol called *Diameter*.

Diameter

With the emergence of new technologies and applications in future mobile wireless networks, it is needed a new protocol capable of fulfilling new access control features while keeping the flexibility for further extension. *Diameter*, proposed as an enhancement to RADIUS, is considered the next generation AAA protocol. Diameter is characterized

by its extensibility and adaptability since it is designed to perform any kind of operation and supply new needs that may appear in future control access technologies. Another cornerstone of Diameter is the consideration of multi-domain scenarios where AAA infrastructures administered by different domains are interconnected to provide an unified authentication, authorization and accounting framework.

▷ Diameter Architecture

The Diameter protocol defines an extensible architecture that allows to incorporate new features through the design of the so-called *Diameter applications*, which rely on the basic functionality provided by the *base protocol*. The Diameter *base protocol* [37], defines the Diameter minimum elements such as the basic set of messages, attribute structure and some essential attribute types. Additionally, the basic specification defines the inter-realm operations by defining the role of different types of Diameter entities. Diameter applications are services, protocols and procedures that use the facilities provided by the Diameter base protocol itself. Every Diameter application defines its own *commands* and *messages* which, in turn, can define new attributes called *Attribute Value Pair* (AVP) or re-use existing ones already defined by some other applications.

The Diameter base protocol does not define any use of the protocol and expects the definition of specific applications using the Diameter basic engine. For example, the interaction of Diameter with NAS for providing authentication during network access is defined in the *Diameter NAS Application* [106]. In turn, this specification is used by the *Diameter EAP Application* [107] to specify the procedure to transport EAP packets between a NAS and a Diameter-based AAA server. Similarly, authorization and accounting procedures are expected to be handled by specific applications.

▷ Diameter Entities

Within a Diameter-based infrastructure, the protocol distinguishes different types of nodes where each one plays a specific role:

1. *Diameter Client*: represents an entity implementing network access control like, for example, a NAS. The Diameter client issues messages soliciting authentication, authorization or accounting services for a specific user.

2. *Diameter Server*: is the entity that processes authentication, authorization and accounting request for a particular domain. The Diameter server must support the diameter base protocol and the applications used in the domain.

3. *Diameter Agent*: is an entity that does not process a request and forwards it to a diameter server or to another agent. Depending on the service provided, we can distinguish:

   (a) *Relay agents*: which forward messages based on routing-related attributes and routing tables.

(b) *Proxy agents*: which act as a relay agent that, additionally, may modify the routed message based on some policy.

(c) *Redirect agents*: instead of routing messages, they inform the sender about the proper way to route the message.

(d) *Translation agents*: which perform protocol translations between Diameter and other AAA protocols such as RADIUS.

▷ Diameter Messages

Diameter messages are used for carrying information between Diameter nodes. Instead of defining a message type, Diameter uses the concept of *command* to specify the type of function a Diameter message intends to perform. Because the message exchange style of Diameter is synchronous, each command consists of a request and its corresponding answer. Tab. 2.1 provides a brief summary of the main Diameter commands defined in the base protocol specification.

Command	**Abbreviation**	**Description**
Capabilities-Exchange-Request /Answer	CER/CEA	Discovery of a peer's identity and its capabilities.
Disconnect-Peer-Request /Answer	DPR/DPA	To inform the sender's intention to shut down the connection.
Re-Auth-Request /Answer	RAR/RAA	Sent to an access device (NAS) to solicit user re-authentication.
Session-Termination-Request /Answer	STR/STA	Sent to the server to inform the provision of a service to a user
Accounting-Request /Answer	ACR/ACA	To exchange accounting information between Diameter client and server.

Table 2.1: Common Diameter Commands

Diameter messages consists of a header and a set of AVPs. As observed in Fig. 2.5, the header contains the following fields:

- *Version*: indicates the Diameter protocol version:
- *Length*: contains the length of the message, including this field.
- *Flags*: specifies special conditions of the message like, for example, if the message is a request or a response.
- *Code*: the value of this field indicates the command associated with the message. The command code is used by the receiver to decide what action is performed to process the message.

- *Application Identifier*: identifies the specific application that uses the message (e.g., NAS Application).
- *Hop by Hop Identifier*: carries an identifier that is used to match request and responses between two Diameter nodes maintaining a direct communication.
- *End to End Identifier*: is an identifier used to detect duplicate messages. The identifier in a response message must match the identifier in the corresponding request message.

The rest of the information carried by a Diameter message is formatted within an AVP. Each AVP has an *AVP Code* identifying the type of information included in the attribute data field (*AVP Data*) of the AVP. Additionally, an AVP includes two mandatory fields: a *Flags* field providing information to Diameter servers or agents routing the message that contains the AVP and the *Length* field containing the length of the whole attribute.

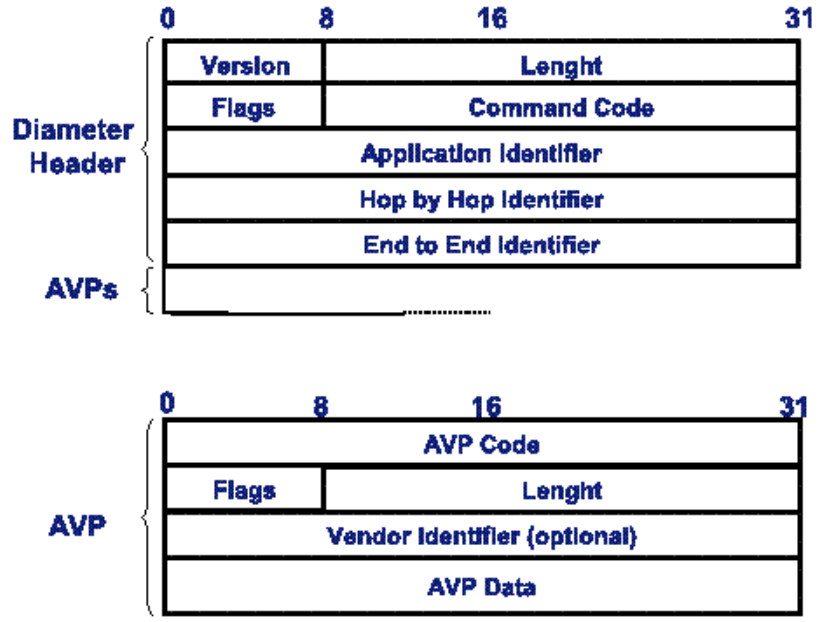


Figure 2.5: Diameter Packet Format

▷ Main Features of Diameter

As discussed within the IETF [108], Diameter is the most complete AAA protocol satisfying a wide set of requirements. For this reason, the adoption of Diameter is recommended in future AAA infrastructures supporting access control in NGN. The most important features provided by Diameter are:

- *Fail-Over.* When the communication with a specific server fails, the Diameter protocol provides a mechanism to change to another backup or secondary server. Diameter fail-over mechanisms are based on application layer acknowledgements to detect lack of activity.

- *Server-Initiated Messages.* Unlike RADIUS which follows a client-server model, Diameter is a peer-to-peer protocol, which means both the client and the server can create either a request or an answer.. Thanks to this feature, for example, a AAA server can explicitly solicit the client to disconnect because the time allowed to access a service has expired.

- *Reliable Transport.* Diameter specification uses TCP or SCTP as transport protocol. Both are connection-oriented protocols that allow to detect packet loss based on the use of acknowledgements.

- *Capability Negotiation.* Diameter includes support for error handling, capability negotiation, as well as ways to indicate support of AVPs.

- *Security.* Diameter defines both transmission-level security and end-to-end security and requires mandatory support of IPSec and optional TLS support at the clients. It is important to clarify that the base protocol only provides a hop-by-hop security, not implementing an end-to-end protection.

- *Inter-Domain Support.* RADIUS does not provide specific support for proxy agents in a inter-domain scenario. This causes each implementation to handle the RADIUS-based roaming in a different manner. Conversely, Diameter defines the role of agents and proxies and their behavior explicitly. By providing explicit support for inter-domain roaming, message routing and transmission-layer security, Diameter addresses the RADIUS limitations.

- *Peer discovery.* Diameter enables dynamic discovery of peers by using the *Domain Name Service* (DNS).

- *Backward Compatibility.* Despite Diameter does not share a common message format with RADIUS, considerable effort has been expended in enabling backward compatibility with RADIUS, so that the two protocols can be deployed in the same network. This interoperability is possible thanks to the translation of RADIUS and Diameter messages performed by the so-called *translations agents.*

2.2 The Extensible Authentication Protocol (EAP)

In previous section we have explained the AAA infrastructure which offer a general framework to carry out the authentication, authorization and accounting processes. In particular, as complement to the AAA infrastructures, the IETF has designed a protocol

named *Extensible Authentication Protocol* (EAP) [33] that permits the use of different types of authentication mechanisms through the so-called *EAP methods* (e.g. based on symmetric keys, digital certificates, etc...). These are performed between an *EAP peer* and an *EAP server*, through an *EAP authenticator* which merely forwards EAP packets back and forth between the EAP peer and the EAP server. From a security standpoint, the EAP authenticator does not take part in the mutual authentication process but acts as a mere EAP packet forwarder.

One of the advantages of the EAP architecture is its flexibility since does not impose a specific authentication mechanism. Additionally, EAP is independent of the underlying wireless access technology, being able to operate in NGNs. Finally, EAP allows an easy integration with existing Authentication, Authorization and Accounting (AAA) infrastructures [38] by defining a configuration mode that permits the use of a backend authentication server, which may implement some authentication methods. The flexibility, wireless technology independence and easy integration with AAA infrastructures are three important aspects that have motivated the success of the EAP authentication protocol.

Since EAP is the foundation on which this PhD thesis is developed, in the following we describe in detail the main features of the protocol such as message format, entities and authentication phases.

2.2.1 EAP Message Format

The EAP protocol consists of request and response messages. Request messages are sent from the authenticator to the peer. Conversely, response messages are sent from the peer to the authenticator. Both request and response messages share a common packet format composed by five fields:

- *Code*: A 1-octet field indicating if the packet is a request (*EAP Request=1*), a response (*EAP Response=2*), a success message (*EAP Success=3*) or a message indicating a failure (*EAP Failure=4*).

- *Identifier*: A 1-octet field used to associate a request with the associated response packet.

- *Length*: A 2-octet field indicating the length (in bytes) of the EAP packet, including this field.

- *Type*: A 1-octet field indicating the type of the request or the response packet. For example, this field can indicate a specific EAP method.

- *Type-data*: This field contains specific authentication data which depends on the type indicated in the *Type* field.

2.2.2 Components

The different messages exchanged during an EAP execution are processed by several components. As observed in Fig. 2.6, these components are conceptually organized in four layers:

- *EAP Lower-Layer*. This layer is responsible for transmitting and receiving EAP packets between the peer and authenticator.
- *EAP Layer*. The EAP layer is responsible for receiving and transmitting EAP packets through the transport layer. The EAP layer not only forwards packets between the EAP transport and peer/authenticator layers, but also implements duplicate detection and packet retransmission.
- *EAP Peer / Authenticator Layer*. EAP assumes that an EAP implementation will support both the EAP peer and the authenticator functionalities. For this reason, based on the code of the EAP packet, the EAP layer demultiplexes incoming EAP packets to the EAP peer and authenticator layers.
- *EAP Method Layer*. An EAP methods implements a specific authentication algorithm that requires the transmission of EAP messages between peer and authenticator.

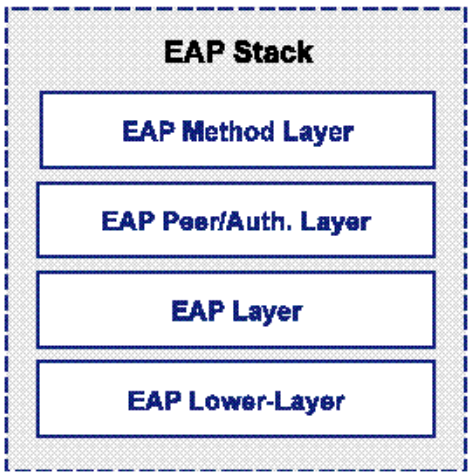
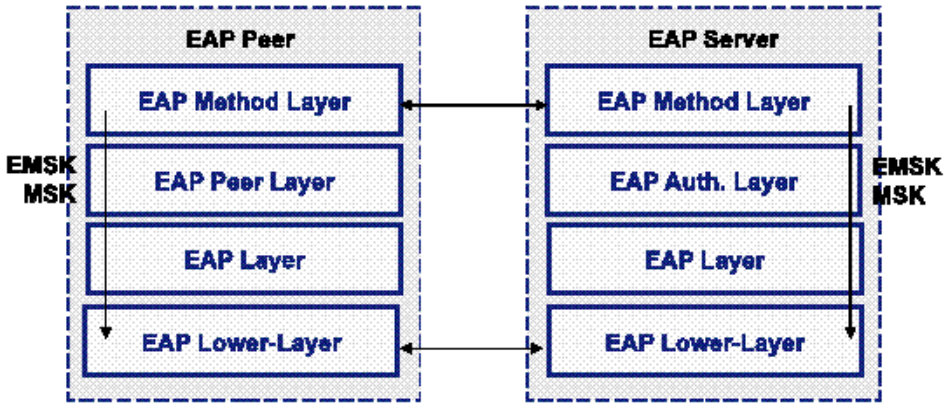


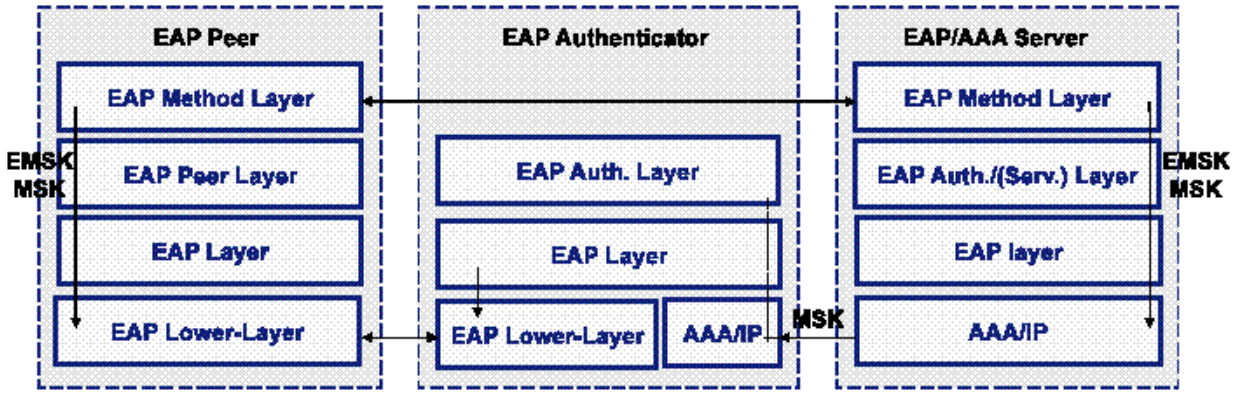
Figure 2.6: Components that Integrate the EAP Stack

2.2.3 Distribution of the EAP Entities

As previously mentioned, an EAP authentication involves three entities: the EAP peer, authenticator and server. Whereas the EAP peer is co-located with the mobile, the EAP authenticator is commonly placed on the *Network Access Server* (NAS) (e.g. an access



(a) Standalone Authenticator Model



(b) Pass-through Authenticator Model

Figure 2.7: EAP authenticator models

point or an access router). Instead, depending on the location of the EAP server, two authenticator models have been defined. Figures 2.7(a) and 2.7(b) show the *standalone authenticator model* and the *pass-through authenticator model*, respectively. On the one hand, in the standalone authenticator model (Fig. 2.7(a)), the EAP server is implemented on the EAP authenticator. On the other hand, in the pass-through authenticator model (Fig. 2.7(b)), the EAP server and the EAP authenticator are implemented in separate nodes.

In general, the pass-through model is the most widely employed configuration since it favours aspects such as scalability and deployment flexibility. For example, let us consider a typical network architecture consisting of a large number of authenticators. In the standalone configuration, each authenticator maintains a database storing the credentials required to authenticate every EAP peer. Additionally, each time a new EAP method appears, all authenticators need to be updated. Similarly, when a new user is registered

in order to allow access to the network, all the databases in the authenticators must be updated with the authentication information about the new user. As we can see, these aspects present serious scalability and management problems in the authentication system. Instead, since the pass-through configuration delegates the authentication process in a centralized backend AAA server which stores the user's information, the management of scenarios with a large number of users and authenticators is easier.

In order to deliver EAP messages, an *EAP lower-layer* (e.g., 802.11 [7]) is used to transport the EAP packets between the EAP peer and the EAP authenticator. The protocol used to transport messages between the EAP authenticator and the EAP server depends on the authenticator model employed. More precisely, in the standalone authenticator model, the communication between the EAP server and standalone authenticator occurs locally in the same node. In the pass-trough authenticator model, EAP protocol requires help of an auxiliary AAA protocol such as RADIUS [103] or Diameter [107]).

2.2.4 EAP Authentication Phases

As mentioned, the *Extensible Authentication Protocol* (EAP) [33] is a request/response protocol which supports only a single packet (request or response) in flight. Each request message (*EAP Request*) is answered with a response (*EAP Response*). As depicted in Fig. 2.8, a typical EAP conversation [1] occurs in three different phases. Initially, in the discovery phase (*Phase 0*), the peer discovers authenticator near to the peer's location with which it desires to start an authentication process. This phase, which is supported by the specific EAP lower-layer protocol, can be performed either manually or automatically.

The authentication phase (*phase 1*) starts when the peer decides to initiate an authentication process with a specific authenticator. This phase consists of two steps. Firstly, the *phase 1a* includes an EAP authentication exchange between the EAP peer, authenticator and server. To start an EAP authentication, as depicted in Fig. 2.8, the EAP authenticator usually starts the process by requesting the EAP peer's identity through an *EAP Request/Identity* message. It is worth noting that the trigger that signals the EAP authenticator to start the EAP authentication is outside the scope of EAP. Examples of these triggers are the EAPOL-Start message defined in IEEE 802.1X [34] or simply an 802.11 association process. On the reception of the *EAP Request*, the EAP peer answers with an *EAP Response/Identity* with its identity. With this information, the EAP server will select the EAP method to be performed. The EAP method execution involves several exchanges of EAP Request and Response messages between the EAP server and the EAP peer. A successful EAP authentication finishes with an *EAP Success* message.

Certain EAP methods [109] are able to generate key material. In particular, according to the *EAP Key Management Framework* [38] (EAP KMF) two keys are exported after a successful EAP authentication: the *Master Session Key* (MSK) and the *Extended Master Session Key* (EMSK). The former is traditionally sent (using the AAA protocol) to the

[1] Without loss of generality, it is assumed an EAP pass-through authenticator model.

authenticator (*Phase 1b*) to establish a security association with the EAP peer (*Phase 2*). Instead, the latter must not be provided to any other entity outside the EAP server and peer. Thus, both entities may use the EMSK for further key derivation.

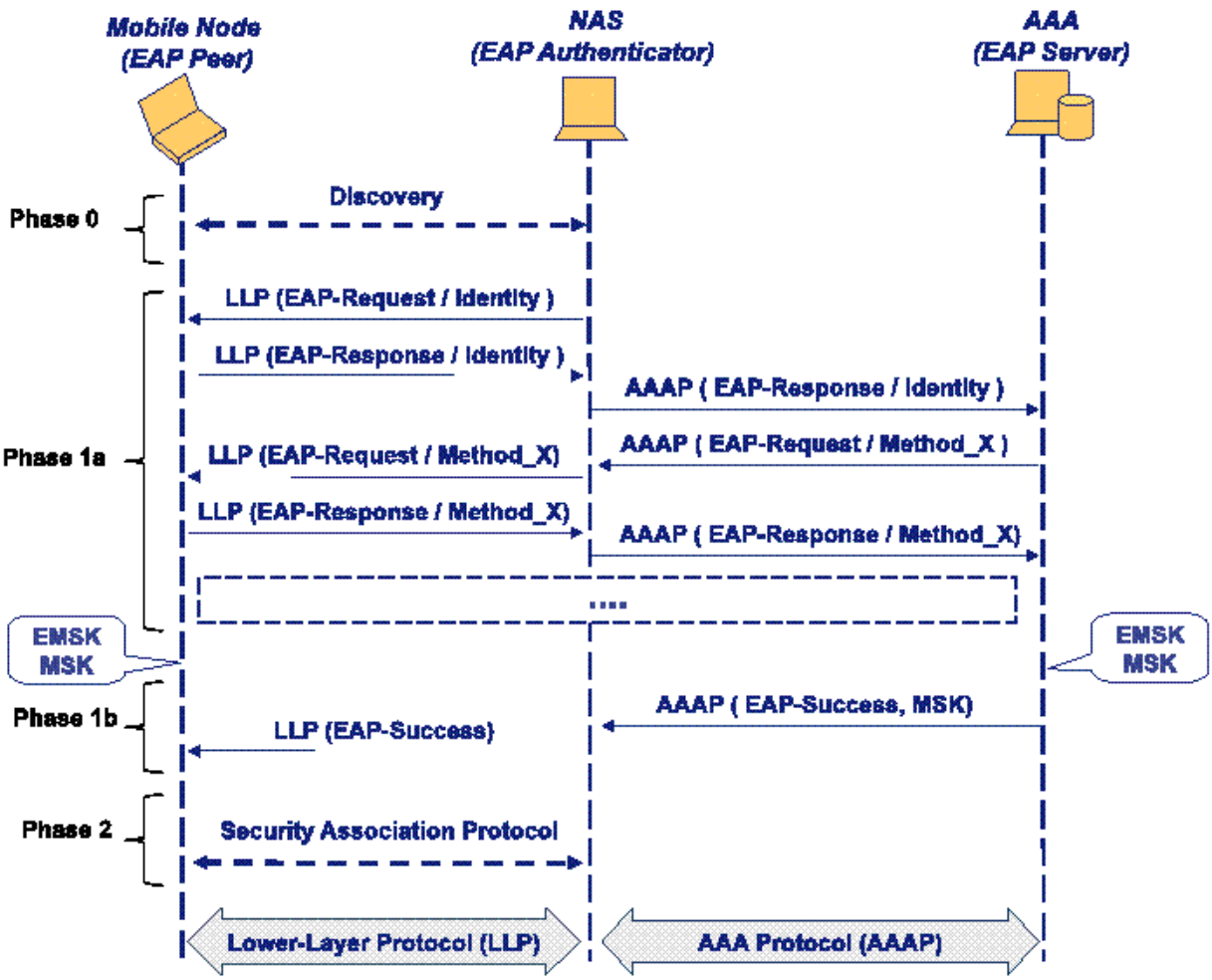


Figure 2.8: EAP Authentication Exchange

EAP is characterized by four main properties, also known as EAP invariants [33]. The first property is *mode independence* by which an EAP server can operate either in the same node as the authenticator (standalone mode) or in a separate node (pass-through mode), but the peer is agnostic about the mode of operation. The second property is *media independence* by which EAP methods operate on any lower-layer. The third property is *method independence* by which any EAP authentication method can be supported without changing EAP itself. The fourth property is *ciphersuite independence* which is a requirement of the second property to be able to handle any lower-layer ciphersuite. These properties allow easy integration of various lower-layers with the AAA infrastructures in order to provide a complete and secure network access service.

2.2.5 Key Material and Parameters Generated by EAP Methods

The EAP KMF [38] mandates that, after a successful EAP authentication, an EAP method must generate the following cryptographic material:

- *Extended Master Session Key* (EMSK) and *Master Session Key* (MSK). As mentioned in section 2.2.4, EAP methods are required to export the MSK and EMSK cryptographic keys of at least 64 octets in length. The MSK key is used to establish a security association between the peer and the authenticator. In pass-through configurations, a key distribution process in needed in order to transfer the key material from the server to the authenticator. Typically this process has been delegated to the AAA protocol used to transport EAP messages between authenticator and server. Conversely, the EMSK key is kept in secret by peer and server and is never shared with any third party. Despite the EAP KMF does not define a specific use for this key, recent work [110] has proposed an EMSK key hierarchy that allows to derive child keys from the EMSK to be used for different purposes like, for example, fast re-authentication usage.

- *Transient EAP Key* (TEK). TEKs are used to establish a secure communication channel between the EAP peer and server during the EAP authentication exchange. In other words, TEKs are destined to protect the EAP conversation. The TEKs are stored locally by the EAP method and are not exported. They are created during an EAP conversation and deleted at the end of the process.

- *Transient Session Key* (TSK). The TSKs key are destined to protect the data exchanged (after a successful EAP authentication) between the EAP peer and authenticator. These keys are derived from the MSK and generated after the execution of a security association protocol between the peer and the authenticator.

Figure 2.9 depicts the internal structure of an EAP method showing the different parameters and key material generated during an EAP method execution. Based on the long-term credential established between the peer and the server and used to authenticate the client, EAP methods derive two types of EAP key material. On the one hand, we find key material computed locally by the EAP method but not exported such as the TEKs. Conversely, other cryptographic material is not stored locally and is exported outside the EAP method: EMSK, MSK and an *Initialization Vector* (IV) for encryption algorithms. Additionally, EAP methods are required to export/import lower-layer parameters known as *channel binding* information as a mechanism to ensure that the authenticator provides the same information (e.g., MAC address) to both peer and server. In addition to the exported key material, EAP methods supporting key derivation should export a method-specific EAP conversation identifier known as the *Session-Id*, as well as one or more method-specific peer identifiers (*Peer-Id*) and method-specific server identifiers (*Server-Id*).

From a security standpoint, only EAP methods which export key material are recommended. Additionally, the generation of key material is an aspect of vital importance

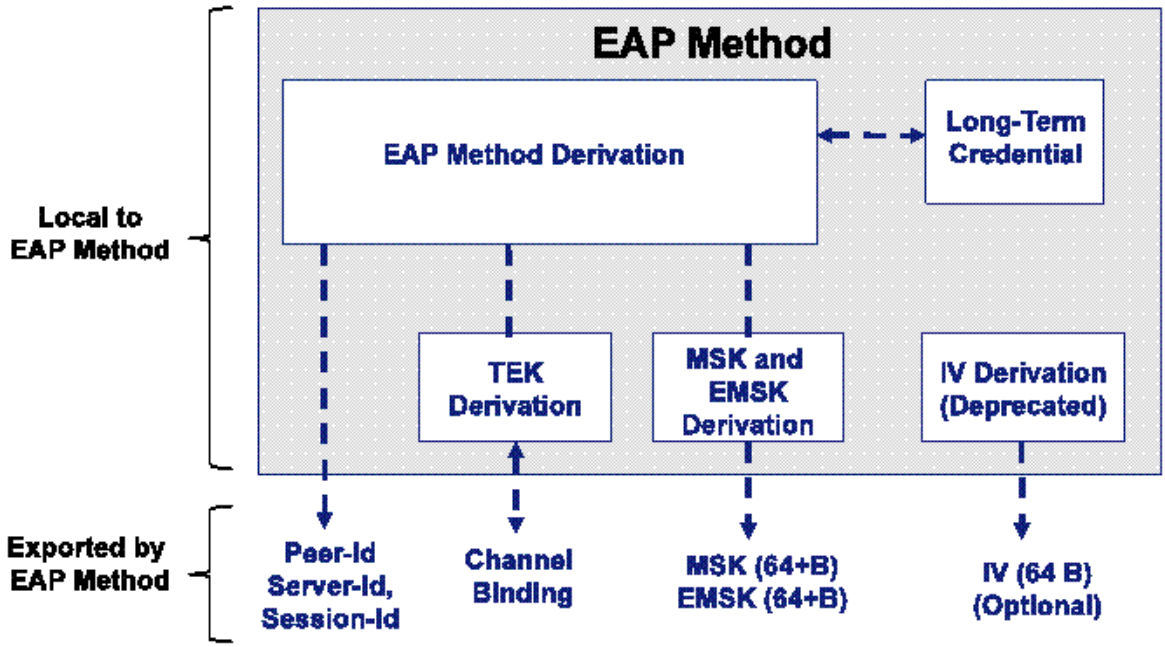


Figure 2.9: EAP Method Internals

for the definition of a secure and fast authentication process. For this reason, following the security requirements defined in [111] for EAP methods in wireless networks, only those EAP methods which are able to generate key material are considered in the context of this PhD thesis.

2.2.6 EAP Lower-Layers

The EAP lower-layer protocol allows an EAP peer to perform an EAP authentication process with an authenticator. Basically, the EAP lower-layer is responsible for transmitting and receiving EAP packets between peer and authenticator. Currently, a wide variety of lower-layer protocols can be found, since each link-layer technology defines its own solution to adopt an EAP-based access control model. However, there are also lower-layer protocols operating at network level which are able to transport EAP messages on top of IP. Finally, some other lower-layer protocols provide an hybrid solution to transport EAP packets either at link-layer or network layer. In the following, the IEEE 802.1X, PANA and IEEE 802.21 MIH protocols are analyzed as representative lower-layer solutions of the link-layer, network-layer and hybrid group, respectively.

IEEE 802.1X

The IEEE 802.1X specification [7] is an access control model developed by the IEEE that allows to employ different authentication mechanisms by means of EAP in IEEE 802 LANs. As depicted in Fig. 2.10, there are three main components in the IEEE 802.1X

authentication system: supplicant, authenticator and authentication server. In a WLAN, the supplicant is usually a mobile user, the access point usually represents an authenticator and an AAA server is the authentication server. 802.1X defines a mechanism for port-based network access control. A port is a point through which a supplicant can access to a service offered by a device. The port in 802.1X represents the association between the supplicant and the authenticator. Both the supplicant and the authenticator have a PAE (*Port Access Entity*) that operates the algorithms and protocols associated with the authentication process.

Initially, as depicted in Fig. 2.10, the authenticator's controlled port is in unauthorized state, that is, the port is *open*. Only received authentication messages will be directed to the authenticator PAE, which will forward them to the authentication server. This initial configuration allows to unauthenticated supplicants to communicate with the authentication server in order to perform an authentication process based on EAP. Once the user is successfully authenticated, the PAE will close the controlled port, allowing the supplicant to access the network service offered by the authenticator's system.

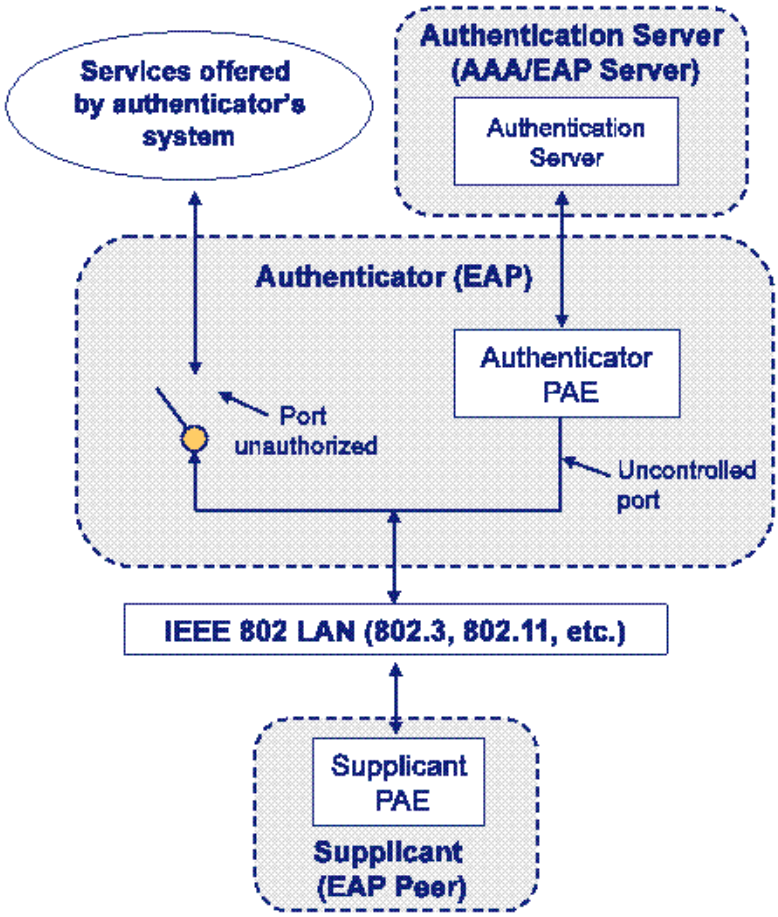


Figure 2.10: IEEE 802.1X Architecture

Apart from the definition of the authentication system, one of the most relevant

contributions of the 802.1X standard is the definition of the EAPOL (*EAP over LAN*) protocol in order to encapsulate EAP messages within IEEE 802 link-layer frames between the supplicant and the authenticator.

IEEE 802.11

Initially, IEEE 802.11 networks offered two different access control models. In the *open system authentication*, any 802.11 station (STA) that associates to an 802.11 access point (AP), is automatically granted access to the network. Therefore, no authentication process is executed. Conversely, the model based on the *Wired Equivalent Privacy* (WEP) protocol defines a simple authentication process using a shared secret key. Nevertheless, posterior security analysis of the WEP protocol [112] revealed some important vulnerabilities. For this reason, researchers developed a security extension called *IEEE 802.11i* in order to address WEP's weakness. This amendment, initially published in [60], has been finally integrated within the IEEE 802.11 core specification [7].

IEEE 802.11i defines the *Robust Security Network Association* (RSNA) concept based on the IEEE 802.1X standard previously described for implementing an access control process in IEEE 802.11 networks. In addition to the controlled and uncontrolled port concepts, the security extension employs algorithms and protocols to protect the data traffic between STA and AP once authentication has been successfully finished. More precisely, the STAs are required to successfully complete an IEEE 802.1X authentication process before gaining access to the network. Unlike IEEE 802.1X, RSNA defines a specific usage for the keys generated during the EAP authentication[2]. In this sense, once the EAP authentication is successfully completed, both STA and AP will share a *Pairwise Master Key* (PMK). This key, derived from the MSK exported by the EAP authentication, is used by a security association protocol (called *4-way handshake*) intended to negotiate cryptographic keys to protect the wireless link between STA and AP.

The authentication process, described in Fig. 2.11, involves three entities: an STA acting as supplicant, an AP acting as authenticator and an authentication server (e.g., an AAA server) that assists the authentication process. The process starts with the so-called *IEEE 802.11 association phase* where the STA firstly discovers the security capabilities implemented by the AP. Through passively monitoring messages (called *beacons*) periodically sent by the AP or actively probing (*Probe-Request/Response*), the AP announces both its identifier (SSID) and certain security information such as supported authentication modes and cryptographic suites used to protect the traffic *(1)*. From this information, the STA concludes that the AP supports the RSNA access control model. Next, the IEEE 802.11 authentication exchange *(2)* is invoked by setting the *Open System* authentication value. As previously described, the IEEE 802.11 Open System authentication provides no security, but it is included to maintain backward compatibility with the IEEE 802.11 state machine [7]. This exchange is followed by an association process *(3)* where the negotiation of the cryptographic suite is performed. In particular,

[2]As recommended in [111], RSNA requires EAP methods exporting keying method after a successful authentication

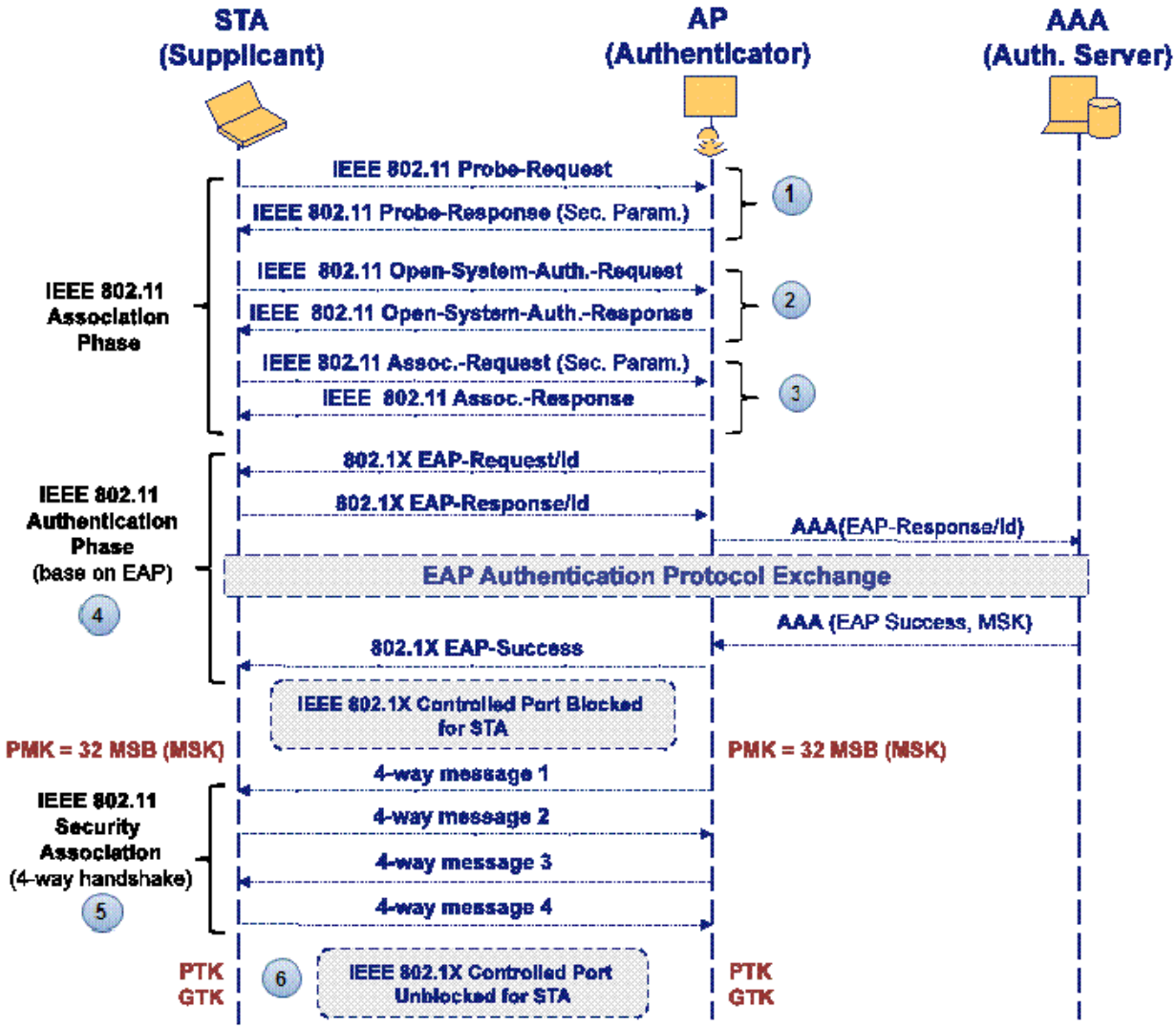


Figure 2.11: IEEE 802.11 Message Flow

while the STA includes within the *Association-Request* message the selected security parameters, the AP successfully completes the association process by answering with an *Association-Response* message.

In the subsequent *IEEE 802.11 authentication phase*, two different mechanisms are available. On the one hand, in the *Pre-Shared Key* (PSK) mode, an EAP authentication is not required since a shared key between STA and AP is used as PMK to establish a security association through the *4-way handshake* protocol. Typically, this key is statically configured in both entities. On the other hand, in the *802.1X/EAP* mode depicted in Fig. 2.11, an EAP authentication is performed where the STA acts as *EAP peer* and the STA acts as *EAP authenticator (4)*. Conversely, the *EAP server* can be co-located with the EAP authenticator (*standalone configuration*) or within an external authentication server

(*pass-through configuration*), in which case an AAA protocol (e.g., RADIUS or Diameter) is used to transport EAP messages between the authenticator and the server. Once the EAP authentication is successfully completed, the 32 *more significant bytes* (MSB) from the exported MSK is used as PMK. Finally, it is worth mentioning that a specific PMK can be cached so that, when the STA re-associates with the same AP in the future, the 802.1X/EAP authentication process is not necessary and a security association can be directly established.

Regardless of whether the PMK is derived from an EAP/802.1X authentication process, based on a PSK, or reused from a cached PMK, a *4-way handshake* protocol must be executed during the *IEEE 802.11 security association phase (5)* to successfully establish a RSNA. Following the establishment of the PMK, this key management protocol is intended to confirm the existence of the PMK and selected cryptographic suites. The protocol generates a *Pairwise Transient Key* (PTK) for unicast traffic and a *Group Transient Key* (GTK) for multicast traffic. Thus, as result of a successful *4-way handshake*, the IEEE 802.1X controlled port in the AP changes to an *authorized state (6)* and a secure communication channel between the STA and the AP is established for secure data transmissions in the wireless link.

IEEE 802.16

Initially, IEEE 802.16 networks assumed a fixed environment where the user, called *subscriber station* (SS), is an entity that does not change of location and connects to a specific IEEE 802.16 base station (BS). The IEEE 802.16e [35] specification is an amendment to the basic standard that relaxes this requirement by allowing the SS to roam between different BSs. This extension, which has been included within the last IEEE 802.16 specification [8], not only enables the mobility support but also enhances the basic access control mechanism defined for fixed scenarios in order to provide authentication and confidentiality in IEEE 802.16-based wireless networks.

In particular, the security architecture is further strengthened in IEEE 802.16e thanks to the *Privacy and Key Management* protocol version 2 (PKMv2) which provides mutual authentication and secure distribution of key material between SS and BS. The authentication can be performed by using an RSA-based or an EAP-based authentication scheme. While the former uses X.509 digital certificates together with RSA public-key encryption algorithms, the latter relies on an EAP authentication to authenticate the user.

Figure 2.12 shows the authentication process using PKMv2 and EAP. As observed, while the SS acts as *EAP peer*, the BS implements the *EAP authenticatior* functionality. Depending on the EAP configuration mode, the *EAP server* can be placed in the BS (*standalone mode*) or in a AAA server (*pass-through*), which is the situation assumed in Fig. 2.12. The SS may trigger the EAP authentication process by sending a *PKMv2 EAP-Start* message to the BS *(1)*. After that, the EAP conversation starts when the BS sends an *EAP-Request/Identity* to the peer *(2)*, which answers with an *EAP-Response/Identity* that is forwarded to the AAA server *(3)*. Next, the specific EAP method is executed *(4)*. While EAP messages exchanged between SS and BS are

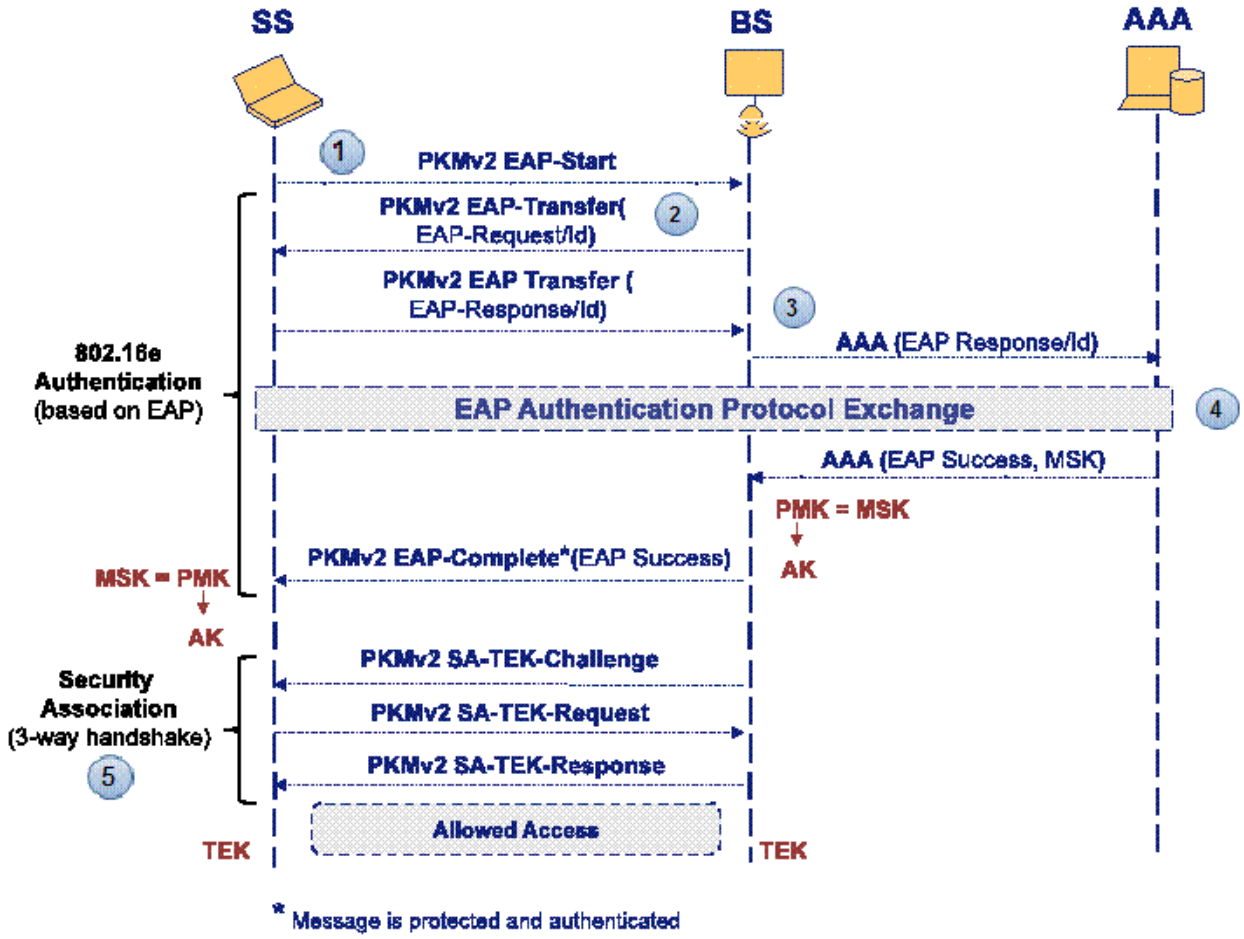


Figure 2.12: IEEE 802.16e Message Flow

transported within the *PKMv2 EAP-Transfer* message, an AAA protocol (e.g., RADIUS or Diameter) is used to convey EAP messages between the BS and the AAA server.

Once the EAP authentication is successfully completed, from the exported MSK a *Pairwise Master Key* (PMK) is derived. In turn, from this PMK, an *Authorization Key* (AK) is generated for the security association establishment. For this reason, as stated in [111], the 802.16e specification requires the use of EAP methods which export key material. Additionally, as happens in IEEE 802.11i, the PMK can be cached for a certain period of time in order to avoid the authentication process when the SS comes back to the BS in the near future.

Finally, as previously mentioned, the AK shared between SS and BS is employed by a security association protocol called *3-way handshake (5)*, which verifies the possesion of the AK and generates a *Traffic Encryption Key* (TEK) used to protect the traffic in the wireless link.

PANA

The *Protocol for carrying Authentication for Network Access* (PANA) [32] is a network-layer transport for authentication information designed by the *PANA Working Group* (PANA WG) within the IETF [113]. PANA is designed to carry EAP over UDP to support a variety of authentication mechanisms for network access (thanks to EAP) as well as a variety of underlying network access technologies (thanks to the use of UDP). PANA considers a network access control model integrated by the following entities:

- The *PANA Client* (PaC) is the client implementation of PANA. This entity resides on the subscriber's node which is requesting network access. A PaC is responsible for requesting network access and participating in the authentication process. The PaC acts as EAP peer according to the EAP model described earlier.

- The *PANA Authentication Agent* (PAA) is the server implementation of PANA. A PAA is in charge of communicating with the PaCs for authenticating and authorizing them to access the network service. The PAA acts as EAP authenticator. As such, the PAA may consult (*pass-through model*) a backend authentication server implementing the EAP server functionality (e.g. AAA server) in order to verify the credentials and rights of a PaC. In that case, AAA protocols are commonly used for communicating the PAA and the AS.

- The *Enforcement Point* (EP) refers to the entity in the access network in charge of inspecting data traffic of authenticated and authorized subscribers. Basically, the EP represents a point of attachment (e.g., access point) to the network and implements non-cryptographic (IP filters) or cryptographic filters (e.g IPsec, link-layer protection) to selectively discard data packets depending on certain parameters associated with the subscriber.

- The *Authentication Server* (AS) is in charge of verifying the credentials provided by a PaC through a PAA. The AS functionality is typically implemented by an AAA server, which also integrates the EAP server. If the AS correctly verifies the credentials, it sends authorization parameters (network access lifetime, quality of service parameters, cryptographic material, etc...) to the PAA.

As highlighted in Fig. 2.13, the architecture allows to physically separate the functionality of network access authentication (PAA) from the one dedicated to filter data traffic (EP). In other words, the PAA can be placed on either the same physical device that implements the EP functionality or a completely different one. For the first case, the PAA can simply use an API to transfer configuration information to the EP after subscriber is successfully authenticated and authorized to use the network. In the second case, the PAA uses a *configuration network protocol* (CNP), such as SNMP [114] (*Simple Network Management Protocol*), to accomplish this task. This potential separation allows to a single PAA to manage several EPs at the same time.

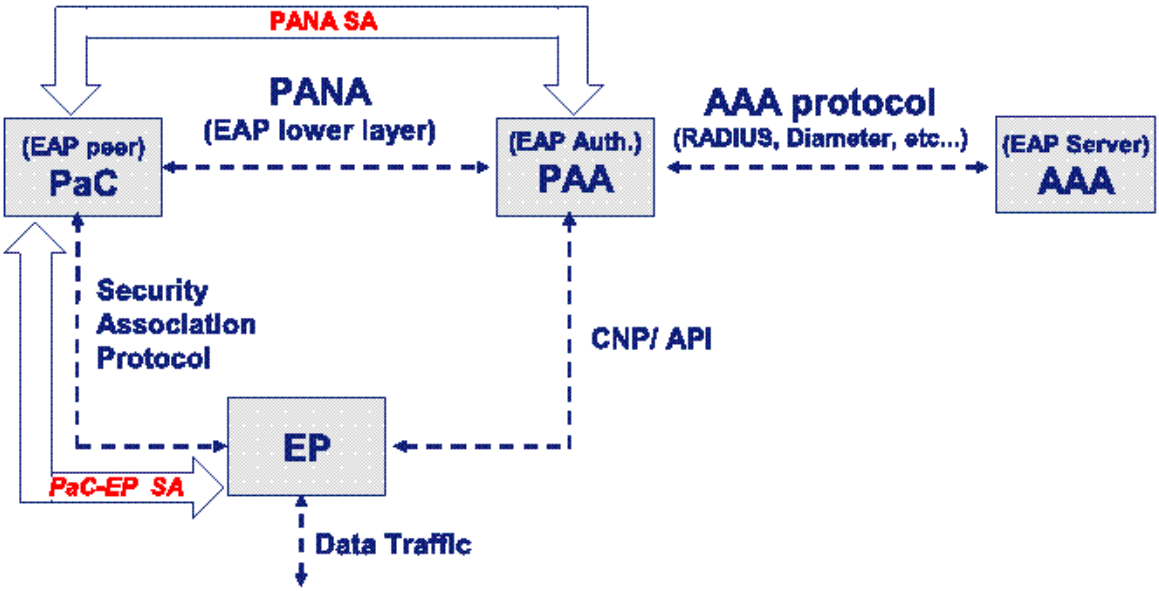


Figure 2.13: PANA Architecture

There are two types security associations related to PaC in the PANA architecture. A *PANA security association* (PANA SA) is established between the PaC and PAA in order to integrity protect PANA messages; and a *PaC-EP SA* is established by performing a security association protocol between the PaC and an EP to protect data traffic.

The PANA operation is developed along four different phases. In first place, the *authentication and authorization phase* happens. In this phase, the PaC and the PAA negotiate some parameters, such as the integrity algorithms used to protect PANA messages. They also exchange PANA messages transporting EAP to perform the authentication and establish a so-called *PANA session*. In the *pass-through* model, the PAA forwards the EAP messages to the backend AS server (where the EAP server is co-located) for verification of the credentials of the PaC.

If the PaC is successfully authenticated, the protocol enters in the *access phase* where the PaC can use the network service by just sending data traffic through the EP. If cryptographic protection is required, all PANA messages from this point to the end of PANA session are integrity protected by using the PANA SA. Similarly, data traffic is protected by means of the PaC-EP SA. If the session is about to expire, typically a *re-authentication phase* happens to renew this session lifetime. The PaC or PAA can terminate the session (e.g. the PaC desires to log out the network access session) during *termination phase*, where resources allocated by the network for the PaC are also removed. If neither PaC nor PAA can complete the termination phase, both entities can release the resources once the PANA session lifetime expires.

During each phase, a different set of messages can be sent. The general format of these messages includes a PANA header and a list of *Attribute Value Pairs (AVPs)* as information containers. For example, *EAP-Payload* AVP is used to transport the EAP

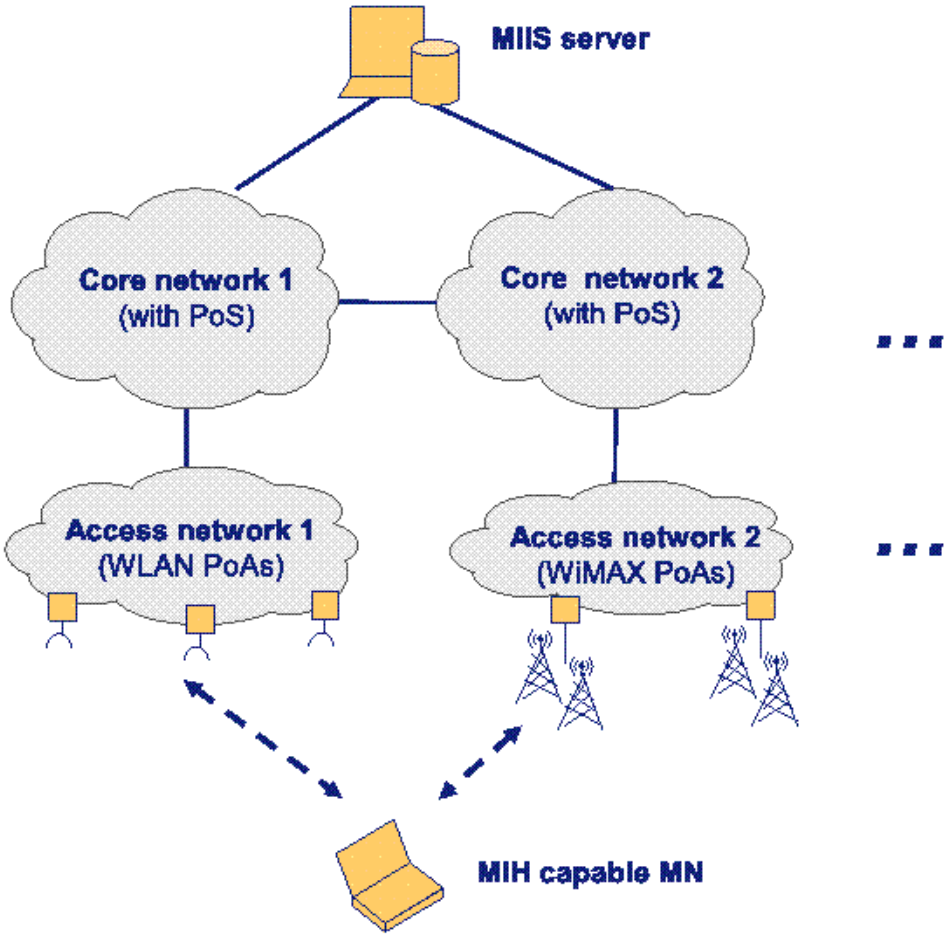
packets in PANA. Basically we can find four types of PANA messages.

- *PANA-Client-Initiation* (PCI). This message is sent by the PaC requesting the PAA start the authentication process.
- *PANA-Auth-Request/Response* (PAR/PAN). These messages are used during the authentication and authorization phase and the re-authentication phase. They allow to negotiate some parameters between the PaC and the PAA and to carry authentication information in the format of EAP packets.
- *PANA-Notification-Request/Response* (PNR/PNA). These messages are exchanged once PaC is authenticated. They are used as keep-alive mechanism of the PANA authentication session or to signal the beginning of a re-authentication process.
- *PANA-Termination-Request/Response* (PTR/PTA). These messages are used to end up a PANA session.

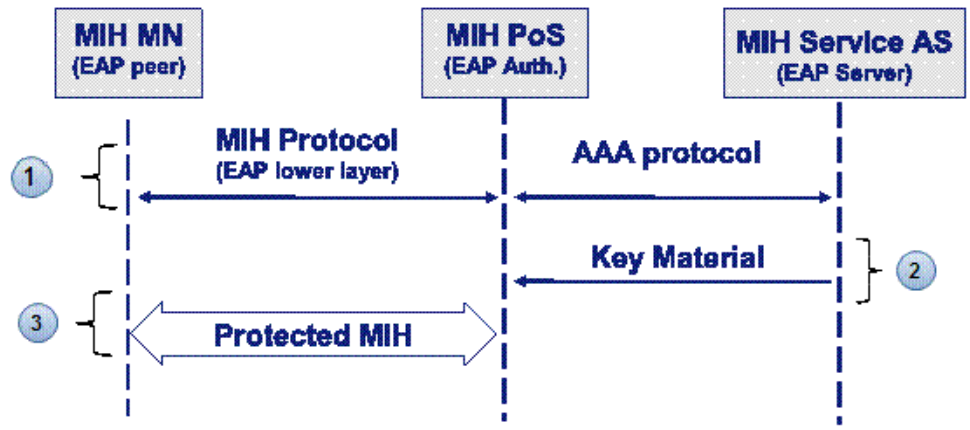
IEEE 802.21 MIH

The IEEE 802.21 is a recent effort of IEEE that aims at enabling seamless service continuity among heterogeneous networks [115, 116]. The standard defines a logical entity, *MIH Function* (MIHF), which facilitates the mobility management and handover process. The MIHF is located within the mobility management protocol stack of a mobile node (MN) or network entity. MIHF is located above various media dependent (link-layer) interfaces and provides single media independent interface to upper layers (network-layer and above). Through the media independent interface, MIHF supports useful services that help in determining the need for initiate a handoff or selecting a candidate network. In particular, the standard considers three primary services:

- *Media Independent Event Service* (MIES). The event service provides information about dynamic changes in link characteristics, link status and link quality. MIES may indicate or predict changes in state and transmission behavior of the physical and link-layers.
- *Media Independent Command Service* (MICS). This service enables upper layers over the MIHF layer to manage and control the link behaviour. For example, the MICS enables higher layers to determine the status of links and control the physical and data link-layers for optimal performance.
- *Media Independent Information Service* (MIIS). The information service is intended to provide network information within a geographical area. The objective is to gain knowledge about all heterogeneous networks in the area of interest to the terminal to facilitate handoffs when roaming across these networks.



(a) Example of network model with MIH services



(b) EAP for MIH Service Authentication

Figure 2.14: MIH Protocol as EAP Lower-Layer

Figure 2.14(a) shows an example of the network model including MIH services. A MIH capable MN has multiple wireless interfaces and is connected to *Point of Attachments* (PoAs). PoAs are network connection points (e.g., 802.11 access point) dependent on the specific underlying technology. Additionally, each access network provides one or more *Point of Service* (PoS) nodes. PoS, which supports remote events and command services, can be co-located with PoAs or located deeper in the core network. Finally, to provide the MIIS, a MIIS server maintains information on neighboring access networks.

Currently, there are several tasks groups (TG) which are defining new extensions to IEEE 802.21. For example, the standardization task group IEEE 802.21a is defining mechanisms that allow to protect the IEEE 802.21 MIH protocol exchanges. The solution [117] designed by the task group proposes that the MN must be authenticated and authorized before granting access to the services offered by the PoS. To carry out the authentication process, the 802.21 TG has agreed to use EAP. Using the key material exported after a successful EAP authentication, subsequent MIH messages exchanged between MN and PoS are protected.

Figure 2.14(b) depicts the general process followed to perform an EAP-based *Media-Independent Authentication Process*. As observed, the MN and PoS acts as EAP peer and authenticator, respectively. The EAP server functionality is implemented by a new entity named *Service Authentication Server* (Service AS). Initially, an EAP authentication (*1*) is performed between the MN and the Service AS through the PoS, which acts as authenticator. While the MIH protocol is used as EAP lower-layer to transport EAP messages between MN and PoS, an AAA protocol is employed between PoS and Service AS for the same purpose. When the credentials provided by the MN are successfully validated and the EAP authentication is completed, the Service AS sends the MSK (*2*) exported by the EAP method to the PoS. From this MSK shared between MN and PoS, the 802.21a TG defines a key hierarchy that allows to protect MIH protocol packets (*3*).

2.3 The Kerberos V5 Protocol

Kerberos [74] is a secure three-party key distribution protocol based on the use of shared secret key cryptography. More precisely, Kerberos provides a secure, single-sign-on, trusted, third-party mutual authentication service with multi-domain support:

- **Secure**. Kerberos security has been widely analyzed [76 78]. Kerberos is unique in its use of tickets, time-limited cryptographic messages that prove a user's identity to a given server without sending passwords over the network or caching passwords on the local user's hard disk.

- **Single Sing-On**. Kerberos provides a single sign-on platform through the so-called *tickets*. A ticket is a piece of encrypted and integrity protected information thanks to which end users only need to be authenticated once using their long-term credentials (e.g., password). Once a user has authenticated to Kerberos at the start of her login

session, her credentials are transparently passed to every other network resources (supporting Kerberos) she accesses during the day.

- **Trusted Third-Party**. Kerberos follows a three-party key distribution model where there exists a trusted entity in charge of authentication and key distribution. In Kerberos, this trusted third-party is called *Key Distribution Center* (KDC).
- **Mutual Authentication**. Mutual authentication ensures that not only is the user who he claims to be, but also proves that the server he is communicating with is who it claims to be.
- **Multi-domain support**. Kerberos offers a flexible support to multi-domain scenarios where a user registered in one domain solicit access to services managed by a different domain. This aspect is of enormous importance in NGNs where mobile users typically roam across different domains.

2.3.1 Kerberos Terminology: Realms and Principals

Each organization deploying Kerberos must establish its own *realm*. By convention, the Kerberos realm for a given DNS domain is the domain converted to uppercase. So, for example, the University of Murcia with domain name *um.es*, would create a Kerberos realm named *UM.ES* [3].

Kerberos messages are exchanged between three types of entities (called *principals*): a *client* that represents a user willing to access a specific service, an *application server* (also referred simply as *service*) providing a specific service, and a *Key Distribution Center* (KDC) in charge of authenticating users and distributing tickets within a specific *realm*. At the same time, the KDC is integrated by two servers: the *Authentication Server* (AS) and the *Ticket Granting Server* (TGS). While a client who wishes to be authenticated in the Kerberos realm firstly interacts with the AS, the TGS is contacted to request access to a specific service.

Kerberos identifies the different participant entities through the so-called Kerberos *principal identifiers* which have the form:

```
component[/component][/component]...@realm_name
```

Principals are globally unique names. To accomplish this, the principal is divided into a hierarchical structure integrated by three types of components, namely: *principal name*, *instance*, and *realm name*. Every principal starts with a *principal_name* component which can be either a username or service name. The name is then followed by an optional *instance* component. The instance is used in two situations. On the one hand, it allows to create special principals for administrative use. For example, administrators can have two principals: one for day-to-day usage, and another (an "admin" principal) to use only when the administrator needs elevated privileges. On the other hand, to distinguish service

[3] In the context of this PhD thesis, the terms *realm* and *domain* will be used interchangeably

principals for the same service but on different machines, the instance component contains the hostname of the machine where the service principal is located on. The username and optional instance, taken together, form a unique identity within a given realm. For this reason, after the "@" symbol, the principal identifier is completed by attaching the specific realm name where the entity is registered.

Therefore, assuming the realm UM.ES, a Kerberos principal assigned to a user named *peter* can be *peter@UM.ES*. Similarly, the principal identifier *printer/server.um.es@UM.ES* unequivocally identifiers a printing service located in the machine server.um.es within the same Kerberos realm. In addition to user and service principal identifiers, the Kerberos system itself contains other several principals. The most important of these *special* principal is the *krbtgt* principal that identifies the TGS service offered by a KDC in a specific realm.

2.3.2 The Concept of Ticket

Kerberos introduces the concept of tickets. Roughly speaking, a ticket is a record that helps a client to authenticate itself to a service. More precisely, a Kerberos ticket is an encrypted [4] data structure that allows to dynamically establish a trust relationship (based on a shared *session key*) between a client and a server. For this reason, tickets include a shared encryption key that is unique for each session. Additionally, a ticket contains the so-called ticket flags that indicate, for example, if the ticket can be forwarded to another service, along with other fields. Tickets are issued by the KDC and are encrypted using the secret key shared with the specific service. Since this key is a secret shared only between the KDC and the application server providing the service, neither the client nor observers can know it or change its contents.

Tickets can be thought as a license (issued by the KDC) that confirms the identity of a user. Just like a license in the real world, each ticket issued by Kerberos includes data about the user, how long the ticket is valid, and restrictions on its use. Kerberos distinguishes between *Ticket Granting Tickets* (TGT) and *Service Tickets* (ST). The first time the user is authenticated by the KDC to *log in* into the Kerberos realm, the KDC provides a TGT. This is the only time the provides its credentials since the TGT is used in posterior communications with the KDC when requesting an ST used to gain access to a service.

According to the Kerberos specification [74], the ASN.1 specification [119] for a Kerberos ticket is defined in Fig. 2.15. As we can see, the ticket contains in cleartext the version number for the ticket format (*tkt-vno*), the realm that issued a ticket (*realm*) and the name part of the server's principal identifier for which the ticket is intended for. The rest of the ticket contains information (*enc-part* field) encrypted with the key shared by the KDC and the server. More precisely, the encrypted information is:

[4] Kerberos employs the framework defined in RFC 3961 [118] which proposes encryption and checksum mechanisms in Kerberos. In this regard, it is important to mention that the encryption mechanism incorporates integrity protection as well, so no additional checksum is required to provide integrity protection.

```
Ticket          ::= [APPLICATION 1] SEQUENCE {
        tkt-vno         [0] INTEGER 5,
        realm           [1] Realm,
        sname           [2] PrincipalName,
        enc-part        [3] EncryptedData -- EncTicketPart
}

-- Encrypted part of ticket

EncTicketPart   ::= [APPLICATION 3] SEQUENCE {
        flags                   [0] TicketFlags,
        key                     [1] EncryptionKey,
        crealm                  [2] Realm,
        cname                   [3] PrincipalName,
        transited               [4] TransitedEncoding,
        authtime                [5] KerberosTime,
        starttime               [6] KerberosTime OPTIONAL,
        endtime                 [7] KerberosTime,
        renew-till              [8] KerberosTime OPTIONAL,
        caddr                   [9] HostAddresses OPTIONAL,
        authorization-data      [10] AuthorizationData OPTIONAL
}

--- Note: basic ASN.1 Kerberos types are defined in RFC 4120
```

Figure 2.15: Kerberos Ticket ASN.1 Specification

- The different flags (*flags* field) that are activated in the ticket (e.g., forwardable, proxiable, renewable,...)
- A shared secret key (session key) to be used for user/server communication.
- The requesting user's principal identifier (*cname* and *crealm* fields).
- The list of Kerberos realms (*transited* field) that have taken part in authenticating the user to whom this ticket was issued.
- The time when the user was initially authenticated in the Kerberos realm (*authtime* field).
- The date and time are a timestamp indicating when the tickets validity starts (*starttime* and *endtime* fields).
- In case of renewable tickets, the *renew-till* field holds the maximum endtime when the ticket can be renewed.

- The IP address of the client machine (*caddr field*) from which the ticket can be used. In Kerberos 5 this field is optional and may also be multiple in order to be able to run clients under NAT or multihomed.

- Some authorization data (*authorization-data* field) that the KDC passes to the server.

Some of these fields are filled in by the KDC. For example, the KDC enforces a maximum ticket lifetime and also generates a unique session key each time it issues a ticket. Other fields (e.g., flags) are filled in by the client and passed to the KDC when it makes a ticket request. It is important to note that, when a ticket is generated by the KDC, relevant information is encrypted and integrity protected to ensure that attackers cannot take a valid ticket and modify it.

As mentioned earlier, tickets have a special attributes called *flags* that determine the behaviour of each ticket. In the Kerberos specification [74], a complete list and associated description of the different flags can be found. Particularly interesting are the new flag types introduced in Kerberos V5:

- *Forwardable tickets*. A forwardable ticket can be forwarded to another entity later. Therefore, the ticket can be used by a new entity. A common special case is the forwardable Ticket Granting Ticket. A forwardable TGT can be forwarded to another entity, and the original TGT can be used to acquire a new TGT on the target entity, without requiring the entity to enter his password in again.

- *Proxiable tickets*. Proxiable tickets are similar to forwardable tickets since they can be transferred to another user. However, a proxiable TGT can only be used to acquire further service tickets.

- *Renewable tickets*. When a user requests a renewable ticket, he receives a ticket with a standard lifetime and a renewable lifetime. The ticket is valid only for the duration of the standard lifetime, but can be submitted back to the KDC for renewal any time before the ticket expires. The KDC can refuse to validate the ticket if, for example, it has been compromised. Otherwise, the KDC validates the ticket and returns another ticket. This process can be repeated until the ticket's renewable lifetime finally expire.

- *Postdated tickets*. A postdated ticket is only valid after some specified date in the future. If a postdated ticket is presented for validation before the start date embedded in the ticket, it will be refused.

2.3.3 Trust Relationships in Kerberos

The Kerberos protocol design assumes that, before starting the protocol execution, some trust relationships (based on a shared secret key) have been pre-established between some entities. More precisely, as depicted in Fig. 2.16, Kerberos requires the establishment of

three shared secret keys that allows a secure communication between some parties. First, between the AS and TGS within the KDC, there must exist a *KDC secret key* used to protect the TGTs issued by the AS and processed by the TGS. Second, it is required the existence of a *service secret key* between the service and the TGS, to protect the STs generated by the TGS and processed by the service. Finally, Kerberos allows users to enter in the Kerberos realm only when they share a *client's secret key* with the AS. Typically, this secret key has been derived from a password (using a *string-to-key* function as explained in [118]) while the other secret keys are set by the administrator.

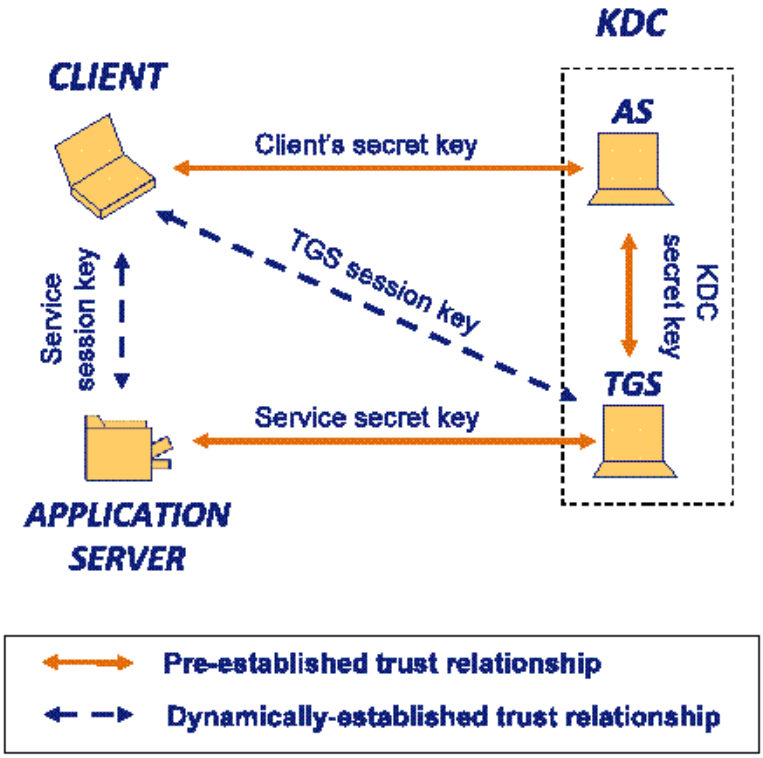


Figure 2.16: Trust Relationships Defined in Kerberos

As a consequence of the protocol execution, Kerberos dynamically establishes trust relationships between the user and some Kerberos entities. As observed in Fig. 2.16, the first trust relationship is established between the user and the TGS. This trust relationship is based on a *TGS session key* and is created when the user acquires a TGT. Similarly, when the user acquires a ST to access a service, another dynamic trust relationship is established between them. In this case, the association is based on the *service session key*.

2.3.4 Kerberos Exchanges

Previous sections have introduced the most important concepts of the Kerberos authentication system, the different entities involved in the protocol execution and briefly

outlined how Kerberos works to achieve a single-sign-on solution. Now, we present the specific protocol operation on a fundamental level.

As described in Fig. 2.17, the Kerberos communication consists of three different exchanges. Initially, by means of the *AS exchange* (KRB_AS_REQ/REP messages) *(1)*, the client contacts the AS to request a TGT. The TGT is a special ticket protected with the KDC secret key (shared between AS and TGS), which is used by the client to request STs. When the KDC receives the KRB_AS_REQ message, it generates the *TGS session key* (shared between client and TGS), which is included in the TGT. Together with other ticket-related information (e.g., validity period), this key is also sent in the KRB_AS_REP message encrypted using the client's secret key. To enhance this authentication exchange, Kerberos implements a mechanism called *pre-authentication* that allows the KDC to authenticate the client before providing the TGT. When using multi-roundtrip authentication mechanisms, client and AS can exchange authentication information through several *KRB_AS_REQ/KRB_ERROR* exchanges. When the authentication process succeeds, the AS responds with a final *KRB_AS_REP* containing the requested TGT.

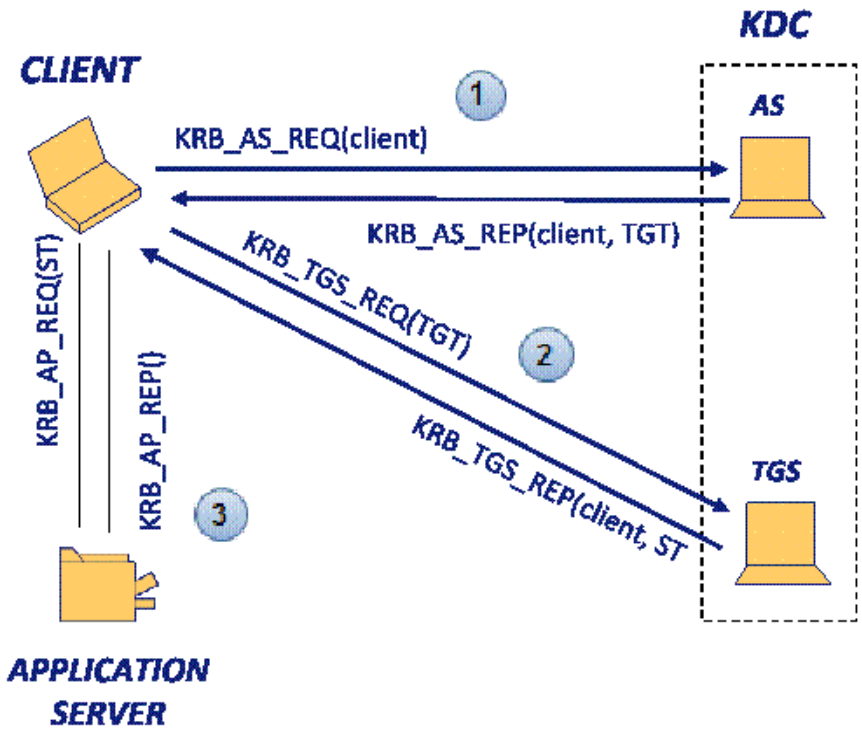


Figure 2.17: Kerberos Standard Signalling

Once the client acquires a TGT, it is ready to solicit STs for accessing different services through the *TGS exchange (2)*. To do so, the client sends a KRB_TGS_REQ message to the TGS. In addition to the the service's identity, this message contains the TGT and an authentication tag, generated with the TGS session key included in the TGT, that allow

the TGS to verify the client's identity. After successful verification of both TGT and the credentials, the TGS generates an ST (protected with the service secret key shared between TGS and service) containing the *service session key* (shared between client and service). Again, this key is also distributed to the client by means of the KRB_TGS_REP, which also transports information (protected with the TGS session key) required by the client to use the ST.

Finally, through the *AP exchange (3)*, the client authenticates itself against the service. In the KRB_AP_REQ, the client sends the previously obtained ST, together with an *authenticator* computed by using the service session key. This authenticator enables the service to verify that the client owns such a key. Optionally, when mutual authentication (authenticating not only the client to the service, but also the service to the client) is required, the service responds with a KRB_AP_REP message. In any case, after successful verification of the ST, the client and the service are able to use the service session key for protecting their application protocol.

In addition to these basic exchanges, Kerberos defines some other exchanges destined to assist the protocol operation. For example, the KRB_CRED exchange allows a client to send credentials from one entity to another (e.g., send a proxiable TGT from one client to another). Similarly, the KRB_SAFE and KRB_PRIV exchanges are used by clients and services to send application data that must be, respectively, integrity and confidentiality protected.

KRB_AS_REQ and KRB_TGS_REQ Message Specification

Both KRB_AS_REQ and KRB_TGS_REQ messages share a common format called *KDC request* (KDC-REQ) structure. As observed in Fig. 2.18, a KDC-REQ message is composed by a header and a body that are sent in cleartext. the header of the message is composed by three elements: the protocol version number (*pvno*), the message type (*msg-type*) and some authentication information transported in the pre-authentication data field (*padata*). Initially, Kerberos conceived this field to transport information that may be used by the KDC to authenticate the client. For example, in the KRB_TGS_REQ message, the padata field transports the TGT (using the *PA-TGS-REQ* padata type) used by the TGS to authenticate the client. Nevertheless, as stated in RFC 4120, the padata field has been used as a typed hole with which to extend protocol exchanges with the KDC.

The remaining fields integrate the body (*KDC-REQ-BODY*) of the message. This is the main part since it contains information regarding the user, the service for which the user demands an authenticated access and some configuration parameters about the requested ticket. In particular, the body is compound by the following fields:

- *kdc-options*: it is a bit-field that indicates the flags that the client wants to activate on the requested ticket. For example, by using this field, a renewable ticket can be solicited.

- *cname* and *sname*: these fields transport the same information as explained in section 2.3.2.

```
AS-REQ          ::= [APPLICATION 10] KDC-REQ

TGS-REQ         ::= [APPLICATION 12] KDC-REQ

KDC-REQ         ::= SEQUENCE {
     pvno            [1] INTEGER 5 ,
     msg-type        [2] INTEGER 10 -- AS -- | 12 -- TGS --,
     padata          [3] SEQUENCE OF PA-DATA OPTIONAL
                         -- NOTE: not empty --,
     req-body        [4] KDC-REQ-BODY
}

KDC-REQ-BODY    ::= SEQUENCE {
     kdc-options             [0] KDCOptions,
     cname                   [1] PrincipalName OPTIONAL
                                 -- Used only in AS-REQ --,
     realm                   [2] Realm
                                 -- Server's realm
                                 -- Also client's in AS-REQ --,
     sname                   [3] PrincipalName OPTIONAL,
     from                    [4] KerberosTime OPTIONAL,
     till                    [5] KerberosTime,
     rtime                   [6] KerberosTime OPTIONAL,
     nonce                   [7] UInt32,
     etype                   [8] SEQUENCE OF Int32 -- EncryptionType
                                 -- in preference order --,
     addresses               [9] HostAddresses OPTIONAL,
     enc-authorization-data  [10] EncryptedData OPTIONAL
                                 -- AuthorizationData --,
     additional-tickets      [11] SEQUENCE OF Ticket OPTIONAL
                                    -- NOTE: not empty
}

--- Note: basic ASN.1 Kerberos types are defined in RFC 4120
```

Figure 2.18: KRB_AS_REQ / KRB_TGS_REQ ASN.1 Specification

- *realm*: this field specifies the realm of the server's principal identifier. In the AS exchange, this is also the realm where the client is registered.

- *from*, *till* and *rtime*: through these fields, the client demands to the KDC a specific lifetime configuration on the requested ticket. Respectively, they allow to specify the desired ticket start time, expiration date and renewal period. Only the *till* time is a

mandatory field.

- *nonce*: this field is used to provide freshness. More precisely, it holds a random number generated by the client. If the same number is included in the encrypted response from the KDC, it provides evidence that the response is fresh and has not been replayed by an attacker.

- *etype*: it is intended to specify the desired encryption algorithm to be used in the response.

- *addresses*: this optional field indicates the addresses from which the requested ticket must be valid. When used, it typically includes the client's host IP address. The contents of this field are usually copied by the KDC into the *caddr* field of the resulting ticket (see Section 2.3.2).

- *enc-authorization-data*: this optional field can be only present in the KRB_TGS_REQ message and transport encrypted authorization data the client desires to be included in the requested ticket. Typically, the TGS session key is used to encrypt the information.

- *additional-tickets*: some advanced features (e.g., user-to-user authentication) require the inclusion of additional tickets when communicating with the TGS. This optional field is destined to be used in these situations.

KRB_AS_REP and KRB_TGS_REP Message Specification

Similarly to the KRB_AS_REQ/KRB_TGS_REQ message specification, the KRB_AS_REP and KRB_TGS_REP messages also share a common format called *KDC response* (KDC-REP) structure. As depicted in Fig. 2.19, in the KDC-REP message we can distinguish the header and the body part. The header is the same as that described for the KDC-REQ structure and contains the protocol version number (*pvno*), the message type (*msg-type*) and pre-authentication information (*padata*).

In the body of the KDC-REP we find both encrypted and cleartext fields. The client's principal identifier (*cname* and *crealm* fields) together with the delivered ticket (a TGT in KRB_AS_REP or a ST in KRB_TGS_REP) are not encrypted. However, the Kerberos protocol protects information associated with the delivered ticket and required by the user to properly use it in the future:

- *key*: this field contains the session key (also contained within the ticket) distributed to the client and server. In the case of a KRB_AS_REP, this field holds the TGS session key. In the case of a KRB_TGS_REP message, this field contains the service session key.

- *key-expiration*: this field specifies the time that the distributed session key is due to expire.

```
AS-REP          ::= [APPLICATION 11] KDC-REP

TGS-REP         ::= [APPLICATION 13] KDC-REP

KDC-REP         ::= SEQUENCE {
        pvno            [0] INTEGER 5,
        msg-type        [1] INTEGER 11 -- AS -- | 13 -- TGS --,
        padata          [2] SEQUENCE OF PA-DATA OPTIONAL
                                -- NOTE: not empty --,
        crealm          [3] Realm,
        cname           [4] PrincipalName,
        ticket          [5] Ticket,
        enc-part        [6] EncryptedData
                                -- EncASRepPart or EncTGSRepPart,
                                -- as appropriate
}

EncASRepPart    ::= [APPLICATION 25] EncKDCRepPart
EncTGSRepPart   ::= [APPLICATION 26] EncKDCRepPart

EncKDCRepPart   ::= SEQUENCE {
        key             [0] EncryptionKey,
        last-req        [1] LastReq,
        nonce           [2] UInt32,
        key-expiration  [3] KerberosTime OPTIONAL,
        flags           [4] TicketFlags,
        authtime        [5] KerberosTime,
        starttime       [6] KerberosTime OPTIONAL,
        endtime         [7] KerberosTime,
        renew-till      [8] KerberosTime OPTIONAL,
        srealm          [9] Realm,
        sname           [10] PrincipalName,
        caddr           [11] HostAddresses OPTIONAL
}

--- Note: basic ASN.1 Kerberos types are defined in RFC 4120
```

Figure 2.19: KRB_AS_REP / KRB_TGS_REP ASN.1 Specification

- *nonce*: this field contains the random number sent by the client in the KDC-REQ message.
- *last-req*: this field is returned by the KDC and specifies the time of the last communication between the client and the KDC. In case it is the first interaction

(e.g., the very first AS exchange), this field does not convey any information.

- *sname*, *srealm*: these fields transport the server's principal identifier for which the delivered ticket is intended to be used.

- *flags*, *authtime*, *starttime*, *endtime*, *renew-till* and *caddr*: these fields are the same as described for the KDC-REQ structure.

KRB_AP_REQ and KRB_AP_REP Message Specification

Conversely to the messages exchanged with the KDC, the KRB_AP_REQ and KRB_AP_REP messages are used by the client to authenticate itself against a service. Moreover, they have a different specification and do not share a common format structure. As depicted in Fig. 2.20, the KRB_AP_REQ message starts with a typical header consisting of the protocol version (*pvno*) and the message type (*msg-type*). The message body consists of three fields. Firstly, the *ap-options* is a bit-field that affects the way the AP exchange will be performed. For example, by means of this field, the client can demand mutual authentication, therefore requiring the service to respond with a KRB_AP_REP message. Secondly, the body of a KRB_AP_REQ transports the ST (previously obtained during a KRB_TGS_REQ/KRB_TGS_REP) authenticating the client to the service. Nevertheless, the ST itself does not authenticate the client. The service needs some additional information in order to certify that the user knows the service session key contained within the presented ST. For this reason, the message includes a final *authenticator* field which, by using the ticket session key, confidentiality and integrity protects the following information:

- *authenticator-vno*: it specifies the version number for the format of the authenticator.

- *cname* and *crealm*: these fields indicate the user's principal identifier.

- *cksum*: this optional field contains a checksum of the application data (if any) that accompanies the KRB_AP_REQ.

- *cuser* and *ctime*: these fields are used to express the current time on the client's host.

- *subkey*: this field contains a specific key to be used to protect the specific application data. When this field is unused, the service session key (present in the ST) will be used.

- *seq-number*: this optional field is destined to specify an initial sequence number to be used in posterior Kerberos messages exchanged between the client and the service. This sequence number is typically used with other Kerberos messages (not explained in this chapter for simplicity) such as KRB_PRIV or KRB_SAFE.

- *authorization-data*: in this field the client can optionally include some authorization data that may help the service to decide if access should be granted to the client.

```
AP-REQ          ::= [APPLICATION 14] SEQUENCE {
        pvno            [0] INTEGER 5,
        msg-type        [1] INTEGER 14,
        ap-options      [2] APOptions,
        ticket          [3] Ticket,
        authenticator   [4] EncryptedData -- Authenticator
}

Authenticator   ::= [APPLICATION 2] SEQUENCE  {
        authenticator-vno       [0] INTEGER 5,
        crealm                  [1] Realm,
        cname                   [2] PrincipalName,
        cksum                   [3] Checksum OPTIONAL,
        cusec                   [4] Microseconds,
        ctime                   [5] KerberosTime,
        subkey                  [6] EncryptionKey OPTIONAL,
        seq-number              [7] UInt32 OPTIONAL,
        authorization-data      [8] AuthorizationData OPTIONAL
}

AP-REP          ::= [APPLICATION 15] SEQUENCE {
        pvno            [0] INTEGER 5,
        msg-type        [1] INTEGER 15,
        enc-part        [2] EncryptedData -- EncAPRepPart
}

EncAPRepPart    ::= [APPLICATION 27] SEQUENCE {
        ctime           [0] KerberosTime,
        cusec           [1] Microseconds,
        subkey          [2] EncryptionKey OPTIONAL,
        seq-number      [3] UInt32 OPTIONAL
}

--- Note: basic ASN.1 Kerberos types are defined in RFC 4120
```

Figure 2.20: KRB_AP_REQ / KRB_AP_REP ASN.1 Specification

Compared with the different messages previously described, the KRB_AP_REP message is the simplest one since the body is only required to contain an encrypted timestamp (using the service session key) indicating the current time on the service's host. When the optional fields *subkey* and *seq-number* are used in the request, the KDC must confirm the received values.

2.3.5 Cross-realm Operation

When a client requests access to a service not controlled by the KDC where the client is registered (*home KDC*), Kerberos defines a special operation mode called *cross-realm* (see Fig.2.21). Cross-realm authentication allows a client from one organization to be authenticated in another, thanks to the definition of trust relationships between TGS/KDCs of different realms. These trust relationships are established through the definition of the so-called *inter-realm keys*. A cross-realm authentication process starts when the client engages in a KRB_AS_REQ/REP exchange (*1*) with its home AS in order to obtain a *single-realm* TGT. This TGT is used in a posterior KRB_TGS_REQ/REP exchange (*2*) with the home TGS where the client requests a special ticket called *cross-realm* TGT. This type of ticket can be used to communicate with a TGS located (*3*) in a different realm thanks to the deployed multi-realm KDC architecture. By repeating this process, the client follows the authentication path from the home to the *visited* realm, where the service is deployed. Finally, the TGS placed in the visited realm delivers a ST (*4*) used by the client to authenticate itself to the service (*5*).

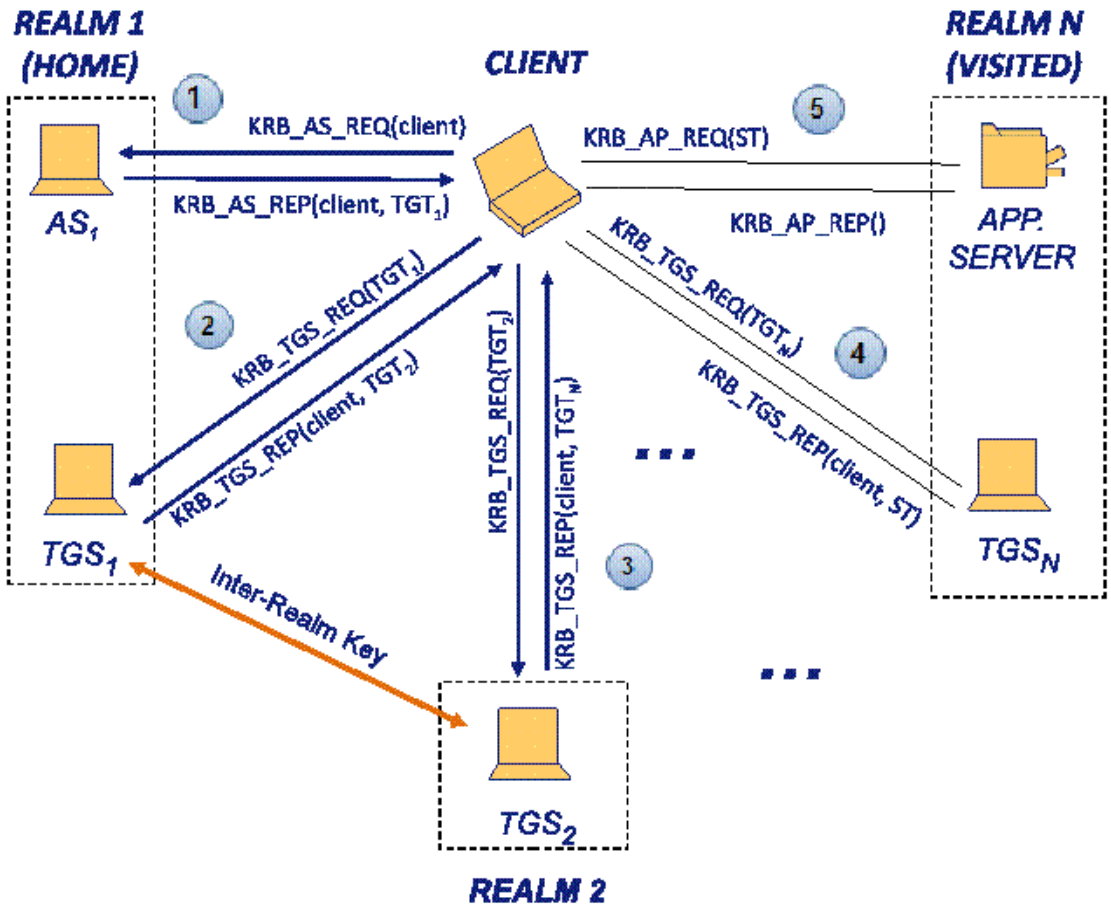


Figure 2.21: Kerberos Cross-Realm Operation

2.4 Existing Fast Re-authentication Solutions

Once it has been described the different technologies related to the EAP authentication process and presented the secure three-party protocol (Kerberos) which we are going to use in the PhD thesis to develop a fast re-authentication architecture, the reader has the necessary background to understand all the contributions that will be presented in the following chapters. Nevertheless, before starting explaining the proposed solution, it is interesting to analyze other efforts that have attempted to both reduce the authentication time during network access control process and preserve the privacy of the user during the process. To the best of our knowledge, any existing solution addresses both objectives at the same time in the context of EAP-based network access authentication. For this reason, while in this section we examine fast re-authentication proposals trying to reduce the EAP authentication time, in next section 2.5 we will present different privacy-enhanced authentication mechanisms.

As explained in section 1.4.1, existing fast re-authentication solutions can be classified in five groups: *context transfer*, *pre-authentication*, *key pre-distribution*, *use of a local server*, and *modifications to EAP*. In the following, we delve into each of them and detail the mechanism proposed to achieve a reduced handoff latency.

2.4.1 Context Transfer

As depicted in Fig. 2.22, the context transfer mechanism tries to reduce the time devoted to network access control by transferring cryptographic material (*1*) from an EAP authenticator (*current*) to a new one (*target*). When the user moves to the new authenticator (*2*), it can use the transferred context (e.g., cryptographic keys and associated lifetimes) to execute a security association protocol with the new authenticator (*3*) to protect the wireless link. Thus, the user does not need to be authenticated and can directly start the security association establishment process based on the transferred cryptographic material.

In order to perform a secure transference between both authenticators, it is assumed the existence of a pre-established security association between them. Additionally, context transfer solutions do not propagate the same cryptographic material (CM) from one authenticator to another. Instead, the transferred cryptographic material is derived (CM') from that owned by the current authenticator where the user is connected. The process employed to generate the derived cryptographic material is followed by both the peer and the authenticator. While the authenticator transfers the derived material to the new authenticator, the peer employs it to start the security protocol execution.

Depending on when the transference is performed, we can distinguish between *reactive* and *proactive* schemes. In the proactive mode, the context transfer is performed before the peer performs the handoff. Therefore, when the peer moves to the new authenticator, the cryptographic material has been already transferred to the new authenticator and the peer can immediately establish the security association. Conversely, in the reactive mode, the context transfer is performed once the user performs the handoff and is under the coverage

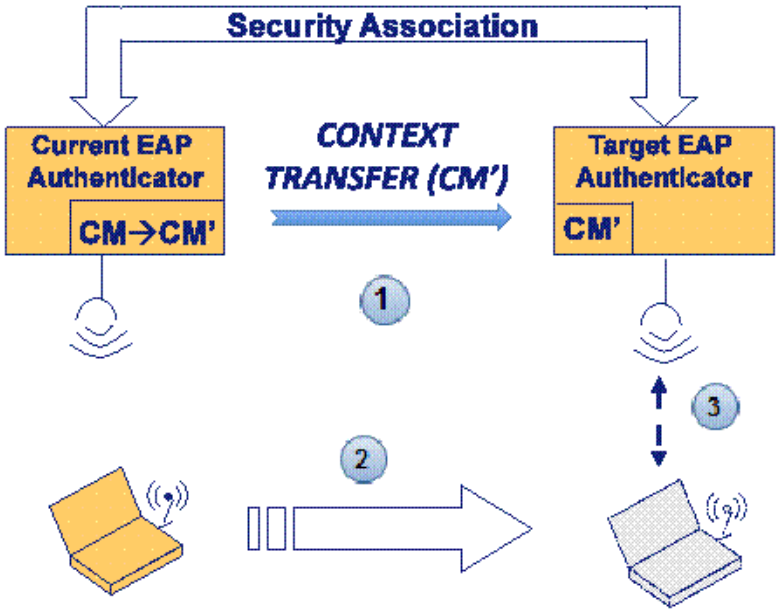


Figure 2.22: Context Transfer Mechanism

area of the new authenticator. The proactive mode introduces less latency to network access control than the reactive mode since the transference of cryptographic material is not performed in advance before the handoff. Nevertheless, reactive solutions are interesting in situations where the handoff happens unexpectedly and there is no anticipation to perform the transference.

One of the first context transfer solutions was presented by Aura et al [56]. This solution proposes a mechanism to enable a fast handoff in IEEE 802.11-based wireless networks. Initially, the base station (EAP peer) obtains from the current access point (EAP authenticator) a credential and a key K. When the base station moves to a new access point, it presents the credential to the new access point. This credential is protected using a key K_{net} shared by all the access points and, in particular, by the new access point. Using this key, the access point obtains the key K contained in the credential. Once the credential has been successfully validated by the access point, both the peer and the authenticator engage in a lightweight authentication process based on the key K. An important advantage of this solution is that network access control process can be completed without contacting an external authentication server located in the network. Nevertheless, there are serious limitations. Firstly, the deployment of this solution requires serious modifications to both 802.11 access points and base stations. Secondly, this optimization can be only employed in 802.11 networks, not being applicable in heterogeneous scenarios. Thirdly, the key K_{net} is a critical element in the system. If this shared key is compromised in some access point of the system, the rest of access points sharing the same key K_{net} are also affected [5]. Furthermore,

[5] As will be discussed later, this problem (known as the *domino effect*) is a security vulnerability inherent to solutions following a context transfer approach.

in practice, the distribution of this key is only feasible over those access points controlled by the same operator, even when these access belong to different networks managed by the same operator. Therefore, this solution has some deployment drawbacks in inter-domain handoffs.

Another relevant solution is the *Mobility-adjustment Authentication Protocol* (MAP) presented by Kim et al [57]. The MAP protocol allows an authentication server (AS) to verify the authenticity of a peer and provide keys to secure the wireless link. The authentication is based on the use of a pre-shared key (PK) between the peer and the AS. The first criticism to this work is that has been designed to be applied only in 802.11i wireless networks. Furthermore, the authentication mechanism is dependent on the technology since it does not operate over EAP. To speed up the authentication process, the solution defines a new entity called *Security Context Node* (SCN). The SCN is logically distinct from the AS, although both may reside on the same physical machine. Once the peer performs an initial authentication, the SCN receives a security context from the AS. By using this security context, the SCN re-authenticate the peer on behalf of the AS. When the peer moves to an authenticator under de control of a different SCN, the security context is transferred from the previous to the new SCN. The SCNs are distributed in each domain so that they can mitigate the authentication latency during inter-domain handoffs. In this case, it is assumed the existence of security association agreements that allows a secure context transfer by means of inter-realm keys. Similarly, Politis et al [18] propose a mobility management architecture (independent of the wireless link technology) that transfers AAA information between access routers located in different domains in order to avoid the AAA state re-establishment time during the handoff. Despite from a technical point of view, the application of context transfer is feasible, in practice it is not realistic to assume an association agreement between every pair of existing domain.

Context transfer mechanisms have been also used by standardization organizations. For example, the *IEEE 802.11F* or *Inter-Access Point Protocol* (IAPP) [59] is a recommendation that describes an optional extension to IEEE 802.11 technology. IAPP proposes to transfer security context between APs in order to speed up the handoff process by avoiding full re-authentication and reusing previously established trust relationships. More precisely, IAPP allows an access point (AP) to communicate over UDP with other APs to exchange relevant information of associated peers. Since this communication is allowed only between APs belonging to the same distribution systems (DS), IAPP optimization is only applicable to intra-network handoffs. IAPP requires the assistance of a RADIUS server which provides two functions: (1) mapping of the BSSID of an AP to its IP address on the DS, (2) distribution of keys to APs to allow secure communications between APs based on IPSec [24]. IAPP defines two modes of operation. In reactive mode, when the peer moves to a new AP, the peer starts a re-association process and provides the BSSID of the old AP. With the help of the RADIUS server, the new AP obtains the IP address of the old AP and contacts him to solicit the context associated with the specific peer. This behaviour is also followed in proactive mode, but with the difference that current access point distributes the security context of the peer to neighbouring access points before the peer moves to any of them. Nevertheless, IAPP does not propose any

algorithm that can guide this context pre-distribution.

Within the *Internet Engineering Task Force* (IETF), also it can be found some standardization efforts that use a context transfer approach to define an optimized network access control process. For example, one representative solution can be found in the work [58] presented by the *PANA Working Group* where the *Context Transfer Protocol* [120] (CxTP) is used to transfer PANA sessions between authenticators. When the peer (*PaC*) moves to a new authenticator (*PAA*), it provides the identifier of the PANA session to be transferred. This identifier contains the identity of the previous authenticator. Using the CxTP protocol, the PANA context is transferred from the previous to the new authenticator. Once the cryptographic material is transferred, both the peer and the authenticator can execute a security association protocol to protect the wireless link. Since PANA is an application level protocol for network access authentication, this context transfer solution is independent of the underlying transmission technology. However, the solution leaves out of scope the security model to protect the context transfer process.

In general, context transfer mechanisms have been widely criticized as a promising technique to achieve a fast network access due to an important security vulnerability known as the *domino effect* [69]. The problem comes from the fact that context transfer re-uses the same cryptographic material (or a derived one following a well-known process) in different authenticators. Therefore, if one authenticator is compromised, the rest of authenticators visited by the same user are also affected.

2.4.2 Pre-authentication

Pre-authentication solutions propose a scheme (see Fig. 2.23) where the mobile user performs a full EAP authentication (*1*) with a candidate authenticator through the current associated one *before* it performs the handoff. In this manner, when the handoff happens (*2*), given that the MSK generated during the pre-authentication process will be already present in the candidate authenticator, the peer only needs to establish a security association (*3*) with it to protect the wireless link. As we see, pre-authentication decouples the authentication and network access control operations from the handoff.

Depending on the role adopted by the current authenticator during the EAP pre-authentication, we can distinguish two scenarios of EAP pre-authentication signalling [121]:

- *Direct pre-authentication.* In this type of EAP pre-authentication, the current authenticator only forwards the EAP lower-layer messages between mobile node and candidate authenticator as it would be data traffic.

- *Indirect pre-authentication.* Here, the current authenticator plays an active role during pre-authentication process. This type of pre-authentication is useful when the mobile node neither has the candidate authenticator address nor is able to access to the candidate authenticator for security reasons. Therefore, there is a signalling from mobile node to/from current authenticator, and from/to the current authenticator

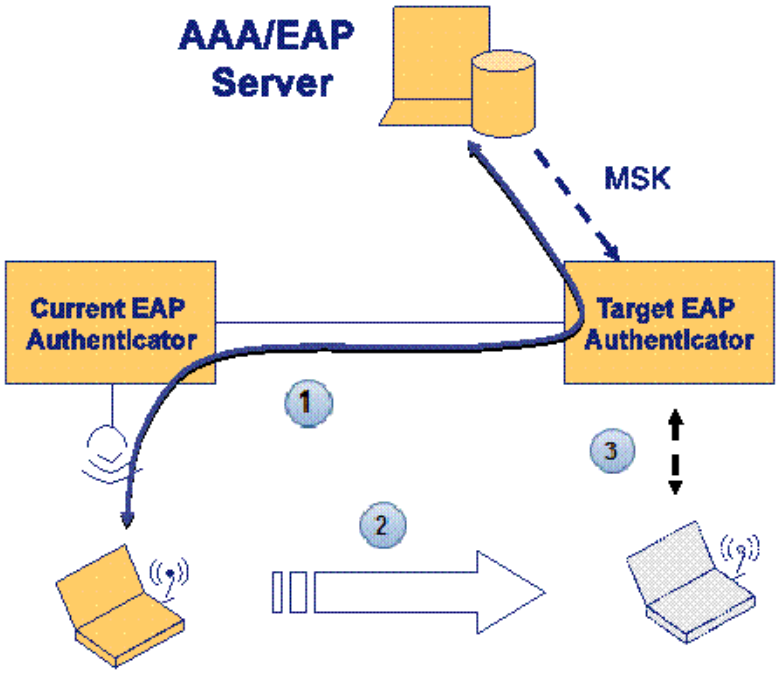


Figure 2.23: Pre-authentication Mechanism

to/from the candidate authenticator. Note that current authenticator does not act as an EAP authenticator; it only translates between different EAP lower-layer protocols.

According to the layer where the pre-authentication solutions are defined, we can distinguish between link-layer and network-layer pre-authentication mechanisms. According to this classification, in the following it is described the most relevant pre-authentication proposals.

Link-layer Pre-authentication

Pre-authentication at link-layer was initially introduced in the IEEE 802.11i [60] technology. IEEE 802.11i provides a stronger security to IEEE 802.11 WLAN by means of a full EAP authentication process and the subsequent link-layer security association. The improvement, which was included in the last revision of the IEEE 802.11 standard [7], offers a pre-authentication mechanism where a 802.11 base station (STA) is able to start a new EAP authentication with a candidate 802.11 access point (AP) through the current associated one. The involved APs are required to belong to the same distribution system. Finally, when the peer roams to the candidate AP, it establishes a security association through the *4-way handshake* protocol. Therefore, EAP authentication is not performed after the handoff.

However, IEEE 802.11i pre-authentication is not a highly optimized process. Each pre-authentication involves a full EAP authentication and consequently, requires a considerable amount of message exchanges with the AAA server. Additionally, a full 4-way

handshake execution is required to be completed after the movement. To overcome these deficiencies, an improvement called IEEE 802.11r [122] has been designed. This link-layer technology introduces a new key hierarchy level which avoids the need to run a full EAP authentication each time the peer roams to a new access point located within the same so-called *Mobility Domain*. Another improvement respect to 802.11i is that 802.11r allows to perform part of the 4-way handshake and some resource reservation (e.g., QoS) at the candidate AP before the movement takes place. This process is performed through the current AP to which the STA is associated and using the distribution system shared by both APs.

Nevertheless, 802.11r still has some drawbacks inherited from 802.11i that make it inappropriate for real mobility scenarios. First, the pre-authentication mechanism only works when the involved APs belong to the same distribution system. Second, the mechanism is only valid for APs which employ the same link-layer technology (802.11 in this case). For this reason, this kind of pre-authentication cannot be used in some type of movements like inter-network, inter-domain or inter-technology handoff.

Network-layer Pre-authentication

Definition of a pre-authentication mechanism at link-layer has some serious limitations since they cannot be applied for cases involving inter-domain or inter-technology handoffs. To avoid this problems, some other solutions have proposed to apply a pre-authentication procedure at network layer. Network layer solutions have the advantage of being capable to work independent of the underlying access technologies and with authenticators located in different networks or domains.

Following this approximation, the *IETF PANA Working Group* has developed the pre-authentication support for PANA [61]. More precisely, the working group participants describe which extensions are required to the PANA protocol for proactively executing an EAP authentication and establishing a PANA SA between a PaC in one access network and a candidate PAA in another access network to which the PaC may move. Authors propose a direct pre-authentication model where, the PaC directly communicates with the new PAA. The PANA pre-authentication exchange is distinguished from a standard one thanks to a *pre-authentication flag* activated by both the PaC and the PAA in every message. This flag is cleared when the PaC performs the handoff and the candidate PAA becomes the current PAA.

The PANA pre-authentication framework is used in [15] to develop an architecture to optimize the handoff process in heterogeneous wireless networks. Authors propose a PANA-based pre-authentication at network-layer to assist link-layer handoff optimization techniques by allowing fast transition in certain types of handoffs where link-layer solutions are not able to operate. The proposed pre-authentication mechanism defines an entity acting as PAA in charge of controlling a set of points of attachments (e.g., a 802.11 access point). The pre-authentication process works as follows. Initially, the PaC pre-establishes a PANA SA (by performing an EAP exchange) with a PAA located in a candidate network, via the current network. From the key material (MSK) generated after the successful EAP

authentication method, the PAA derives different keys per point of attachment. Using the SNMP protocol [114], the PAA installs these keys in the different point of attachments. At the same time, the user is provided with the required information to derive those keys. Therefore, when the user moves to any of these pre-configured point of attachments, the user runs a specific security association protocol by using the specific key generated during the pre-authentication process.

The *Media-Independent Pre-Authentication* (MPA) [17] is another solution that uses the pre-authentication concept to propose a secure handoff optimization scheme. MPA is a mobile-assisted architecture that allows a user to proactively perform the necessary configuration tasks with a candidate network before the handoff. An important advantage of MPA is that not only allows a mobile user to securely obtain an IP address and other configuration parameters for a candidate network, but also to send and receive traffic using that IP address before it attaches to the candidate network. Therefore, for example, the user can authenticate itself against a candidate point of attachment, establish a security association to protect the wireless link traffic in the candidate network and complete the binding update of any mobility management protocol.

To achieve this objective, MPA defines four steps. Firstly, during the *pre-authentication* procedure, the user establishes a security association with the candidate network to protect subsequent protocol signalling. The second procedure is referred to as *preconfiguration* and securely executes a configuration protocol to obtain an IP address and other parameters from the candidate network. During third step a tunnel management protocol is executed to establish a proactive handover tunnel (PHT) between the mobile user and an access router in the candidate network. This PHT is used to securely transmit IP packets (e.g., application data packets) containing the IP address acquired during the preconfiguration phase as part of the inner tunnel address. Finally, the last step deletes the PHT immediately before attaching to the candidate network and reassigns the inner address of the deleted tunnel to its physical interface immediately after the mobile device attaches to the target network. The third and fourth steps are referred to as *secure proactive handover*.

MPA can potentially provide a reduced handoff latency since, in the aforementioned third step, the user can complete tasks such as authentication and mobility management procedures before the user moves to the candidate network. In other words, MPA decouples higher-layer handoff procedures such as authentication from the critical link-layer handoff to establish connectivity with the candidate network. To proactively perform these procedures, MPA requires an efficient engine able to predict movements sufficiently in advance. Nevertheless, in some situations, connectivity may be suddenly lost and the pre-authentication cannot be completed. These situations provokes that pre-authentication cannot operate and their advantages cannot be obtained.

Despite pre-authentication solutions can potentially achieve an important reduction in the latency introduced by the authentication process during the network access control, this technique presents important drawbacks. First, pre-authentication requires the existence of network connectivity to carry out the pre-authentication process which is

a requisite that may not always be satisfied. Second, pre-authentication requires a precise selection of the authenticator with which perform a pre-authentication process. If the user performs a pre-authentication with authenticators where the user finally does not move, the technique may incur in an unnecessary use of network resources. The third disadvantage is related to the previous one. Since pre-authentication implies the pre-reservation of resources in candidate authenticators, in practice, operators are reluctant to pre-reserve resources for users that may or may not roam in the future. Therefore, pre-authentication may have a limited application, specially in inter-domain handoffs. Finally, given that pre-authentication involves a full EAP authentication, special care must be taken to determine the moment to start the pre-authentication process. As a consequence, pre-authentication needs to be performed with a considerable anticipation to the handoff.

2.4.3 Key Pre-distribution

Key pre-distribution solutions propose the pre-installation of cryptographic material (e.g., keys) in candidate authenticators so that the keys required for secure association are already available when the peer moves to the authenticators. As depicted in Fig. 2.24, the mobile user initially performs an EAP authentication (*1*) with the AAA server. Once the EAP authentication is successfully completed, the AAA server pre-distributes keys (*2*) to authenticators which the user can potentially associate to in a near future. Therefore, when the peer moves to a new authenticator (*3* and *5*), it is not required to perform a full EAP authentication. Instead, using the key material already present in the authenticator and known by the peer, a security association is established between both entities (*4* and *6*). As happen with pre-authentication, a critical aspect in key pre-distribution solutions relies on the correct selection of those authenticators to which pre-distribute cryptographic material. To avoid unnecessary waste of resources in authenticators where the peer finally does not move to, it is required an effective selection mechanism that identifies the set of authenticators where the peer, with high probability, will move in the future.

One of the earliest solutions based on key pre-distribution were proposed in [70] and [62]. Both proposals, designed for 802.11 networks, describe an algorithm that steer the key installation process based on the peer's movement. To carry out the pre-distribution process, authors in [70] introduce the concept of *neighbours graph*. A neighbour graph is a structure destined to collect information about the peer's movements. Each node of the graph represents an access point and an edge between a pair nodes $[i, j]$ represents the ability of the peer to perform a re-association with the involved access points AP_i and AP_j. This solution has two main disadvantages. On the one hand, to construct the neighbour graph and achieve an optimal operation of the solution, the mobile user has to perform a full EAP authentication with every AP of the network. On the other hand, as clearly recognized by authors, the solution is restricted to both intra-technology and intra-domain handoffs.

Assuming an access network integrated by N access points, authors in [62] use an NxN matrix to guide the key pre-installation process. A specific position $[i, j]$ in the

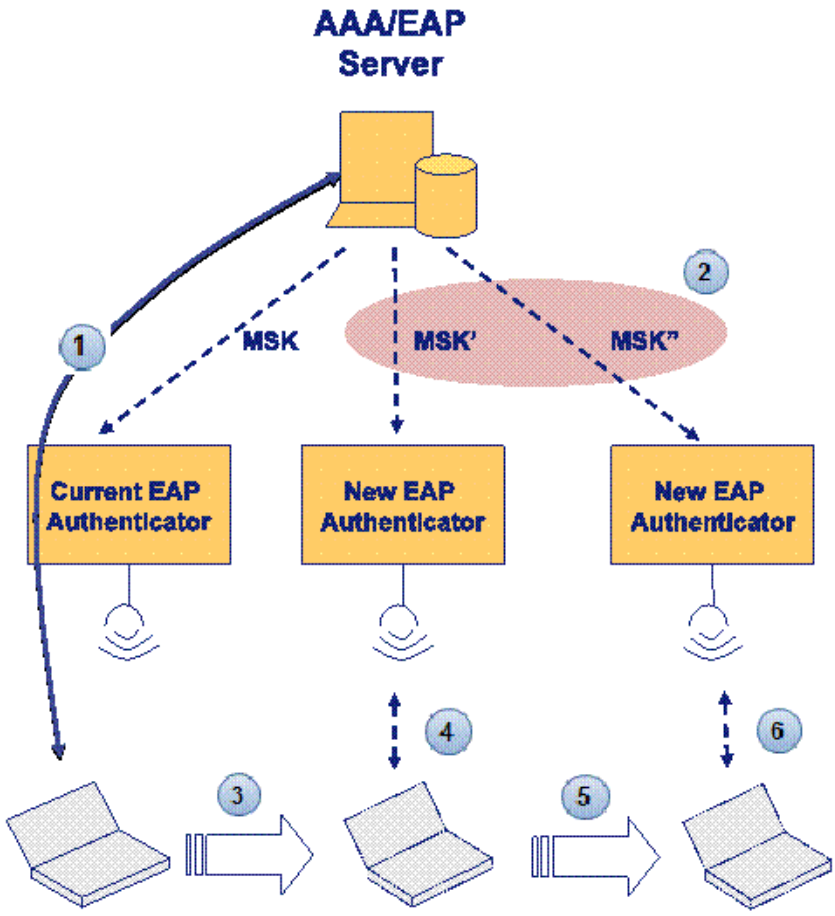


Figure 2.24: Key Pre-distribution Mechanism

matrix stores the probability that the peer performs a handoff from access point AP_i to AP_j. Thanks to this information, the selection algorithm calculates those regions where, with high probability, the peer will move in a near future. Similarly to the solution proposed in [70], two important disadvantages are found in this proposal. Firstly, the matrix demands an excessive memory consumption in the order $O(n)$. Secondly, this solution is only applicable for intra-technology and intra-domain handoffs.

Within standardization organizations, we can find some research efforts to reduce the latency introduced by network access process by adopting a key pre-installation strategy. For example, in the IETF, the *Handover Keying Working Group* (HOKEY WG) has recently proposed the so-called *Authenticated Anticipatory Keying Model* (AAK) [63]. AAK is a mechanism that uses EAP to establish authenticated key material in one or several candidate authenticators. This operation happens *before* the arrival of the peer at the access network managed by that authenticator. Unlike the aforementioned solutions, AAK is intended to work in inter-subnet, inter-domain or inter-technology handoffs. The AAK mechanism proposes an optimized access control model based on the use of the *EAP Re-authentication Protocol* [68] (ERP). ERP, which will be explained in detail in

section 2.4.5, specifies extensions to EAP and the EAP keying hierarchy to support a method-independent re-authentication.

The AAK-based key pre-distribution works as follows. Initially, the peer it is assumed to have previously completed a full EAP authentication. The authenticator starts the ERP/AAK execution and sends the peer a list neighbouring authenticators to which the peer could be interested to roam in the near future. Once the peer decides the list of candidate authenticator to which pre-distribute a key, it sends and *early-authentication request* containing such list. On the reception, once the AAA server verifies the correctness of the request, it derives a specific key *pMSK* for each candidate authenticator. Each key is distributed to the corresponding authenticator by using the AAA infrastructure. Once the pre-distribution process is completed, the AAA server responds with an *early-authentication finish*, informing the peer the set of candidate authenticators with which the pre-distribution process was successfully completed. Note that AAK proposes to carry out an EAP re-authentication process with the current authenticator to establish key material in other candidate authenticators. However, as outlined in [96], this model completely changes the key management model described in the EAP Keying Management Framework [38].

The IEEE 802.21 standardization group is another organization where the key pre-installation has attracted the attention of its members to enable a seamless handoff in NGN. IEEE 802.21 standard defines media-access independent mechanisms to facilitate and enable optimizations to improve handoffs between heterogeneous wireless networks. In particular, the IEEE 802.21a standardization task group is currently studying mechanisms to reduce the latency during network access control after an IEEE 802.21-assisted handoff. The architecture specified in [123] defines an 802.21 access control model that distinguishes between *Media Independent Authenticator* (MIA) and *Media Specific Authenticator* (MSA). MIA is an entity that controls a set of MSAs and facilitates the mobile user to perform a proactive authentication before moving to a target MSA. In particular, the MIA offers key distribution services to optimize the network access process to the different MSA under its control. According to this approach, the mobile node should be authenticated and authorized to use the MIA services.

In Ref. [64], authors describe the integration of three well-known key distribution mechanisms taking into account the particularities of the MIA/MSA architecture. These mechanisms have been considered in a future amendment developed by the 802.21a TG, in order to address the different security risks [124] that appear in 802.21-based handoffs. In particular, authors propose a key pre-installation approach called *Push Key Distribution*. In this key distribution mechanism, the MIA pushes a key into a candidate MSA based on a mobile node's request or an autonomous decision taken by the MIA itself. The solution defines a key hierarchy to generate those keys that are pre-installed in candidate MSAs. This key hierarchy stems from a key generated after the successful authentication of the mobile user against the MIA. Once a key has been installed on a specific MSA, the mobile user can perform the handoff to such MSA and establish the link-layer association and the security association to protect link-layer traffic. By using a key hierarchy derived from the initial authentication with the MIA, this solution enables to access different MSAs

without performing an EAP authentication each time the mobile user handoffs to a new MSA controlled by the same MIA.

Fast re-authentication solutions based on key pre-distribution have two main disadvantages. On the one hand, they require a precise selection of those authenticators to which pre-distribute key material. If the user pre-distributes key material to authenticators where the user finally does not move, the technique may incur in an unnecessary use of resources. Nevertheless, this is a complex problem given the difficulty of predicting future movements of the user. On the other hand, key pre-installation solutions have a significant deployment cost since a modification in existing lower-layer technologies is required in order to allow pushing a key provided by an external entity instead of being produced as a consequence of a successful EAP authentication executed through the EAP authenticator.

2.4.4 Use of a Local Server

According to the EAP authentication model [33], each time a user needs to be authenticated, a full EAP authentication must be performed with the AAA/EAP server located in the user's home domain. This is a serious limitation for roaming scenarios, specially in mobility contexts. The reason is that each time the visited network needs to re-authenticate the client, the home domain must be contacted. This introduces a considerable latency during network access process since the home EAP server could be located far from the current user's location. Furthermore, taking into account that typical EAP methods (e.g., EAP-TLS [54]) require multiple round trips, the home domain needs to be contacted several times in order to complete the EAP conversation, resulting in unacceptable handoff times.

To solve this issue, some solutions have proposed the use of a local server near the area of movement of the peer to speed up the re-authentication. The basic idea is to allow the visited domain to play a more active role in network access control by allowing the home AAA server to delegate the re-authentication task to the local AAA server placed in the visited domain. As depicted in Fig. 2.25, the user firstly performs a full EAP authentication (*1*) with the home AAA/EAP server using the *long-term* credentials that the home domain provides to their subscribers. This initial EAP authentication, commonly named *bootstrapping phase*, is performed the first time the user connects to the network. Next, once the EAP authentication is successfully completed, the home AAA/EAP server sends (*2*) some key material (KM) to the visited AAA/EAP server. This key material, which is used as *mid-term* credential between the mobile and the visited AAA/EAP server, allows to locally perform re-authentication (*3,4*) when the peer moves to other authenticators located in the visited domain, thus avoiding AAA signalling with the home AAA/EAP server.

The idea of using a local server has been widely employed in 3GPP networks. In particular, an entity located in the visited domain called *Visitor Location Register* (VLR) is able to perform authentication tasks without contacting the home domain. To enable this delegation, 3GPP [67] proposes that the VLR receives a set of *authentication vectors* from the home domain. Each authentication vector, which is employed by the VLR

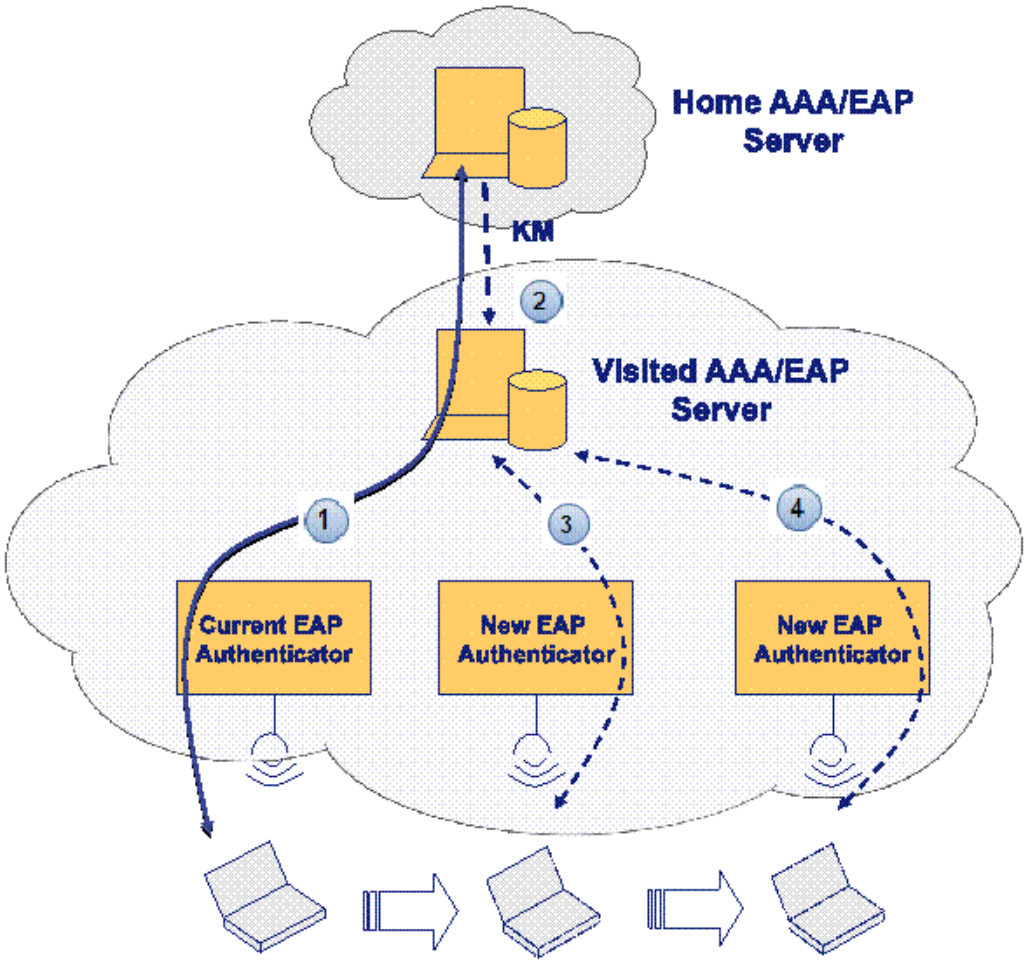


Figure 2.25: Use of a Local Server Mechanism

to authenticate a mobile user, has the form $< challenge, expected\_response >$. The VLR starts the authentication process by sending a *challenge* to the user and waits for a response. If the received response is equal to the expected, the user is successfully authenticated. However, whereas this authentication model has been successfully adopted within cellular networks, it cannot be applied in EAP networks straightforward since they consider neither the EAP authentication model nor the EAP Keying Framework [38]. In particular, EAP specifically distributes a key (MSK) instead authenticator vectors. Furthermore, EAP does not assume the presence of any entity in the visited domain in charge of the re-authentication and key distribution tasks.

Being aware of this problem, authors in [66] propose an enhanced framework which adds an extra level in the key hierarchy defined in the EAP keying framework. This new level allows to establish, based on a shared key, a trust relationship between the visited domain and the mobile user. This relationship is used for faster re-authentication of roaming user in the same visited domain. Initially, the mobile user (EAP peer) starts an EAP authentication with the home AAA/EAP server. During the EAP method execution, an

AAA-Key is generated and distributed from the home to the local AAA/EAP server. The local server uses this key to generate the *NAS-Key* which is finally sent to the authenticator. On the other side, the mobile user computes its own version of AAA-Key during authentication process. Then the AAA-Key is used to generate the NAS-Key. By using this key, peer and authenticator execute a security association protocol to protect data link traffic. According to this proposal, at the end of the EAP authentication, the local AAA/EAP server shares a secret key with the mobile. This key is used to re-authenticate it (e.g., re-run an EAP method based on shared secret key) when moves to another authenticator controlled by the specific local server.

In addition to the key pre-distribution mechanism (see previous section 2.4.3), the local server optimization has been also considered as possible solution for IEEE 802.21 networks. More precisely, authors in [64] propose an access control model called *Reactive Pull Key Distribution* where the MIA is used as local re-authentication server for accessing the different MSAs under its control. Initially, the peer starts a *media-independent authentication* to access the MIA services. To carry out this authentication, the participation of the home EAP/AAA server can be required. Once the authentication is successfully completed, both the peer and the MIA share a *Media-Independent Pairwise Key* (MI-PMK). From the MI-PMK, a MS-PMK is derived for each different technology (e.g., 802.11). Next, when the peer moves to a certain MSA (e.g., an access point), the MS-PMK is used as a *mid-term* credential to perform an EAP re-authentication where the MIA and MSA act as EAP/AAA server and EAP authenticator, respectively.

In conclusion, despite this kind of fast re-authentication solutions do not require to contact the home domain to re-authenticate the user, they do not define any optimization for the re-authentication process with the local server. For example, authors in [66] propose the use of an EAP method based on shared secret key like EAP-GPSK [125] which requires two message exchanges with the authentication server. Another serious disadvantage is found in the process followed to distribute the key that establishes a trust relationship between the peer and the local server. The solutions previously analyzed [64, 66] use a two-party model to carry out a key distribution process which involves three entities: peer, local re-authentication server and home AAA/EAP server. Since the use of a two-party model is known to be inappropriate [71] from a security standpoint, a three-party approach is recommended. In this sense, Kerberos thanks to the cross-realm operation is able to enable a local KDC by using a three-party key distribution model.

2.4.5 Modifications to EAP

A final group of solutions propose the modification of the EAP protocol itself. The most relevant contributions following this approach are the *EAP Extensions for EAP Re-authentication Protocol* (ERP) and the *3-Party for Fast Handoff Protocol* (3PFH). Both protocols are a method-independent solution that modifies the EAP protocol to achieve a lightweight authentication process. Additionally, they agree that it is necessary the existence of a local re-authentication server to optimize the process and show concern over the definition of a secure key distribution process to the local server and authenticator.

EAP Extensions for EAP Re-Authentication Protocol (ERP)

Within the *HandOver KEYing Working Group* (HOKEY WG) [126], the IETF has proposed a solution to reduce the latency of EAP authentications. The solution proposes the extension of the EAP protocol itself and the definition of a key hierarchy specially designed for handover keying purposes. Additionally, the HOKEY WG has defined an entity, named *HOKEY server*, which will be in charge of both fast EAP re-authentication and key distribution tasks. These tasks can be carried out by either a server in the peer's home domain, normally co-located in the AAA/EAP server (*home HOKEY server*); or can be delegated to a local server in other domain (*local HOKEY server*). To permit this delegation, a mechanism has been designed to transfer cryptographic material from the home HOKEY server to the local one following a two-party key distribution model. Next, we introduce the different elements that constitute the solution.

▷ Key Hierarchy and Derivation

For supporting a key distribution process during the handoff, a key hierarchy has been designed. This key hierarchy uses the EMSK, exported during a full EAP method authentication, as root key. In order to derive the rest of the hierarchy, a general key derivation framework is defined in [110]. This framework is based on a *Key Derivation Function* (KDF) which derives further keys from the EMSK. In particular, as defined in [127], three *root keys* (RK) are derived from the EMSK:

- USRK (*Usage Specific Root Key*). This key is used for an specific usage, as for example, for key distribution purposes during the handoff.
- DSRK (*Domain Specific Root Key*). This key is generated and sent to another HOKEY server placed in other domain. It shall be used as root key to derive further keys (DSURK).
- DSUSRK (*Domain Specific and Usage Specific Root Key*). This key is similar to USRK, except that its scope is restricted to the same domain as the parent DSRK. In other words, the DSUSRK can be used only within that particular domain, however that restriction is not applied to USRK.

Apart from these keys, each time a peer moves to a new authenticator a new *re-authentication Master Session Key* (rMSK) is derived and sent to the authenticator. The details of this key explained in the following section.

▷ EAP Extensions for EAP Re-authentication Protocol

The *EAP Extensions for EAP Re-authentication Protocol* (ERP) [68] has been designed for fast EAP authentication. It describes a set of extensions to EAP, which enables efficient re-authentication for a peer that has previously established EAP key material with the EAP server through the so-called *bootstrapping phase*. These extensions include three new

messages: *EAP-Initiate/Re-auth-Start*, *EAP-Initiate/Re-auth* and *EAP-Finish/Re-auth*. The support of these new messages assumes the upgrade of the current EAP state machine specification [128] in peers, authenticators and servers.

The ERP negotiation involves the peer, the authenticator and the HOKEY server, named ER (*Efficient Re-authentication*) server. Beforehand, it is assumed that the peer performs a full EAP authentication with the EAP server and both entities share a EMSK. From the EMSK, the peer and the EAP server, acting as home ER server, derives either a USRK, named *rRK*, or a DRUSRK, named *DS-rRK*, following the scheme explained in [110]. For simplicity, both keys are named rRK. From the rRK, a new key named *Re-authentication Integrity Key* (rIK) is derived to provide proof of possession and authentication during the re-authentication process (see Fig. 2.26).

The rIK is used to protect the ERP exchange between peer (which sends *EAP-Initiate/Re-auth*) and the ER server (which replays with a *EAP Finish/Re-auth*). Together with this last message, the server derives a rMSK (from the rRK) and sends it to the authenticator to establish a security association with the peer. The process is initiated by the authenticator by sending *EAP-Initiate/Re-auth-Start* to the peer. Furthermore, a sequence number (SEQ) is included to provide replay protection to the exchange.

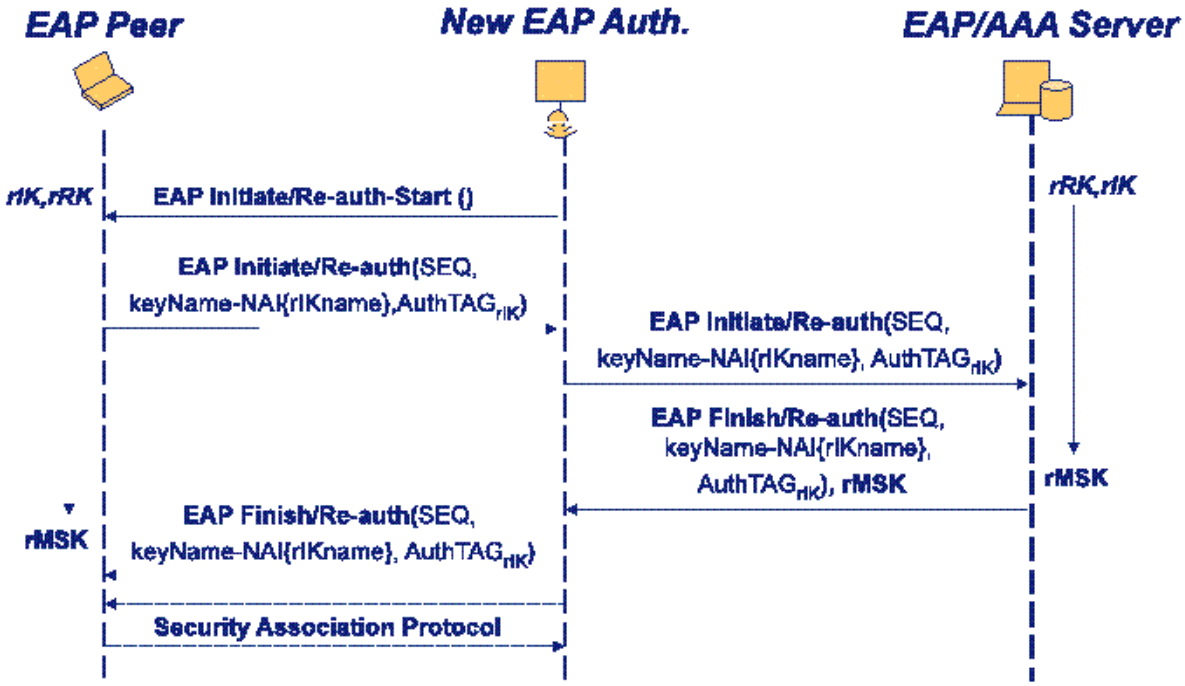


Figure 2.26: ERP Protocol

▷ Bootstrapping in ERP

ERP provides certain *bootstrapping* capabilities which address the need of obtaining certain information to allow the peer to use ERP. For example, the peer may need to know

the specific ER server which will be in charge of handoff keying operations or its identity for re-authentication purposes in the local domain. Furthermore, it may need to start the distribution of a DSRK to a local ER server.

ERP identifies two bootstrapping mechanisms: implicit and explicit. In the *implicit* bootstrapping, the required information is delivered to the EAP peer by means of some out-of-band mechanism. The details of this type of bootstrapping are out of scope of ERP. On the other hand, the explicit bootstrapping it is supported by ERP itself. Basically, just after full EAP method execution, peer immediately starts ERP by sending *EAP-Initiate/Re-auth* with a flag B(oostrapping) activated. When the server receives *EAP-Initiate/Re-auth* with B flag activated, sends a *EAP Finish/Re-auth* with required information for the peer.

▷ Key Distribution of EAP-based Keys for Re-authentication

Originally, EAP-based network access authentication and authorization have been based on a two-party trust model, where the peer and the server shared long-term credentials for mutual authentication. This model is suitable for network access authentication since it involves two parties: the peer and the server. However, with the inclusion of key management capabilities in EAP [38] and the use of *pass-through* model, it is required to distribute a key (e.g. MSK) from a centralized AAA server to the authenticator, since it cannot generate the same key material generated during the authentication. Thus, under key distribution standpoint, there are *three* parties involved and not two.

Even so, EAP has traditionally followed a two-party model for key distribution instead three-party. The reason is that, although three-party model offers better security features, the two-party model allows a simple key distribution even when there are intermediate entities (e.g. AAA proxies) between the authenticator and the server. Although, these entities can observe the distributed key material, it is assumed that they form a chain of trust where the entities are trusted to not use the key material they have access. Based on these arguments, two-party model is still advocated, despite the context of the distributed key is not correctly defined [71].

The security problems with using a two-party model for key distribution have been generally referred to as a problem with *channel binding* [33]. Basically, the peer infers the identity of the authenticator but has no direct indication of the identity of the entity to whom the authentication server transmitted the key material. In this way, the peer may believe that distributed key (e.g. MSK) is shared with one entity, whereas it is actually shared with another.

Although this problem was not finally handled within EAP WG, the HOKEY WG has tried to fix it, at least, in the handoff keying field. Basically, handoff keying involves the distribution of key material to three parties [71]: the peer that wants to access the network, the ER server in charge of key distribution, and the authenticator to which the handoff will be performed. Additionally, when the ER server does not have a key (e.g. a local ER server in a roaming scenario), it is also possible to use a three-party model

to distribute a DSRK, since there also are three parties involved: the peer, the local ER server and the home ER server. Nevertheless, despite the security problems associated with a two-party model, ERP still uses a two-party model to distribute the rMSK. To distribute a DSRK from home a ER server to the local ER server, a three-party model was initially considered [129] with the definition of the *Key Distribution Exchange* (KDE) protocol. However, a two-party model has finally been adopted [127], by relaying in the underlying AAA protocol security to achieve a secure distribution.

3-Party for Fast Handoff Protocol (3PFH)

To achieve an efficient solution to the EAP authentication problem in mobile environments it has been recognized [65] that a fast a secure key distribution process is required. Such key distribution process will provide keys (derived from the key material exported by a previous full EAP execution) to both the mobile user and the authenticator from a trusted server without running lengthy full EAP authentications. Additionally, it has been concluded that a three-party model is the most appropriate model to achieve a secure key distribution process [71].

Following these recommendations, the *3-Party for Fast Handoff Protocol* [16] (3PFH) is a secure three-party authentication protocol for network access specially adapted for EAP-based networks. More precisely, 3PFH is a server-based key establishment protocol that can significantly reduce the EAP authentication time. Unlike ERP, 3PFH has been proven to provide strong security properties and it is applicable even during handoffs that involve several administrative domains (inter-domain handoff). The protocol also requires an initial *bootstrapping phase* which is executed before distributing any key material to the peer and authenticator during the *handoff phase*.

- *Bootstrapping phase*. The peer runs a full EAP authentication with its home EAP server through a specific authenticator. This step allows both the peer and the home EAP server to share a fresh EMSK. From the EMSK, a key hierarchy is derived to support the key distribution process carried out by 3PFH during the subsequent handoff phase. This step is performed only once (e.g., the first time the peer gets network access) as long as the EMSK lifetime is valid.

- *Handoff phase*. The peer, the authenticator and a server named *Key Distribution Server* (KDS), which is in charge of performing the key distribution within a domain, are involved when running 3PFH. The protocol run will establish a shared key between the authenticator and the peer. The KDS can be placed on either the peer's home domain (*home KDS*) or, for optimization purposes, on a local server (*local KDS*) near the peer's area of movement. The assumption is that the home KDS is the home EAP server co-located within home AAA server, whereas the local KDS is located in a local AAA server placed near the peer's area of movement.

The following are notations that help to describe 3PFH in detail.

- A: a party which refers to the EAP peer.
- B: a party which refers to the EAP authenticator.
- L: a party which refers to a local key distribution server.
- H: a party which refers to the home key distribution server.
- S: a party which refers to either a local (L) or home (H) key distribution server.
- $\{X\}_K$: X encrypted with key K providing confidentiality and integrity.
- K^i_{XY}: A symmetric key shared between parties X and Y used for the purpose i.
- N_X: Pseudo-random number acting as nonce and provided by the party X.
- T_X: Timestamp generated by the party X.
- SEQ_{XY}: Sequence number maintained by parties X and Y.
- $[x]$: x is optional.

▷ **Intra-KDS Handoff**

During an intra-KDS handoff, it is assumed that the peer (A) shares a fresh key K_{AS} with the KDS (S). This key is derived according to key hierarchy defined in 3PFH [16]. This KDS can be either a home server in the home domain (H) or the local server in the visited domain (L) when the peer is roaming within the visited domain. From K_{AS}, the two keys K^{auth}_{AS} and K^{deriv}_{AS} are derived. It is also assumed that a pre-established key K^{auth}_{BS} is shared between the authenticator (B) and the KDS (S). Based on these keys, three parties (the peer, the authenticator and the KDS) are involved when running the basic 3PFH as shown in Fig. 2.27. The basic 3PFH execution will distribute a session key K_{AB} between the peer and the authenticator.

▷ **Inter-KDS Handoff**

3PFH defines an extended version of the protocol which is intended to support handoffs where the peer roams between authenticators that are under the control of different KDSs. When this kind of handoff happens, it is desirable to distribute a shared key to the new local KDS (L) in such a manner that, inter-authenticator handoffs under this KDS do not require to contact the home KDS. In order to provide a fast and efficient key distribution in this case, the extended 3PFH version allows performing the key distribution of K_{AL} (to be shared between A and L and distributed by H) and K_{AB} (to be shared between A and B and distributed by L) in a single exchange during an inter-KDS handoff. Compared to intra-KDS handoff, inter-KDS handoff involves four entities (see Fig. 2.28): the peer (A), the new authenticator (B), the new KDS (L) which controls the new authenticator and it

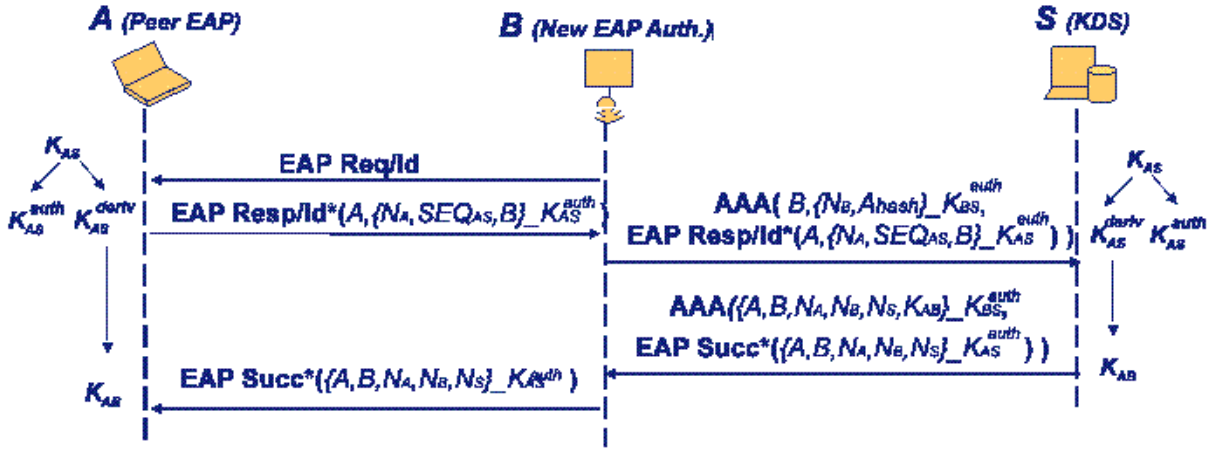


Figure 2.27: Basic 3PFH for Intra-KDS Handoff

is located in the visited domain, and the home KDS (H).

Thanks to the use of a three-party model, 3PFH achieves a secure fast re-authentication process. Nevertheless, this proposal still presents some drawbacks that prevent us from considering it as a proper solution to achieve a fast network access in NGNs. First, the modification of EAP is required in order to transport the 3PFH messages between the involved entities. More precisely, as observed in Fig. 2.27 and Fig. 2.28, authors assume that the *EAP Response/Identity* and *EAP Success* messages can transport authentication data. Second, despite the protocol execution is completed in a single exchange, 3PFH needs to contact a server (KDS) located in the AAA infrastructure to carry out the key distribution process. If this communication was not necessary, the latency introduced by the re-authentication process could be considerable reduced. Third, 3PFH does not handle the inter-domain scenario in a flexible manner since, the extended version of the protocol, requires the existence of a direct trust relationship between the visited and home domains. In this sense, it would be beneficial to allow the existence of an indirect trust relationship between the home and visited domain through intermediary domains as happens with Kerberos. Finally, 3PFH defines a new key distribution protocol which has not been standardized. Operators prefer the adoption of standardized solutions whose security properties have been widely validated.

2.5 Existing Privacy-Enhanced Authentication Solutions

The issue of user privacy protection has been the subject of study by researchers, especially in mobile environments [81]. On the one hand, in the literature we can find a wide set of authentication protocols for wireless communication networks where user anonymity

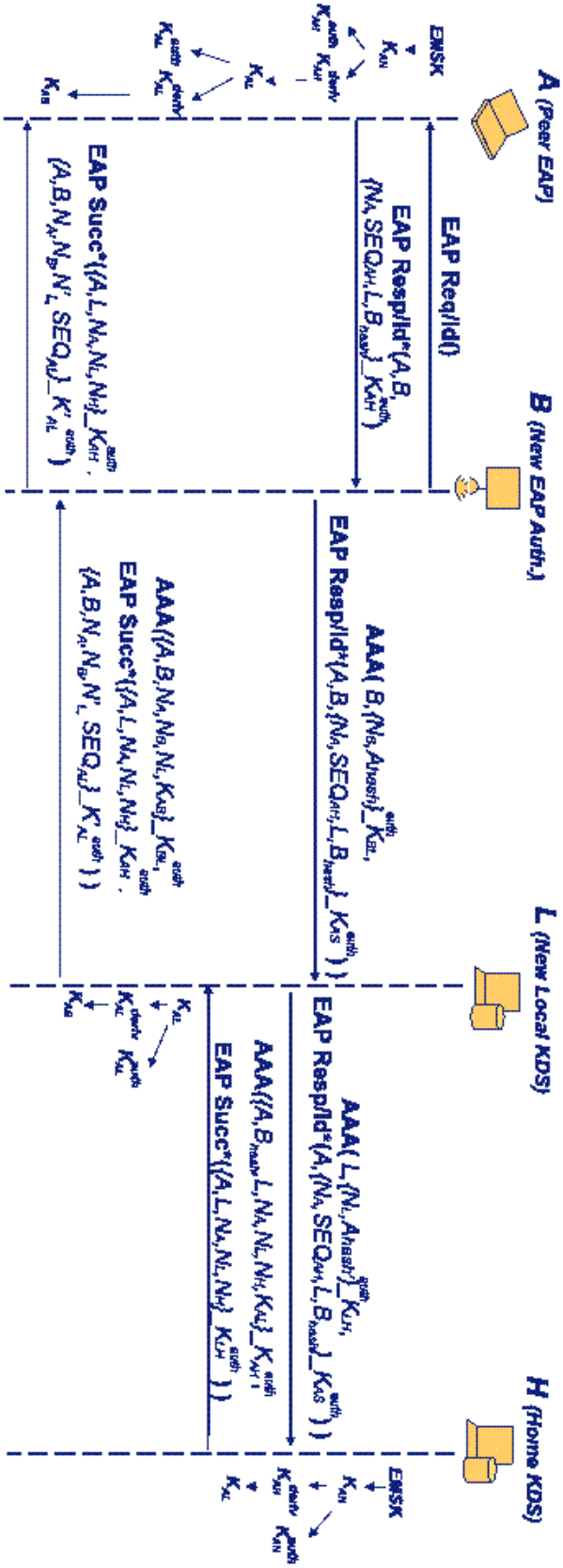


Figure 2.28: Extended 3PFH for Inter-KDS Handoff

is integrated in the protocol design. Typically, user's anonymity has been achieved by using a fictitious identity (called *pseudonym*) instead of using the user's real identity in the authentication protocol. Following this approximation, references [88] and [89] define two different authentication schemes where user anonymity is preserved by using pseudonym that does not change. Despite the user's real identity is not revealed, eavesdroppers can profile the whole activity of a single (anonymous) user by tracing the fixed pseudonym employed by the user. This problem is solved by [90] and [91] that incorporate a pseudonym renewal feature to allow the mobile user to remain untraceable. Nevertheless, both authentication protocols do not consider any lightweight procedure to avoid contacting the mobile user's home domain in each execution. This drawback is fixed in [92], which defines a re-authentication process with the visited domain while maintaining user anonymity and untraceability. However, none of these solutions can be applied in EAP networks straightforward since they consider neither the EAP authentication authentication analyzed in section 2.2 nor the problem of fast re-authentication associated with EAP-based wireless networks. In fact, these aspect have influenced the design of our proposal.

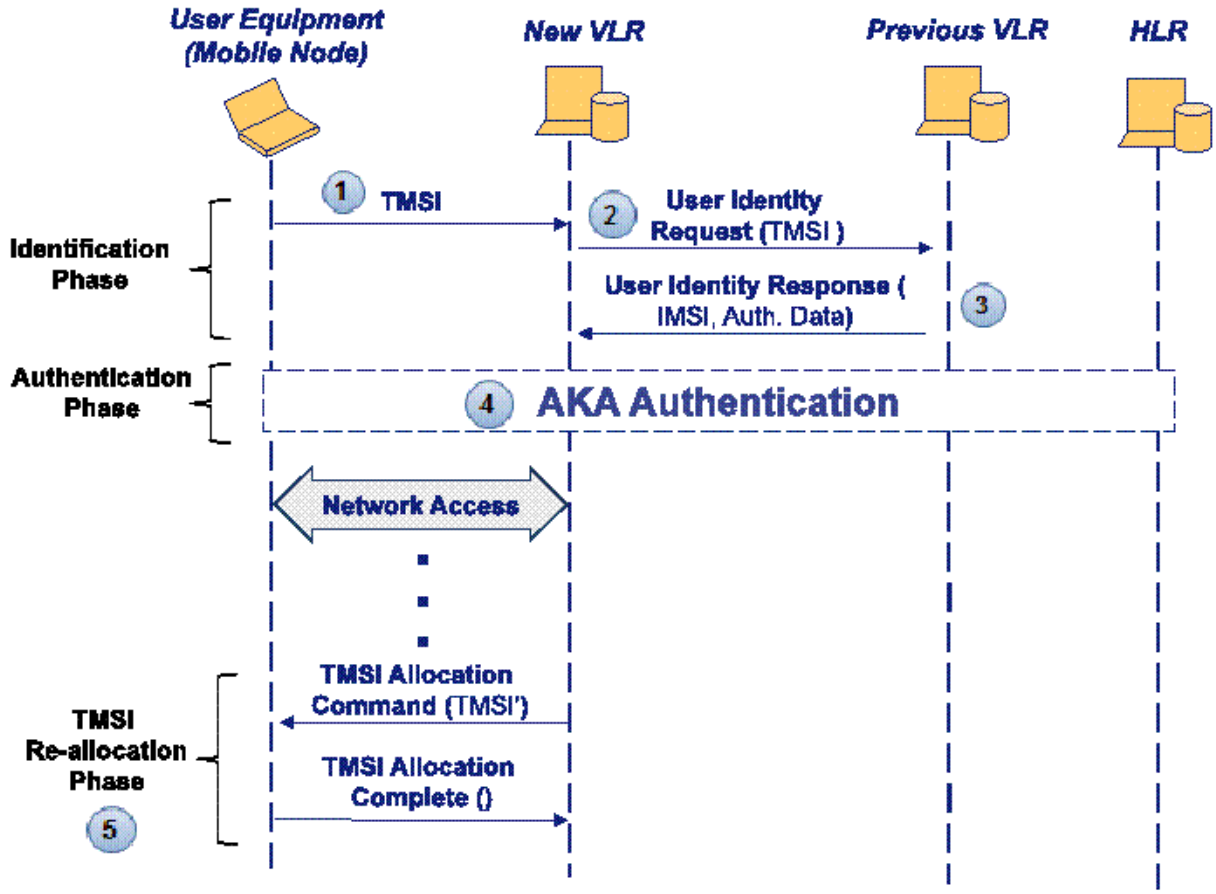


Figure 2.29: User Privacy Protection in UMTS AKA

On the other hand, the *Universal Mobile Telecommunication System* (UMTS) architecture has important security features like user identity confidentiality, achieved through the adoption of the *Authentication and Key Agreement Protocol* (AKA) specified by the *Third-Generation Partnership Project* (3GPP) [130]. In UMTS networks, a mobile user is recognized by either a permanent identity called *International Mobile Subscriber Identity* (IMSI) or by a temporary one known as *Temporary Mobile Subscriber Identity* (TMSI) which allows to provide user anonymity. The authentication process flow is depicted in Fig. 2.29. Initially, when the mobile user attaches to a network controlled by a new *Visitor Location Register* (VLR), it sends the TMSI to the new VLR *(1)*. The TMSI is formatted in such manner that contains location information of the previously VLR visited by the mobile user. Using this information, the new VLR solicits the previous VLR *(2)* information regarding the permanent user identity. The previous VLR not only responds *(3)* with the IMSI but also may provide some temporary authentication data. Since the subsequent AKA authentication *(4)* is performed by using the TMSI, the real user's identity (IMSI) is not revealed to unauthorized parties. Finally, in the so-called *TMSI Re-allocation Phase (5)*, the TMSI can be renewed by a new one in order to assure anonymous user untraceability. Nevertheless, AKA violates some properties of user identity confidentiality. More precisely, there are two particular cases where a user is required to send its real identity. One is that the user performs the first authentication since the user does not have a valid TMSI. The other is when the new VLR cannot contact the previous one. As explained in [131], this deficiency allows an attacker to correlate real and temporary identities by using traffic analysis tools. To solve these problems, there have been some efforts like in [131] and [132]. Nevertheless, it is important to note that these improvements are specific to 3G networks, leaving undefined a privacy optimization for other technologies.

In the EAP context, there are EAP methods that define some mechanism to provide anonymity during the authentication process. For example, the last revision of EAP-TLS [54] has a special privacy extension destined to hide the peer's identity from observers. This objective is achieved by *(1)* using an anonymous identity (*anon@anon*) in the *EAP-Response/Identity* messages and *(2)* by allowing the entire peer's certificate to be sent within a TLS session providing confidentiality. Nevertheless, a privacy-enhanced EAP-TLS authentication results in a heavy process that requires up to twelve messages between peer and server to complete. This problem is partially solved by methods such as EAP-SIM [48] and EAP-AKA [49] which define an authentication process with both a reduced number of messages and identity privacy support. More precisely, these methods specify an optional identity privacy support by using *pseudonym usernames* and *fast re-authentication usernames*. While the former are only used on full authentications, the latter are employed during the fast re-authentication operations. Both types of temporary identities are generated by the EAP server and securely delivered to the peer. Furthermore, they are only valid for one use and cannot be re-used.

A privacy-enhanced authentication based on EAP-SIM or EAP-AKA works as follows. Initially, when the peer firstly attaches to the network, it engages in a full EAP-SIM/EAP-AKA authentication using the pseudonym username as identification.

During the authentication, the server generates and delivers to the peer both a new pseudonym identity (that will be used in the next full EAP-SIM/EAP-AKA authentication) and an initial fast re-authentication username. After that, when the peer roams to a new authenticator, the peer will perform an EAP-SIM/EAP-AKA fast re-authentication by using the fast re-authentication identity previously acquired. Again, during the fast re-authentication process, the server will sent to the peer a new fast re-authentication identity to be used in the next fast re-authentication operation and so on. Despite both methods define a fast re-authentication procedure to minimize the signalling, they suffer from their inefficiency to achieve an efficient re-authentication since they always require contacting the home network. Furthermore, they do not achieve an effective privacy protection since, in the first EAP-SIM/EAP-AKA authentication, the user is required to employ its real identity.

As we can observe, existing privacy proposals in the field of authentication and access control are oriented towards 3G networks or are not applicable to EAP-based access control or do not take into account the problem of EAP when used in mobile environments (e.g., always contacting the user's home domain). In this way, the access control solution developed within this PhD thesis represents an important contribution to this field by defining an EAP-based fast re-authentication mechanism that protects the privacy of the user.

2.6 Conclusions

This chapter provides a general overview about the state-of-art of technologies and protocols related to network access control. In particular, the interest has been focused on the process intended to authenticate users requesting access to the network service. On the one hand, we have extensively reviewed technologies and protocols related to network access control which is the research area where this PhD thesis is developed. On the other hand, existing solutions trying to provide a fast re-authentication process are analyzed in depth, in order to justify the necessity of defining a new mechanism able to reduce the authentication time during network access.

The description of technologies related to network access starts with the explanation of the AAA infrastructures, which provide an unified framework to handle the authentication, authorization and accounting processes. Once it is described the different entities and interaction models in a typical AAA scenario, a detailed explanation of the most relevant AAA protocols (RADIUS and Diameter) is provided.

To implement the authentication service in AAA-based scenarios, attention is directed to the EAP protocol designed within the IETF. Apart from being easily deployable within existing AAA infrastructures, EAP exhibits important features such as flexibility to select an authentication mechanism and independence from the underlying wireless technology. We have described the EAP components, entities and authentication phases since EAP is the basis on which the proposed access control system is developed so that the reader can know the internal mechanisms of this authentication protocol.

	SUPPORTED HANDOFF TYPES								Strong Security	Low Deployment Impact	Avoid Standard Modifications	Privacy Support
	Intra-Tech.	Inter-Tech.	Intra-Netw.	Inter-Netw.	Intra-Dom.	Inter-Dom.	Intra-Sec.	Inter-Sec.				
CONTEXT TRANSFER SOLUTIONS												
Politis et al [18]	↑	↑	↑	↗	↑	⟶	↑	↗	↘	↘	↗	↓
Kim et al [57]	↑	↓	↑	↗	↑	⟶	↑	↓	⟶	↘	↓	↓
Aura et al [56]	↑	↓	↑	⟶	↑	↓	↑	↓	↘	↘	→	↓
IAPP [59]	↑	↓	↑	↓	↑	↓	↑	↓	⟶	⟶	↓	↓
CxTP PANA [58]	↑	↑	↑	↑	↑	⟶	↑	↗	↘	↗	↗	↓
PRE-AUTHENTICATION (LINK-LAYER)												
802.11i [60]	↑	↓	↑	↓	↑	↓	↑	↓	⟶	↗	↓	↓
PRE-AUTHENTICATION (NETWORK-LAYER)												
PANA Pre-auth. [61]	↑	↑	↑	↑	↑	↗	↑	↑	↑	⟶	↗	↓
Marin et al [15]	↑	↑	↑	↑	↑	↗	↑	↑	↑	↘	⟶	↓
MPA [17]	↑	↑	↑	↑	↑	↗	↑	↑	↑	↘	↗	↓
KEY PRE-DISTRIBUTION												
Mishra et al [70]	↑	↓	↑	↓	↑	↓	↑	↓	⟶	↗	↓	↓
Pack et al [62]	↑	↓	↑	↓	↑	↓	↑	↓	⟶	↗	↓	↓
AAK [63]	↑	↑	↑	↑	↑	↑	↑	↑	↗	↘	↓	↓
802.21 [64]	↑	↑	↑	↑	↑	↑	↑	↑	↗	⟶	↓	↓
USE OF A LOCAL SERVER												
3GPP [67]	↑	↓	↑	↓	↑	↓	↑	↓	↗	↓	↓	↘
Marin et al [66]	↑	↑	↑	↑	↑	↓	↑	↑	⟶	↗	↗	↓
802.21 [64]	↑	↑	↑	↑	↑	↑	↑	↑	↗	⟶	↓	↓
MODIFICATIONS TO EAP												
ERP [68]	↑	↑	↑	↑	↑	↘	↑	↑	↗	↓	↓	↓
3PFH [16]	↑	↑	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓

** Arrows represent the capacity to accomplish a requirement: ↑ = high, ↗ = moderate, ⟶ = middle, ↘ = low, ↓ = null

Table 2.2: Detailed Comparison of the Most Relevant Fast Re-authentication Proposals

The review of technologies finalizes with the description of the Kerberos protocol which is the secure three-party key distribution protocol we use in this PhD thesis to define a fast re-authentication process. Kerberos is a standardized protocol widely used to control the access to resources and whose security has been well verified. In particular, in the context of this PhD thesis, Kerberos is employed to control the access to the network service. For this reason, it is provided a complete revision of the protocol operation that goes from the basic concepts to the Kerberos exchanges through the entities and message format.

In general, in the revision of AAA infrastructures, EAP and Kerberos, it is done a special emphasis on explaining the extensibility mechanisms available for each protocol. This is because the proposed access control system aims to not inflict modifications on existing standardized protocols by proposing enhancements strictly based on the extensibility mechanisms defined by each protocol.

As mentioned, the second part of the chapter is devoted to revise and analyze the different solutions that have tried to reduce the latency introduced by network access control during the handoff. The different proposals are grouped in five categories according to the strategy followed to reduce the authentication time: context transfer, pre-authentication, key pre-distribution, use of a local server and modifications to EAP. Table 2.2 provides a comparative overview of the analyzed solutions following the same criteria presented in the section 1.6. As observed, despite it can be found solutions which are able to support every type of handoff, the main problem appears with the level of security provided by the solution, deployment impact and compatibility with standardized technologies. Furthermore, a common deficiency to existing solutions is the inability to protect the privacy of the user during the authentication process.

After revising existing solutions, we reach the conclusion that there exists an important gap that needs to be covered regarding the definition of an efficient network access control system for EAP-based NGNs. By using a secure three-party key distribution process, the mechanism must be able to reduce the authentication latency in mobile environments while preserving the user privacy. In order to overcome deficiencies present in existing solutions, the easy deployment and compatibility with current standardized technologies are key features to be achieved. In the following chapters, we present our contribution to cover this gap.

Chapter 3

EAP-FRM: Architecture for Fast Re-authentication in EAP-based Wireless Networks

After presenting the main objectives of this PhD thesis, basic concepts and technologies related to network access, as well as existing solutions dealing with the fast re-authentication problem, we present in this chapter the first component of our contribution. More precisely, we develop a generic architecture aimed at reducing the time spent on providing network access in EAP-based mobile networks.

The chapter begins clarifying the requirements which motivate the definition of a new architecture for fast re-authentication. On the one hand, it is necessary a solution able to operate independent of the underlying link-layer technology. This requirement is highly important in NGN environments where users may access through heterogeneous wireless access networks. On the other hand, the fast re-authentication solution must respect existing standardized protocols in order to not inflict modifications that complicate the deployment of the solution. In order to accomplish the first requisite, EAP has been selected as authentication framework since it is a protocol independent of the underlying media access technology. The second requisite is satisfied by using the extensibility mechanism available at EAP, which is the definition of new EAP methods.

In particular, the architecture for fast re-authentication presented in this chapter is based on the design of a new EAP method called *EAP-FRM* which works on standalone mode. The description of the architecture not only covers the different phases that integrate a re-authentication process based on EAP-FRM, but also specifies the EAP-FRM method and the minimal extensions to AAA protocols in order to implement the architecture. Following the description, the chapter discusses some interesting aspects regarding the EAP-FRM method itself. For example, the process followed to derive the key material exported by the method, message protection or details about the security of the re-authentication process are examined. Next, the chapter tries to demonstrate the capabilities of the re-authentication architecture. With this objective, through several use cases, we describe how EAP-FRM operates in conjunction with different protocols for fast

re-authentication.

The chapter finalizes showing a proof-of-concept testbed of the architecture. Based on a implemented prototype of the EAP-FRM architecture, we analyze the performance in order to demonstrate that is able to achieve a fast re-authentication process while solving common deployment deficiencies present in existing re-authentication solutions.

3.1 Introduction

As surveyed in section 2.4, there have been numerous efforts addressing the problem of using EAP in mobile environments, where a reduced handoff latency is required. Despite these attempts try to solve the problem in various ways, researchers agree [65] that the problem of efficient re-authentication must be solved by using a fast and secure key distribution process. Considering that an EAP authentication can generate key material valid for a certain period of time, the execution of an initial full EAP authentication is proposed for generating valid keys which are used to enable efficient re-authentication. The re-authentication process is based on a fast and secure key distribution protocol through which it is verified that the peer was successfully authenticated and to distribute valid keys to both the peer and the authenticator. The optimization not only contemplates the use of a key distribution process but also the existence of a local re-authentication server placed near the mobile user. This server is typically placed in the visited domain and is in charge of re-authenticating users, thus avoiding the participation of the home authentication server in the re-authentication process.

The most relevant contributions following this strategy to optimize the EAP authentication during network access control are *EAP Extensions for EAP Re-authentication Protocol* [68] (ERP) and the *3-Party for Fast Handoff Protocol* [16] (3PFH). As described in section 2.4.5, both protocols define the existence of a local re-authentication server with which a user can complete an EAP-based re-authentication process in a single message exchange (*round-trip*). Additionally, both solutions take advantage of the key material exported by a successful EAP authentication to define an efficient re-authentication protocol. However, these solutions present important drawbacks. In particular, ERP uses a two-party model to perform the key distribution. As explained in [71], this model is appropriate for authentication purposes but introduces serious security vulnerabilities when used for key distribution where three parties are involved. This deficiency of ERP is solved by 3PFH which proposes a novel key distribution protocol for EAP based on a three-party model. Nevertheless, both solutions suffer from requiring modifications to the existing lower-layers standards or the EAP protocol itself. For example, in ERP a modification to the standard EAP state machine [128] and lower-layers is assumed. Similarly, 3PFH considers a modification to the *EAP Success* message to transport authentication data in the final exchange. This situation also provokes the modification of existing lower-layer technologies since, according to the standard EAP specification, the *EAP-Success* represents a successful finalization of the authentication but it is not allowed to carry any data.

Considering the deployment issues in these approaches due to the nature of the required modifications to standardized protocols, these assumptions may cause a harder deployment of these solutions. In order to alleviate the impact of these potential problems and keeping a reduced latency for network access authentication, we propose an architecture for fast EAP re-authentication based on the design of a new EAP method named *EAP Fast Re-Authentication Method* (*EAP-FRM*). This EAP method works on standalone EAP authenticators but the method itself can contact a backend authentication server if necessary. EAP-FRM can transport the payload of any key distribution protocol which, in the context of EAP-FRM, is referred to as *fast re-authentication protocol* (*FRP*).

As we will discuss later, given that the definition of new EAP methods is a standard mechanism to extend the EAP authentication framework, this new architecture provides the following benefits:

- No modification to EAP is required because EAP is designed to carry any EAP method.
- No modification to lower-layer specifications (such as IEEE 802.11 and IEEE 802.16e) is required because EAP has the media independence property.
- No modification to the *EAP Key Managament Framework* defined in [38] is required.
- Minimum extension to AAA protocols is needed. When an AAA protocol is used between the authenticator and a backend authentication server, new AAA attributes may be needed to carry the authentication information between an authenticator and the backend authentication server.
- Assuming a well designed FRP, only one roundtrip beyond access network is needed at the most.

As we can see, a novel feature not present in previous fast re-authentication solution is that EAP-FRM decouples the protocol used (FRP) to re-authenticate the user from the transport employed to convey the FRP messages. In the context of this PhD thesis, this FRP is assumed to be a well-designed secure three-party key distribution protocol. In particular, as we have described in the introductory chapter 1, we propose the use of standardized Kerberos protocol.

3.2 Description of the Proposed Architecture

The proposed architecture for fast re-authentication uses a new EAP method named EAP-FRM that operates in the standalone authenticator mode, that is, the EAP-FRM method will be implemented by the peer and the authenticator. EAP-FRM is destined to transport authentication information between the peer and the authenticator. More precisely, a *Protocol Data Unit* (PDU) of any fast re-authentication protocol can be the authentication information encapsulated in EAP-FRM.

As explained in section 2.2.3, in the standalone mode, the EAP conversation takes place between the EAP peer and the EAP authenticator which also implements the EAP server. Typically, an EAP authenticator working on standalone configuration for a specific EAP method, is expected to locally process the authentication data received from the peer. Nevertheless, if necessary, the EAP-FRM method implementation executed by the standalone authenticator can communicate with an backend authentication server (e.g., an AAA server) for verification of the authentication information originated by the peer and encapsulated in EAP-FRM messages. The protocol used to convey the FRP messages between the authenticator and the backend authentication server is referred to as *backend protocol*. Figure 3.1 shows a general overview of the proposed architecture where an AAA protocol such as RADIUS or Diameter is used as the backend protocol.

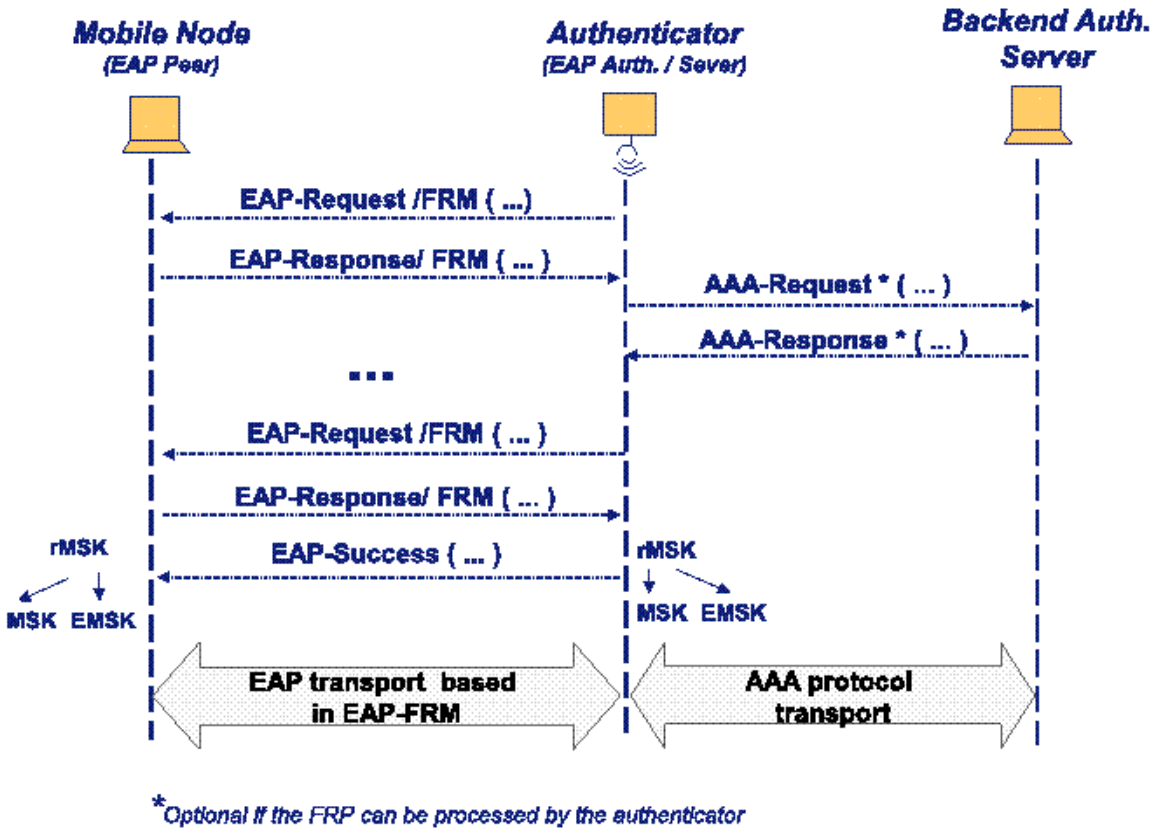


Figure 3.1: Proposed architecture

EAP-FRM assumes that the execution of the FRP will generate cryptographic material. In particular, the FRP is expected to export a key which, for generality, we denote as *re-authentication Master Session Key* (rMSK). With this cryptographic material, the EAP-FRM method will derive the MSK and the EMSK that will be exported to the lower-layers. The details of derivation of both keys are explained in 3.5.2.

Figure 3.2 shows the implementation framework for the EAP-FRM fast

re-authentication architecture. As observed, if the specific FRP requires it, the EAP-FRM method implementation in the EAP Auth/Server may interact with an AAA client implementation (e.g., through an API) in order to contact with the backend authentication server. After performing the FRP, cryptographic material (rMSK) is held by the EAP-FRM implementation for key derivation in the peer and the server. When the EAP-FRM server needs to contact a backend authentication server for verification of the authentication information, the latter is expected to distribute the rMSK to the EAP-FRM implementation in the EAP Auth/Server. Depending on the key distribution model followed by the FRP, the rMSK can be distributed by the FRP itself or by using the AAA implementation (as depicted in Fig. 3.2).

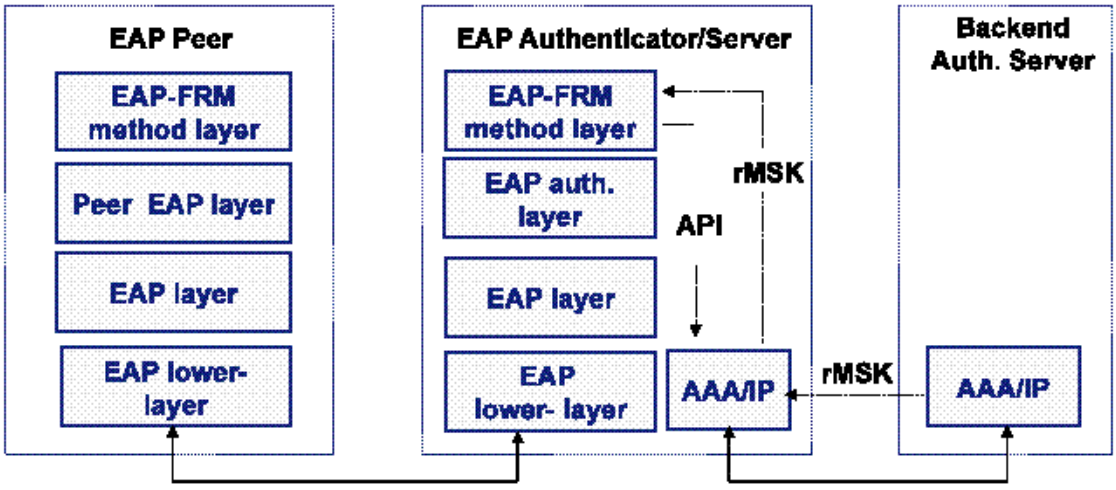


Figure 3.2: EAP-FRM Architecture General Overview

Typically, FRPs achieve a fast re-authentication operation based on the key material exported by an initial EAP authentication performed during the so-called *bootstrapping phase*. This EAP authentication can be based on any EAP method (*bootstrapping EAP method*) which exports key material and requires the participation of the home authentication server (where the user is registered). In general, the bootstrapping phase is performed when the peer initially connects to the network and once throughout the lifetime of the generated EAP key material. The designed FRP may use the EMSK exported by the bootstrapping EAP method as root of a key hierarchy to create some shared cryptographic key material following the guidelines provided in [110]. This cryptographic material will be used by the FRP during the *fast re-authentication phase* each time the peer authenticates with a new EAP-FRM capable authenticator.

3.2.1 Bootstrapping Phase

If the cryptographic material used by the FRP is derived from the EMSK, it is required the use of a bootstrapping EAP method (e.g. EAP-TLS) which exports the EMSK before using EAP-FRM. The bootstrapping EAP method requires the use of long-term

credentials of the peer or *Transient EAP Keys* (TEKs) derived from the long-term credentials [38]. Given that the bootstrapping EAP method is typically executed in a pass-through configuration mode, the EAP server for the bootstrapping EAP method resides in the backend authentication server.

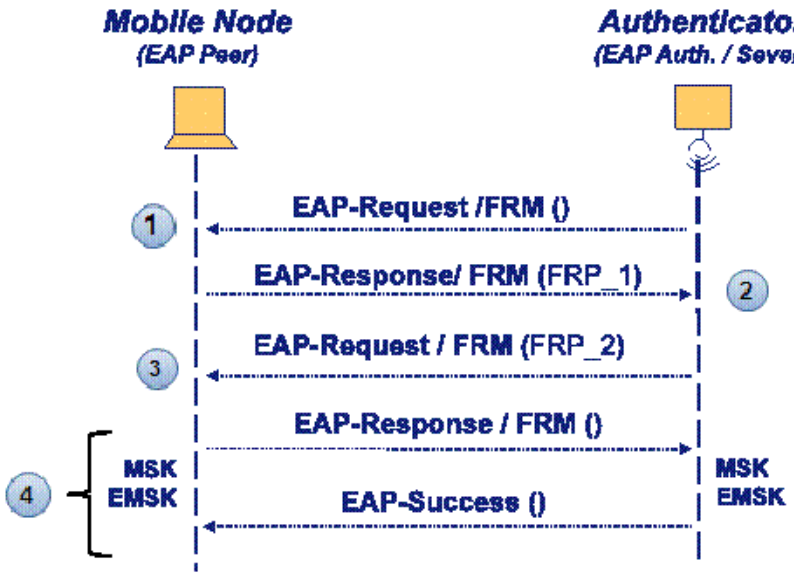
When the EAP authentication for the bootstrapping EAP method succeeds, a MSK and an EMSK are generated as the EAP key material. While the MSK is sent to the authenticator to establish a security association with the peer, the EMSK is held by the backend authentication server and the peer and used for generating the credentials needed by the FRP in the fast re-authentication phase. The use of EAP-FRM and related parameters may be negotiated between the peer and the authentication server in the bootstrapping phase.

It is worth noting that to carry out the bootstrapping phase, EAP-EXT [133] is a candidate method which has been specially conceived to assist this process. Without modifying EAP, this method not only allows to perform an authentication process based on existing EAP methods but also provides a secure exchange of bootstrapping information between the EAP peer and server. In the chapter 4, we will extend further the concept of bootstrapping.

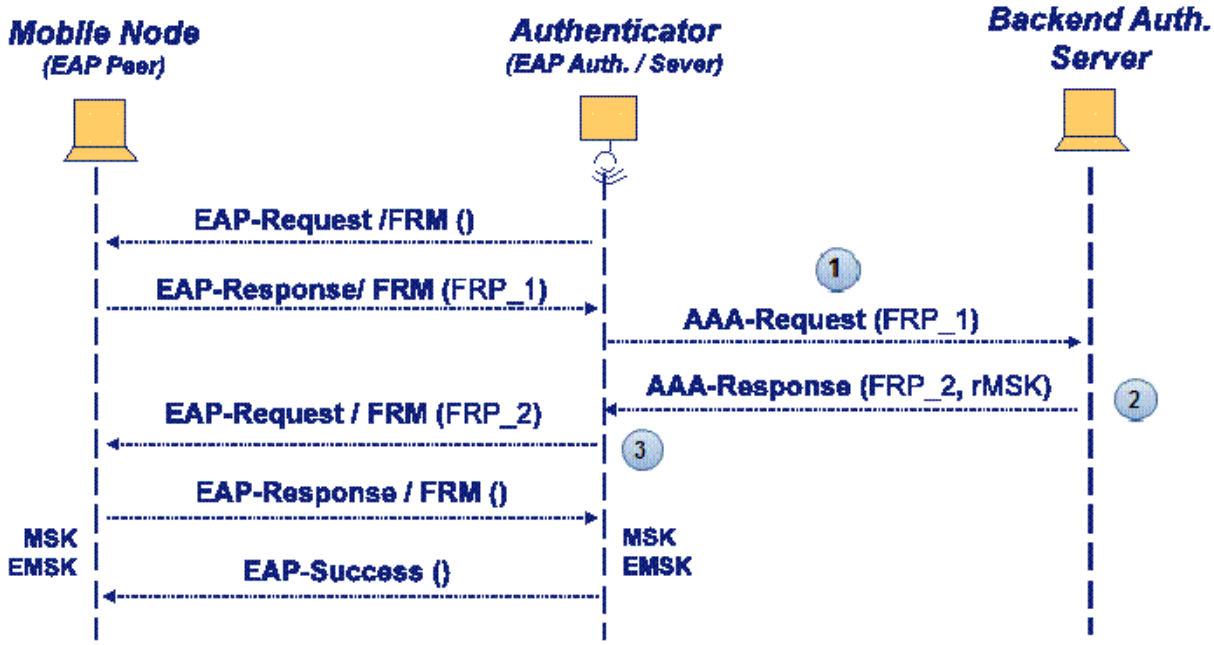
3.2.2 Fast Re-authentication Phase

As Fig. 3.1 depicts, the peer after the handoff engages an EAP-FRM exchange with the EAP authenticator. The EAP authenticator must be configured to start EAP-FRM, in such a way that the EAP authenticator sends automatically an *EAP-Request/FRM* message instead of *EAP Request/Id*. If the peer does not support EAP-FRM, it can send an *EAP-Response/Nak* message. This shall provoke that the authenticator sends back an *EAP-Request/Id* in order to start a traditional EAP authentication process. In this way, we allow to support authentication for non EAP-FRM peers. Alternatively, if the peer answers an *EAP-Response/FRM* with FRP data, the authenticator extracts this information from the *EAP-Response/FRM* and either processes it to provide access or forwards it (e.g., through an AAA protocol) for verification to the backend authentication server. This will depend on the specific FRP used. If a backend authenticator server has to verify the FRP and it is correctly verified, the backend authentication server answers to the authenticator with the response obtained as a consequence of processing the FRP, which will include a rMSK. That FRP response message is forwarded to the peer through an *EAP-Request/FRM*.

Assuming a FRP that involves, for example, two messages *FRP_1* and *FRP_2* to re-authenticate a user, Fig. 3.3 shows two general usage examples of EAP-FRM depending on whether a backend authentication server needs to be contacted by the authenticator or not. For the first case (Fig. 3.3(a)), the authenticator starts the fast re-authentication process by sending an *EAP-Request/FRM* to the peer (*1*), which answers with an *EAP-Response/FRM* containing the first message of the FRP (*FRP_1*). When the authenticator receives this message, extracts the authentication information and processes it locally (*2*). Once the information is successfully validated, the authenticator answers



(a) Communication when a Backend Server is not required



(b) Communication when a Backend Server is required

Figure 3.3: General Example of EAP-FRM Operation

the peer with the final FRP message (*FRP_2*) transported within an *EAP-Request/FRM* (*3*). At this point, the execution of the FRP has been successfully completed and an rMSK is locally exported by the FRP to the EAP-FRM method in both the peer and the authenticator. Nevertheless, to conclude the EAP conversation, a final *EAP-Response/FRM* and *EAP-Success* exchange is required (*4*) in order to maintain compatibility with current EAP specification and avoid any modification to existing EAP lower-layers.

If the FRP needs a backend authentication server to verify the authentication data sent by the peer, the fast re-authentication process is slightly different. As observed in Fig. 3.3(b), the authenticator forwards *FRP_1* to a backend authentication server by using an AAA protocol (*1*). After processing the received FRP message, the backend server generates *FRP_2* and computes the rMSK. Depending on the key distribution model followed by the FRP, different approaches can be used to distribute the rMSK to the authenticator. Assuming that the FRP uses a two-party key distribution model, the rMSK can be sent, for example, using the AAA protocol together with message *FRP_2* (*2*). On the reception of the AAA response, while *FRP_2* is forwarded to the peer through an *EAP-Request/FRM*, the rMSK is processed by the EAP-FRM method to derive the MSK and EMSK keys (*3*).

Nevertheless, since the use of a two-party model has some security weakness [71] for key distribution, the use of a three-party approach is recommended when designing a FRP. In this case, instead of using the AAA protocol, the FRP securely distributes the rMSK to the authenticator.

It is worthy noting that, whereas EAP-FRM enables the transport of authentication data, the conveyed authentication information is required to provide a fast processing and reduced number of messages in order to obtain a single round-trip between the authenticator and the backend server in the worst case. In section 3.6, we describe some examples about how EAP-FRM can transport different fast re-authentication protocols. Nevertheless, before that, we detail the EAP-FRM method packet format and AAA protocol extensions to relevant protocols such as RADIUS and Diameter.

3.3 EAP-FRM Format

The packet format for the EAP-FRM messages (depicted in Fig. 3.4) follows the EAP packet format defined in [33]. The *Code*, *Identifier* and *Length* fields are common to any EAP message. The *Type field* contains the EAP method type value corresponding to EAP-FRM. Currently, this value is undefined since it must be assigned by the *Internet Assigned Numbers Authority* (IANA), an organization in charge of assigning protocol names and number registries used by Internet protocols. Following, the data field includes a single-octet *FRP-Type* field which identifies the transported FRP followed by an optional *payload* field consisting of a set of TLVs (*Type-Length-Value* field) which, for example, contains the specific data of the FRP.

Next, we provide a detailed description of each field and detail the different values that

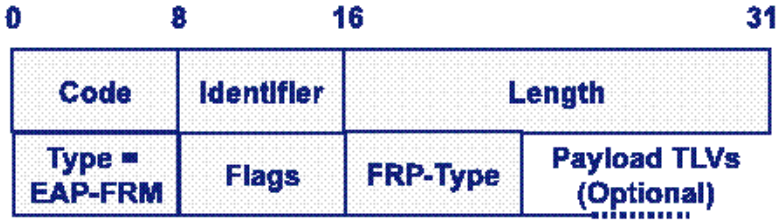


Figure 3.4: EAP Fast Re-authentication Method (EAP-FRM)

can be used.

- *Flags*: It is a 1-octet field reserved for flags. Currently, no flags are defined in EAP-FRM.

- *FRP-Type*: The FRP-Type is 1-octet field which identifies the transported fast re-authentication protocol. The next FRPs have been defined in EAP-FRM:

  1. Reserved
  2. 3PFH-based. This type refers to a FRP based on the 3PFH protocol (see section 3.6.1).
  3. ERP-based. This type refers to a FRP based on the ERP protocol (see section 3.6.2).
  4. Kerberos. This type refers to the FRP presented in chapter 4 as contribution of this PhD thesis

- *Payload TLVs*: the rest of the EAP-FRM packet is optionally composed of a set of TLVs. Each TLV is formed by 1-octet type field, two octet length field and n-octet data field. The length field indicates the length of the data field in number of octets. Next, there are some TLVs initially considered:

  Nonce TLV. It contains a pseudo-random nonce sent by the EAP-FRM peer or the EAP-FRM server. As explained in section 3.5, these values are used to create the *Method-Id* identifier which, in turn, is employed to generate the key material exported by EAP-FRM. This is a mandatory TLV that must be only used in the first *EAP-Request/FRM* or first *EAP-Response/FRM*. When included in the first *EAP-Request/FRM*, it contains a pseudo-random nonce generated by the EAP-FRM server. Otherwise, when included in the first *EAP-Response/FRM*, it contains a pseudo-random nonce generated by the EAP-FRM peer. The management and size of the nonces follows the recommendations in [32].

  FRP-Payload TLV. It transports the specific PDU of the FRP. This TLV can be optionally used in *EAP-Request/FRM* or *EAP-Response/FRM* messages.

Auth-Server TLV. It contains the backend server's *Network Address Identifier* (NAI) or *Fully Qualified Domain Name* (FQDN) that manages the current EAP authenticator. This optional TLV can be only used in the first *EAP-Request/FRM* message sent by the EAP server to the peer.

User-Id TLV. This TLV, destined to transport the user's NAI, is primarily used for routing purposes similarly to the *EAP-Response/Identity* message defined in the EAP specification [33]. The user's NAI is allowed to be truncated or modified by, for example, omitting the name portion of the NAI. This optional TLV, if used, must be included in an *EAP-Response/FRM* message.

Auth TLV. It is included to integrity protect an EAP-FRM message. This optional TLV can be included either in a *EAP-Request/FRM* or *EAP-Response/FRM* message once an rMSK is generated by the FRP. In section 3.5.3, we describe the process followed to compute the value transported by this TLV.

Integrity-Algorithm TLV. It indicates the integrity algorithm used to compute the value contained in the AUTH TLV. This TLV can be optionally used in conjunction with the AUTH TLV. If it is not included, the default cryptosuite is used. Some allowed cryptosuites are:

1. Reserved
2. HMAC-SHA256-64
3. HMAC-SHA256-128 (default)
4. HMAC-SHA256-256

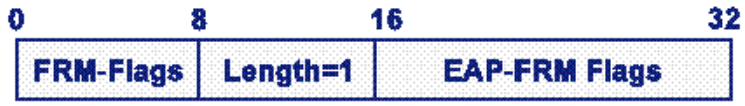
KDF TLV. This TLV is optionally included in the first *EAP-Request/FRM* sent by the EAP server and specifies the KDF that EAP peer has to use to generate the key hierarchy defined for EAP-FRM (described in section 3.5.2). If this TLV is not included, it means that the default value is used. The following values are considered:

1. Reserved
2. PRF+ key expansion specified in [31] based on HMAC-SHA-256 [134].

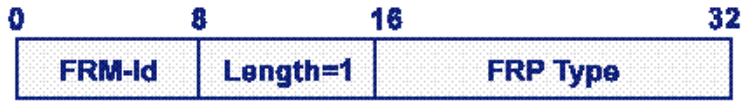
3.4 AAA Protocol Extensions

If the authenticator is not able to process the FRP payload sent by the peer within an *EAP-Response/FRM*, it may need to contact an AAA server to process the FRP (see Fig. 3.1). Thus, the FRP may need to be carried from the authenticator to the backend authentication server (AAA server) by means of an AAA protocol such as RADIUS [36] or Diameter [37].

For RADIUS, the definition of three new attributes is required (see Fig. 3.5): *RADIUS FRM-Flags*, *RADIUS FRP-Id* and *RADIUS FRP-Payload-Attr*. The first transports one



(a) RADIUS FRM-Flags attribute



(b) RADIUS FRP-Id attribute

(c) RADIUS FRP-Payload-Attr attribute

Figure 3.5: Definition of new RADIUS attributes

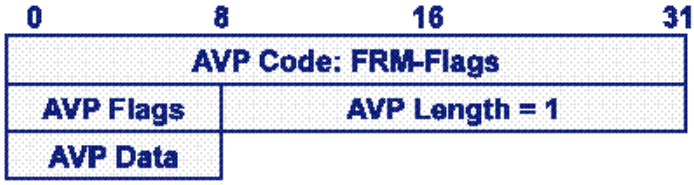
octet with the content of the flag field in the EAP-FRM packet; the second transports the FRP type (one of the numbers defined for the *FRP-Type* filed in the EAP-FRM packet) and the third carries the FRP payload. These attributes are included in both *RADIUS Access-Request* and *RADIUS Access-Accept* messages. It is important to note that the *FRP-Id* attribute allows the backend authentication server to determine which specific FRP is contained in the attribute *FRP-Payload-Attr*.

In case Diameter is selected to convey authentication information between the authenticator and the AAA server, it is necessary the definition of a new Diameter application that we have called *Diameter Fast Re-authentication Application*. The main reason of defining a new application it is motivated by the fact that current Diameter applications related to authentication and authorization such as Diameter EAP [107] and Diameter NASREQ [106] are not applicable in our architecture for the following reasons. On the one hand, while Diameter EAP assumes a pass-through configuration mode where EAP messages are forwarded by the authenticator to the server, our architecture is based on an standalone EAP method (EAP-FRM). On the other hand, Diameter NASREQ supports CHAP for authentication which does not match with our requirements where any strong key distribution protocol might be transported.

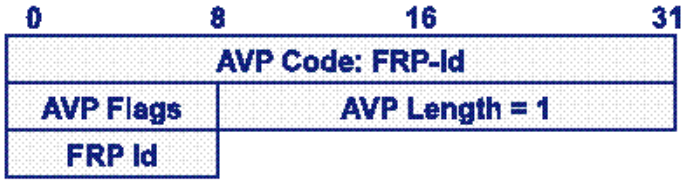
In the *Diameter Fast Re-authentication Application* we define two commands:

- *Diameter-FR-Request*: to carry the payload of an EAP-FRM message from the authenticator to the authentication server.
- *Diameter-FR-Answer*: for the same purpose in the opposite direction.

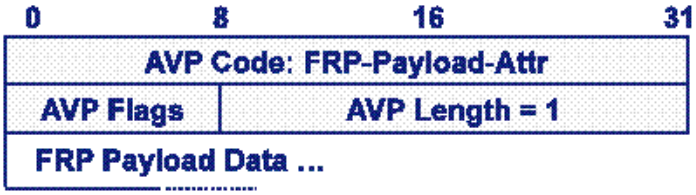
Both messages include three new Diameter AVPs (FRM-Flags, FRP-Id and



(a) Diameter FRM-Flags AVP



(b) Diameter FRP-Id AVP



(c) Diameter FRP-Payload-Attr AVP

Figure 3.6: Definition of new Diameter AVPs

FRP-Payload-Attr) which transports the flags field in EAP-FRM, the FRP type and the FRP payload, respectively. These AVPs are defined in Fig. 3.6.

It is worth noting that these extensions to the AAA protocols are not considered as a deployment barrier of the proposed re-authentication architecture for several reasons. First, AAA protocols are defined in a modular manner such that the required extensions on the AAA protocols do not affect the basic implementation of the AAA protocols. Second, the required changes to the AAA protocols can be dealt with by software upgrades for AAA clients and servers.

3.5 Discussion

3.5.1 Exported Parameters

In addition to the key material, EAP-FRM exports some additional parameters such as the *Peer-Id*, *Server-Id* and *Session-Id*. Additionally, an EAP method may optionally export an *Initialization Vector* (IV). Nevertheless, exporting an IV has not been considered in EAP-FRM since its use has been deprecated.

The *Peer-Id* and *Server-Id* unequivocally identify the EAP peer and server. Despite several formats can be used to represent both identifiers, the *EAP Keying Management Framework* (EAP KMF) recommends the use of a *Full Qualified Domain Name* (FQDN).

The *Session-Id* is a method-specific identifier used to identify the authentication between the EAP peer and server. In particular, EAP-FRM generates this identifier by concatenating a byte that represents the method type (*Type-Code*) and an unique identifier known as *Method-Id*.

$$Session\text{-}Id = Type\text{-}Code \,||\, Method\text{-}Id$$

In particular, in the context of EAP-FRM, the *Method-Id* is obtained from the concatenation of the pseudo-random values *Nonce-Peer* and *Nonce-Server* generated by the EAP-FRM peer and the EAP-FRM server, respectively. We have followed this approximation since it combines two pseudo-random values exchanged between the peer and the authenticator.

$$Method\text{-}Id = Nonce\text{-}Peer \,||\, Nonce\text{-}Server$$

3.5.2 Key Derivation

As mentioned in previous section 3.2, EAP-FRM is able to generate cryptographic material from the key (rMSK) exported by the fast re-authentication protocol. In particular, EAP-FRM uses this key as root of a key hierarchy generated following the guidelines provided in [110]. In particular, this reference explains a general scheme for the derivation of keys based on the use of a *Key Derivation Function* (KDF) to generate new keys from a root key. The new keys, referred to as USRK (*User Specific Root Key*), are derived according to following formula:

USRK = KDF(root key, key label, additional data, key length)

Although in [110] it is assumed that the root key is the EMSK generated after a successful EAP authentication, the scheme is general enough to be applied to any key selected as root key. Thus, other key different from the EMSK can be used as input parameter in the KDF function. In particular, EAP-FRM uses the rMSK exported by the FRP as root key to generate a key hierarchy. Additionally, as we can observe, the KDF function includes: a key label associated with the derived key; some additional data that may be necessary to perform the derivation process and a value indicating the length of the derived key. It is assumed that the KDF function generates the same output for the same input parameters. By default, the KDF is inherited from the key expansion function PRF+ defined in IKEv2 [31]. The default pseudo-random function used by PRF+ is HMAC-SHA-256 [134]. Nevertheless, EAP-FRM allows the negotiation of a different key derivation function through the *KDF TLV*.

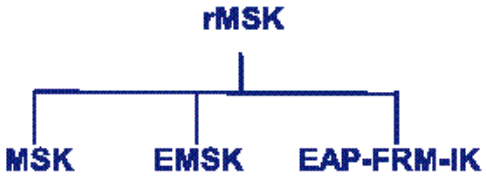


Figure 3.7: EAP-FRM Key Hierarchy

To construct the key hierarchy used by EAP-FRM, we have employed this general scheme where, the KDF function is recursively used to generate the different keys that integrate the key hierarchy depicted in Fig. 3.7. As observed, rMSK acts as root key and is used to generate the keys exported by EAP-FRM (MSK and EMSK) as well as a key (EAP-FRM-IK) necessary within the EAP-FRM method to provide integrity protection to EAP-FRM messages.

Integrity Key EAP-FRM-IK

The EAP-FRM-IK is used to compute the *Auth TLV* destined to integrity protect the EAP-FRM messages. EAP-FRM-IK is exclusively used within the EAP-FRM method and is never exported. Under the *EAP Keying Management Framework*, this key is considered as a *Transient EAP Key* (TEK). The EAP-FRM-IK is derived from the rMSK as follows:

EAP-FRM-IK = KDF(rMSK, Session-Id || "EAP-FRM-Integrity-Key", length);

Again, the key label used as input parameter is obtained from the concatenation of the *Session-Id* exported by EAP-FRM and the string "EAP-FRM-Integrity-Key". The length field will depend upon the integrity algorithm selected by the EAP-FRM server during the EAP-FRM exchange. For example, when HMAC-SHA-256 [134] is used for the integrity algorithm, length=32.

MSK and EMSK Exported by EAP-FRM

The EAP-FRM method, following the recommendations of the EAP KMF [38], exports two 64-bytes length keys: MSK and EMSK. Both keys are derived from the rMSK applying the KDF function with the following input parameters:

(MSK, EMSK) = KDF(rMSK, Session-Id || "EAP-FRM-EAP-Keying-Material", 128)

As observed, the key label is formed by concatenating the Session-Id and the string "EAP-FRM-EAP-Keying-Material". The KDF is asked to generate 128 bytes from which the first 64 bytes correspond to the MSK and the other 64 bytes to the EMSK.

3.5.3 Message Authentication

If the EAP-FRM peer or the EAP-FRM server requires integrity protection for some of the EAP-FRM messages, an *Auth TLV* is included. The *EAP-FRM-IK* is used to compute Auth TLVs. The process followed to derive the EAP-FRM-IK is described in section 3.5.2. The value of the *Auth TLV* can be generated once the *EAP-FRM-IK* is derived, in the following way:

Auth TLV value = Integrity-Algorithm(EAP-FRM-IK, EAP-FRM-Packet)

The value field in the TLV contains an authentication tag computed over the entire EAP-FRM packet (*EAP-FRM-Packet*), starting from the first bit of the code field to the last bit of the Auth TLV with the value field of the Auth TLV filled with all 0s for the computation. The *Integrity-Algorithm* used to create the *Auth TLV* value is specified in the *Integrity-Algorithm TLV* sent in the first *EAP-Request/FRM* message.

3.5.4 Capabilities Negotiation

Although the proposed solution is able to transport any FRP, current specification of EAP-FRM does not allow the negotiation of the FRP. The authenticator sets the specific FRP that is going to be used in the *EAP-Request/FRM*. If the peer supports EAP-FRM but it does not support such FRP, it must respond with an *EAP-Response/Nak* message. The same procedure is applied when the integrity algorithm (specified in the *Integrity-Algorithm TLV*) and the key derivation function (specified in the *KDF TLV*) indicated by the authenticator in the first *EAP-Request/FRM* are not supported by the peer. Nevertheless, the inclusion of a FRP negotiation in the architecture is considered motive of future work.

3.5.5 TLVs Usage

The following Table 3.1 summarizes the TLVs defined in EAP-FRM and indicates which TLVs may or may not be found in the EAP-FRM messages, and the quantity. In both *EAP-Request/FRM* and *EAP-Response/FRM*, we distinguish between the first (*F*) and last (*L*) messages, referring to the remaining exchanged messages as intermediate (*I*) messages. The table uses the following symbols:

- *0*: The TLV must not be present in the message.
- *0-1*: Zero or one instance of the TLV may be present in the message. It is considered an error if there is more than one instance of the TLV.
- *1*: One instance of the TLV must be present in the message.

	MESSAGE TYPE					
	EAP-Request / FRM			*EAP-Response / FRM*		
TLV NAME	**F**	**I**	**L**	**F**	**I**	**L**
Nonce	1	0	0	1	0	0
FRP-Payload	0-1	0-1	0-1	0-1	0-1	0-1
Auth-Server	0-1	0	0	0	0	0
User-Id	0	0	0	0-1	0-1	0-1
Auth	0	0-1	0-1	0	0-1	0-1
Integrity-Algorithm	0-1	0	0	0-1	0	0
KDF	0-1	0	0	0-1	0	0

Table 3.1: Usage of TLVs defined in EAP-FRM

As observed, the first *EAP-Request/FRM* message sent by the EAP server is expected to contain a pseudo-random value generated by the EAP server within the *Nonce TLV*. Optionally, this initial message may contain the following TLVs: a *FRP-Payload TLV* transporting the first FRP message; an *Auth-Server TLV* indicating the NAI or FQDN of the backend server controlling the target authenticator; an *Integrity-Algorithm TLV* used by the authenticator to indicate the integrity algorithm that will be used to compute the *AUTH TLV* in future messages, and a *KDF TLV* used by the authenticator to suggest a specific function to generate the EAP-FRM key hierarchy.

The first *EAP-Response/FRM* message is also expected to contain a nonce generated by the peer in the *Nonce TLV*. Recall that, as explained in section 3.5.1, the nonces exchanged in the first *EAP-Request/FRM* and *EAP-Response/FRM* are used to generate the *Method-Id* identifier which, in turn, is necessary to generate the EAP-FRM key hierarchy. Similarly to the first *EAP-Request/FRM* message, the first *EAP-Response/FRM* message can optionally contain the following TLVs: a *FRP-Payload TLV* containing the first FRP payload generated by the peer; an *User-Id TLV* to indicate the authenticator the specific domain to which the authentication data should be routed; an *Integrity-Algorithm TLV* used by the authenticator to confirm the integrity algorithm that will be used to compute the *AUTH TLV* in future messages, and a *KDF TLV* used by the peer to confirm the key derivation function that will be used to generate the EAP-FRM key hierarchy.

After these initial messages, a set of intermediary *EAP-Request/FRM* and *EAP-Response/FRM* messages are exchanged between the peer and server. On the one hand, intermediary *EAP-Request/FRM* messages may contain a *FRP-Payload TLV*.

On the other hand, intermediary *EAP-Response/FRM* messages may contain not only *FRP-Payload TLV* but also *User-Id TLV* to specify where the FRP message should be routed. Additionally, once the FRP exports a valid rMSK, the *Auth TLV* may be used to integrity protect EAP-FRM messages.

As depicted in Table 3.1, the same behaviour described for the intermediary *EAP-Request/FRM* and *EAP-Response/FRM* messages is also valid for the final ones.

3.5.6 Security Considerations

Given that EAP-FRM is an EAP method respecting the EAP authentication model [33], EAP-FRM itself does not introduce any new vulnerability to EAP since the definition of new EAP methods is conceived as the extensibility mechanism available at EAP to design new authentication mechanisms according to the standardized protocol. EAP-FRM is a generic transport that can carry different fast re-authentication protocols. Thus, the security of the overall EAP-FRM process strongly depends on the inherent security of the fast re-authentication protocol in use. Thus, it is highly recommendable to design the transported fast re-authentication protocol with suitable security properties.

3.6 Use cases

In this section, we describe how our architecture can be used in conjunction with different protocols for fast re-authentication, by providing a transport that avoids any link-layer or standard modification. In particular, we explain how the 3PFH protocol can employ EAP-FRM to provide fast network access without any modification to the EAP standard unlike it happens in the reference [16]. Finally, we analyze how the recent standardized ERP protocol can be adapted to our EAP-FRM based solution and show that it is possible to provide a fast re-authentication solution without modifying EAP standard protocol and associated wireless technologies, reducing the impact in future deployments. As we will analyze in chapter 4, this is the case of Kerberos, which is the secure three-party protocol we use in our solution.

3.6.1 3PFH-based Fast Re-authentication Protocol with EAP-FRM and RADIUS

ERP defines a key distribution protocol following a two-party model inherited from the EAP authentication process. The two-party model is valid for mutual authentication but it is inappropriate for key distribution where three entities are involved: peer, authenticator and server. In fact, two-party solutions have been known to be inappropriate since they introduce some security vulnerabilities as described in [71]. Taking into account this problem, 3PFH [16] proposes a novel key distribution protocol for EAP-based networks based on the principles of a three-party key distribution model.

Despite from a security point of view **3PFH** can be considered as an improved solution with respect to **ERP**, **3PFH** requires a transport based on the modification of the **EAP** authentication protocol. In particular, as described in section 2.4.5, **3PFH** messages are transported in both *EAP-Response/Identity* and *EAP-Success*, which deviates from the standard behaviour described in **EAP** specification [33]. In particular, while the *EAP-Response/Identity* only contains the user's identity, the *EAP-Success* is not allowed to carry any data. These modifications to the standardized **EAP** protocol complicates the adoption and deployment of the solution. In the following we demonstrate that this inconvenient can be avoided by using the **EAP-FRM** method as transport for the **3PFH** messages. In particular, the analysis is performed depending on whether the peer roams between authenticators controlled by the same *Key Distribution Server* (KDS) or not, which is the entity in charge of performing the key distribution within a domain.

Basic 3PFH for Intra-KDS handoff

The basic protocol version of **3PFH** is used when the peer roams between authenticators controlled by the same KDS (intra-KDS handoff). The basic protocol interaction is based on the exchange of four different messages between the peer (A), the authenticator (B) and the KDS (S) according to the following flow:

1. $\mathbf{A} \Rightarrow \mathbf{B}$: A,$\{N_A, SEQ_{AS}, B\}_{K_{AS}^{auth}}$ (Message 3PFH_1)
2. $\mathbf{B} \Rightarrow \mathbf{S}$: B,$\{N_B, A_{hash}\}_{K_{BS}^{auth}}$,A,$\{N_A, SEQ_{AS}, B\}_{K_{AS}^{auth}}$ (Message 3PFH_2)
3. $\mathbf{S} \Rightarrow \mathbf{B}$: $\{A, B, N_A, N_B, N_S\}_{K_{AS}^{auth}}$,$\{A, B, N_A, N_B, N_S, K_{AB}\}_{K_{BS}^{auth}}$ (Message 3PFH_3)
4. $\mathbf{B} \Rightarrow \mathbf{A}$: $\{A, B, N_A, N_B, N_S\}_{K_{AS}^{auth}}$ (Message 3PFH_4)

Figure 3.8 explains how the basic **3PFH** protocol execution can be performed by using the **EAP-FRM** architecture. The process starts when the authenticator sends to the peer an *EAP-Request/FRM* message (*1*) with the *FRP-Type* field set to **3PFH**-based code, and containing two TLVs: *Nonce TLV* with a pseudo-random number (P_B) specifically generated by the authenticator for the **EAP-FRM** message and independent of the pseud-random numbers used in **3PFH**; and *Auth-Server TLV* which contains the specific domain ("local" in this example) to which the authenticator belongs.

After that, the peer sends an *EAP-Response/FRM* (*2*) containing message *3PFH_1* within a *FRP-Payload TLV*. This message also includes a *Nonce TLV* with a pseudo-random number (P_A) specifically generated by the peer for the **EAP-FRM** message and independent of the pseudo-random numbers used in **3PFH**; and an *User-Id TLV* that has the NAI of *@local* which is the domain where the authentication information should be routed. On the reception, the authenticator creates a *RADIUS Access-Request* (*3*) that includes the *User-Name* attribute carrying the content of the *User-Id TLV* ; the *FRP-Id* attribute indicating the **3PFH**-based protocol as the **FRP**; and the *FRP-Payload-Attr* containing message *3PFH_2*.

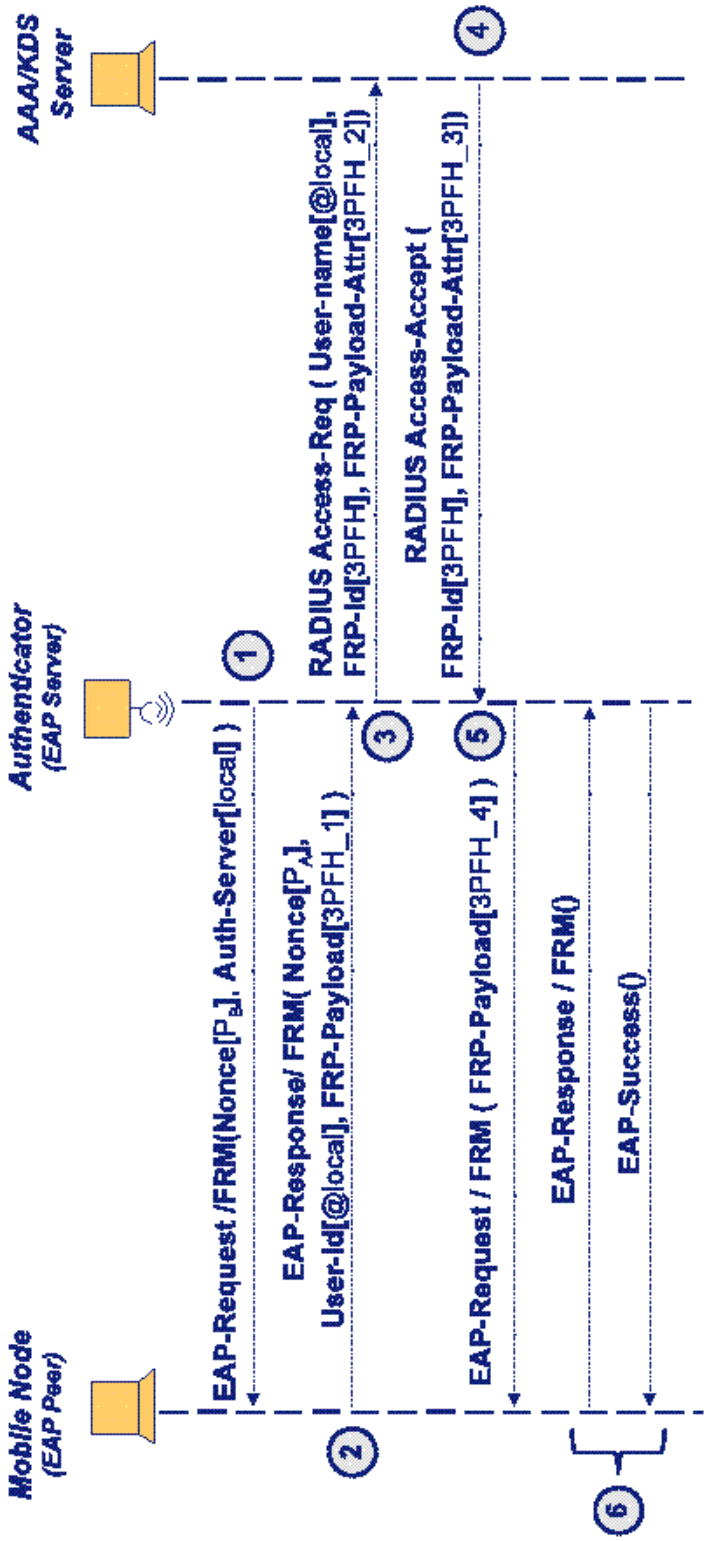


Figure 3.8: 3PFH-based FRP with EAP-FRM: Basic Protocol Version

When the KDS successfully re-authenticates the peer based on message *3PFH_2*, it performs the key distribution process by generating message *3PFH_3* which is composed of two tokens: one to be processed by the authenticator and another one for the peer. This message is inserted in a *FRP-Payload-Attr* attribute and sent back to the authenticator through a *RADIUS Access-Accept* (*4*). The **EAP-FRM** method engine in the authenticator will process its token and forward the other one (message *3PFH_4*) through an *EAP-Request/FRM* to the peer (*5*). After that, the **3PFH** execution is completed and a session key (rMSK) is securely distributed to both the peer and the authenticator. The rMSK is used by the **EAP-FRM** method to derive the MSK and EMSK exported to the lower-layer. Finally, the **EAP-FRM** method concludes through an additional *EAP-Response/FRM / EAP-Success* exchange (*6*) which is necessary to maintain backward compatibility with current **EAP** specification.

Extended 3PFH for Inter-KDS handoff

The extended protocol version of **3PFH** is destined to be applied when the peer roams between authenticators controlled by different KDSs (inter-KDS handoff). The extended protocol interaction is based on the exchange of four different messages between the peer (A), authenticator (B), a local KDS (L) located in the visited domain and a home KDS (H) located in the user's home domain according to the following flow:

1. **A** $\Rightarrow$ **B**: A,B,$\{N_A, SEQ_{AH}, L, B_{hash}\}_{K_{AH}^{auth}}$ (Message 3PFH_1)
2. **B** $\Rightarrow$ **L**: B,$\{N_B, A_{hash}\}_{K_{BL}^{auth}}$,A,B,$\{N_A, SEQ_{AH}, L, B_{hash}\}_{K_{AH}^{auth}}$ (Message 3PFH_2)
3. **L** $\Rightarrow$ **H**: L,$\{N_L, A_{hash'}\}_{K_{LH}^{auth}}$,A,$\{N_A, SEQ_{AH}, L, B_{hash}\}_{K_{AH}^{auth}}$
   (Message 3PFH_3)
4. **H** $\Rightarrow$ **L**: $\{A, L, N_A, N_L, N_H\}_{K_{AH}^{auth}}$,$\{A, B_{hash}, L, N_A, N_L, N_H, K_{AL}\}_{K_{LH}^{auth}}$
   (Message 3PFH_4)
5. **L** $\Rightarrow$ **B**: $\{A, L, N_A, N_L, N_H\}_{K_{AH}^{auth}}$,$\{A, B, N_A, N_B, N_L^{'}, [SEQ_{AL}]\}_{K_{AL}^{auth}}$,
   $\{A, B, N_A, N_B, N_L^{'}, K_{AB}\}_{K_{BL}^{auth}}$ (Message 3PFH_5)
6. **B** $\Rightarrow$ **A**: $\{A, L, N_A, N_L, N_H\}_{K_{AH}^{auth}}$,$\{A, B, N_A, N_B, N_L^{'}, [SEQ_{AL}]\}_{K_{AL}^{auth}}$
   (Message 3PFH_6)

The integration of the extended version of **3PFH** with the **EAP-FRM** architecture (depicted in Fig. 3.9) is very similar to that explained in previous section for the intra-KDS handoff , except that now the home AAA/KDS server must be contacted. As observed, the authenticator starts by sending an *EAP-Request/FRM* message (*1*) indicating the domain ("local") where the authenticator is located. The peer detects that the authenticator is under the control of a different KDS and, consequently, the extended version of **3PFH** must

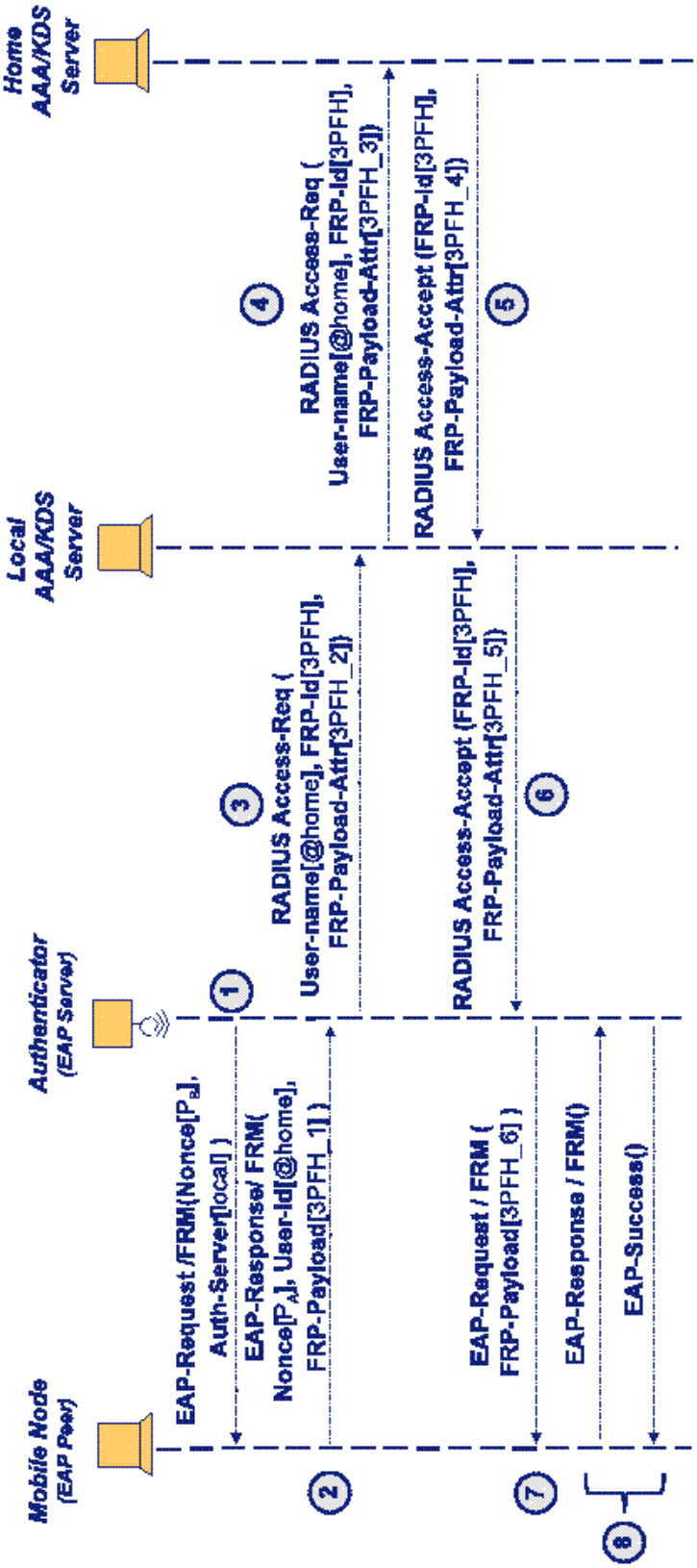


Figure 3.9: 3PFH-based FRP with EAP-FRM: Extended Protocol Version

be executed in order to distribute a key to the new KDS and authenticator. For this reason, the peer builds an *EAP-Response/FRM* (*2*) containing message *3PFH_1* (*FRP-Payload TLV*) and the home domain is indicated (*User-Id TLV*) as the domain towards the authentication data must be routed. Subsequent messages *3PFH_2*, *3PFH_3*, *3PFH_4* and *3PFH_5* travel through the AAA infrastructure within *RADIUS Access-Request* (*3 - 4*) and *RADIUS Access-Accept* (*5 - 6*) messages. The last message *3PFH_6* is delivered to the peer through an an *EAP-Request/FRM* (*7*) previous to the final *EAP-Response/FRM* / *EAP-Success* exchange (*8*).

3.6.2 ERP-based Fast Re-authentication Protocol with EAP-FRM and RADIUS

As described in section 2.4.5, ERP (*EAP Extensions for EAP Re-authentication Protocol*) [68] extends EAP by defining three new EAP messages *EAP-Initiate/Re-auth-Start*, *EAP-Initiate/Re-auth* and *EAP-Finish/Re-auth*. When the peer moves to a new authenticator and receives an *EAP-Request/Identity* (or *EAP-Initiate/Re-auth-Start* instead), it sends an *EAP-Initiate/Re-auth* to the server through the new authenticator. If the message is successfully verified, the server answers with an *EAP-Finish/Re-auth* to the peer. When the peer verifies that it is recent and correctly signed, the the re-authentication process is successfully completed.

However, this mode of operation requires modifications in both the standardized EAP state machine and current wireless technologies [68]. This fact has been widely criticized due to the impact on future ERP deployments. By using our architecture for fast re-authentication to carry the content of ERP within EAP-FRM, this problem can be solved. Indeed, we present an example where EAP-FRM is used to transport a FRP based on the ERP protocol. In particular, for this example, the FRP payload is composed by the fields defined in ERP messages [68], starting from the Type field to the end. Since there are three messages in ERP (*EAP-Initiate/Re-auth-Start*, *EAP-Initiate/Re-auth* and *EAP-Finish/Re-auth*), three different payloads are transported as the FRP payload: *IRS** (fields of *EAP-Initiate/Re-auth-Start* from the Type field), *IR** (fields of the *EAP-Initiate/Re-auth* from the Type field) and *FR** (fields of the payload *EAP-Finish/Re-auth* from the Type field). In terms of this example, the backend authentication server is called ER server.

Fast Re-authentication Phase

Figure 3.10 shows the protocol exchange that takes place when a peer performs a handoff between authenticators that are under the control of the same ER server. The process starts when the authenticator sends an *EAP-Request/FRM* (with the *FRP-Type* field set to ERP-based code) to the peer (*1*), containing an *Auth-Server TLV*. In this example, this TLV can carry the same value that the so-called *Domain-Name TLV* carries in ERP. Moreover, a *FRP-Payload* is included to contain the *IRS** payload. On the reception of this message the peer knows that is under an authenticator of the domain "local".

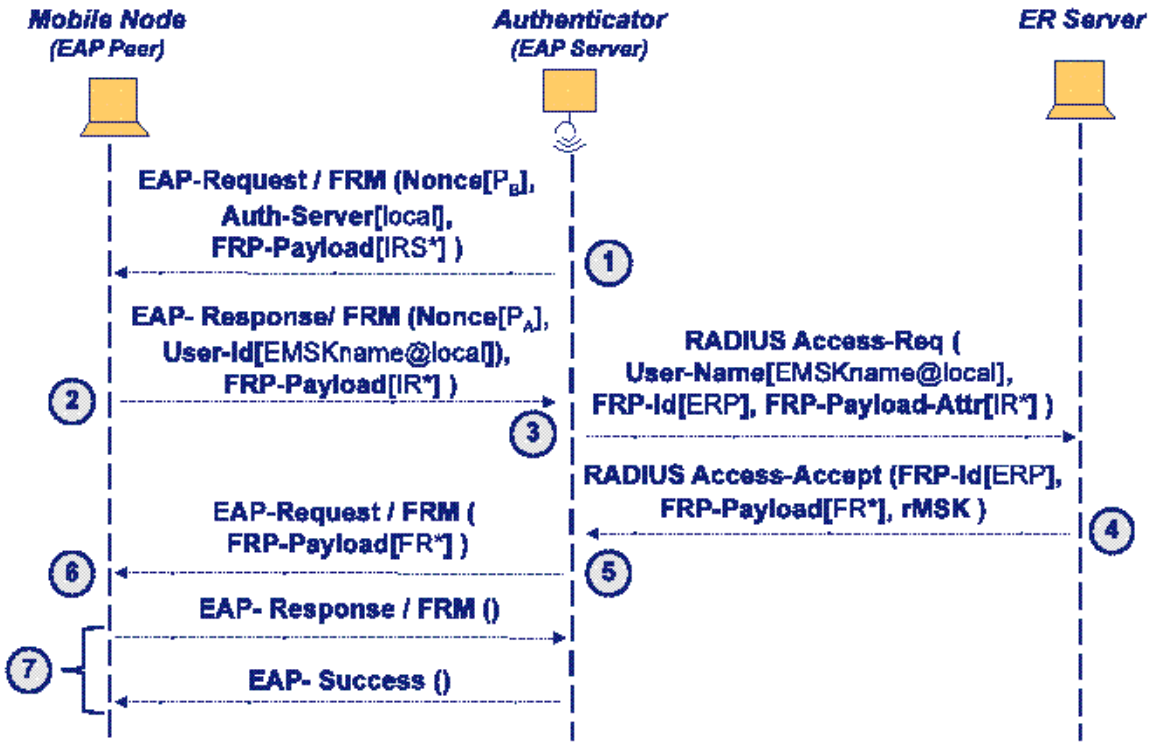


Figure 3.10: ERP-based FRP with EAP-FRM: Handoff Operation

Then, the peer sends an *EAP-Response/FRM* containing the *IR** payload (*2*). The message includes an *User-Id TLV* that has the value of the *keyName-NAI* (e.g., EMSKname@local that ERP implements). On the reception, the authenticator creates a *RADIUS Access-Request* (*3*) that includes the *User-Name* attribute carrying the content of the *User-Id TLV* ; the *FRP-Id* attribute indicating the ERP-based protocol as the FRP; and the *FRP-Payload-Attr* containing the ERP payload (*IR**). Note that, in this case, the authenticator is not required to understand the data contained in the *FRP-Payload TLV* because, as described in the ERP specification [68], the ERP-based FRP needs to contact a backend authentication server in charge of verifying the authentication data generated by the peer (IRS* payload).

When the ER server performs a successful re-authentication based on the information contained in the *FRP-Payload-Attr*, it generates the associated *RADIUS Access-Accept* (*4*). This message includes: the *FRP-Id* attribute (indicating ERP as FRP); the FRP response message (*FR**) that is included in a *FRP-Payload-Attr* attribute; and the rMSK. The rMSK will reach the EAP-FRM method at the authenticator, and the EAP-FRM method engine will derive the MSK and EMSK from the rMSK and will export them to the lower-layer (*5*). Additionally, the authenticator forwards the content of the FRP response message (*FR**) to the peer through an *EAP-Request/FRM message*. When the peer processes the *FR** payload contained in the received *FRP-Payload TLV*, the execution of the FRP (in this case the ERP-based protocol) ends (*6*). As a consequence of a successful

ERP-based FRP execution, the EAP-FRM method in the peer will obtain the same rMSK that the one received in the authenticator and will derive and export the same MSK and EMSK to the lower-layer. The EAP-FRM method concludes through an additional *EAP-Response/FRM* and *EAP-Success* messages (*7*) to maintain backward compatibility with current EAP specification.

Bootstrapping Phase

The ERP specification defines the so-called explicit bootstrapping. As described in [68], this operation is intended to support those situations where, for example, a local ER server needs to request a DSRK from the home ER server. In terms of our architecture, nothing changes with respect to the signalling showed in previous section, except that now the home ER server must be contacted (*1- 2*). Figure 3.11 shows the signalling involved in our architecture taking into account an ERP-based FRP that includes explicit bootstrapping.

3.7 Performance Evaluation

The architecture for fast re-authentication based on EAP-FRM offers a generic transport independent of the underlying technology which does not require any modification either to the EAP specification or existing lower-layer technologies. Since EAP-FRM is merely a transport to convey authentication data, the efficiency of the re-authentication process depends on the conveyed authentication information. In other words, in order to obtain a fast network access, the FRP is required to provide a fast processing and reduced number of messages.

Therefore, the evaluation of the reduction in the time required to complete a re-authentication will be performed in next chapter 4 where we propose a specific FRP based on Kerberos. At this point, our objective is to verify that the EAP-FRM transport itself is able to transport any re-authentication protocol without compromising the network access time. In particular, since EAP-FRM is a standalone method implemented by peer and authenticator, we are simply interested in evaluating the EAP-FRM behaviour in the wireless access network. The communication between the authenticator and the backend authentication server is not going to be analyzed yet since it depends on aspects such as the number of message exchanges or the location of the AAA server, both dependent on the specific FRP in use. We will also take a look at this in chapter 4 with the use of Kerberos as FRP.

To carry out our analysis, we have implemented a prototype of our architecture for fast re-authentication. Using this implementation, we have deployed a scenario which is used to test EAP-FRM.

3.7.1 Implementation Details

To conduct real experiments, we have implemented a prototype of our architecture for fast re-authentication based on EAP-FRM. The implementation has been developed

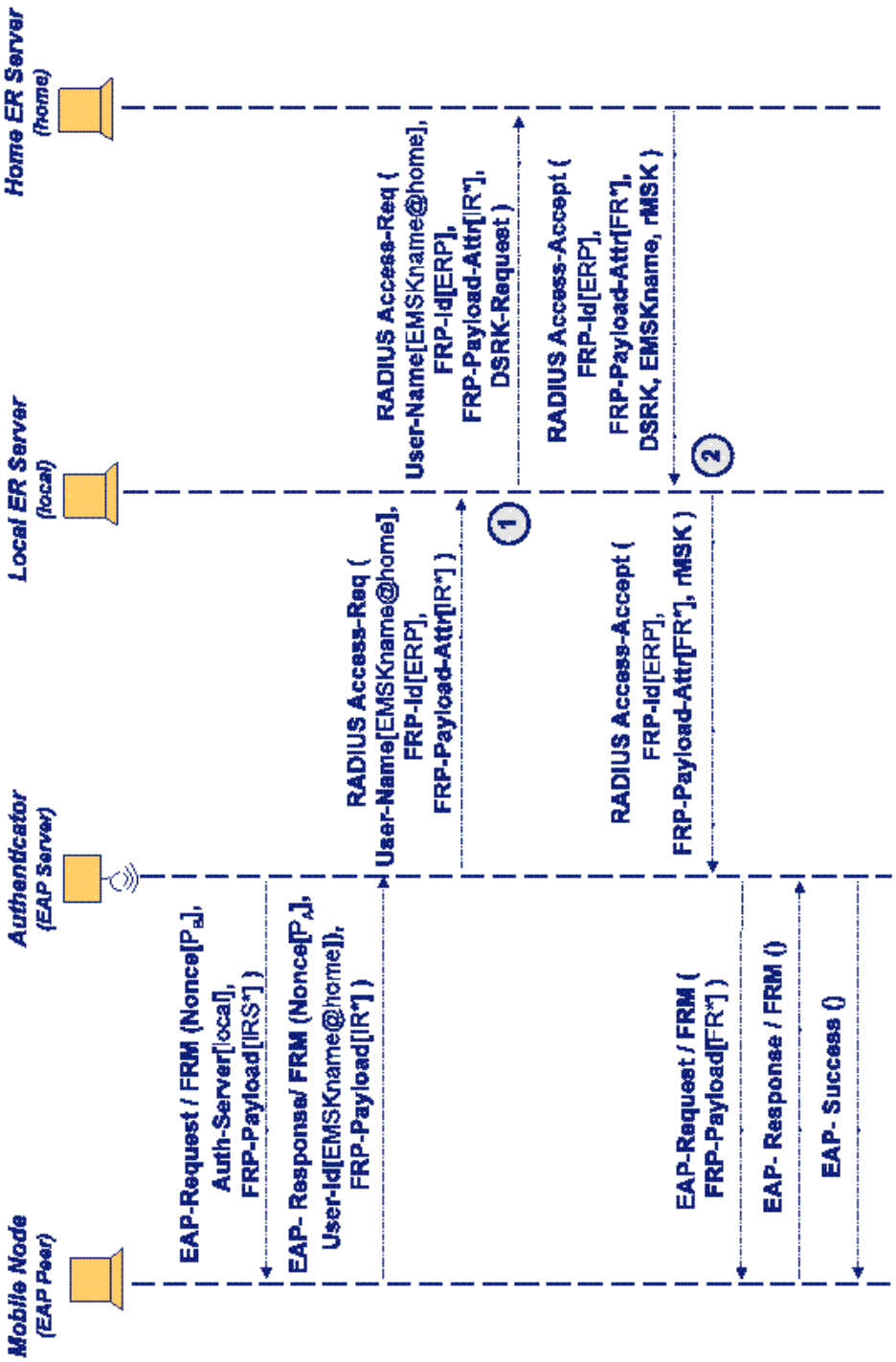


Figure 3.11: ERP-based FRP with EAP-FRM: Explicit Bootstrapping

following the EAP-FRM packet format and AAA protocol extensions proposed in previous sections 3.3 and 3.4, respectively.

On the one hand, the EAP-FRM implementation for both the peer and the authenticator has been developed using the open-source *wpa_supplicant* implementation version 0.6.4 [135] and *hostapd* implementation version 0.6.4 [136], respectively. On the other hand, regarding the communication between EAP authenticator and backend authentication server, we have selected RADIUS as AAA protocol. The RADIUS client implementation has been run by using the API provided by hostapd. This API, called from the EAP-FRM server implementation, allows the creation of the RADIUS messages. The backend authentication server is a RADIUS server implemented with *Free RADIUS* version 2.0.2 [137]. For managing the information carried by the *FRP-Payload-Attr* attribute described in 3.4, a new module has been developed in Free RADIUS.

3.7.2 Developed Testbed

A well-designed FRP is expected to provide a reduced operation time by, at most, only requiring one message exchange with a backend authentication server. For the purpose of this proof-of-concept testbed, we have employed 3PFH (see section 3.6.1) as fast re-authentication protocol since it provides a fast network access with high security levels [16]. Similarly, as representative wireless technology, in the testbed we employ IEEE 802.11.

As depicted in Fig. 3.12, we have deployed a scenario consisting of three Linux systems: a laptop which runs *wpa_supplicant* acting as the peer; two boxes which run *hostapd* acting as authenticators and a box acting as AAA/KDS server with *Free Radius* placed very near the authenticators (the average ping time is about $\approx$ 0.15ms). Details about the different machines are provided in Tab. 3.2.

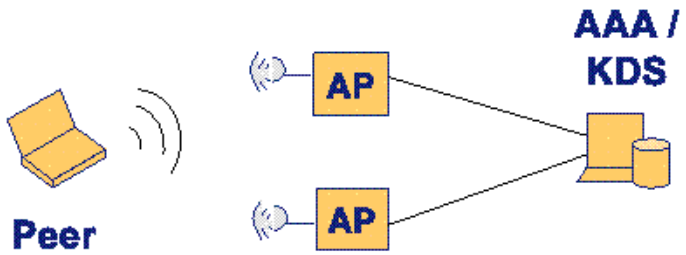


Figure 3.12: EAP-FRM Deployed Testbed

Machine	CPU Type	Freq(MHz)	RAM(MB)
Peer	VIA Nehemiah	1,200	488
Access Points	VIA Nehemiah	1,200	488
AAA/KDS	Pentium Dual	2,000	2048

Table 3.2: Testbed machines

As previously mentioned, our objective is to analyze the behaviour of EAP-FRM in the wireless access network. For this reason, we do not distinguish between intra or inter-domain scenarios since the communication between the authenticator and the AAA/KDS server depends on the specific FRP.

3.7.3 Performance Results

We have emulated around 200 handoffs involving re-authentications between the peer and the AAA server where the fast-re-authentication protocol messages are transported over EAP-FRM between peer and authenticator. In Fig. 3.13 we can observe an example of trace obtained from the authenticator ethereal [138] output[1].

No.	TIME	SOURCE	DESTINATION	PROTOCOL	INFO
1	98.075284	Aironet_b2:18:7b	Aironet_b2:13:4e	EAP	Request, Unknown type
2	98.077293	Aironet_b2:13:4e	Aironet_b2:18:7b	EAP	Response, Unknown type
3	98.079631	192.168.1.5	192.168.1.3	RADIUS	Access-Request(1)
4	98.080718	192.168.1.3	192.168.1.5	RADIUS	Access-Accept(2)
5	98.081216	Aironet_b2:18:7b	Aironet_b2:13:4e	EAP	Request, Unknown type
6	**98.081884**	**Aironet_b2:13:4e**	**Aironet_b2:18:7b**	**EAP**	**Response, Unknown type**
7	**98.082396**	**Aironet_b2:18:7b**	**Aironet_b2:13:4e**	**EAP**	**Success**

Figure 3.13: Ethereal Authenticator Output for EAP-FRM Authentication

Using this information, we have measured the time each node employs to process authentication messages. Table 3.3 shows a 95% confidence interval of total time that each entity devotes to message processing. As observed, our solution does not introduce high latency to the re-authentication process, which mainly depends on the efficiency of the re-authentication protocol. In fact, the critical point, in comparison with ERP or 3PFH, is the final *EAP Response* and *EAP Success* messages. This exchange is required in order to maintain compatibility with current EAP specification. However, our measurements have revealed that mobile requires an average time of 3.4 ms (+/- 1.37 ms) to complete this exchange. As we expected, this time does not impact in high degree in comparison with the complete re-authentication time. Nevertheless, we achieve significant advantages like compatibility with current standards, and the expected trade-off between fast re-authentication and simpler deployment.

[1]Since Ethereal has not registered the EAP-FRM type yet, it shows the *Unknown type* text.

Entity	Mean Processing Time (ms)	Confidence Interval (ms)
Mobile Node	1.5	±0.026
Authenticator	2.3	±0.203
AAA Server	0.96	±0.009

Table 3.3: Mean Processing Time and 95% Confidence Interval for 3PFH

3.8 Conclusions

This chapter has presented the first component of the fast re-authentication solution developed in the context of this PhD thesis. In particular, we have designed a generic architecture for fast re-authentication based on a new EAP method (EAP-FRM) which works on standalone configuration mode. This framework does not introduce any modification to link-layer technologies and is fully compatible with current standards.

Initially, in this chapter, we clarify that researchers have agreed [65] that the inefficiency of EAP in mobile environments must be solved by using a fast and secure key distribution process. Through a key distribution protocol, it can be verified that the user was previously authenticated in a full EAP authentication and, as a consequence, a key is distributed to both the peer and the authenticator for executing a security association protocol used to protect the wireless link. Unfortunately, relevant solutions following this approximation such as 3PFH [16] and ERP [68], have shown important drawbacks. In particular, both solutions require modification to the existing lower-layers standards or the EAP protocol itself which complicates their deployment.

To avoid this deployment issue caused by the modification of standardized technologies, we propose a generic fast re-authentication architecture for EAP-based wireless networks. Compared with previous solutions, the novelty of this proposal relies on its ability of not requiring modifications to EAP or existing lower layer protocols. This is achieved through a new EAP method called EAP-FRM which works on standalone authenticators and it can communicate, if necessary, with a backend authentication server.

After describing the different phases that integrate the fast re-authentication framework, the EAP-FRM method is extensively described by indicating messages and information elements expressed in TLV format. Similarly, extensions to the most relevant AAA protocols (RADIUS and Diameter) are defined to allow the authenticator to transmit authentication information to a backend authentication server. The EAP-FRM explanation is concluded with a detailed discussion about internal aspects of EAP-FRM, such as key derivation or message protection.

Next, through several use cases, it is demonstrated the ability of EAP-FRM to provide a fast re-authentication operation without requiring any modification to current standards. More precisely, the use of EAP-FRM is demonstrated by means of two examples of FRPs: 3PFH and ERP. Finally, through a performance evaluation performed over a proof-of-concept testbed, it is verified that the EAP-FRM architecture is able to transport any fast re-authentication protocol without compromising network access time.

In next chapters, we focus on the secure three-party key distribution protocol which will be used as FRP. In particular, instead of designing a new key distribution protocol, we have opt by using a well-known and standardized authentication and key distribution protocol such as Kerberos [74].

Chapter 4

Definition of a Kerberos-based Fast Re-Authentication Protocol for Next Generation Networks

In the previous chapter we have described the generic architecture for fast re-authentication based on the EAP-FRM method, which will be the basis of our fast re-authentication solution. This architecture, without requiring any modification to EAP or existing lower-layer technologies, acts as a vehicle to transport any key distribution protocol[1] intended to reduce the EAP authentication time. In this chapter we deal with the integration of EAP-FRM with a secure three-party key distribution protocol which, ultimately, is responsible for achieving a reduction in the latency of the re-authentication process.

In the literature we can find different solutions trying to optimize the EAP authentication through a fast key distribution process. Nevertheless, these proposals present common deficiencies that do not allow to achieve a fully optimized re-authentication. First, the *proactive*[2] execution of this solutions imposes resource pre-reservation on the authenticator. This requisite may represent a serious problem in NGNs where authenticators are expected to handle an high number of mobile users at the same time. Second, these solutions always require contacting a backend authentication server located in the AAA infrastructure to carry out the key distribution process. Third, since these proposals define new protocols which are not standardized, their acceptance and success in real deployments is limited.

The contribution of this PhD thesis to this area consists in the definition of a fast re-authentication architecture based on the Kerberos protocol. Kerberos is a well-known secure three-party authentication protocol which has been extensively used to control the access to network resources. In particular, in the context of our solution, the use of

[1] Let us recall that, in the context of EAP-FRM, the key distribution protocol is referred to as *fast re-authentication protocol* (FRP).

[2] The term *proactive* refers to a process that is executed before the handoff. On the contrary, the term *reactive* refers to a process that is executed after the handoff.

Kerberos is adapted to control the user access to the network service. As will be described, the solution results in the definition of a novel framework supporting different modes of operation which allows to significantly reduce the authentication time.

The chapter not only focus on the integration of Kerberos into network access control scenario but also discusses important aspects related to authorization and accounting. Moreover, a representative wireless testbed is developed in order to prove the feasibility of the architecture and to obtain experimental results. Those results are used in simulations to evaluate the performance of the proposed architecture for different deployment scenarios and parameters. Additionally, a mathematical analysis is performed with the objective of computing the authentication delay and validating the simulation results. Performance comparison based on the experiment, simulation and analysis show that the proposed architecture can reduce the authentication time compared to other alternatives in typical deployment scenarios.

4.1 Introduction

In the chapter 3 we have defined a generic architecture for fast re-authentication specially adapted for EAP networks. This architecture can be considered as the basis to transport any key distribution protocol (FRP) without requiring modifications to EAP or existing lower-layer technologies. Nevertheless, the architecture only defines the framework (basically the EAP-FRM standalone method and the corresponding AAA protocol extensions) that permits to carry out an efficient re-authentication process. In other words, EAP-FRM is a generic method to transport any FRP and, consequently, it does not define any specific FRP which, ultimately, is responsible for achieving a fast re-authentication process by introducing a minimal latency during the handoff. Precisely, the objective of this chapter is to define a proper FRP adapted to the EAP-FRM architecture to reduce the authentication time not only compared with traditional EAP authentication but also with existing fast re-authentication protocols. It is important to clarify that the aspect of user privacy is still not considered here. This issue will be covered in next chapter 5.

As surveyed in chapter 2, ERP [68] is a fast re-authentication protocol based on a two-party key distribution model which creates several security issues [96]. 3PFH [16] solves these security weakness by defining a key distribution protocol following a three-party model. Leaving aside aspects related to the specific key distribution model, these fast re-authentication solutions present some drawbacks:

- To re-authenticate the user, a backend authentication server located in the AAA infrastructure needs always to be contacted. Neither ERP nor 3PFH are able to re-authenticate the user with a communication only between peer and authenticator. Instead, in both solutions the authenticator needs help from a backend authentication server during the re-authentication process. This server is located in the network core and is in charge of validating the authentication information presented by the user.

- Definition of new key distribution protocols. The design process of new cryptographic protocols, providing security services such as key distribution, is a complex and error-prone process [75]. A significant number of protocols have been demonstrated to contain security vulnerabilities several years after their publication [139]. This highlights the complexity of the process of verifying that protocols are secure and do not contain undetected flaws. For this reason, operators are reluctant to adopt solutions based on new protocols which have not been extensively deployed and used. This situation causes a limited success of solutions like **3PFH** and **ERP**.

- Resource reservation on the authenticator. The proactive execution of these solutions always requires the reservation of some resources (e.g., the distributed key) in the authenticator. Nevertheless, this requisite constitutes a serious concern in NGNs where authenticators are expected to handle an high number of mobile users at the same time.

To solve all these problems, the second contribution of this PhD thesis proposes the definition of a **FRP** based on Kerberos. Kerberos is a secure three-party key distribution protocol based on the use of shared secret key cryptography destined to control the access to network resources. Since it was originally developed by the *Massachusetts Institute of Technology* (MIT) within the *Athena* project in 1987, Kerberos has grown to become one of the most widely deployed systems for authentication and authorization in modern computer networks. In fact, Kerberos is currently shipped with all major computer operating systems and is uniquely positioned to become a universal solution to control service access.

Although Kerberos has been traditionally used to restrict the access to services such as printers or web servers, it is a general-purpose protocol that can be used with other network services. In particular, we propose the application of Kerberos to control the access to the network service. According to this approximation, we develop a *Kerberized fast re-authentication architecture* where the point of attachment to the network (e.g., an 802.11 access point) acts as a Kerberos *application server* offering network access service. Unlike other proposals using Kerberos during network access control (discussed in next section 4.2), one of the most important novelties of our contribution is to not require modifications to either **EAP** or exiting lower-layer protocols thanks to the use of **EAP-FRM** as transport.

Considering the problem addressed in this PhD thesis, Kerberos presents interesting features that make it an excellent candidate to be applied for achieving fast network access. On the one hand, Kerberos is a lightweight protocol providing a single-sign-on framework based on the so-called *tickets*. The protocol follows a *proactive* operation mode where the process of obtaining a ticket is decoupled from the access to the service. Before accessing a service, users are expected to request tickets that are stored in the so-called *credentials cache*. Only when the user decides to access a service, the ticket is recovered from the credential cache and presented to service. The ticket is used by the service to autonomously authenticate the user in order to decide if the access is granted or denied. That is, the service does not need to contact any backend server (e.g., the KDC) to verify the ticket

presented by the user. Only when access to the service is granted, the service reserves resources for the specific user.

On the other hand, Kerberos is a standardized protocol within the IETF. The first *Request for Comments* (RFC) specifying the Kerberos protocol was published in 1993 [140], obsoleted by a new protocol version in 2005 [74]. In addition to be a protocol extensively accepted within the standardization community, Kerberos has been widely deployed by service providers [141] mainly due to the numerous open source implementations that can be found [142 144]. Its extensive adoption and use throughout years, together with formal security analysis performed to the protocol [76 78], guarantee the security and robustness of the protocol.

Furthermore, Kerberos satisfies the security requirements defined in chapter 1 (section 1.4.3). The protocol achieves an authenticated key distribution process where the user is authenticated only once by using its long-term credentials (*Req. S1*). Additionally, Kerberos can be easily integrated with an authorization system in charge of determining if the user can access to a service and the type of access allowed (*Req. S2*). In fact, protocol messages and tickets can transport authorization data between the different entities. On the other hand, Kerberos implements a robust key distribution process based on strong security principles: first, the distributed key has a precise context that indicates, for example, the entities that are authorized to use the distributed key or the period of time when the distributed key is valid (*Req. S3*); second, by means of both pseudo-random numbers and timestamps, it is assured that the distributed key is fresh (*Req. S4*), and third, the distributed keys are cryptographically independent since they are randomly generated (*Req. S5*). Finally, Kerberos is agnostic to the transport used to deliver its messages between the different entities (*Req. S5*). This independence allows us to transport Kerberos messages using EAP-FRM.

As we can observe, Kerberos is able to solve all the problems previously identified in current fast re-authentication solutions enabling a key distribution process. In the following section we provide an overview of existing work related to the use of Kerberos during network access control. After that, we will present the Kerberos-based fast re-authentication architecture based on the EAP-FRM transport described in chapter 3. Using experimental results extracted from an implemented prototype, we simulate the behaviour of the solution in both intra and inter-domain scenarios. As we will shown, compared with existing solutions, the proposed architecture introduces a minimal latency during network access.

4.2 Related Work

In the literature, there exits a wide set of solutions that have proposed the idea of using Kerberos to control the access to the network service. Nevertheless, as we will discuss, they have not applied the Kerberos authentication protocol in a suitable way that allows an effective reduction on the EAP authentication time. Even more, some solutions require modifications to EAP, link-layer technologies or even to the Kerberos protocol in order to

implement the proposed mechanism.

One of the first proposals in this field is found in [145] where authors propose an authentication architecture for IEEE 802.11 wireless networks called *Wireless Kerberos* (*W-Kerberos*). W-Kerberos is a solution based on the Kerberos authentication protocol and the 802.1X model with the intention of achieving fast and secure handoffs. The W-Kerberos system is composed by three main entities: the client trying to access the network, the access point acting as a Kerberos enabled service offering access to the network and a W-Kerberos server providing authentication and tickets delivery. The W-Kerberos authentication process consists of three main phases. Initially, during the *initial authentication phase*, the client requests to the W-Kerberos server a service ticket (ST) valid for the current access point (AP) through a KRB_AS_REQ/KRB_AS_REP exchange (note that this behaviour deviates from the standard Kerberos operation since STs are delivered by means of the KRB_TGS_REQ/KRB_TGS_REP exchange). After that, the client is finally authenticated after presenting the ST to the AP through an KRB_AP_REQ/KRB_AP_REP exchange. The key shared between client and AP can be renewed during the *key refreshment phase* or distributed to another AP (*handoff phase*) where the user moves to. The latter can be considered as a security context transfer (with associated security problems discussed in [69] and chapter 2) performed through a new Kerberos exchange performed between the new access point and the W-Kerberos server. W-Kerberos present several drawbacks. First, it is only valid for intra-domain handoffs since the solution assumes the existence of an unique W-Kerberos server able to generate STs for any AP. In practice, this assumption is only feasible within a domain where a network operator may deploy a W-Kerberos server controlling all the APs. Second, W-Kerberos requires heavy modifications to both EAP and Kerberos in order to implement the solution. For example, the EAP state machine specification is violated since an *EAP Request/Identity* is not answered with the corresponding *EAP Response/Identity*. Also the acquisition of a ST during the initial KRB_AS_REQ/KRB_AS_REP exchange (instead by using the standard KRB_TGS_REQ/KRB_TGS_REP exchange) or the Kerberos-based context transfer between services (APs) is not allowed in the current Kerberos specification. Third, the implementation of the solution requires the integration of RADIUS server in the EAP authenticators, which is a very rare and unrealistic deployment configuration, apart from adding the need of additional resources in the NASes (e.g. access points) to run the RADIUS server. Finally, the re-authentication process always requires contacting an external server (W-Kerberos server) in order to carry out the security context transfer.

Authors in [146, 147] have also studied the integration of Kerberos as an authentication mechanism for existing EAP-based authentication frameworks. One significative contribution has been the definition of *EAP-Kerberos* [146], a new EAP method specially designed to transport Kerberos messages. The EAP-Kerberos method operates in an authentication framework where an entity called *Kerberized Network Access Server* (KNAS) controls the network access service provided by a set of *Access Nodes* (AN) such as an access point. An EAP-Kerberos execution may consist of an initial KRB_AS_REQ/KRB_AS_REP exchange to acquire an initial TGT and one or more KRB_TGS_REQ/KRB_TGS_REP exchanges to obtain a ST. This ST is finally processed by KNAS that decides if access to

the network is granted to the user through a specific AN. The EAP-Kerberos method is also applied by the same authors in [148] to define an access control solution in vehicular networks.

The same authors refine this solution in [147] where a new approach is proposed that considers the EAP server as a Kerberos service. This new authentication architecture called *EAP-FRAP* (*EAP Fast Re-Authentication Protocol*) is composed of two phases. Initially, during the *bootstrapping phase*, after the successful execution of a typical EAP method (e.g., EAP-TLS), a temporal Kerberos account (Kerberos principal name and password) for the client is created on the local KDC placed in the visited domain by means of a mechanism that is left out of scope. These temporary credentials and the EAP-FRAP extensions are used by the client during the *fast handoff phase*. More precisely, using a new EAP method very similar to EAP-Kerberos, the client executes the standard Kerberos exchanges to obtain a ST that is finally presented to the AAA/EAP server through a KRB_AP_REQ/KRB_AP_REP exchange. In other words, as we can observe the AAA/EAP server is acting as application server. When the EAP server successfully validates the ST, the EAP authentication is completed and an MSK is distributed to the EAP authenticator by using an AAA protocol, thus following a two-party model for key distribution [71]. Despite both works present a network authentication solution fully compatible with current EAP and Kerberos specifications, they do not achieve a fully optimized fast network access since additional signalling is required with a backend authentication server. Assuming the best case, where the user already owns a ST valid for the target service, the authentication process needs up to three exchanges with the backend AAA/EAP server to be completed.

Kerberized Handover Keying (KHK) [149] is a novel key management architecture for fast network access. In KHK, a mobile node and an authenticator act as a Kerberos client or service depending on the operation mode. In *proactive mode*, through the standard KRB_AS_REQ/KRB_AS_REP and KRB_TGS_REQ/KRB_TGS_REP exchanges, the mobile node acquires ST for authenticators in the neighbourhood. Next, when the user roams to one authenticator, the ST is presented by using a KRB_AP_REQ/KRB_AP_REP exchange. Conversely, in *reactive mode*, the roles of client and authenticator are reversed, that is, the mobile node and the authenticator act as Kerberos service and client, respectively. In this case, the authenticator executes a KRB_TGS_REQ/KRB_TGS_REP exchange with the KDC to obtain a ST that is used to authenticate the mobile user through a KRB_AP_REQ/KRB_AP_REP exchange. KHK is an interesting proposal that does not require AAA signalling during the handoff as long as the mobile user proactively obtains ST for the authenticators. Nevertheless, the solution presents several drawbacks. First, it requires modifications to EAP or link-layer protocols to carry Kerberos messages. Second, the solution requires the authenticator to act as Kerberos client in the reactive mode and as a service in the proactive mode, which complicates the implementation in this entity. Finally, the AAA usage in its reactive mode significantly deviates from what has been widely deployed, where authorization needs to take place before authenticating the mobile, which has significant impact on the existing AAA deployment and the authorization model.

Similarly to [148], authors in [150] propose Kerberos to control the access to services offered to users in vehicular scenarios. The network authentication model follows the

reactive model previously described for KHK where the authenticator acts as Kerberos client and the mobile user as Kerberos service. Initially, the authenticator solicits a ST for the mobile user through a KRB_AP_REQ/KRB_AP_REP exchange. This ST is used in a posterior KRB_AS_REQ/KRB_AS_REP exchange where both the mobile user and the authenticator mutually authenticate each other. Nevertheless, as previously mentioned for KHK, this operation mode requires the authorization process to be performed before authenticating the user which significantly deviates from the standard AAA usage in existing deployments.

The application of Kerberos has also been considered in [151] where authors propose an architecture called *EAP Fast Mobile* (EAP-FAMOS which provides not only fast network authentication but also efficient mobility management. EAP-FAMOS defines a network architecture similar to W-Kerberos where a set of authenticators are controlled by a central entity called *Residencial Gateway* (RGW). The fast re-authentication operation consists of two phases. Firstly, during an initial *bootstrapping phase*, the mobile user obtains a TGT after a successful full EAP authentication. This ticket is issued by a *Mobility Broker* (MB), an entity with two major roles within a specific domain: providing the KDC functionality and provisioning location-related information. By using this TGT, the mobile user proactively obtains STs for RGWs controlling authenticators close to the mobile user's location. When the user roams to a new authenticator, the *fast handoff phase* is initiated where access to the network is achieved by presenting a ST (KRB_AP_REQ/KRB_AP_REP exchange) to the RGW controlling the target authenticator. Again, as previously mentioned proposals, EAP-FAMOS do not achieve a fully optimized fast network access since an entity (RGW) located in the core network needs to be contacted.

In conclusion, despite there is a wide set of solutions integrating Kerberos into the network access control scenario, these proposals present important drawbacks. In summary, the problem is that some solutions ([145 147,150,151]) always involve additional signalling (at least two round trips) between the authenticator and the server, which increases latency when providing network access. Additionally, some proposals ([145, 149]) may require modification to EAP or link-layer protocols to carry Kerberos messages.

4.3 Kerberized Architecture for Fast Re-authentication

In order to achieve an efficient fast re-authentication operation and to solve the problems presented by existing solutions, we propose an architecture that integrates Kerberos and associated entities with EAP framework. An overview of this integration, together with the required interfaces, is depicted in Fig. 4.1. The architecture is composed of the following entities:

- The *mobile user*, which integrates the functionality of EAP peer and Kerberos client.
- The *network access server* (NAS), which acts as EAP authenticator and as the

application server in Kerberos. For example, this entity can be an access point or access router. From now on, we refer to this entity as *the authenticator*.

- The *AAA server*, which is responsible for authenticating and authorizing a mobile user. It may contact a Kerberos KDC for Kerberos authentication during a fast re-authentication process.

- The *Key Distribution Center* (KDC), which implements the AS and TGS functionality in Kerberos. It may contact the AAA server to obtain some authorization information for the mobile.

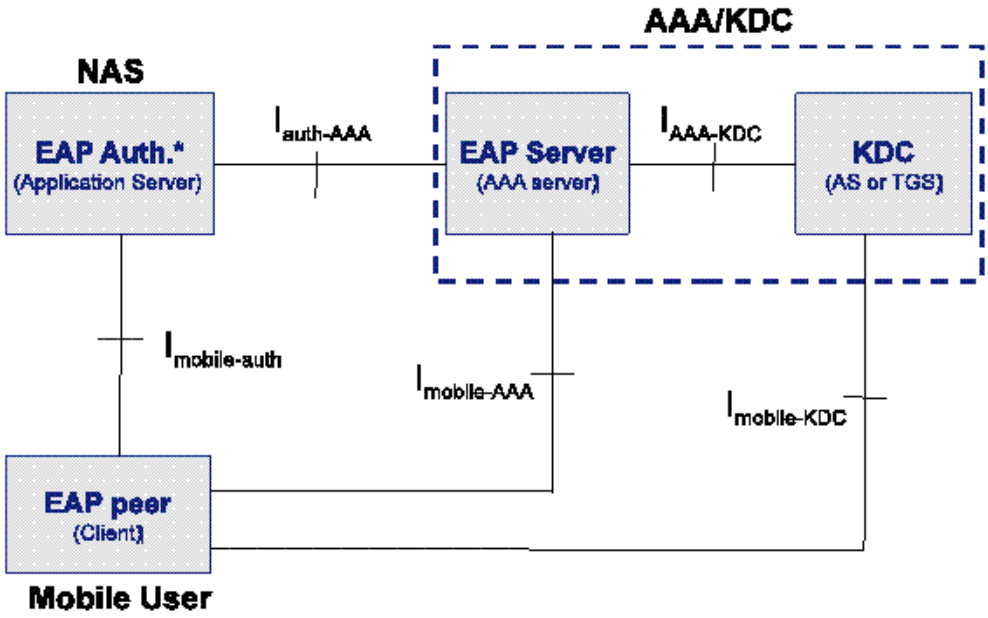


Figure 4.1: Proposed Architecture and Interfaces.

In order to reduce the network access latency, these entities are involved in two different phases: *bootstrapping phase* and *fast re-authentication phase*. During the bootstrapping phase, based on a full EAP execution with the home domain, the user obtains information that is necessary to perform a fast network access during the *fast re-authentication phase*. The specific operation within our architecture, carried out during these phases, requires the entities to interact by means of the following communication interfaces:

- The interface $I_{mobile\ \ auth}$ is implemented through the EAP-FRM method defined in chapter 3, which is adapted to transport Kerberos messages during fast re-authentication operation. In this case, the NAS acts as EAP authenticator in standalone mode.

- The interface $I_{auth\ \ AAA}$ is implemented through an AAA protocol such as RADIUS [36] or Diameter [37].

- The interface $I_{mobile\ AAA}$, destined to assist the bootstrapping process, is implemented by using EAP. In this case, the mobile user acts as EAP peer, the NAS as EAP authenticator in pass-through mode (only forwarding EAP packets between EAP peer and EAP server), and EAP server is located in the AAA server (EAP/AAA server).

- The interface $I_{AAA\ KDC}$ is used to carry out communication between Kerberos KDC and the AAA server. Its purpose is twofold: to deliver Kerberos messages carried in AAA messages to the KDC; and to allow the KDC to contact the AAA server to obtain authorization information for the network access service. This interface can be implemented in several ways: through an AAA protocol or simply UNIX sockets when both entities are in the same machine.

- The interface $I_{mobile\ KDC}$ allows the mobile user to contact directly with the KDC by performing Kerberos exchanges over UDP.

Thus, as we may observe, the architecture assumes the deployment of the Kerberos protocol and related entities (e.g. KDCs) in each participating realm. It also assumes the deployment of an AAA infrastructure in each realm which interacts with the deployed Kerberos entities by means of the defined interfaces.

4.3.1 Bootstrapping Phase

In order to enable the fast re-authentication operation, the mobile user (acting as EAP peer) needs to perform an initial full EAP authentication with the home EAP/AAA server. The authentication involves a *bootstrapping EAP method* with the use of either long-term credentials of the mobile user or the so-called *Transient EAP Keys* (TEKs) [38] derived from the long-term credentials. We assume that the EAP method used for bootstrapping is able to export key material (MSK and EMSK), as recommended by [111] for security reasons. The main objective of this authentication is to establish the key material necessary for Kerberos. More precisely, the *EAP-KRB-KEY* to be shared between the mobile user and the AAA/KDC in the home AAA realm is derived from the EMSK during bootstrapping. This EAP-KRB-KEY is used as the *client's secret key* in Kerberos, used to protect Kerberos messages between the mobile user and the home AAA/KDC. Additionally, during the bootstrapping both the length and the lifetime of EAP-KRB-KEY may also be established.

In order to perform the bootstrapping phase, we have considered two main alternatives:

1. *EAP-EXT* [133] *is used as the bootstrapping EAP method for bootstrapping Kerberos.* The bootstrapping process of EAP-EXT is depicted in Fig. 4.2. EAP-EXT is a tunneling EAP method that encapsulates one or more inner EAP methods *(1)*, and other related information, in TLVs (*Tag-Length-Value* format). After successful completion of one EAP method, other inner EAP methods are protected within the EAP-EXT tunnel. These inner EAP methods, which are transported in *Method* TLVs (i.e. **Method**(Type X)), are assumed to export key material (at

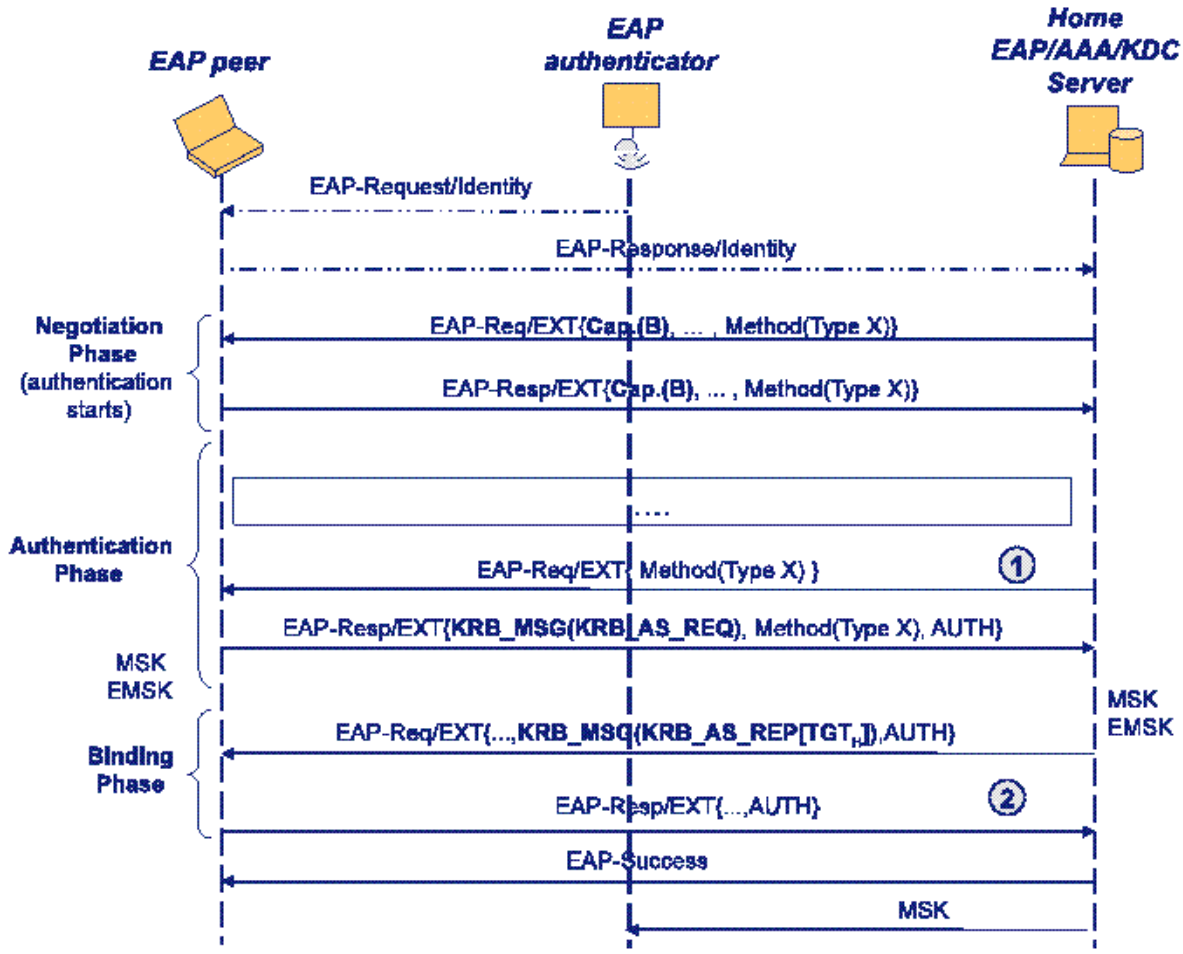


Figure 4.2: Bootstrapping Phase with EAP-EXT.

least the MSK). By using the facilities offered by EAP-EXT to perform a secure exchange of bootstrapping information, in addition to the length and the lifetime of the EAP-KRB-KEY, the authenticator might provide the peer with the following information:

- *Realm and IP addresses.* The realm of the home AAA/KDC and an IP address of the home AAA/KDC can be provided.
- *Synchronization Information.* Given that Kerberos provides freshness to its messages through timestamps, the entities that participate in the protocol execution are required to be loosely synchronized. For this reason, during bootstrapping, the home AAA/KDC may optionally provide timing information to the mobile user for a proper synchronization with the home AAA/KDC.
- *Initial Kerberos TGT.* As depicted in Fig. 4.2, the KRB_AS_REQ/KRB_AS_REP exchange (to obtain a valid TGT TGT_H

for the home AAA/KDC) may be performed within the EAP-EXT method *(2)*. The exchange is transported within the new defined **KRB_MSG** TLVs and is protected by means of the **AUTH** TLV generated with the key material exported by the inner EAP method X.

2. *Any other EAP method which is able to generate EAP key material is used for bootstrapping Kerberos.* In this case, the default EMSK lifetime (i.e., 8 hours [38]) is used as the EAP-KRB-KEY lifetime and other bootstrapping parameters are set with a default value or are provided by using other methods such as IEEE 802.21 Information Service [115]. In this case, the KRB_AS_REQ/KRB_AS_REP exchange is performed over UDP once the EAP method has successfully finished and the mobile user has obtained network access.

Although both alternatives are valid, in the context of this PhD thesis, the first one has been implemented. The main reason is that EAP-EXT has been specifically designed to extend EAP functionality for bootstrapping purposes. Thus, it can be easily extended to provide explicit capability negotiation for bootstrapping Kerberos. In particular, a new bit **Cap(B)** is defined in the negotiation phase for indicating the capability of bootstrapping Kerberos. Moreover, as already mentioned, a new **KRB_MSG** TLV has been specified for carrying the Kerberos messages required to bootstrap a valid TGT for the home AAA/KDC (TGT_H).

Nevertheless, it is important to mention that other EAP methods may also be extended to support a secure capability negotiation for the Kerberized handoff scheme. Typically, tunneled EAP methods [53] may provide this kind of secure channel by establishing a TLS security association between the EAP peer and the EAP server. Through asymmetric cryptography, this tunnel is usually established without authenticating the EAP peer (only server authentication is required).

Finally, it is important to clarify one aspect related to the identity used by the mobile user. During the bootstrapping phase, the mobile user is required to provide its identity in two different situations. First, the authenticator initiates the EAP authentication by sending an *EAP-Request/Identity* to request the mobile user's identity. The mobile user answers with an *EAP-Response/Identity* that contains its identity, represented using the *Network Address Identifier* (NAI) format [152]. On the one hand, the authenticator uses the identity of the user to route the EAP messages (through an AAA protocol) to the correct AAA server of the mobile node's domain. On the other hand, the EAP server selects a proper bootstrapping EAP method based on the mobile node's identity. Second, in the initial Kerberos authentication exchange (KRB_AS_REQ/KRB_AS_REP), the mobile node's must indicate its identity in the *cname* field of the KRB_AS_REQ message. The same field is used by the KDC in the KRB_AS_REP message to confirm the mobile node's identity.

Initially, it is assumed that the same identity is used by the AAA and KDC servers to identify the user in the EAP authentication and Kerberos initial authentication, respectively. Nevertheless, different identities may be employed if this option better satisfies

the network operator's needs. Finally, we would like to highlight that, in both processes, the mobile node's identity is not protected, being transmitted in cleartext. Thus, it is quite easy for an external observer to determine which specific user is performing a bootstrapping process, creating some privacy issues which will be tackled in next chapter 5.

4.3.2 Fast Re-authentication Phase

Once the mobile has finished the bootstrapping phase, it can change its point of attachment to a new authenticator in the *fast re-authentication phase*. In this phase we propose a fast re-authentication solution based on the use of Kerberos.

The architecture involves all the entities defined in Kerberos. On the one hand, authenticators act as Kerberos application servers, offering network access service to the clients. On the other, the mobile users act as Kerberos clients, interested in obtaining connectivity through the authenticators. For this reason, every operator is expected to deploy at least one KDC to administer its subscribers and authenticators installed in the operator's access networks. Without loss of generality, we assume that the KDC is co-located with the AAA server in the same physical entity.

When the mobile user starts the fast re-authentication phase, we assume that it has at least one TGT obtained from its home KDC (TGT_H) during the bootstrapping phase. Based on this TGT or on others stored in the mobile user's credential cache, the mobile user can request a valid ST (ST_{auth}) for the target authenticator during the fast re-authentication phase. To achieve this goal, we employ the standardized Kerberos exchanges (see section 2.3). Depending on the type of credentials held by the mobile user and the moment as to when the ST_{auth} is acquired, we distinguish between proactive mode and reactive mode. Next section 4.3.3 provides details of both.

4.3.3 Carrying Kerberos in EAP

In order to achieve a fully compatible solution with the current standards, the Kerberos transport should not require modifications to EAP or existing wireless technologies. Moreover it should not add any significant additional latency to the authentication process.

The generic framework presented in chapter 3 fully satisfies these requirements. As the reader may recall, the generic fast re-authentication architecture is based on a new EAP method called EAP-FRM, which is able to transport a PDU of any fast re-authentication protocol. Moreover, EAP-FRM works in standalone authenticator mode which, if necessary, can communicate with a backend AAA server by using an AAA protocol for verification of the fast re-authentication information originated by the mobile user and encapsulated in EAP-FRM messages.

In this work we use EAP-FRM to transport Kerberos as the fast re-authentication protocol in EAP-FRM. As we will see in the following sections, it can be used in both proactive and reactive modes.

Single-Realm Proactive Mode

During a single-realm proactive handoff (movements under the same AAA/KDC), the proactive mode assumes that the mobile user already owns a valid ST_{auth} and the *service session key* for accessing the authenticator to which the mobile user is moving. Both ST_{auth} and service session key could be already stored in the kerberos tickets cache maintained by the mobile user; or they could be obtained through an KRB_TGS_REQ/KRB_TGS_REP exchange before the mobile user moves to the new authenticator. In fact, the mobile user may perform several of these exchanges to obtain different ST_{auth}'s for several authenticators *before* the movement.

In this manner, when the mobile user moves to the new authenticator, it only has to send the ST_{auth} to the authenticator to perform network access authentication. Specifically, the mobile user sends the KRB_AP_REQ message which contains the ST_{auth} and other useful information for proving that the message is originated by the correct entity that the ST_{auth} was issued to, it was recently generated and it is not a replay. Basically, this information consists of an authentication tag encrypted with the service session key and which includes a timestamp such as described in the standard Kerberos [74].

In order to reduce the network access latency further, we allow the authenticator not to reply with a KRB_AP_REP message, as Kerberos specification permits [74]. This does not introduce any security issue as long as a secure association protocol (e.g. *4-way handshake* in IEEE 802.11) is performed after successful EAP authentication. In any case, this is the most common case to avoid further security problems [38]. In this way, once the authenticator successfully validates the ST_{auth} sent by the mobile user, it directly answers with an *EAP-Success*, finalizing the EAP authentication and initiating a security association protocol.

Figure 4.3 depicts how the proactive authentication mode works using the EAP-FRM transport. As observed, the authenticator starts EAP-FRM by sending an *EAP-Request/FRM* message (*2*) to the mobile user containing a nonce ($Nonce[N_A]$) generated by the authenticator (that will be used for key derivation purposes within EAP-FRM) and an *Auth-Server TLV* indicating the authentication server (*server@realm*) which controls the target authenticator. Typically, this first message is sent by the authenticator thanks to some kind of trigger (*1*) that tells the EAP authenticator to start the authentication. This trigger is out of scope of EAP and, for example, the specific trigger may depend on the underlying technology. An example is the *EAPOL-Start* message defined in IEEE 802.1X [34].

On the reception of this initial EAP-FRM message, if the mobile user does not support EAP-FRM or it has not yet performed the bootstrapping phase, it sends an *EAP-Response/Nak* message. This may cause that the authenticator to send *EAP-Request/Id* in order to start a traditional EAP authentication process. On the contrary, the mobile node responds with *EAP-Response/FRM* (*3*) which contains a nonce generated by the mobile node ($Nonce[N_S]$) and a *FRP-Payload TLV* that transports the KRB_AP_REQ message.

When the authenticator successfully validates the ST and the authentication tag present

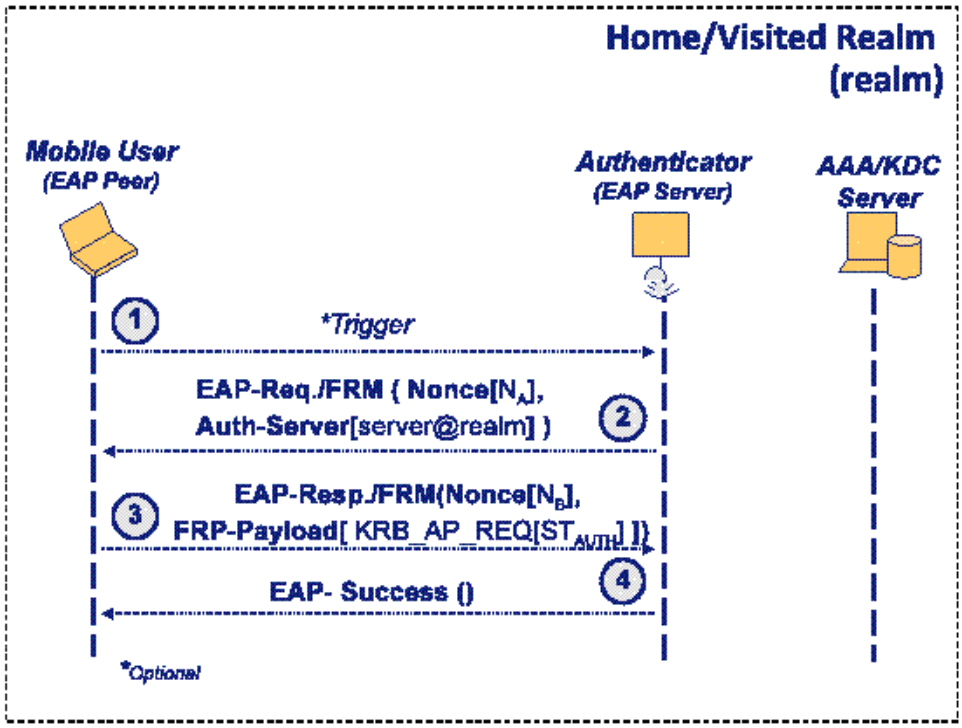


Figure 4.3: Single-Realm/Cross-Realm Proactive Mode

in the KRB_AP_REQ message, the fast re-authentication protocol execution finishes and the *service session key* distributed by Kerberos to both the peer and the authenticator is provided to the EAP-FRM method. This key is considered as the rMSK described in chapter 3 and used as root key to construct the key hierarchy generated within EAP-FRM. Once the EAP-FRM method execution ends, the EAP authentication is completed with a final *EAP-Success* message (*4*) sent to the peer.

Single-Realm Reactive Mode

Unlike the proactive mode, the reactive mode is useful when the mobile user attaches an authenticator without having a valid ST_{auth} for that authenticator, but owning a TGT (TGT_V) with the KDC which provides STs for that authenticator (and others in the vicinity). For this reason, as we can see in Fig. 4.4, the main part is related to the ST_{auth} acquisition (*1*). If the mobile user owns a TGT (TGT_V) with the KDC which provides STs for that authenticator (and others in the vicinity), the ST_{auth} can be acquired through a single KRB_TGS_REQ/KRB_TGS_REP exchange with that KDC. The communication between the mobile user and the authenticator is based on EAP-FRM; and between authenticator and AAA/KDC it takes place through an AAA protocol. Without loss of generality, we select RADIUS to describe the process in Fig. 4.4. Once the mobile user has the ST_{auth} for the authenticator, it sends it to the authenticator via EAP-FRM (*2*).

The EAP-FRM TLVs usage in the reactive mode is the same as previously explained

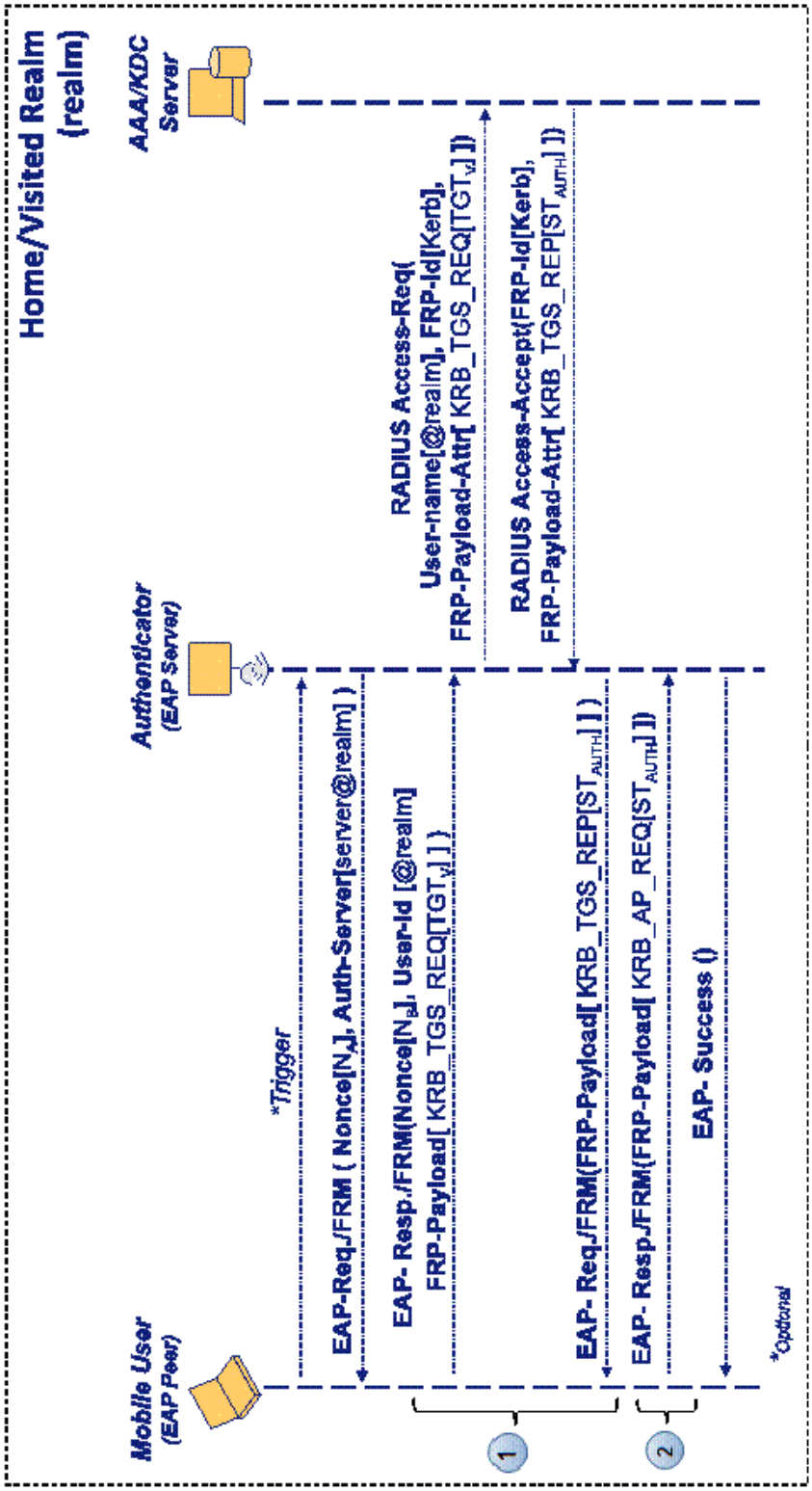


Figure 4.4: Single-Realm Reactive Mode

for the proactive mode. Now, since the authenticator needs to contact the AAA/KDC server, a AAA protocol (RADIUS) is used for this purpose. In particular, the *FRP-Id* and *FRP-Payload-Attr* RADIUS attributes are used to indicate the fast re-authentication protocol type (*Kerberos*) and transport the fast re-authentication protocol payload, respectively. For simplicity, the *FRP-Flags* attribute is omitted since, in this example, no flags are used to indicate special conditions.

In the reactive mode, one important aspect is related to how the message KRB_TGS_REQ transported in the AAA protocol from the authenticator is routed to the correct AAA/KDC. In other words, the AAA infrastructure should be able to route the AAA message containing the KRB_TGS_REQ properly to the particular KDC which is able to process the TGT_V. To achieve this, we have used a similar idea to that presented in [153][3]: each AAA/KDC server defines its own subdomain where the single entity in the domain is the AAA/KDC server itself. Let us illustrate this with an example. In the general domain *um.es*, there might be several AAA/KDC servers. Let us suppose the AAA/KDC server *kdc1* is the KDC that can process the TGT. The KDC server *kdc1* is placed on the subdomain *kdc1.um.es*. Only that AAA/KDC server is in that subdomain.

Following this approach, and based on the guidelines of the EAP specification [33] regarding the identity contained in the *EAP-Response/Identity*, we have used the *User-Id* TLV for routing purposes to indicate the domain to which a KRB_TGS_REQ must be routed. More precisely, as observed in Fig. 4.4, the *User-Id* TLV contains an abbreviated identity where: (a) the username portion of the NAI has been omitted and; (b) the domain name part indicates the domain where the content of the *FRP-Payload* should be routed. The content of the *User-Id* TLV is set by the mobile user and the information is used by the authenticator as the routing AAA information (*User-name* RADIUS attribute) when sending an AAA request. It is worth noting that this subdomain information may be provided by the AAA/KDC during the *bootstrapping phase* or through a discovery mechanism, as discussed in section 4.3.4. In this way, institutions are free to select this subdomain name. Therefore, despite the user's identity is not transmitted at EAP-FRM level within the *User-Id* TLV, this information is contained in the TGT_V (*cname* field) sent within the KRB_TGS_REQ message.

Cross-Realm Proactive Mode

As in the single-realm proactive mode, the cross-realm proactive mode assumes the mobile user has obtained an ST_{auth} before the movement to the target authenticator. Thus, the sequence of messages is equal to that one shown in Fig. 4.3. However, that implies obtaining a TGT_V for the KDC in the visited realm that provides STs for the authenticators before the movement. Assuming that the mobile user at least owns a TGT_H with its home AAA/KDC, several cross-realm exchanges KRB_TGS_REQ/KRB_TGS_REP may be required until the TGT_V is obtained. Fortunately, this operation can be automatically started immediately the mobile user gets network access. Once the mobile user owns

[3]In this reference, the term *realm* and *domain* are used interchangeably

the TGT_V, it can obtain one or several STs beforehand for different authenticators with additional KRB_TGS_REQ/KRB_TGS_REP exchanges with the visited AAA/KDC.

Cross-Realm Reactive Mode

In the single-realm scenario, we assumed at least that the mobile user already owns a TGT with one KDC in the realm. In the cross-realm reactive mode, it is desirable to maintain the same assumption with the TGT_V for the KDC in the visited realm. The reason is this assumption represents a best case in a cross-realm reactive mode and, therefore, a faster operation is expected in the reactive mode. Taking into account the Kerberos operation (see section 2.3), this means requesting beforehand a TGT (TGT_V) for the new KDC. In order to keep this assumption valid, it is necessary to discover a new KDC in the mobile user's neighborhood. This request will trigger the required (if any) cross-realm KRB_TGS_REQ/KRB_TGS_REP exchanges to obtain the TGT_V beforehand.

However, this general assumption may fail in some specific cases; for example, the mobile user may move unexpectedly to an authenticator under a KDC with which the mobile user does not own a valid TGT_V. In this case, different cases are possible:

1. The mobile user does not own a valid TGT_H.

2. The mobile user owns a valid TGT_H.

In case *1*, a regular EAP authentication by using long-term credentials is required. In case *2*, the mobile user may perform a Kerberos *cross-realm* operation in order to get a valid TGT_V for the KDC in the visited realm by using, as initial TGT, the TGT_H with the home AAA/KDC. This process is shown in Fig. 4.5. As we may observe, the KRB_TGS_REQ/KRB_TGS_REP exchanges are performed through the authenticator. They are transported by EAP-FRM between the mobile user and the authenticator and, using an AAA protocol, between the authenticator and the AAA/KDCs involved in the process. Every KRB_TGS_REQ message is routed to the correct KDC thanks to the information contained in the *User-Id* TLV that allows to specify the domain to which the information contained in the *FRP-Payload TLV* should be routed. The EAP-FRM implementation of the peer can easily know the correct domain by, for example, examining the value contained in the *server* field of the KRB_TGS_REQ message.

Another alternative for case *2* is to perform a bootstrapping operation, as described in section 4.3.1 which involves a full EAP authentication with the home AAA. Depending on the number of realms between the visited AAA/KDC and the home AAA/KDC and the potential EAP method used to complete the bootstrapping, the use of cross-realm operation will be more or less beneficial than performing a bootstrapping phase. Since the mobile user would require additional information to determine which option is better (e.g. it may need to know the number of realms between visited KDC and home AAA/KDC or the number of round trips involved in the bootstrapping EAP method), we establish by default the policy of performing a bootstrapping phase in case *2*. Nevertheless, in

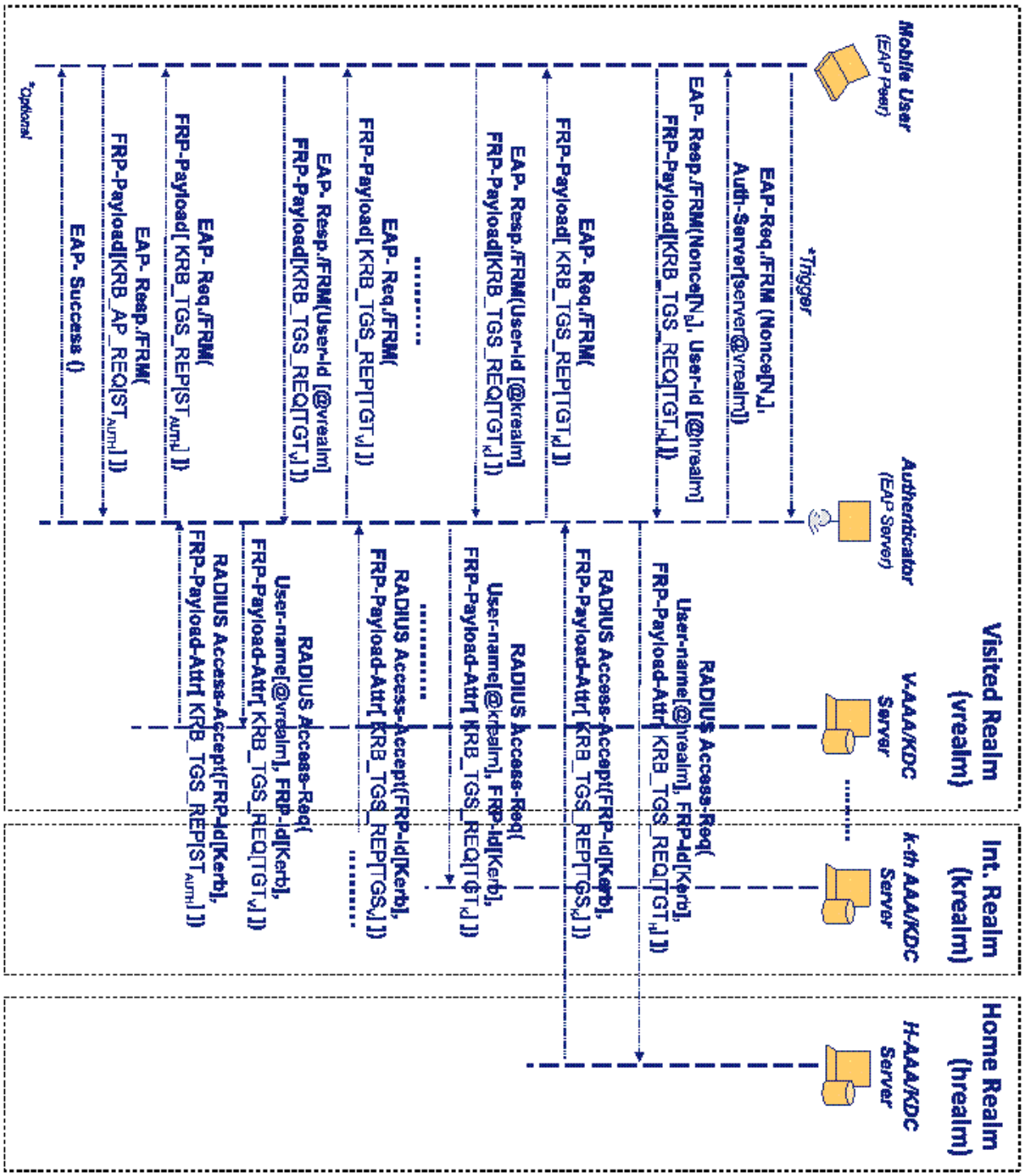


Figure 4.5: Cross-Realm Reactive Mode

section 4.6.3, we analyze the impact of the number of realms in the decision to choose either a bootstrapping operation or a cross-realm operation.

In general, these cases introduce undesired additional latency and must be avoided as far as possible. An effective mechanism for network discovery that allows the discovery of authenticators and KDCs in the neighborhood of the mobile user is of vital importance to reduce the impact of these cases.

4.3.4 Authenticator and KDC Discovery

In the (single and cross-realm) proactive mode, the mobile user needs to discover authenticators in neighboring networks. Once the mobile user knows these authenticators, it can start to request STs for the authenticators through KRB_TGS_REQ/KRB_TGS_REP exchanges before attaching to them.

In cross-realm reactive mode, it has been outlined the importance of knowing the KDCs in the neighborhood of the mobile user. This basically means discovering an authenticator in the neighborhood that is under a KDC with which the mobile user does not own a valid TGT. If the mobile user detects this KDC, it can start the process to obtain a TGT (TGT_V) for that KDC. Taking this into account, a network discovery mechanism needs to provide at least the following information.

- The identity of the authenticator, including the realm, so that the mobile user can identify it to obtain a ticket for that authenticator from the KDC of the realm.
- A link-layer or IP address of the authenticator so that the mobile user can communicate with the authenticator for a KRB_AP_REQ message.
- Certain authenticator capabilities; for example, if it supports the EAP-FRM method and Kerberos based fast re-authentication.
- The identity of the KDC that provides STs for a particular authenticator. In this case, the mobile user can discover whether it already has a TGT for the KDC or not.
- The IP address of the KDC that provides STs for a particular authenticator. This is useful for the mobile user to contact the KDC through the interface $I_{mobile\ \ KDC}$.
- Whether the visited realm has a roaming agreement with the mobile user's home realm. If this agreement is not established, the mobile will not be granted access to the network service in the visited realm so that performing the handoff is useless.

To be independent of the particular wireless technology in use, a media-independent authenticator discovery mechanism is highly demanded to provide all pieces of information described above. For example, IEEE 802.21 *Information Service* [115] is considered to be such a mechanism because it provides various pieces of information on neighboring networks to facilitate handoff decision making process and is designed to work over any media. The details of how the general information detailed above is translated into the IEEE 802.21 is a motive of future work.

4.3.5 Authorization Aspects

So far, we have focused on the authentication aspect. Yet, authorization is also important and must be performed after authenticating the peer. However, Kerberos does not explicitly deal with authorization [74].

In Kerberos, certain authorization information can be carried in tickets within the *authorization-data* field. However, Kerberos does not define an explicit format for this information, leaving this choice to the developer of a concrete kerberized application. To authorize network access service, we have used this feature to include certain authorization information within the ST_{auth}. This authorization information is included in the ST_{auth} by the KDC when it receives a KRB_TGS_REQ from the mobile user. Since Kerberos itself does not deal with authorization, our architecture enables the KDC to request this information to the AAA through the interface I_{AAA_KDC}. This communication is based on the specific AAA protocol implemented by the AAA server. For example, assuming RADIUS, the KDC obtains the authorization data from the AAA server in a single *RADIUS Access-Request/RADIUS Access-Accept* exchange.

When the mobile user sends a KRB_AP_REQ, the authorization information contained in the ST_{auth} will help the authenticator to take the decision to provide or not network access. In fact, after verifying the ST_{auth}, the authenticator will verify the authorization information contained in the ST_{auth}. Considering that authenticators are NASes on which AAA clients are also implemented, we express this authorization information in terms of either RADIUS attributes or Diameter AVPs. The specific format used for the authorization information depends on the AAA protocol used by the authenticator to communicate with the AAA server. For example, if the authenticator is an IEEE 802.11 access point implementing a RADIUS client to communicate with a RADIUS server, RADIUS attributes shall be included in the ticket. As observed, this approach allows reuse of existing AAA client software code to process and enforce these attributes, which gives, a substantial advantage from an implementation perspective.

In proactive mode, the authenticator can verify the ST_{auth} sent by the mobile user without communicating with any AAA/KDC server. This avoids any signalling from the authenticator to the AAA/KDC server and, thus, reduces latency. However, to keep this advantage, the authenticator must also be able to perform the authorization process without contacting any external backend server. Providing authorization information in the ST_{auth} achieves this objective. On the other hand, the proactively generated ST_{auth} may not be used immediately (or it may not even be used at all). Therefore, when the client finally connects to the authenticator and uses the ST_{auth}, the authorization information contained in the ST_{auth} might be considered as invalid (e.g., the general authorization policy has changed since the ST_{auth} was issued). In this case, the authenticator still may decide to interact with the AAA infrastructure to obtain the latest authorization information.

In reactive mode, the authenticator always contacts with the backend AAA infrastructure and, hence, it obtains always recent authorization information. Note that once the mobile user obtains the ST_{auth}, the proactive mode is used for subsequent attachments to the same authenticator during the ST_{auth} lifetime.

In cross-realm operations, each realm may wish to apply different authorization policies to the authorization information generated by the home realm and included in the TGT_H. To do this, each KDC on the Kerberos authentication path may modify the authorization data contained in a cross-realm TGT received from the mobile user and generate, as a consequence, a cross-realm TGT for the next KDC hop with the modified authorization data. As a result, the authorization information introduced in the TGT_V is the authorization information applicable to the mobile user in the visited realm and may differ from the one contained in the TGT_H.

The format of that authorization information contained in the TGTs does not need to be interpreted by the mobile user or the authenticator and can be expressed with higher level authorization languages such as SAML [154], though, the particular format is subject for future work.

4.3.6 Accounting Aspects

Accounting is started by the authenticator through an AAA protocol once the mobile user has obtained network access. However, it is necessary to associate accounting records with the appropriate authorization session of the mobile user. In reactive mode, the authenticator must communicate with the AAA/KDC to perform Kerberos operations after the mobile user attaches to it. This allows the authenticator to initiate the authorization session and relate it with the subsequent accounting messages. However, in the proactive mode, there is no AAA signalling for authorization (see section 4.3.5). To solve this problem, we include a simple authorization session identifier in the authorization part of the distributed ST. With this information, the authenticator can start accounting signalling with the AAA/KDC server by including this authorization session identifier, extracted from the ST_{auth}.

4.4 Deployed Wireless Testbed

4.4.1 Deployed testbed

A wireless testbed prototype has been developed in order to accomplish two main objectives: to prove the feasibility of the architecture and to obtain experimental results for the bootstrapping and fast re-authentication phases (proactive and reactive modes) described in sections 4.3.1 and 4.3.2. At the same time, the experimental values will be used as the input parameters for the simulations described in section 4.6.2.

Figure 4.6 shows the architecture of the wireless testbed where IEEE 802.11 has been used as representative wireless technology. Access points implement the IEEE 802.1X authentication model for EAP authentication. Moreover, the *4-way handshake* secure association protocol is enabled in the IEEE 802.11 wireless link to improve security.

The scenario is composed of three institutions, two acting as visited institutions and the other as the mobile user's home institution. Each institution has a different number

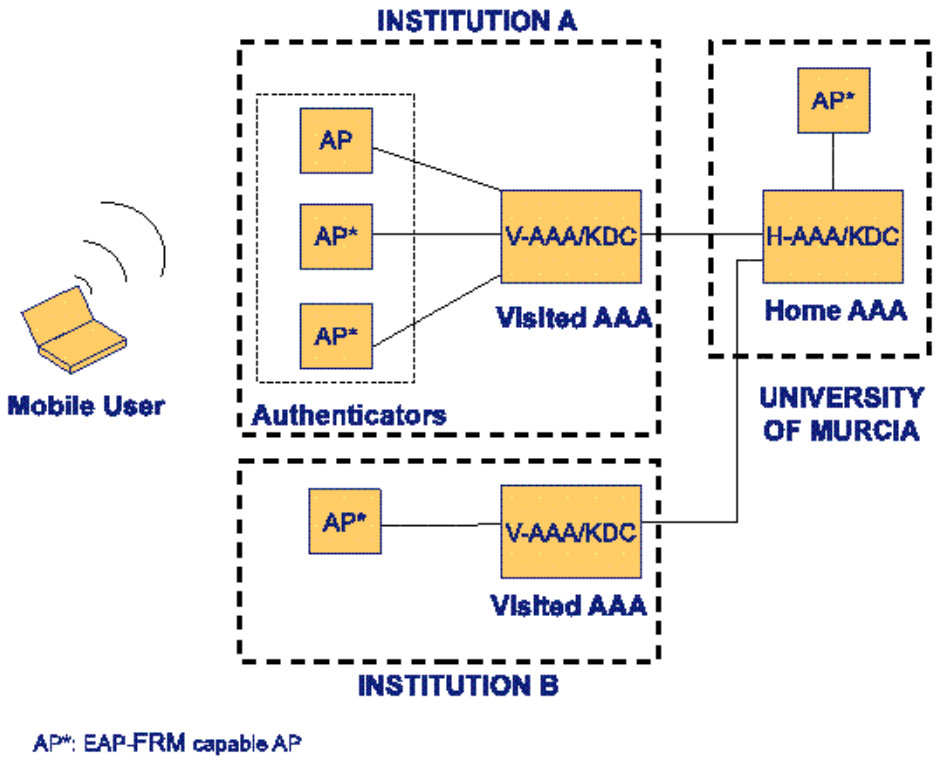


Figure 4.6: Deployed Wireless Testbed.

of access points which may be EAP-FRM capable. For example at institution A, we have used two PCs that play the roles of authenticators acting as EAP-FRM capable access points (AP*'s). The AP*'s deploy the open-source software *hostapd* (version 0.6.4) [155] that implements the EAP authenticator functionality and a RADIUS client. We have also deployed a linksys access point (AP) which does not support EAP-FRM but can be used for bootstrapping operations. Finally, three PCs deploy the open-source software *FreeRadius* (version 2.1.3) [137] to implement RADIUS server functionality and *MIT's Kerberos implementation* (version 1.6.3) [142] for Kerberos operations. Two of the PCs acts as a visited RADIUS/KDC server (*V-AAA/KDC*) in institution A and B, respectively. Finally, the third (*H-AAA/KDC*) acts as the RADIUS/KDC in the mobile user's home institution.

Machine	CPU Type	Freq(MHz)	RAM(MB)
Peer/AP*	VIA Nehemiah	1,200	483.376
L-AAA	Pentium Dual	2,000	2058.052
H-AAA	Pentium 4	3,000	516.572

Table 4.1: Testbed machines

The mobile user deploys the open-source software *wpa_supplicant* (version 0.6.4) [155]

that implements the EAP peer functionality. The mobile user also uses the same *MIT's Kerberos implementation* (version 1.6.3) for Kerberos operations. Details of the features of each machine are shown in Table 5.2. A summary of the most relevant MIT's Kerberos routines and RADIUS API routines used in the testbed implementation is provided in Table 4.2.

4.4.2 Implementation Details

This section provides some important implementation details about how the most important component of the architecture have been developed.

Kerberos

As described in section 4.3, the design of the fast re-authentication architecture relies on the Kerberos protocol. For this reason, one important objective has been to develop an engine which is able to provide the required Kerberos functionality to the experimental testbed. To achieve this, the open-source MIT Kerberos implementation [142] has been selected.

The source code of the MIT Kerberos implementation has a layered structure [156]. There is an external layer providing high-level functionalities to control the Kerberos protocol behavior, e.g., authentication against a KDC, service ticket management, access to a specific service or manipulation of the credentials cache. This external layer is accessible from Kerberos applications. On the other hand, there is an internal layer realizing low-level Kerberos functionalities. This internal layer is accessible only from the external layer and is not accessible from Kerberos applications.

However, the external layer is designed to work with the presence of IP network connectivity between the Kerberos entities through TCP/UDP sockets. Since, in certain circumstances, the fast re-authentication application may need to work without the presence of IP network connectivity (e.g. the mobile user connects to a new authenticator that requires a successful authentication before providing network access), modifications to the external layer of the MIT Kerberos source code are required to work under this condition.

As Table 4.2 shows, the external layer has been modified by creating new routines (*extended Kerberos API*) and extending an existing routine as follows. First, four new routines have been added (*krb5_build_*$\{TGS,AS\}REQ$, *krb5_process_and_check_*$\{TGS,AS\}REP$) to access Kerberos messages formatted in byte string and perform its validation, leaving apart the transport of these messages. This routine is only necessary for the KRB_AS_REQ/KRB_AS_REP and KRB_TGS_REQ/KRB_TGS_REP exchanges. Second, an existing routine (*krb5_get_credentials*) has been modified to handle the authorization information contained in the *AuthorizationData* field of a *ST*.

EAP-EXT

The EAP-EXT method has been implemented to perform bootstrapping operations as described in section 4.3.1. In the first place, we have developed EAP-EXT method in *wpa_supplicant* (EAP peer part) and *FreeRadius* (EAP server part) for bootstrapping purposes. For testing, EAP-TLS implementation has been integrated in order to use it as the inner EAP method of EAP-EXT. Additionally, the *MIT's Kerberos implementation* has been also integrated within EAP-EXT, following the signalling in Fig. 4.2.

Note that in the server side of EAP-EXT, implementation integrates the communication with a KDC server in order to perform the bootstrapping phase and retrieve the necessary data from the KDC server. To achieve this communication between RADIUS server and KDC server, *MIT's Kerberos implementation* has been integrated in our RADIUS server. Kerberos routine *krb5_sendto_kdc* is used to perform the communication with KDC to retrieve the Kerberos TGT ticket.

On the mobile user side, Kerberos is used differently. The EAP-EXT implementation adds Kerberos functionality to build the KRB_AS_REQ (*krb5_build_ASREQ*) and process the KRB_AS_REP (*krb5_process_and_check_ASREP*). After a successful execution of EAP-EXT and subsequent secure association at the link-layer, a program has been developed to obtain a ST_{auth} for the attached authenticator. This program makes use of the original Kerberos API's routine *krb5_get_credentials* to obtain ST_{auth}.

EAP-FRM

The same EAP-FRM implementation developed in chapter 3 has been re-used here. As the reader may recall, the mobile user and authenticator implementation of EAP-FRM are based on *wpa_supplicant* and *hostapd*, respectively. Since EAP-FRM is designed to operate in standalone mode, the EAP server for EAP-FRM is implemented on the authenticator with *hostapd*.

A set of functions provided by the *MIT's Kerberos implementation* are used to support both proactive and reactive modes. For reactive mode, *krb5_is_tgs_req* and *krb5_is_ap_req* functions are used by the authenticator for determining whether the payload is a KRB_TGS_REQ and a KRB_AP_REQ, respectively. For proactive mode, *krb5_rd_req* is used for validating KRB_AP_REQ message including the ST.

RADIUS implementation

For reactive mode it is necessary to extend the RADIUS implementation to manage the Kerberos data. Firstly, in the authenticator, the EAP-FRM implementation interacts with a radius client implementation by calling the API provided by *hostapd*. The API has only been used to manage the creation of the RADIUS messages. The API functions used have been *radius_msg_new* to create a new RADIUS message, *radius_msg_make_authenticator* for adding the shared key between a RADIUS client and a RADIUS server, and *radius_msg_add_attr* to add attributes in the message. In particular, the RADIUS attribute *FRP-Payload-Attr* is used to transport the Kerberos data between the authenticator and

			Kerberos API		**RADIUS API**
	Interface	*Implementation*	*Original*	*Extended*	
Mobile User	mobile-auth	EAP-FRM	krb5_get_credentials krb5_mk_req_extended krb5_mk_rep	krb5_build_TGSREQ krb5_process_and_check_TGSREP	X
	mobile-aaa	EAP-EXT	X	krb5_build_ASREQ krb5_process_and_check_ASREP	X
	mobile-kdc	Kerberos/UDP	X	krb5_get_credentials	X
Auth.	mobile-auth	EAP-FRM	krb5_rd_req krb5_mk_rep krb5_is_tgs_req krb5_is_ap_req	X	X
	auth-aaa	RADIUS	X	X	radius_msg_new radius_msg_make_authenticator radius_msg_add_attr
Server	mobile-aaa	EAP-EXT	X	X	X
	auth-aaa	RADIUS	X	X	X
	aaa-kdc	Kerberos/UDP	krb5_sendto_kdc	X	X
KDC	mobile-kdc	Kerberos/UDP	X	X	X
	aaa-kdc	RADIUS	X	X	radius_msg_new radius_msg_make_authenticator radius_msg_add_attr

Table 4.2: Summary of main functions used in the implemented testbed.

the RADIUS server. To manage the Kerberos data, a new authentication module has been developed in FreeRadius. The implementation in the RADIUS server makes use of the interface I_{AAA_KDC} (depicted in Figure 4.1) for communicating with the KDC server using the *MIT's Kerberos implementation* routine *krb5_sendto_kdc* which communicates with the KDC through an UDP socket.

4.5 Performance Analysis

In this section, a mathematical analysis is performed to compute approximate authentication delay used for performance evaluation. The following notation is used. Let n denote the number of authentication messages exchanged between the mobile user and authenticator. Let p denote the number of messages between the authenticator and the AAA infrastructure. Let denote r the number of messages exchanged between the mobile user and authenticator when performing a secure association protocol. Let h denote the number of AAA/KDCs over the authentication path in the AAA infrastructure. Let $m_{1,i}$ and $a_{1,i}$ denote the processing delay spent in each authentication message i by the mobile user and the authenticator, respectively. Let D denote the propagation delay between the mobile user and the authenticator. Let $a_{2,j}$ and s_j^k denote the processing delay spent for message j by the authenticator and the k-th AAA/KDC, respectively. Let $D_j^{k\ 1,k}$ denote the propagation delay for any message j between the $(k-1)$-st and the k-th AAA/KDC for $k > 2$, and the propagation delay between the authenticator and the first KDC for $k = 1$, where the first KDC denotes the KDC in the visited realm and the h-th KDC denotes the KDC in the home realm. Let $m_{3,p}$ and $a_{3,p}$ denote the processing delay spent for message p of the *secure association protocol* (SAP) by the mobile user and authenticator, respectively.

In the analysis, it is considered for simplicity that transmission and queueing delays are negligible. It is assumed that Kerberos messages are transported over AAA protocol between each AAA/KDC in the path between the authenticator and the home AAA/KDC (h-th AAA/KDC). Note that different AAA/KDCs may be involved in different AAA messages depending on the target KDC of the Kerberos message carried in the AAA message.

Then, the authentication latency can be computed as follows:

$$\begin{aligned} T_{auth_{total}} &= \sum_{i=1}^{n}(m_{1,i} + a_{1,i} + D) \\ &+ \sum_{j=1}^{p}(a_{2,j} + \sum_{k=1}^{h}(s_j^k + D_j^{k\ 1,k})) \\ &+ \sum_{q=1}^{r}(m_{2,q} + a_{3,q} + D) \end{aligned} \tag{4.1}$$

The *first term* of the equation shows the time spent for the authentication message

exchanges between the mobile user and the authenticator. The *second term* shows the time spent for the AAA message exchanges. It is assumed $D_j^{k\ 1,k} = 0$ when message j is not exchanged between the KDC $k-1$ and k; and $s_j^k = 0$ when k-th KDC is not involved in processing message j. The *third term* shows the time spent for the execution of the SAP. As is evident, this term will be 0 when no SAP is performed.

4.6 Performance Evaluation

In order to obtain accurate performance evaluation, we have taken experimental data by measuring processing delays of authentication messages and the SAP messages per each participant entity in the implemented testbed described in section 4.4. Second, simulation results are taken to evaluate different sets of parameter values and more complex network configurations, by including in the simulation the experimental processing time results obtained from the testbed. Third, analytic results have been computed from the formula provided in section 4.5 to validate the simulation results. Note that a specific set of n and p values is determined for each authentication scheme and used for the formula and simulations.

4.6.1 Measurement

To develop more realistic simulations, several experiments have been conducted over the implemented wireless testbed described in section 4.4, in order to obtain the authentication delay component in each protocol entity. To achieve this goal, several experiments have been performed with fast EAP-TLS [54], EAP-GPSK [125] and the proposed Kerberized EAP-FRM approach (assuming $h = 2$).

It is important to mention that since EAP-FRAP implementation [157] uses a different version of software, we have approximated EAP-FRAP processing times in each entity from the results obtained from the Kerberized EAP-FRM implementation. In particular, these values have been obtained taking into account the processing delays of different Kerberos exchanges in our implementation.

Around 200 authentication tests have been carried out involving handoffs of a mobile user within the visited realm and the home realm. Through the *Wireshark* tool [138] and information obtained from the source code, it has been obtained the time that each node spends in processing the authentication messages.

4.6.2 Simulation

Given the complexity of a real deployment scenario, simulation has been chosen as an alternative way to compare the performance of our architecture with other fast re-authentication proposals. In this way, we have developed a simulation scenario based on IEEE 802.11 as representative wireless technology. The version 2.29 of the *ns-2 Network Simulator* [158] is used as simulation tool, by using NIST mobility package [159] to simulate

802.11 infrastructure mode. The simulation model consists of several wireless access points (APs) acting as EAP authenticators distributed over a square area of 210×210 m^2. Over this area, five mobile users will perform handoffs between different authenticators while they have active communications. The duration of the simulation is fixed at 3000 seconds.

During each handoff, mobile users try to gain network access when they are under the coverage area of a new access point. The APs, using 802.11 at 11Mb/s and 30m coverage, are evenly distributed according to the following coordinates: AP 1 (30,35); AP 2 (80,35); AP 3 (130,35); AP 4 (180,35); AP 5 (55,80); AP 6 (105,80); AP 7 (155,80); AP 8 (30,125); AP 9 (80,125); AP 10 (130,125); AP 11 (180,125); AP 12 (55,170); AP 13 (105,170); AP 14 (155,170). This distribution tries to implement a scenario where mobile users are under the coverage of, at least, one AP (e.g. inside a faculty). Nevertheless, there are some regions (peripheral area) that are not under the coverage of any AP and, when a mobile user moves to these regions, it may not get network access.

In the simulation model, several ns-2 nodes have been implemented acting as AAA/KDC servers, placed to carry out single-realm and multi-realm operations. Specifically, we have considered a scenario where there is a visited AAA/KDC and a home AAA/KDC ($h = 2$). As representative values, we have considered $D_j^{0,1}$ =10 ms between the authenticator and the AAA/KDC located in the visited realm, and two values of 25 ms and 60 ms for $D_j^{1,2}$ between the visited AAA/KDC and the home AAA/KDC. The mobility patterns of the mobile users have been generated by the BonnMotion tool [160] The *RandomWayPoint* mobility model has been used. Assuming that mobile nodes are users who move on foot, the maximum speed for mobile nodes has been fixed at a realistic value of 2 m/s.

4.6.3 Performance Comparison

Table 4.3 summarizes the values for n and p for an arbitrary number h of AAA/KDCs for different proposals that have been compared against the solution described in this chapter (*Kerberized EAP-FRM*). The value r is not included since it is the same for all schemes. Nevertheless, we have set r =4, assuming the *4-way handshake* in IEEE 802.11 protected networks.

In particular, we have compared against *non-Kerberized* methods such as fast EAP-TLS [54] and EAP-GPSK (based on symmetric key); and against a representative *kerberized method* like EAP-FRAP [147]. It is important to note that, with EAP-FRAP, the application server is the EAP/AAA server behind the authenticator (thus the mobile user presents an ST to the EAP/AAA server that controls the authenticator). In contrast, in Kerberized EAP-FRM, the application server is the authenticator. Thus, the mobile user presents a valid ST to the authenticator to obtain network access service instead.

For the kerberized methods, the values of n and p are summarized according to three different cases at the moment the mobile user connects to the authenticator:

- case TGT_V & ST, when the mobile user owns a TGT_V with the visited KDC and an ST for accessing the service;

	TGT_V&ST				TGT_V&$noST$				$onlyTGT_H$			
	Proact.		*React.*		*Proact.*		*React.*		*Proact.*		*React.*	
	n	p	n	p	n	p	n	p	n	p	n	p
KERBERIZED METHODS												
EAP-FRAP	–	–	5	4	–	–	7	6	–	–	2h+7	2h+6
Kerberized EAP-FRM	3	0	–	–	–	–	5	2	–	–	2h+3	2h
NON KERBERIZED METHODS												
Fast EAP-TLS	–	–	7	6	–	–	7	6	–	–	7	6
EAP-GPSK	–	–	7	6	–	–	7	6	–	–	7	6

Table 4.3: Number of messages (n,p) involved in the authentication process (for an arbitrary h).

- case TGT_V & $noST$, when it owns a TGT_V but not a valid ST for accessing the service;
- case $onlyTGT_H$, when it only owns a valid TGT_H with the home AAA/KDC.

These cases can be proactive (only supported by Kerberized EAP-FRM) or reactive. For the non-Kerberized methods, only the reactive mode is tested. In terms of number of messages, the proactive mode of Kerberized EAP-FRM is always better than other schemes, and the reactive mode of Kerberized EAP-FRM is no worse than other schemes when $h < 2$.

Table 4.4 shows the measurement results for mean message processing delay per protocol entity with 95% confidence interval obtained from the wireless testbed. For the Kerberized solutions (EAP-FRAP and Kerberized EAP-FRM), the processing time data has been collected from the three cases described at the beginning of the section: case TGT_V & ST; case TGT_V & $noST$ and case $onlyTGT_H$.

Fig. 4.7 shows the mean authentication and SAP establishment time required by mobile users, for each authentication scheme, when they perform a handoff in a visited realm. Specifically, Fig. 4.7(a) shows the case TGT_V & ST where $D_j^{0,1}$ is 10 ms and $D_j^{1,2}$ is 25ms. As can be observed, since the kerberized solutions finishes the authentication process in the visited realm, they both offer a smaller network authentication delay compared with fast EAP-TLS and EAP-GPSK. Moreover, in this case the proactive Kerberized EAP-FRM mode can be used and the results indicate that we reduce the network access delay drastically. In comparison with EAP-GPSK and fast EAP-TLS, we reduce the network access latency by ≈8 times and, in comparison with EAP-FRAP by about ≈3 times.

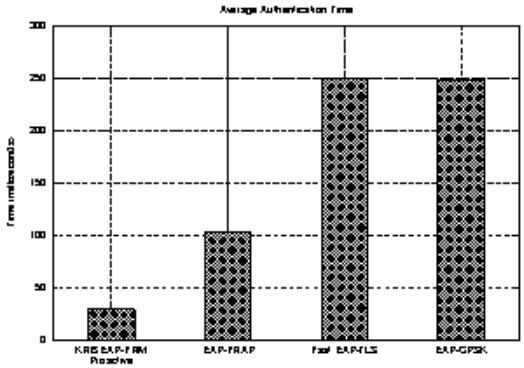
For the case TGT_V & $noST$, the kerberized solutions need some additional time to retrieve an ST for the target authenticator in Kerberized EAP-FRM; and an ST for the visited EAP/AAA server in EAP-FRAP. However, note that this access is performed within the visited realm, avoiding contacting the home AAA/KDC server. Also, in this case, only

	KRB EAP-FRM(**)			***EAP-FRAP(*/**)***		
	TCT_V & ST	TCT_V & $noST$	$onlyTCT_H$	TCT_V & ST	TCT_V & $noST$	$onlyTCT_H$
$\sum_{i=1}^{n} m_{1,i} + \sum_{q=1}^{r} m_{3,q}$	9.36 ± 0.08	15.11 ± 0.20	22.01 ± 0.39	11.82 ± 0.12	17.62 ± 0.20	24.51 ± 0.39
$\sum_{i=1}^{n} a_{1,i} + \sum_{j=1}^{p} a_{2,j} + \sum_{q=1}^{r} a_{3,q}$	13.05 ± 1.47	16.05 ± 0.25	18.88 ± 0.26	9.41 ± 0.23	10.52 ± 0.24	11.32 ± 0.24
(V-AAA/KDC)$\sum_{j=1}^{p} s_j^1$	0 ± 0.0	12.45 ± 4.32	13.34 ± 4.32	4.32 ± 0.83	16.80 ± 4.32	17.69 ± 4.32
(H-AAA/KDC)$\sum_{j=1}^{p} s_j^2$	0 ± 0.0	0 ± 0.0	12.45 ± 4.32	0 ± 0.00	0 ± 0.0	12.45 ± 4.32
	EAP-GPSK			***Fast EAP-TLS***		
$\sum_{i=1}^{n} m_{1,i} + \sum_{q=1}^{r} m_{3,q}$	11.22 ± 0.59			9.70 ± 0.06		
$\sum_{i=1}^{n} a_{1,i} + \sum_{j=1}^{p} a_{2,j} + \sum_{q=1}^{r} a_{3,q}$	9.023 ± 0.20			11,96 ± 0.13		
(V-AAA/KDC)$\sum_{j=1}^{p} s_j^1$	7.36 ± 0.82			4.23 ± 0.23		
(H-AAA/KDC)$\sum_{j=1}^{p} s_j^2$	0.12 ± 0.01			2.19 ± 0.03		

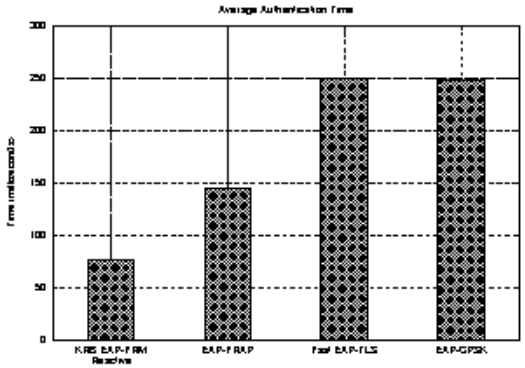
*These values have been approximated based on the results obtained from Kerberized EAP-FRM results

**Times include the communication between KDC and AAA to obtain authorization information for the network access service

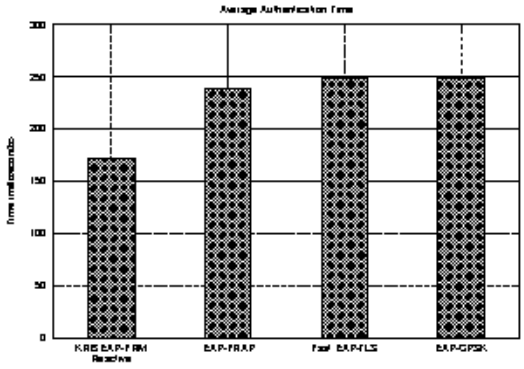
Table 4.4: Mean Processing Time (including security association establishment) and 95% Confidence Interval (in milliseconds) for EAP-GPSK, FAST EAP-TLS, EAP-FRM KRB and EAP-FRAP.



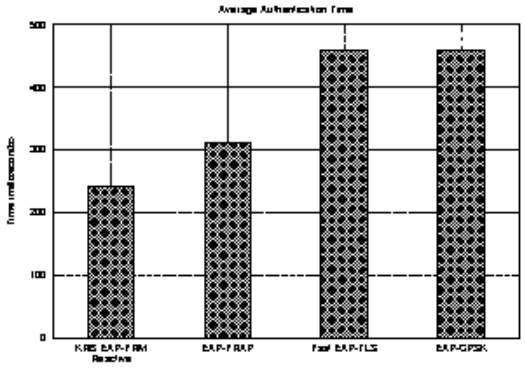
(a) Case TGT_V & ST, $h = 2$, $D_j^{D,1} = 10ms$, $D_j^{1,2} = 25ms$



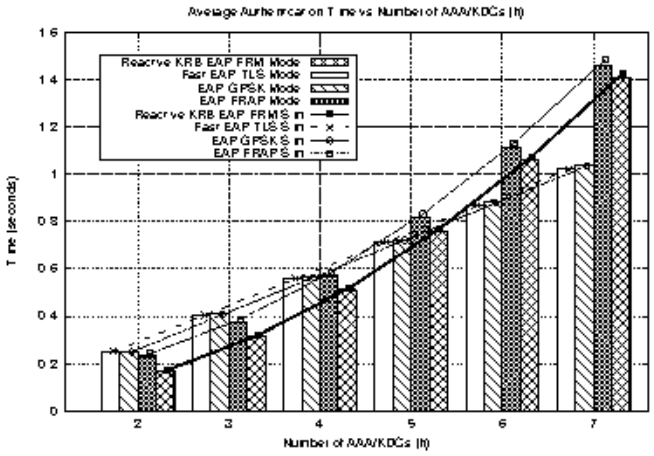
(b) Case TGT_V & $noST$, $h = 2$, $D_j^{D,1} = 10ms$, $D_j^{1,2} = 25ms$



(c) Case TGT_H, $h = 2$, $D_j^{D,1} = 10ms$, $D_j^{1,2} = 25ms$



(d) Case TGT_H, $h = 2$, $D_j^{D,1} = 10ms$, $D_j^{1,2} = 60ms$



(e) Analysis of impact of h ($h \geq 2$, $D_j^{D,1} = 10ms$, $D_j^{k-1,k} = 25ms$)

Figure 4.7: Comparison and Performance Results

reactive Kerberized EAP-FRM mode can be used. As depicted in Fig. 4.7(b), Kerberized EAP-FRM still achieves a faster authentication process than the rest of the alternatives. Taking into account the reactive mode, the average authentication time is reduced by ≈3 times with respect to EAP-GPSK and fast EAP-TLS; and ≈2 times with respect to EAP-FRAP. However, it is important to highlight that if we had applied the proactive Kerberized EAP-FRM mode (see Fig. 4.7(a)), the reduction would have increased even further with respect to the other alternatives: ≈8 times (EAP-GPSK and fast EAP-TLS); ≈5 times (EAP-FRAP) and ≈2.5 times with respect to our reactive mode.

Finally, for the case $onlyTGT_H$, the kerberized solutions perform a cross-realm operation. Fig. 4.7(c) shows the simulation results when $D_j^{1,2}$ is set to ≈25ms. In this case, only reactive Kerberized EAP-FRM can be used, but it reduces the network access latency in ≈1.5 times with respect to EAP-GPSK and fast EAP-TLS, and ≈1.4 times with respect to EAP-FRAP. Nevertheless, as before, if we had applied the proactive mode to avoid this case, the reduction would have been even higher: ≈8 times (EAP-GPSK, fast EAP-TLS and EAP-FRAP) and ≈5.6 times with respect to our reactive mode.

Even when the home AAA/KDC is placed further ($D_j^{1,2}$ is set to ≈60ms) as Fig. 4.7(d) shows, Kerberized EAP-FRM behaves better than others. For example, the reactive mode improves ≈2 times with respect to EAP-GPSK and fast EAP-TLS, and ≈1.2 times with respect to EAP-FRAP. Again, if we had performed the proactive mode we would have obtained better results and improved: ≈15 times (EAP-GPSK and fast EAP-TLS); ≈10 times (EAP-FRAP) and ≈8 times with respect to our reactive mode.

It is important to note though that, in EAP-FRAP and Kerberized EAP-FRM, the case $onlyTGT_H$ only occurs once to recover the TGT_V. Once the mobile user owns the TGT_V, the cases in Fig. 4.7(a) and Fig. 4.7(b) apply.

Finally Fig. 4.7(e) shows the impact of value h in the average authentication time. The results obtained with the mathematical model by applying the mean values for all parameters in the equation have been contrasted against the simulation results. The following values have been assumed: $D_j^{0,1}$ ≈10ms and $D_j^{k\ 1,k}$ ≈25ms. As observed, the mathematical model and simulations remain consistent. Moreover the information in Fig. 4.7(e) is useful to determine when a reactive Kerberized EAP-FRM cross-realm operation is better than performing a bootstrapping operation that contacts the home AAA/KDC (e.g. by using fast EAP-TLS or EAP-GPSK), as discussed in section 4.3.3. As we may observe, it is better to perform Kerberized EAP-FRM than a bootstrapping operation when $h < 4$. However, with EAP-FRAP this number is less, $h < 3$. This means that with EAP-FRAP the improvement is less in comparison with traditional bootstrapping methods based on a full EAP authentication with the home AAA (with a lower number of AAA/KDCs, traditional bootstrapping methods behave better).

4.7 Conclusions

This chapter has studied the definition of a fast re-authentication architecture to effectively reduce the latency introduced by the authentication process during the handoff. This

architecture constitutes the second contribution of this PhD towards the definition of an efficient access control system for NGNs able to preserve the user privacy.

Initially, it has been analyzed that existing fast re-authentication solutions, based on a key distribution process, present common deficiencies like resource reservation on the authentication, requiring to contact a backend authentication server located in the AAA infrastructure or designing new key distribution protocols which is a complex and error-prone process. In this sense, to avoid these drawbacks, we have proposed an architecture which integrates the EAP-FRM framework (described in chapter 3) with Kerberos as a standard, robust and secure three-party key distribution protocol (FRP). Compared with other solutions applying Kerberos for network access, the novelty of our proposal is the adaptation of Kerberos to control the access to the network service without requiring modifications to EAP, lower-layer protocols, the Kerberos protocol itself or adding new entities apart from those ones which are mandatory in the standards we have used. These features ensure an easier deployment of the solution.

Through a detailed description, the Kerberized architecture has been introduced: involved entities, defined interfaces and phases that integrate the fast re-authentication operation. Special interest has been devoted to describe the two fast re-authentication operation modes: proactive mode in which the mobile user can obtain a session ticket for a candidate authenticator before connecting to the authenticator, and reactive mode in which the mobile user obtains a session ticket after connecting to the authenticator. On the one hand, thanks to the use of the EAP-FRM framework, no modification to EAP or link-layer technologies are required. On the other hand, the presented solution is based on the standard Kerberos protocol, the robustness and security properties from Kerberos are guaranteed through its extensive deployment, are thus inherited by our the Kerberized fast re-authentication architecture.

The final part of the chapter has been devoted to test and evaluate the proposal. Initially, in order to show the benefits of the architecture, we have presented a prototype developed by using open-source software. The main findings and conclusions extracted from the development experience have been outlined. This prototype is used to extract real values that are provided to a simulation performed through the NS-2 tool. Comparing the simulation results with a mathematical analysis, we can conclude that the proposed architecture can achieve the goal of fast re-authentication. The performance of the proactive mode, in terms of number of messages and authentication delay, is better than other fast re-authentication schemes. Similarly, the performance of the reactive mode is better than other schemes when the number of realms in Kerberos authentication path is small.

With this chapter, the designed access control system is able to satisfy all the requirements enumerated in the introductory chapter 1 and used to elaborate the comparison table 1.1: *support of any type of handoff* (thanks to the use of EAP which is independent of the underlying technology), *strong security* (thanks to the use of Kerberos which is a well-known and tested secure three-party key distribution protocol), *low deployment impact* and *avoid standard modifications* (since our proposal does not require modifications in current standardized protocols and employ the extensibility mechanisms

available in both **EAP** and **AAA** protocols). Nevertheless, the issue of *privacy* has still not been covered. Precisely, next chapter 5 studies this aspect in depth.

Chapter 5

Providing User Privacy in Kerberized Environments

With the contributions presented in chapters 3 and 4, we have designed an access control system which is able to provide a reduced latency in any kind of handoff, compatible with current standards, easy to deploy and accomplishing strong security properties. Nevertheless, up to now, the aspect of user privacy has not been covered yet. For this reason, we study in this chapter enhancements to the Kerberized EAP-FRM architecture that allow to protect the exposure of sensitive data to unauthorized parties.

In particular, in order to clearly state the context of the distributed key, key distribution protocols such as Kerberos explicitly indicate in their messages the identity of the entities to which the key is distributed [93]. In the specific context of our fast re-authentication based on EAP-FRM and Kerberos, this means that the identity of both the user and the authenticator can be observed by tracing the Kerberos protocol execution. This situation creates some privacy issues for the user since, for example, an eavesdropper can easily determine which specific user is performing a certain fast re-authentication process. As explained in the introductory chapter 1 (see section 1.5), in the long term, this situation allows the existence of attacks such as activity monitoring, movement tracking or user profiling that clearly violate the user's private sphere.

Thus, this chapter addresses this problematic in Kerberos since it is the key distribution protocol which our fast re-authentication solution is based on. Specifically, to solve this problem, we develop a novel privacy architecture called *PrivaKERB* that preserves the privacy of the user during its activity with Kerberos. The architecture provides a flexible solution offering three different levels of privacy: *level 1* which provides user anonymity by means of the use of pseudonyms, so that an eavesdropper cannot determine the real identity of the user; *level 2* where, apart from user anonymity, service untraceability is assured, so that an eavesdropper cannot collect information about per-user access patterns to different services; and *level 3* that represents the highest level of privacy achieving exchange untraceability, where an eavesdropper cannot know whether a set of Kerberos exchanges are performed by the same anonymous user or not.

As we will show, no modifications to the standard Kerberos are required since the

privacy framework is implemented by using the extensibility mechanisms available in the protocol. Furthermore, the use of pseudonyms allows that, in certain controlled situations, trusted parties can perform operations (e.g., to charge the user for service access) which may demand some association with the specific user. To verify the efficiency of the privacy extensions, through an implemented prototype over different scenarios, we demonstrate that the overhead imposed by the privacy extensions are almost negligible in comparison with the standard Kerberos protocol. In other words, our privacy extensions will not harm the fast re-authentication operation defined in previous chapters.

5.1 Kerberos Protocol Privacy Issues

As already pointed out, Kerberos assigns an identifier (or *identity*) to the different entities that participate in the protocol operation. According to the format specified in [74], these identities are in the form "principal_name@realm_name". The first part is a multi-component string (each component is separated by the "/" symbol) that unequivocally identifies an entity in the realm specified by the second part. For example, assuming the realm is UM.ES, *peter@UM.ES* and *printer/server.um.es@UM.ES* are valid to identify a user and a service, respectively.

Nevertheless, principal identifiers are transmitted in cleartext during Kerberos execution. On the one hand, tickets contain in cleartext the service identity for which the ticket has been generated. On the other, Kerberos messages also define fields in cleartext to convey the identities of the principals [74]. More specifically, the KRB_AS_REQ message defines a client name (*cname*) field that contains in cleartext the identity of the client requesting a TGT for the TGS service specified in the service named (*sname*) field. The latter is also included in the KRB_TGS_REQ to specify the identity of the service for which a ST is being solicited. Similarly, both the KRB_AS_REP and KRB_TGS_REP messages include the identity of the client (*cname* field) for whom a ticket has been issued. Therefore, an eavesdropper can easily obtain client's real identity and observe which services are being accessed, thus violating the principle of *user anonymity* [80].

It turns out that, even in the hypothetical case that the client's real identity may remain unknown, an eavesdropper may obtain some general information about behavioural patterns of service access of specific anonymous users in the network. The reason is that, according to Kerberos specification [74], a client typically uses the same TGT to access multiple services (by requesting the corresponding STs). As a consequence, an eavesdropper can determine that the same (anonymous) client is accessing these services just by tracing the use of the same TGT, therefore not accomplishing *service access untraceability* [161,162].

Moreover, for accessing services, Kerberos defines a linked operation where (1) the client engages in message exchange with the KDC to obtain a ticket, which (2) is used in a subsequent communication with the service. This situation provokes that, an eavesdropper is still able to relate all the exchanges followed by a specific (anonymous) user to access a service. This circumstance can be exploited by eavesdroppers, for example, to determine

the origin realm of (anonymous) foreign users when they first access to a service in the visited realm. In other words, an eavesdropper can determine by just observing the Kerberos exchanges that certain users coming from one domain access some particular services in the foreign domain. This specific problem of Kerberos could be avoided through a mechanism providing *exchange untraceability*.

Thus, our goal is to design a lightweight solution to handle these three specific problems in Kerberos: *user anonymity*, *service access untraceability* and *exchange untraceability*. Our intention is to develop a generic privacy solution valid not only in our fast re-authentication architecture based on Kerberos but also to any scenario where Kerberos is deployed to control the access to other network resources such as web servers, printers, etc. Additionally, it is important to clarify that we will focus on Kerberos protocol itself, not dealing with the user privacy protection at other network layers such as, for example, user tracing due to the IP or link-layer address which are out of scope of this work. In the following sections, we provide some important related work and the details of our contribution to solve this problem.

5.2 Related Work

The provision of client privacy in Kerberos is an aspect that has not been completely ignored by researchers. One of the earliest works in this area is found in [163], where the concept of *anonymous ticket* is introduced. An anonymous ticket is a regular Kerberos ticket which does not contain any information about the client's real identity. Thus, when a client uses an anonymous ticket to access a service, the client's real identity is not revealed either to the service or to eavesdroppers. However, the deficiencies of this solution stem from the methods proposed to deliver anonymous tickets to the clients. The first method assumes that the client does not share any secret with the KDC and proposes the use of *Public Key Cryptography for Initial Authentication in Kerberos* (PKINIT) [164] in order to securely deliver the session key associated with the anonymous ticket. Nevertheless, this method requires the existence of a *Public Key Infrastructure* (PKI) that, unfortunately, may not always be available. The second method only allows registered users within a Kerberos realm to obtain an anonymous ticket. More precisely, the client obtains an anonymous ticket by performing a standard AS exchange with its real identity. Thus, the real client's identity and the distributed anonymous ticket are visible for an eavesdropper during the AS exchange, which allows the eavesdropper to establish a relationship between them. Consequently, an eavesdropper can easily infer the client's real identity by just tracing the use of the specific anonymous ticket.

On the basis of this initial work, the IETF Kerberos Working Group defines a solution [165], also based on the use of anonymous tickets, to completely hide the client's identity from KDCs and external observers. This work mainly focuses on the definition of the Kerberos protocol extensions required to implement the anonymity solution presented in [163]. For example, the working group defines the well-known anonymous principal name (*WELL-KNOWN/ANONYMOUS*) as an identifier having a special meaning other than

identifying a particular user. Similarly, new flags are proposed to distinguish anonymous tickets from standard ones and to allow users to solicit an anonymous ticket. Nevertheless, since the technical solution defined by the working group remains the same as proposed in [163], the same deficiencies previously mentioned for each anonymous ticket acquisition method apply here.

Another way to offer client privacy in Kerberos is to transmit messages through a protected TLS tunnel [166]. Once the TLS tunnel is established between two entities (e.g., client and KDC), Kerberos exchanges can be performed within the tunnel. Thus, sensitive information such as the client's real identity is not revealed to eavesdroppers. However, this solution requires the client to successfully establish a TLS tunnel with any entity (KDCs and services) with which it desires to exchange Kerberos messages. Excluding the overhead that a full TLS handshake may impose [167], this requirement is especially problematic during cross-realm operation where the client has to establish TLS tunnels with intermediary KDCs. Since we cannot assume a pre-established trust relationship between the client and intermediary realms, a multi-domain PKI infrastructure [168] is required to assist a typical certificate-based TLS authentication, increasing the deployment cost of the solution. Regardless of this inconvenience, it is important to note that this solution is just shifting the privacy problem to the TLS protocol where, during the authentication phase prior to the TLS tunnel establishment, it is necessary to define a mechanism to preserve the client's privacy.

The *Generalized Framework for Kerberos Pre-authentication* [169] proposes an architecture to assist the design of authentication mechanisms which allow the client to be authenticated before granting a ticket. An additional objective of this solution is to enhance the security of the protocol by protecting information that Kerberos sends in cleartext. In particular, the client's identity is confidentiality protected in the messages sent from the client to the KDC (KRB_AS_REQ and KRB_TGS_REQ). Using this improvement in conjunction with the extensions defined in [165], the solution allows a client to solicit an anonymous ticket without revealing its real identity. Nevertheless, the pre-authentication framework solution presents some drawbacks related to the so-called *armor TGT*, which clients must obtain before starting to use the pre-authentication extensions with a specific KDC. In particular, the three methods proposed to obtain such special TGT present some deficiencies. First, a client can perform a standard AS exchange using its real identity to request an armor TGT. Nevertheless, it allows eavesdroppers to relate the acquired armor TGT with the client's identity. Thus, when the client re-uses the armor TGT to request an anonymous ticket, an observer can infer the real identity associated with the anonymous ticket. Secondly, assuming the KDC owns a valid certificate, the user can use anonymous PKINIT [164] to obtain an armor TGT. When a PKI infrastructure is not available, a final method proposes to acquire the armor TGT using anonymous PKINIT without KDC authentication. Nevertheless, as recognized by authors of the work, this option third is vulnerable to man-in-the-middle attacks.

Thus, to the best of our knowledge, none of these proposals define a solution which at the same time allows: a) the client to remain anonymous and untraceable from eavesdroppers; b) to identify the client in the specific cases that important processes such as accounting

and charging operations require it; and c) to eliminate the need of other infrastructures different than the Kerberos ones, such as a PKI. As seen, proposals such as [163,165,169] aim for the complete anonymity of the client, which makes it unfeasible to perform some vital operations like the accounting process performed by trusted entities. Conversely, the work in [166] requires the support of a PKI infrastructure to operate that makes its deployment harder. In this way, the presented privacy framework for Kerberos is a novel approach which gives flexibility between the level of privacy provided to the user and reduces the cost required to deploy the solution. Furthermore, introducing an almost negligible penalization to the protocol, our proposal achieves an effective privacy protection of the client without affecting other vital processes where the client needs to be identified such as accounting and charging operations. Next, we provide details about our proposal.

5.3 Proposed Privacy Framework

5.3.1 General Overview

Motivated by the privacy issues described in section 5.1, in this work we focus on hiding client's real identity from unauthorized parties and reduce the information about service access behaviour of specific (anonymous) users that an eavesdropper is able to collect from looking at Kerberos exchanges. More precisely, our solution must fulfil three important requirements:

1. *User anonymity.* During its activity with Kerberos, a client must remain anonymous not only to eavesdroppers but also to any entity in a particular realm. Only the KDC in the home realm will have access to the client's real identity.

2. *Service access untraceability.* Eavesdroppers must be unable to trace the different services accessed by a specific *anonymous* user. This information will be only known by the specific and trusted KDC controlling the set of services accessed by the client.

3. *Exchange untraceability.* Observers cannot deduce that two different Kerberos exchanges are related and performed by the same *anonymous* user. Only trusted KDCs and services with which the client interacts can know this information.

The reason of providing this flexibility is that, as we will analyze, it is expected that a higher level of privacy shall require more extensions and implementation efforts. Therefore, if only user anonymity is required/implemented within a realm there is no need to implement any extension related to, for example, service access traceability. Moreover, the selection of a specific level of privacy can also consider factors such as deployment cost, since a higher level of privacy will imply more changes in existing Kerberos infrastructures.

It is worth noting that these three general requirements are interrelated. Service untraceability assumes that user anonymity is ensured. At the same time, the existence of exchange untraceability implies that accesses to services cannot be traced [80]. Instead of

proposing a single solution supporting the highest level of privacy (*exchange untraceability*), our objective is to design a flexible framework which allows to select which specific requirement must be accomplished.

To achieve this flexibility our solution therefore works in three modes of operation. Each one, which is identified with a specific privacy level, satisfies one of the previously described requirement. More specifically, we distinguish:

- *Level 1.* Lowest privacy level providing only user anonymity (*req.1*).
- *Level 2.* Intermediate privacy level providing both user anonymity and service untraceability (*req.2*).
- *Level 3.* Highest privacy level that, in addition to the facilities provided by level 2, offers exchange untraceability (*req.3*).

These privacy levels have been designed considering the following guidelines. Firstly, we favor the interoperability of the solution with current implementations of Kerberos (e.g. Massachusetts Institute of Technology - *MIT* - Kerberos Implementation [142]). In other words, the proposed solution respects the standard Kerberos specification [74], and modifications to the protocol (e.g., definition of new message fields) are not required. Indeed, we employ the extensibility mechanisms available in the standard Kerberos [74]: a) definition of new flags in the messages; b) design of new pre-authentication data (hereafter *padata*); and c) new authorization elements [74] which can carry any valuable information for new applications, such as our privacy extensions. Using these mechanisms, our solution integrates smoothly with current Kerberos implementations without privacy support. Secondly, for the solution we demand the support of the cross-realm operation. This aspect becomes important in the context of NGNs, where users frequently change their point of attachment to the network and solicit access to services located in foreign realms.

In the following, we present the notation we employ to describe the enhancements to the Kerberos protocol:

- $name_i@realm$: ith principal name employed by a user in the specified Kerberos realm.
- $TGT_X^i[Y]$: ith TGT for KDC X that contains the information Y confidentiality and integrity protected.
- $ST_X[Y]$: service ticket for service X that contains the information Y protected providing confidentiality and integrity.
- N_i: ith random number generated by the client.
- $PA-NAME(Y)$: a padata type named NAME containing the information Y.

- $AD(X, Y, ...)$: pieces of information X,Y,... are transported in the authorization-data (AD) field of a ticket by using different authorization data elements.
- $FlagX$: a flag named X.

5.3.2 Level 1: User Anonymity

Preliminaries

Level 1 is the lowest level of privacy that we have designed to accomplish requirement 1. That is, level 1 keeps the user anonymous during its activity with Kerberos. One possibility is to use an anonymous identity which will be shared by all users desiring to be anonymous, such as implemented in [165]. Nevertheless, this kind of solution does not allow to perform some operations that require some association to the specific user, as for example, charging for the received services. In our opinion, a more convenient solution would be to allow the user to act using pseudonyms so that its real identity is not revealed. Moreover, the association between these pseudonyms and the real user identity is only known by a trusted party of the realm where the user has a subscription (*home KDC*).

One approximation may be based on assigning to the user an unique and permanent pseudonym, which would be used as its identity. Nevertheless, if the association *(real identity, pseudonym)* is revealed to unauthorized third parties (e.g., some security leak), the whole user's activity can be traced. To avoid this problem, we opt by dynamically assigning pseudonyms, which are only valid for a specific period of time. In particular, we follow a similar approach to that introduced in our contribution in [101]. With this approach, the user owns a unique and temporary pseudonym which is used as identity for a period. Specifically, we bind this time to the TGT lifetime, so that the pseudonym for which a TGT is distributed must be renewed before the TGT lifetime expires. This binding is possible since we propose a *KDC-controlled pseudonym generation* mechanism where a KDC generates and distributes the pseudonyms to the client, which the client will use as client name (*cname*) in subsequent accesses with Kerberos.

It is worth noting that the pseudonyms generated by the KDC are unique, so different users do not employ the same pseudonym. In this manner, trusted entities like KDCs can easily identify which activity has been performed by a certain user through a set of pseudonyms. By associating these pseudonyms with the real user's identity, the home KDC may generate, for example, an invoice specifying the resources consumed by a client. Taking into account these general aspects, we proceed to describe the details of the operation of level 1.

Detailed Operation

Figure 5.1 illustrates the operation and the extensions to Kerberos required to implement privacy level 1 in a single-realm scenario, (i.e. the client accesses services in the home realm). As we observe, the process starts when the client sends a KRB_AS_REQ message to the home KDC using the pseudonym CP_1 as the name (the realm associated with this

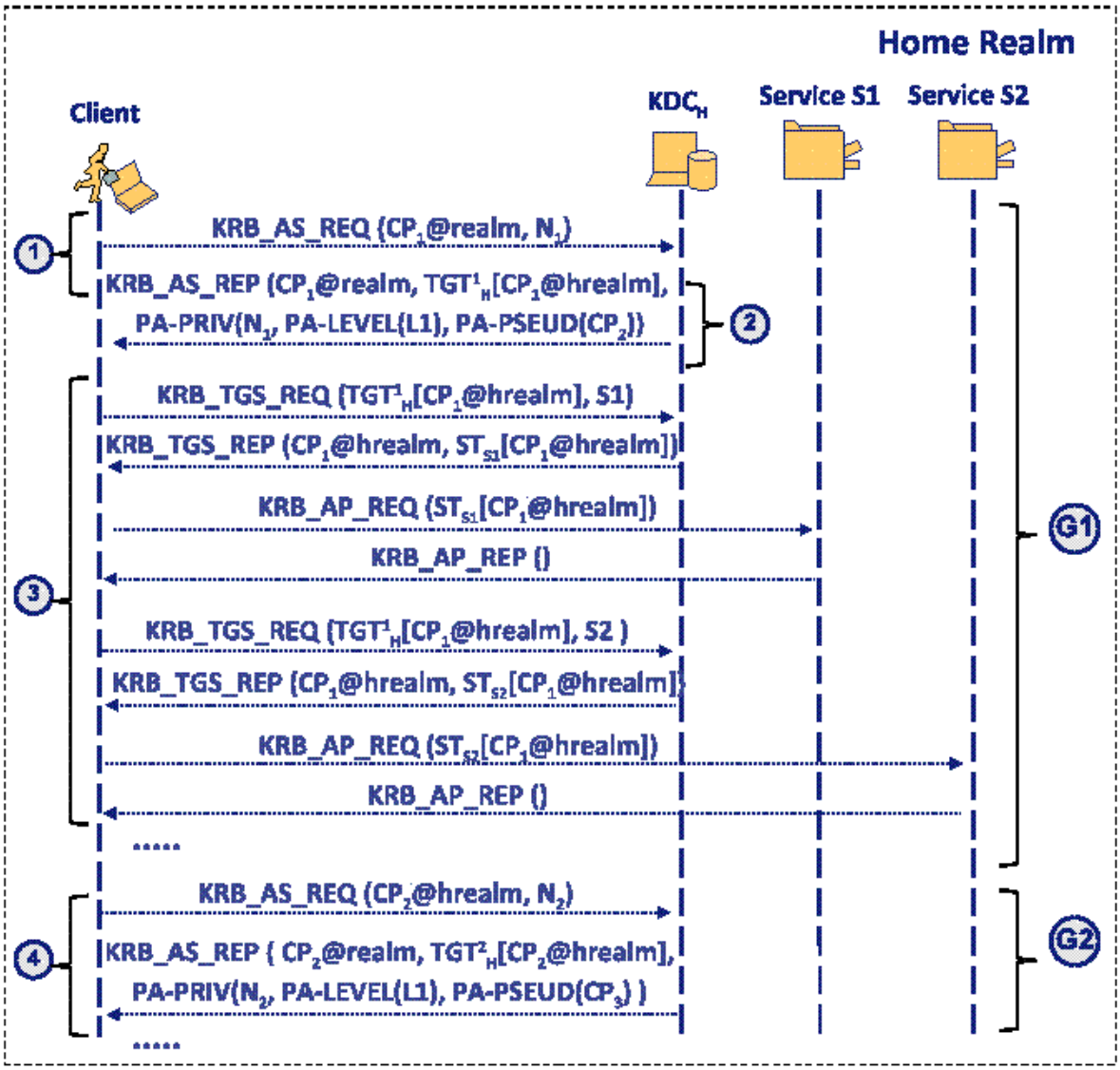


Figure 5.1: Privacy level 1: single-realm scenario

name is *hrealm* in this example) *(1)*. When the KDC receives the message and retrieves from its user database the profile associated with the user identified as CP_1, it realizes that this identity is a pseudonym and that privacy level 1 is enabled for this user.

From this point, the KDC follows the standard Kerberos operation, assuming CP_1 as client's name (i.e. CP_1 is set in the field *cname* of Kerberos messages). In this manner, the KDC answers the client with the KRB_AS_REP *(2)* containing a TGT for the KDC (TGT^1_H). Nevertheless, this message is extended in our contribution to deliver the privacy level assigned by the KDC to the client as well; and a new pseudonym (i.e. CP_2), which the client must employ in the subsequent AS exchange. To convey these values, we propose

the use of the *padata* field defined for those messages exchanged with the KDC. According to Kerberos standard specification [74], this field can be used to extend the protocol exchanges between the client and the KDC. Specifically, we define two new padata types named *PA-LEVEL* and *PA-PSEUD* (see appendix A.1). The former contains a numeric value informing the client about the privacy level assigned by the KDC (denoted as *L1*). The latter contains the new pseudonym (i.e. CP_2) for the subsequent AS exchange. It is worth noting that the distribution of these pieces of information must be protected in such a manner that third parties cannot observe or modify their values. To provide a secure transmission between KDC and client of these values we define a new padata type named *PA-PRIV* (see appendix A.1). PA-PRIV contains a sequence of padatas that are confidentiality and integrity protected under the same key employed to compute the so-called encrypted part (*enc-part*) field defined in the KRB_AS_REP message, as specified in [74]. In addition to the encrypted sequence of padatas, the PA-PRIV padata includes the nonce N_1 (already defined in the Kerberos specification [74]) sent by the client in the KRB_AS_REQ to provide freshness to the padata content.

Once this AS exchange has been successfully performed, the client follows the standard protocol operation to access services, employing CP_1 as the client's identity (that is, by performing TGS exchange *(3)* with the TGT^1_H). When TGT^1_H expires and the client solicits a new TGT, the client performs another AS exchange *(4)* using now pseudonym CP_2 as the new identity. The home KDC operates as previously described, that is, the client receives the assigned privacy level and a new pseudo-random pseudonym CP_3 is distributed for the next AS exchange, and so on. Details regarding the pseudonym management are provide in section 5.4.4.

Given that an eavesdropper cannot relate pseudonyms CP_1 and CP_2 since they are distributed encrypted to the client, user traceability is broken each time a new TGT is acquired. That is, as depicted in Fig. 5.1, an eavesdropper is unable to infer from looking at Kerberos exchanges that groups of messages *G1* and *G2* are actually exchanged by the same user.

Finally, it is worth highlighting a special case that occurs when the client starts an AS exchange for the very first time. In this case, we assume that client is provided with an initial pseudonym by the home network through some out-of-band mechanism (e.g., pre-installed in the device). The details of such mechanisms are outside the scope of this work.

5.3.3 Level 2: Service Access Untraceability

Preliminaries

Although the use of pseudonyms prevents the client's real identity being revealed, an eavesdropper can easily determine that the same client with a specific pseudonym (e.g., CP_1) is accessing a set of services. This is possible by simply observing the *cname* field in each KRB_TGS_REP. Thus, although the real client's identity is not accessible, eavesdroppers are able to infer [161,162] which services have been requested by a specific

anonymous user (*pseudonym-based service access traceability*).

We may hide the client's pseudonym from eavesdroppers to avoid this problem. However, this is a necessary but not sufficient condition to solve it. Indeed, a client typically re-uses the same TGT several times to request access to multiple services. An eavesdropper can use this TGT as a reference to discover the service access pattern of the specific anonymous user. Therefore, although the real client's identity is not accessible, eavesdroppers can infer which services have been requested by a specific anonymous user (while the same TGT is employed) and obtain anonymous profiles [161, 162] during the TGT lifetime (*TGT-based service access traceability*). In other words, the simple use of pseudonyms enables user anonymity but does not prevent eavesdroppers from tracing the different services accesses performed by a concrete anonymous user during the period of time that such a user employs the same TGT. When this information is systematically collected, other valuable information like service access behavioral patterns of the specific anonymous clients can be inferred [162]. As a consequence our requirement of *service access untraceability* is not met with mere pseudonyms. In fact, to achieve this requirement, we need to accomplish two goals: 1) to hide the pseudonym from the eavesdropper; and 2) to avoid the use of the same TGT each time an anonymous client requests a ST.

For the first goal, we enhance the pseudonym-based approach followed in level 1 by employing the so-called *anonymous ticket* defined in [165]. The anonymous tickets have a flag designating ($Flag_A$) that the ticket is anonymous (see appendix A.2), and are always associated with the anonymous client *anon@anon*, which is the client's identity that can be observed by an eavesdropper in the KRB_TGS_REP messages. In particular, we extend the concept of anonymous ticket by including both the privacy level assigned to the client and its pseudonym in the *authorization-data* field (see appendix A.3) defined in [74]. With this new type of ticket, we protect the client's pseudonym so that it is only revealed to authorized parties like KDCs or services, which may require it to perform, for example, charging operations.

For the second goal, we introduce the concept of *self-renewed TGT*, which has been specifically designed for our solution. While traditional TGTs in Kerberos are generated by the AS and processed by the TGS [74], self-renewed TGTs are generated and processed by only the TGS. In this sense, we define a new secret key named TSRK (TGT self-renewal key) generated and only known by the TGS in charge of creating self-renewed TGTs. Each time a client uses a self-renewed TGT to request a ST for a service, the TGS will not only distribute the ST but also a new self-renewed TGT to the client, which the client will use to request the new ST for another service. In other words, the self-renewed TGTs are used only once (one-use ticket) for ST solicitation. Moreover, the distribution of self-renewed tickets is performed confidentially, so that an eavesdropper cannot know that the new self-renewed TGT is related in any way with the recently used self-renewed TGT. This is key to achieving service access untraceability, since the eavesdropper will observe different (one-use tickets) and unrelated TGTs each time the client accesses a service. Thus, the eavesdropper will be now unable to use of the same TGT as a pointer to obtain any service access pattern of a specific anonymous user. In the following section, we describe in detail how these tickets are used to achieve service access untraceability.

Detailed Operation

Figure 5.2 shows an example about how privacy level 2 works assuming that a user is accessing services in its home realm (typical single-realm scenario). The process starts when the client is authenticated through an AS exchange *(1)* following a process similar to privacy level 1. The client selects pseudonym CP_1 as principal name and sends a KRB_AS_REQ message to the AS/KDC. Upon reception, the AS/KDC realizes that CP_1 is a pseudonym by consulting its client database and determines that privacy level 2 must be enabled for the client (e.g., according to its profile). Since privacy level 2 is based on privacy level 1, the same extensions proposed for the latter are also applied here. For this reason, the AS/KDC generates a new random pseudonym (CP_2), which is sent to the client together with the assigned privacy level 2 (denoted as *L2*). Additionally, the AS/KDC generates an anonymous TGT ($TGT^1_H[FlagA...$) where both the privacy level and the client's pseudonym CP_1 are encrypted as part of the authorization data elements in the ticket.

Next, when the client needs an ST for accessing service S1, it builds a KRB_TGS_REQ *(2)* that contains TGT^1_H. Upon reception, the TGS/KDC examines the TGT presented and realizes that privacy is enabled since the anonymous ticket flag ($FlagA$) is set. Although, the anonymous TGT is associated with the anonymous user (*anon@anon*) the TGS/KDC becomes aware that the client with pseudonym CP_1 is employing privacy level 2 extensions by looking at the authorization data field in the ticket. When the request is successfully validated, the TGS/KDC generates two tickets and sends them to the client in the KRB_TGS_REP *(3)*: a) an anonymous ST ST_{S1} (whose format is defined in [165]) associated with client *anon@anon* containing the same authorization data in the anonymous TGT; and b) an anonymous self-renewed TGT ($srTGT^1_H$) that contains the same values of the anonymous TGT, but an updated *starttime* field. As we can observe, $srTGT^1_H$ is transported through a new padata type which we define called PA-SR-TGT that, in turn, is contained in a PA-PRIV padata for achieving a secure distribution process. For this reason, the distributed self-renewed TGT is neither visible to eavesdroppers nor can it be modified by an attacker. As a consequence, an eavesdropper cannot know what ticket the client will employ for the next ST solicitation since the self-renewed TGT is not related in any way with the TGT TGT^1_H presented. Finally, the ST_{S1} is delivered to the service through a standard KRB_AP_REQ/KRB_AP_REP exchange *(4)*. In this case, if the service recognizes our extensions and needs to know the client's pseudonym (e.g., to charge the client for the offered service), it can obtain it from to the authorization data contained in the ST. Conversely, if the service does not support them, it will be unable to interpret the authorization data containing the client's pseudonym. The service could be provided to the anonymous user (*anon@anon*), but no charging operations will be carried out.

The same process we have explained for accessing service S1 is also applied when the client requests access to another service S2, but now the client sends $srTGT^1_H$ in the KRB_TGS_REQ to the TGS/KDC *(5)*. An eavesdropper tracing the communication is unable to deduce that this TGT belongs to the same client which sent the previous

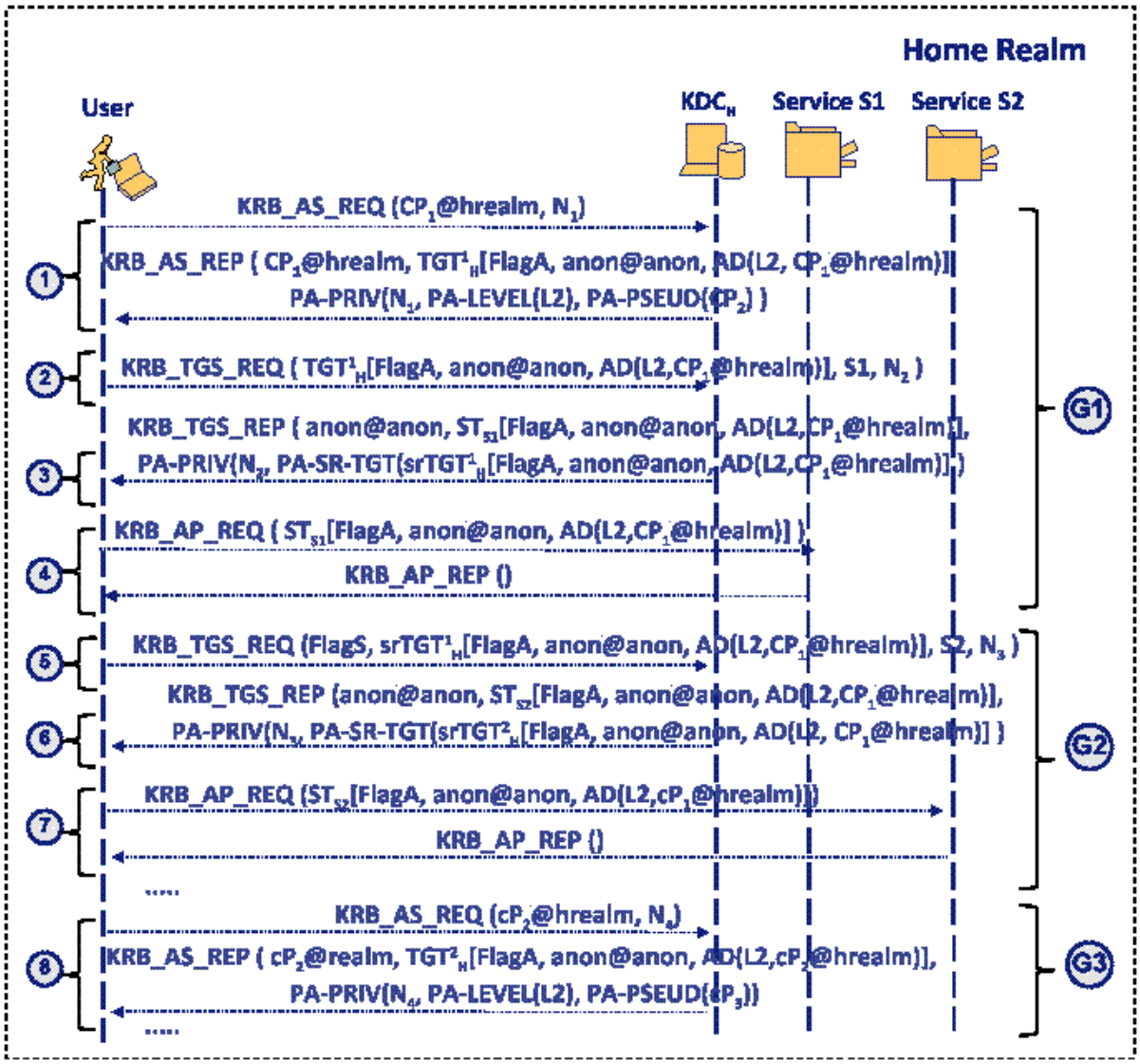


Figure 5.2: Privacy level 2: single-realm scenario

TGT TGT^1_H. The reason is twofold: a) the resulting self-renewed TGT is completely different from the previous one TGT^1_H; and b) the eavesdropper is unable to determine that the distributed self-renewed TGT $srTGT^1_H$ is related to TGT^1_H, since the former is confidentiality protected by using the PA-PRIV padata during the distribution process. After successful validation, the client receives an anonymous ST $ST_{S2}[FlagA...]$ and a new self-renewed TGT $srTGT^2_H$ for the next access to the TGS/KDC *(6)*. While the former is used to access service S2 *(7)*, the latter replaces the previous $srTGT^1_H$. Given that self-renewed TGTs are neither re-used (they are considered as one-use tickets) nor sent in cleartext, an eavesdropper cannot relate the different ST solicitations performed by the same user. More precisely, according to Fig. 5.2, accesses to services S1 and S2 (represented

by group of messages *G1* and *G2*, respectively) are performed by non-related users from the eavesdropper's point of view.

When the self-renewed TGT expires, the client must be authenticated through a new AS exchange *(8)*. Then the client starts using CP_2 as its identity, and obtains a new pseudonym for the next AS exchange. As already discussed in section 5.3.2, the change of pseudonym prevents an eavesdropper from relating messages exchanged under a new pseudonym (*G3*) with previous ones (*G1* and *G2*).

5.3.4 Level 3: Exchange Untraceability

Preliminaries

For accessing services, Kerberos defines an atomic operation where the client engages in a message exchange with the KDC to solicit a ticket that is presented to the service in a subsequent exchange. This operation is applied twice in the protocol. Firstly, the AS exchange is destined to provide the client a TGT to be used in the TGS exchange. Secondly, in the TGS exchange the client obtains an ST that is sent to the service through the AP exchange. Since tickets are transmitted in cleartext, this particularity allows an eavesdropper to relate the different messages sent or received by a client to access a service.

As depicted in Fig. 5.2, given that privacy level 2 provides service untraceability, an eavesdropper capturing the communication maintained by the client cannot infer that accesses *G1* and *G2* are being performed by the same anonymous client. Nevertheless, by using as reference the ticket (either TGT or ST) distributed in one exchange and used in a subsequent one, an eavesdropper can relate the group of messages exchanged to access, for example, service S1. On the one hand, the AS and TGS exchanges are related thanks to TGT TGT_H^1, which is distributed to the client in the AS exchange and used in the TGS exchange. On the other hand, TGS and AP exchanges are related thanks to ST ST_{S1} which is obtained by the client in the TGS exchange and presented to the service in the AP exchange. Therefore, both TGT_H^1 and ST_{S1} allow an eavesdropper to infer that the these AS, TGS and AP exchanges (group of messages *G1*) are performed by the same anonymous user. The same situation happens with the group of messages exchanged to access service S2 where in this case, ST_{S2} reveals an eavesdropper that the TGS and AP exchanges (group of messages *G2*) are performed by the same anonymous user. Nevertheless, as previously mentioned, thanks to the self-renewed feature of privacy level 2, an eavesdropper cannot infer that accesses (*G1* and *G2*) are performed by the same anonymous client.

At first sight, we might consider that the ability of relating the different exchanges performed to access a service may not pose any risk for privacy. In fact, in the most typical service access (based on a self-renewed TGT), an eavesdropper cannot obtain useful information since the anonymous identity *anon@anon* is exposed in the KRB_TGS_REP message (see group of messages *G2* in Fig. 5.2). However, a special situation occurs when the client accesses the first service after being authenticated through the KRB_AS_REQ/REP exchange. As we can observe in Fig. 5.2 (group of messages *G1*), an eavesdropper can determine that *S1* is the first service accessed by an anonymous user

from *hrealm*. This situation is more important when the first accessed service after a KRB_AS_REQ/REP exchange is located in a foreign realm (cross-realm scenario) because an eavesdropper can observe that an anonymous user from a specific realm is accessing to a service offered by a visited one. Tracing this activity gives valuable information to eavesdroppers like, for example, the most attractive services for roaming users offered by a domain.

In this sense, with privacy level 3 we go a step further by improving the aforementioned privacy level 2 so as to provide exchange untraceability as well. In other words, an eavesdropper cannot relate the different Kerberos exchanges, therefore being unable to relate any set of messages exchanged by an anonymous client to access a specific service. Since the ticket (either TGT or ST) distributed (in cleartext) to the client in one exchange and used in a subsequent one[1] is the element used by eavesdroppers to relate the different exchanges to access a service, if we define some procedure to hide this information, we will achieve exchange untraceability. For this reason, we propose to hide (e.g., through encryption) the distributed ticket, so that only the client can recover it. Additionally, the field of the message where an eavesdropper expects to find the requested ticket, will contain a *fake ticket*. A fake ticket is a new type of ticket where, except the *flags* field, all the fields belonging to the part that is confidentiality and integrity protected (*EncTicketPart*) contain invalid information (e.g., fields are null or randomly initialized) so that the resulting ticket is different from the real one sent to the client. Additionally, a new ticket flag named *fake flag* is defined to indicate that a ticket is a fake one. Upon reception, a client must discard the fake ticket and process the real one issued by the KDC and located in the *padata* field of the response (either KRB_AS_REP or KRB_TGS_REP).

Detailed Operation

Figure 5.3 shows how privacy level 3 works when a user solicits access to services located in its home realm. In general, we observe that the process is very similar to that described for privacy level 2 (see Fig. 5.2). The only difference between both privacy levels is the way tickets are delivered to the client in privacy level 3 by using the fake ticket mechanism.

Initially, when the client performs the AS exchange *(1)*, we observe that the issued TGT ($TGT^1_H[FlagA,...]$) is transported through a new padata type named *PA-TICKET*. Together with TGT^1_H, this padata contains information (omitted for simplicity) that will normally appear in the *enc-part* of the KRB_AS_REP and necessary for the client (e.g., TGT lifetime). Details about the ASN.1 specification of this padata type are provided in appendix A.1. Additionally, as we can see, the PA-TICKET belongs to the sequence of padatas protected by the PA-PRIV padata. As described in section 5.3.2, recall that PA-PRIV provides confidentiality, integrity and freshness to the set of padatas that contains. For this reason, an eavesdropper cannot associate the issued TGT^1_H with the anonymous client. On the other hand, the field of the KRB_AS_REP expected to transport the issued TGT (and visible to everyone) contains $FakeTGT[FlagF...]$, a fake ticket with

[1] While a TGT allows to relate an AS exchange with a posterior TGS exchange, an ST allows to relate a TGS exchange and the posterior AP exchange.

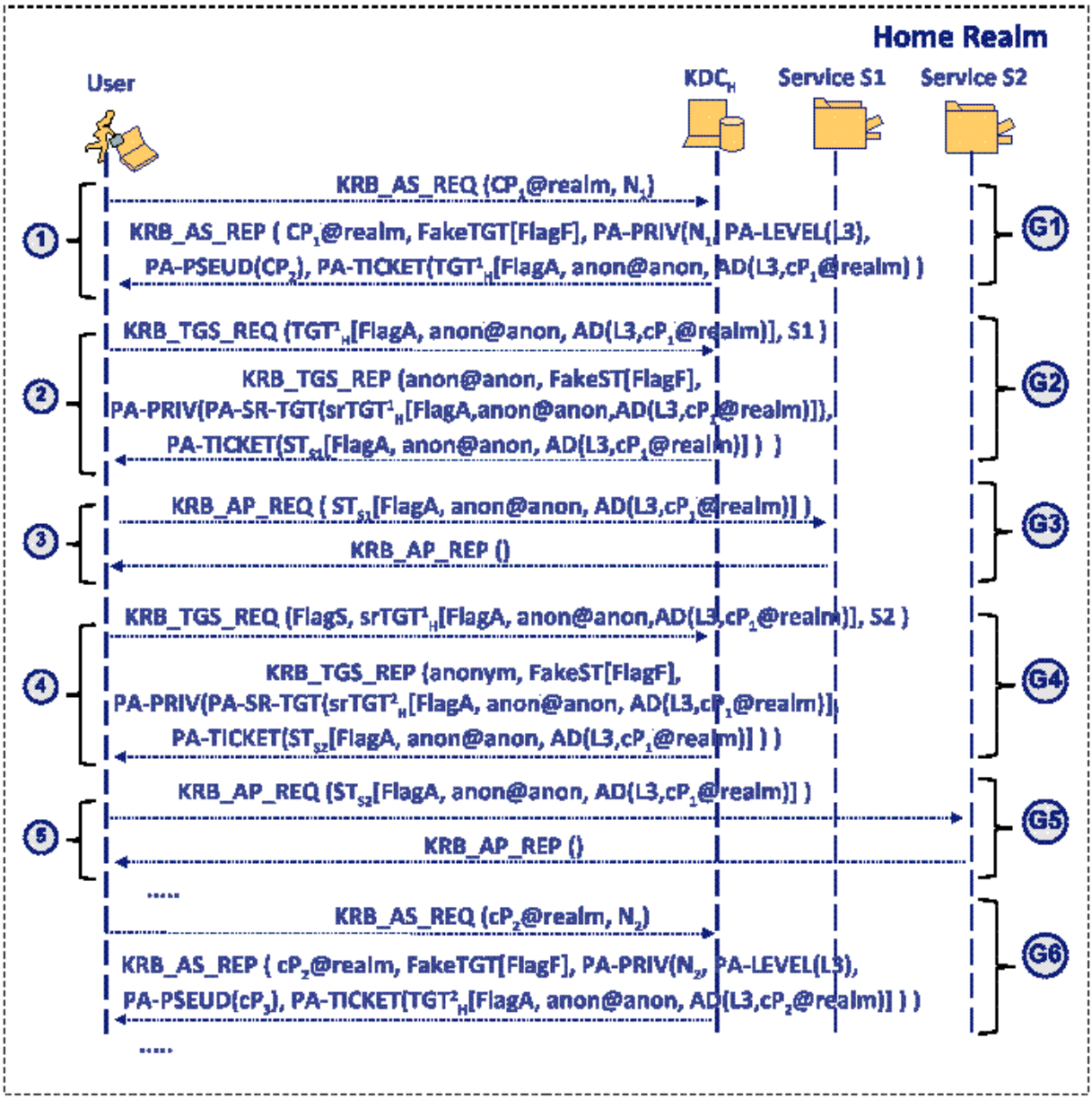


Figure 5.3: Privacy level 3: single-realm scenario

the fake ticket flag enabled ($FlagF$) containing invalid information that, therefore, will be ignored. Upon delivery, when the client process the *enc-part* field of the KRB_AS_REP and detects that $FlagF$ is set in the attached ticket, it will automatically discard this fake ticket, expecting to find the real one in a PA-TICKET padata type.

This process is also applied in the posterior TGS exchanges *(2 and 4)* where the KDC distributes service tickets $ST_{S1}[FlagA...]$ and $ST_{S2}[FlagA...]$ to the client, respectively. Thanks to the fake ticket feature employed in the KRB_TGS_REP message, an eavesdropper can only observe the fake ST since the real one is encrypted by means

of the PA-PRIV padata and can be only recovered by the client using the same key (typically the secret key shared with the TGS) employed to process the *enc-part* field of the response. Therefore, it is impossible for an eavesdropper to know the ST delivered to the client. For this reason, when the client presents the received ST to the service through a KRB_AP_REQ/REP *(3 and 5)*, an eavesdropper cannot deduce that this operation is performed by the same anonymous user that performed the previous KRB_TGS_REQ/REP because it cannot infer that ST was previously acquired by the same client.

As a result, by combining the pseudonym-based client identification introduced in level 1, both the anonymous tickets and the self-renewed TGTs presented in level 2, and the fake ticket feature described in this level 3, we achieve a higher level of privacy in Kerberos where an eavesdropper is unable to relate the different Kerberos exchanges in the protocol. According to Fig. 5.3, an eavesdropper tracing the communication cannot deduce that the different Kerberos exchanges (from *G1* to *G6*) are performed by the same anonymous user.

5.3.5 Operation in Cross-realm Scenarios

In a cross-realm authentication, our objective is to minimize the deployment cost of the solution and favour its adoption. Indeed, we have designed a privacy-enhanced cross-realm operation where only KDCs where the client has a subscription (*home KDCs*) and the KDCs that the client visits in another realm (*visited KDCs*) must be updated to support our privacy extensions. Thus, intermediary KDCs (which are placed to intermediate between home and visited KDCs in a Kerberos cross-realm infrastructure, as depicted in Fig. 2.21) can use existing implementations based on the Kerberos specification [74] without support of our privacy extensions. Next, we describe how the three levels of privacy operate in a cross-realm scenario.

Privacy Level 1

Level 1 can be adapted straightforwardly to the cross-realm operation as it only introduces some extensions to the initial AS exchange performed with the AS/KDC server. So basically, intermediate TGS/KDCs are not affected by our extensions.

Figure 5.4 details how privacy level 1 works over a cross-realm scenario. In our example, we describe the process when a user accesses services S1 and S2 controlled by the visited KDC (KDC_V), other than its home KDC (KDC_H). To simplify the analysis, we assume that there only exists an intermediary KDC KDC_I. After AS exchange *(1)*, the client is informed about both the privacy level assigned and the new pseudonym to be employed in the next AS exchange. Next, a typical cross-realm process based on several KRB_TGS_REQ/REP exchanges is performed following the standard Kerberos *(2)*. Once the client acquires a valid ST for service *S1*, it authenticates itself against the service *(3)*.

By re-using the acquired cross-realm TGT (TGT_V^1), the client is able to solicit another ST for accessing service *S2 (4)*. As observed, the use of privacy level 1 is transparent to the KDC in the visited realm KDC_V and the intermediary KDC KDC_I. However, although user anonymity is achieved, the client employs the same pseudonym CP_1 inside the visited

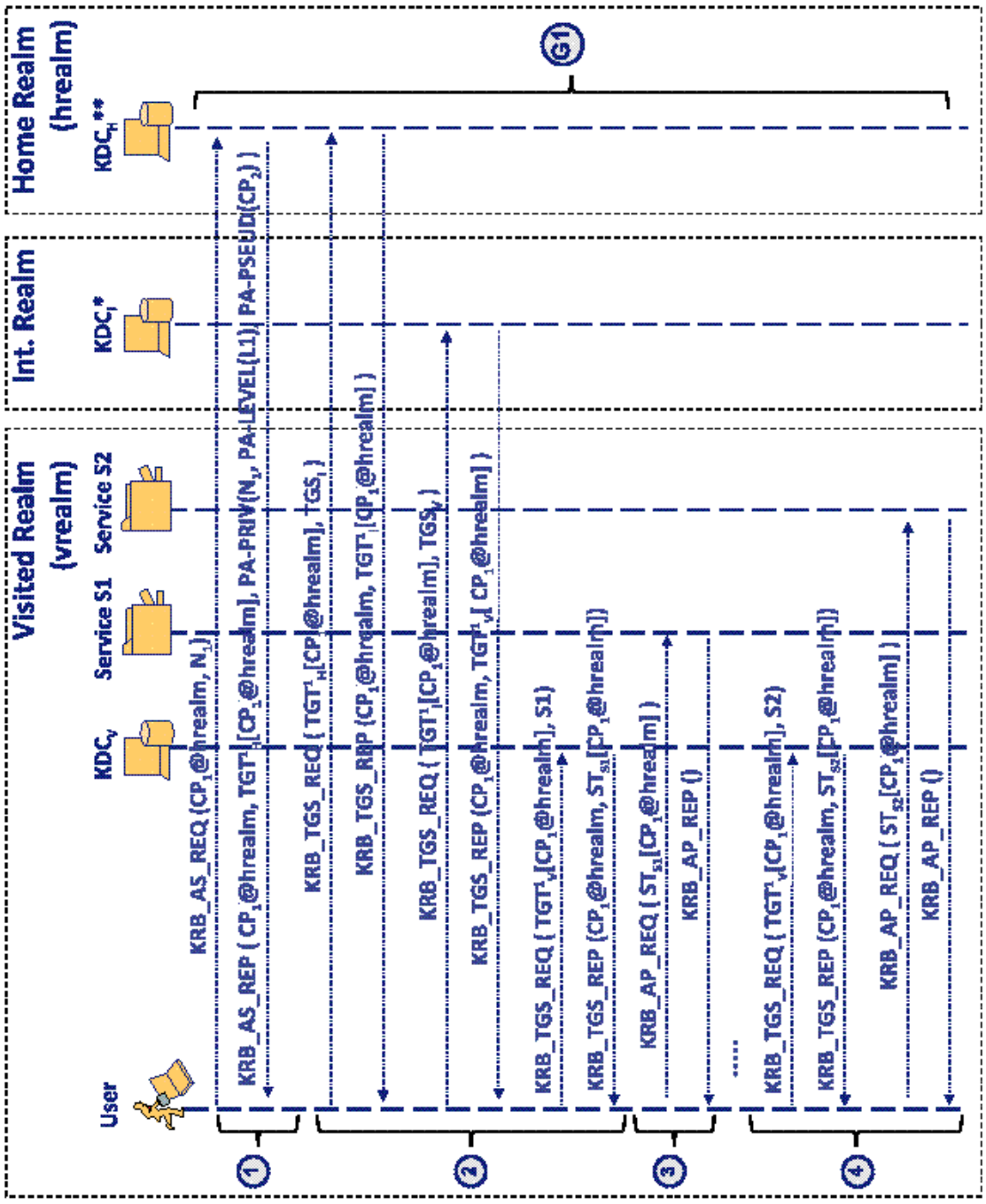


Figure 5.4: Privacy level 1: cross-realm scenario

realm, so an eavesdropper can determine that all Kerberos messages (denoted as group of messages *G1*) for the cross-realm operation and service accesses are performed by the same user. To solve this issue, which violates service access untraceability, privacy level 2 must be used instead.

Privacy Level 2

Figure 5.5 shows how privacy level 2 works in a cross-realm operation. Initially, if the client does not possess a valid TGT for the home KDC, it performs an AS exchange to obtain one following the process described in section 5.3.3 by using CP_1 as name *(1)*. In this step, the KDC generates an anonymous TGT (TGT^1_H) and a new pseudonym CP_2 for the next AS exchange.

After that, the client starts a cross-realm operation involving (three) TGS exchanges to obtain a service ticket for S1, i.e. one with every KDC (KDC_H, KDC_I and KDC_V). In the first TGS exchange with KDC_H *(2)*, the client solicits a cross-realm TGT for KDC_I. According to the privacy level 2 operation, KDC_H issues an anonymous TGT (TGT^1_I) for the KDC_I and a self-renewed TGT ($srTGT^1_H$). While the client replaces TGT^1_H (acquired in the initial AS exchange) by $srTGT^1_H$, the former is sent to KDC_I *(3)* in order to obtain a cross-realm TGT for KDC_V (TGT^1_V). Since it is assumed that KDC_I does not support privacy, it will process the request following the base Kerberos specification [74], that is: (a) since KDCs ignore unknown flags, the anonymous flag ($FlagA$) will not be set in the TGT^1_V issued by the intermediary KDC (KDC_I); and (b) since authorization data elements defined in our solution and contained in TGT^1_I are not recognized by intermediate KDCs (which may not support our privacy extensions), they propagate those data types to derivative tickets (TGT^1_V in this case). In summary, only authorization data is propagated through cross-realm TGTs.

Once the client acquires TGT^1_V, it is ready to perform the final TGS exchange with KDC_V *(4)*. On the reception of the KRB_TGS_REQ, KDC_V starts analyzing TGT^1_V. At first sight, it observes that is a cross-realm TGT issued to the well-known anonymous client (*anon@anon*). Assuming that KDC_V is privacy-enabled, more information can be extracted from the authorization data ($AD(L2, CP_1)$) contained in the ticket. With this information, KDC_V deduces that a client named CP_1 coming from realm *hrealm* is demanding privacy level 2 support. Therefore, following the privacy level 2 operation, KDC_V distributes an anonymous ST ($ST_{S1}[FlagA...$) and self-renewed TGT ($srTGT^1_V$). Note that KDC_V re-establishes the use of the anonymous flag in both tickets. Finally, the client can use the anonymous ST ST_{S1} to access the service *(5)*. If the service needs the client's pseudonym for charging purposes, it can obtain it from the authorization data in the ST_{S1}.

As we can observe, even if intermediary KDCs do not support privacy, the service access untraceability is respected. In particular, when the service access implies a cross-realm operation, an eavesdropper will be able to determine that all exchanges grouped by label *G1* belongs to the same client. Nevertheless, once the client communicates with the visited KDC, the latter provides a self-renewed TGT that allows traceability with subsequent

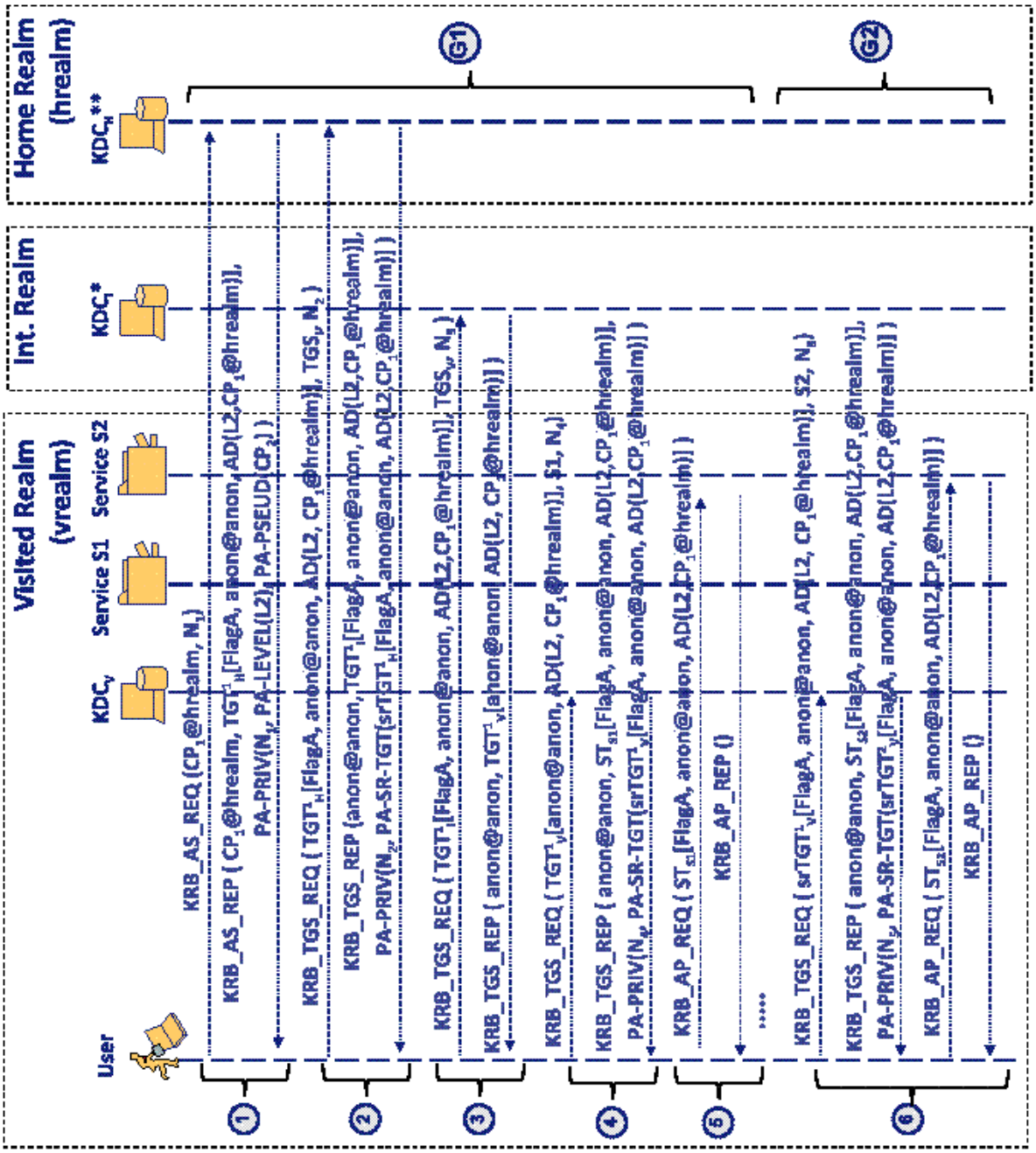


Figure 5.5: Privacy level 2: cross-realm scenario

access to a service (*G2*) located in the visited realm to be broken, by accomplishing *requirement 2*. Thus, when the client uses the self-renewed TGT to solicit a ST *(6)* to access another service (e.g. S2) in the visited realm, an eavesdropper is unable to deduce that the access is performed by the same anonymous client.

Privacy Level 3

Privacy level 3 offers the highest privacy protection since an eavesdropper cannot infer that two exchanges are performed by the same client. As explained in section 5.3.4, this objective is achieved thanks to the concept of fake ticket that hides from eavesdroppers both TGTs and STs delivered to the client.

More details are provided in Fig. 5.6 where, following the same scenario used in privacy level 2 (see Fig. 5.5), we describe a cross-realm operation enhanced with privacy level 3 extensions. Initially, the AS and TGS exchanges with the home KDC (KDC_H) are equal to those explained in section 5.3.4. As observed, an eavesdropper cannot infer that the anonymous user requesting a TGT for KDC_I *(2)* is the same that has been previously authenticated *(1)*. Furthermore, this first TGS exchange *(2)* cannot be related to the second TGS exchange *(3)* where, following the standard Kerberos protocol, the client contacts KDC_I to request a cross-realm TGT for KDC_V. Since KDC_I does not support our privacy extensions, TGT_V^1 is distributed in the KRB_TGS_REP without using the fake ticket feature, thus being visible for anyone. As a consequence, it allows eavesdroppers to relate the second *(3)* with the third TGS exchange *(4)*, where the client solicits a ST for service S1. Nevertheless, only these two exchanges are affected because KDC_V restores the use of privacy level 3 extensions and the KRB_TGS_REP message distributes ST_{S1} through the fake ticket extensions. For this reason, from the eavesdropper's point of view, access to service S1 *(5)* cannot be associated with previous exchanges. Finally, subsequent exchanges to access services in the visited realm *(6 and 7)* cannot be related to the same (anonymous) user thanks to the privacy level 3 extensions. As a result, thanks to the fake ticket feature, an eavesdropper tracing the communication cannot deduce that the different Kerberos exchanges (from *G1* to *G6*) are performed by the same anonymous user.

In conclusion, when privacy level 3 is applied to access services located in foreign realms, exchange untraceability prevents eavesdroppers to trace the whole process followed by the user to obtain a service ticket in the foreign realm. Therefore observers cannot obtain information regarding the roaming of the users like, for example, which are the preferred services solicited by foreign users in a realm. Furthermore, since we do not require intermediary KDCs to support our privacy extensions, our solution can be easily integrated in existing deployments. Despite this advantage allows eavesdroppers to trace those KRB_TGS_REQ/KRB_TGS_REP messages exchanged from the first intermediary KDC to the visited KDC, it is important to note that it neither knows the identity of the user nor the realm to which the user belongs to. Furthermore, since this situation only affects to this specific set of messages in the cross-realm infrastructure but not when accessing to the service, an eavesdropper cannot obtain valuable information to elaborate anonymous service access profiles. Therefore, the decision of not requiring intermediary

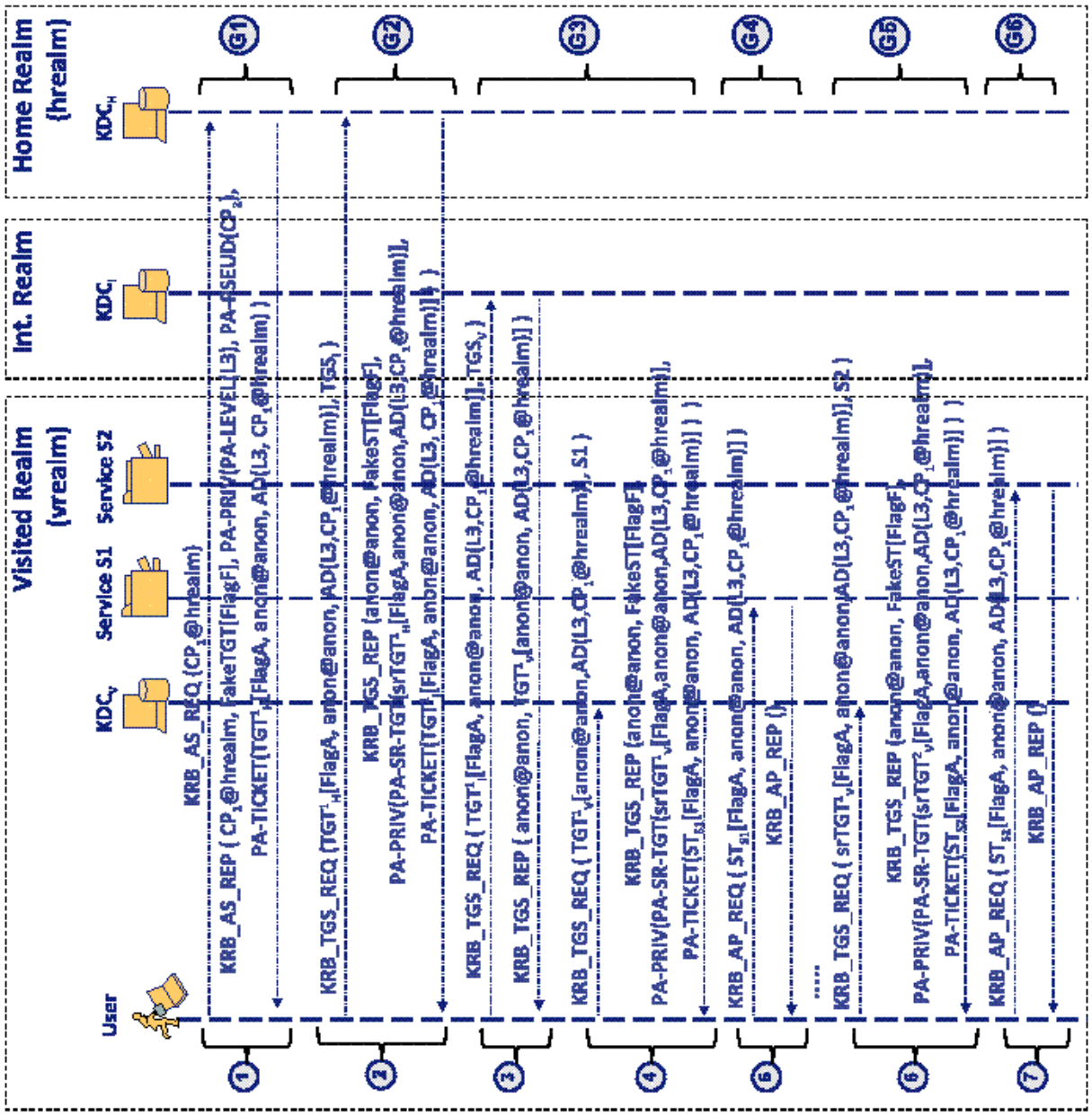


Figure 5.6: Privacy level 3: cross-realm scenario

KDCs to support privacy level 3 is a trade-off between deployment cost and privacy provided to the user.

5.4 Discussion

5.4.1 Privacy Levels Analysis

PrivaKERB is a privacy framework which operates in three different modes of privacy. This aspect gives flexibility to our solution since a specific level can be selected depending on the necessities of both the client and the KDC. In the following, we analyze and compare privacy levels (see Table 5.1) according to aspects like type of privacy supported or deployment cost in order to envisage under which conditions is better to apply a specific level of privacy.

Privacy level 1 is a trade-off between privacy provided to the user and deployment cost. In this level, the user remains anonymous by using periodically renewed pseudonyms instead of the real identity. This simple but effective solution requires the Kerberos protocol to be extended by defining three new padata types, one of which is especially destined to protect with confidentiality and integrity the information sent from the KDC to the client. Furthermore, the main advantage of level 1 is that these extensions only affect the AS exchange, so only the AS module of the home KDC needs to be updated. This feature becomes more important in cross-realm scenarios since neither intermediary nor visited KDCs need to be modified. For this reason, level 1 is suitable for those scenarios where it is desirable to minimize the deployment impact.

On the other hand, although users remain anonymous, just using pseudonyms enables eavesdroppers to trace the activity of a certain user during the period of time that it employs the same pseudonym and to extract anonymous profiles and behavioural patterns. This situation does not pose a risk for a particular client since profiles cannot be linked to its real identity. Nevertheless, it must be carefully considered by operators and service providers since observers are able to identify common behavioural patterns of their clients [162]. If this circumstance represents a serious concern, the use of privacy level 2 is recommended. As a consequence, level 2 is a qualitative improvement on level 1, since it offers not only user anonymity but also service access untraceability. Nevertheless, a higher privacy protection is achieved at the expense of increasing the deployment cost. In our opinion, this is assumable though, taking into account that intermediary KDCs are not required to be privacy-enabled servers in cross-realm operations.

Finally, privacy level 3 provides a full obfuscation of the protocol where eavesdroppers cannot relate Kerberos exchanges among them. This level improves the level 2 solution by introducing the fake ticket feature that allows issued tickets to be only observed and recovered by clients. Regarding the deployment cost, it is similar to level 2 since the same entities must be updated. However, the fake ticket feature introduced in level 3 requires the response messages (KRB_AS_REP and KRB_TGS_REP) to transport the fake ticket in addition to the real one delivered to the client. Although this situation increases the

size of the messages, the resulting process does not involve costly processing operations as we further analyze in section 5.5.2.

	LEVEL 1	LEVEL 2	LEVEL 3
Type of Privacy	Anonymity	Anonymity + Service Untraceability	Anonymity + Service Untraceability + Exchange Untraceability
Eavesdropper Knowledge	The behaviour of an anonymous user can be profiled while using the same TGT	The set of messages exchanged to access a service can be traced	Full obfuscation: Kerberos exchanges cannot be related.
Features	Pseudonyms	Pseudonyms, anonymous tickets and self-renewed TGTs	Pseudonyms, anonymous tickets, self-renewed TGTs and fake ticket
Kerberos Extensions	New padata types (*PA-LEVEL*, *PA-PSEUD*, *PA-PRIV*)	Level 1 extensions + well-known anonymous identity (*anon@anon*) + anonymous flag (*FlagA*) + new padata type (*PA-SR-TGT*) + new authorization data types (level and client's identity)	Level 2 extensions + fake flag (*FlagF*) + new padata type (*PA-TICKET*)
Deployment	Easier deployment. Only the home AS/KDC must be updated	Home and visited KDCs must be updated. Implementation of anonymous tickets and self-renewed TGTs.	Home and visited KDCs must be updated. Implementation of anonymous tickets, self-renewed TGTs and fake tickets.

Table 5.1: Privacy levels comparison and related issues

5.4.2 Kerberos Extensibility and Security

Our privacy framework for Kerberos enhances the basic protocol introducing new features based on the definition of new flags, pre-authentication and authorization data types which are standard mechanisms offered to designers for protocol extensibility. For this reason, PrivaKERB respects the Kerberos protocol operation (e.g., we do not define new messages) and the semantic of messages (e.g., message fields are not used for other purpose or messages are not modified with new fields).

On the other hand, another important aspect is to assure that the extensions proposed do not compromise the security of the protocol. In this regard, we have to clarify some interesting points regarding the PA-PRIV padata and self-renewed TGTs. The PA-PRIV padata is used in KRB_AS_REP and KRB_TGS_REP messages to confidentiality and integrity protect a sequence a padatas sent from the KDC to the client. To encrypt this information, we propose to follow the same process used by Kerberos to protect the *enc-part* field which contains information associated with the distributed ticket. Furthermore, we

propose to use the same key, which will depend on the Kerberos messages at hand. In particular, in a KRB_AS_REP, the client's long-term key or another key selected via pre-authentication mechanisms [169] will be selected. In a KRB_TGS_REP, the TGS session key or a TGS authenticator subkey will be used instead. Note that according to [170], it will be necessary to define new key usage numbers for these keys, though this is out of the scope of this work.

Unlike ordinary TGTs, self-renewed TGTs are generated and processed by the TGS module. To protect this new type of ticket we use the so-called TSRK (TGT Self-Renewal Key), a secret key generated by the TGS and not shared with any other entity. The TSRK can be pre-established or randomly created when the KDC is initialized for the first time. Therefore, the same security properties achieved with ordinary TGTs are also fulfilled with self-renewed TGTs.

5.4.3 Error Handling

In general, Kerberos controls abnormal situations that may arise during the protocol operation through the KRB_ERROR message. This message, returned by the KDC or service, is intended to inform the client about an error that occurred when processing the client's request. Given that the base Kerberos protocol defines an error handling mechanism, we consider that the most natural way to control privacy errors is to extend it by introducing new error types.

For example, a typical error could appear in a cross-realm scenario when the visited KDC (see Fig. 5.5) does not support the privacy level assigned to the user. In this situation, the KDC must respond to the client with a KRB_ERROR informing about the error and supported privacy levels.

However, the contents of the KRB_ERROR message are not protected [74]. Therefore, a client cannot detect replays or modifications and has no means to distinguish between a valid error message sent from the KDC and one sent by an attacker. This problem has already been highlighted in [171], and the IETF Kerberos Working Group is currently leading an effort to solve this security weakness (among others). A partial solution to signal an error related to privacy is to define a new padata PA-ERROR which can be included in the KRB_TGS_REQ/REP and KRB_AS_REQ/REP protected by a PA-PRIV padata. Nevertheless, we believe that to define privacy-specific errors will be the right choice once a complete resolution of this general problem is achieved in the context of the IETF Kerberos Working Group.

Additionally, the client may also interact with a KDC that does not support our privacy extensions. It is obvious that at least the home KDC will support privacy because, otherwise, the client will not have a valid pseudonym (provided by its home realm) to start the AS exchange. Nevertheless, in a cross-realm scenario, the visited KDC may not support privacy of the client. In this situation, and taking that we have used the mechanism of extensibility provided by Kerberos, the visited KDC will process the request following the standard Kerberos operation. This means that it will ignore unknown flags and authorization data types contained in the presented ticket and answer with a

KRB_TGS_REP, which does not include any of the our privacy extensions. Then, the client can notice this situation by just verifying that the KRB_TGS_REP and can decide to abort the process since privacy level in which she is enrolled is not provided.

5.4.4 Pseudonym Management

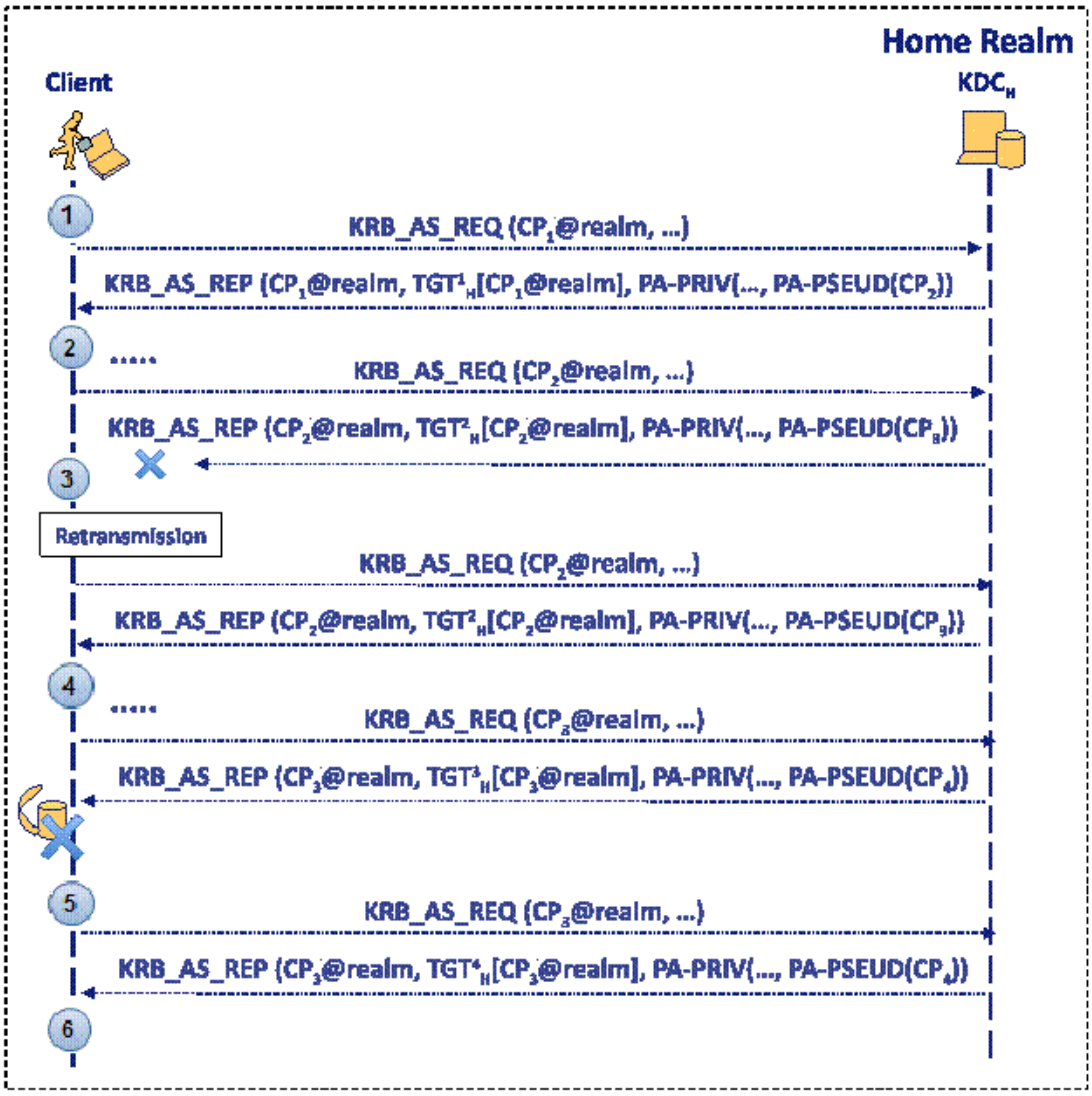
PrivaKERB ensures user anonymity by means of pseudonyms which are dynamically assigned and valid during the home TGT lifetime. As described in section 5.3.2, when a client participates in an AS exchange uses a pseudonym as client's identity and a new one is distributed for the next AS exchange. Therefore, two different pseudonyms are associated with a specific user: (a) the current pseudonym (*cpseud*), which is currently being used as client's identity (e.g., CP_i), and (b) the new pseudonym (*npseud*), which will be used as client's identity in the next AS exchange to solicit a new TGT (e.g., CP_{i+1}). The 2-tuple of pseudonyms (*cpseud*, *npseud*) are maintained by both the client and the home KDC. Whereas the *cpseud* can be maintained in volatile memory in the client, *npseud* is stored in permanent memory (i.e. a smart card) so that it is available when client's device is reinitialized. On the other side, the KDC stores the 2-tuple in its database as identities associated with a particular user, though only the home KDC knows the association between these pseudonyms and the client's real identity.

To illustrate this, let us assume that the client and KDC shares the 2-tuple (CP_{i-1}, CP_i) and the client starts an AS exchange by using CP_i as client's identity. Upon receiving the KRB_AS_REQ and verify that CP_i is the expected pseudonym, the KDC generates a new *npseud* (CP_{i+1}) to be used by the client in the next AS exchange, and updates the 2-tuple to (CP_i, CP_{i+1}). Then it sends the KRB_AS_REP which contains the CP_{i+1} to the client. In the reception of CP_{i+1}, the client also updates the 2-tuple to (CP_i, CP_{i+1}).

As we may observe, it is important to note that the KDC cannot confirm that client really received and stored the distributed CP_{i+1}. The only way to confirm this is when the client really uses CP_{i+1} as identity in a subsequent AS exchange. If the KDC receives the CP_i instead of CP_{i+1}, it can determine that client did not receive the CP_{i+1}. In this case, we propose the KDC to re-distribute the same CP_{i+1} securely instead of generating a new pseudonym. Subsequently, the client will be identified as CP_{i+1} while the acquired TGT is valid.

As an example, Fig. 5.7 shows the management of pseudonyms in two different abnormal situations. Let us assume a user is registered in a Kerberos realm administered by KDC_H and privacy support is enabled in the client's subscription. Therefore, an initial pseudonym CP_1 is provided to the client (*1*). This pseudonym is used in the first AS exchange to solicit the first home TGT. When KDC_H receives the KRB_AS_REQ message, it issues TGT_H^1 (associated with CP_1) and randomly generates a new pseudonym CP_2 which are sent in the KRB_AS_REP. Once the AS exchange is successfully completed (*2*), the 2-tuple of pseudonyms are updated to (CP_1, CP_2) in both entities.

When TGT_H^1 expires, the client engages in a new AS exchange to solicit a new TGT using pseudonym CP_2 as client's identity. KDC_H processes the KRB_AS_REQ issuing TGT_H^3 (associated with CP_2) and generating a new pseudonym CP_3 to be used in the next



Pseudonym Management						
	1	2	3	4	5	6
Client	(-,CP$_1$)	(CP$_1$,CP$_2$)	(CP$_1$,CP$_2$)	(CP$_2$,CP$_3$)	(-,CP$_3$)	(CP$_3$,CP$_4$)
KDC	(-,CP$_1$)	(CP$_1$,CP$_2$)	(CP$_2$,CP$_3$)	(CP$_2$,CP$_3$)	(CP$_3$,CP$_4$)	(CP$_3$,CP$_4$)

Figure 5.7: Pseudonym management in PrivaKERB

AS exchange. Once the request is processed and the KRB_AS_REP message is sent to the client, the KDC updates the pseudonyms tuple to (CP_2, CP_3) *(3)*. Due to a problem in the communication, the KRB_AS_REP message is not received by the client and, consequently, the tuple of pseudonyms cannot be updated. Following the Kerberos protocol operation, the client retransmits the same KRB_AS_REQ after a specific period of time. In this case, a special situation occurs in the KDC since it receives a KRB_AS_REQ using CP_2 as

client's identity while CP_3 is the new pseudonym expected to be used by the client. In this situation, the KDC follows the default behaviour defined in the Kerberos specification [74] retransmitting the response previously sent. Now, on the reception of the KRB_AS_REP, the client can update the 2-tuple to (CP_2, CP_3) *(4)*.

In the subsequent AS exchange, the client renews CP_2 and starts using CP_3 as client's identity in the KRB_AS_REQ. The KDC follows the same procedure described in previous exchanges: it issues TGT_H^3 associated with CP_3 and generates a new pseudonym CP_4. Once the request is processed and the KRB_AS_REP message is sent to the client, the KDC updates the pseudonym tuple to (CP_3, CP_4) *(5)*. Nevertheless, while the client is processing the response, a problem arises before the new pseudonym CP_4 is stored in permanent memory and the device is turned off. Once the device is re-initialized, CP_3 is recovered (recall that only new pseudonyms are stored in permanent memory) by the client to start an AS exchange with its home KDC. When KDC_H receives this request, it observes a new request (not a retransmission of the previous request message) coming from the client using CP_3 instead of the expected pseudonym CP_4. In this situation, KDC_H process the request following the privacy extensions but a new pseudonym is not generated and CP_4 is re-distributed to the client. After that, both the client and the KDC share the same tuple of pseudonyms (CP_3, CP_4) *(6)* and both are synchronized.

5.5 Performance Evaluation

In this chapter we propose some extensions to the Kerberos protocol that allow to preserve the privacy of the user at different levels. It is expected that these extensions will introduce some additional latency in respect to the base protocol, thus expecting access and/or service time to increase. This aspect is more important in the context of NGNs where service quality must be optimized in terms of access time and continuity. Therefore, the key question is whether the additional cost imposed by our privacy extensions is affordable for next generation networks and for a fast re-authentication operation. To perform this evaluation, we have developed an implementation prototype that is used to evaluate the performance of our privacy framework.

5.5.1 Deployed Testbed and Implementation Details

To implement privacy levels 1, 2 and 3, we have selected the open-source MIT Kerberos implementation v.1.6.3 [142]. The main implementation work has focused on extending the message construction and processing message engine since most changes affect the information transported by messages. Additionally, special effort has been devoted to the encoding and decoding routines in order to support new padatas, flags and authorization data types described in appendix A.1, A.2 and A.3, respectively. These modifications are available through the Kerberos library for clients, services and KDC.

To conduct real experiments we have developed different programs that simulate a generic client and service employing the Kerberos *Application Programming Interface*

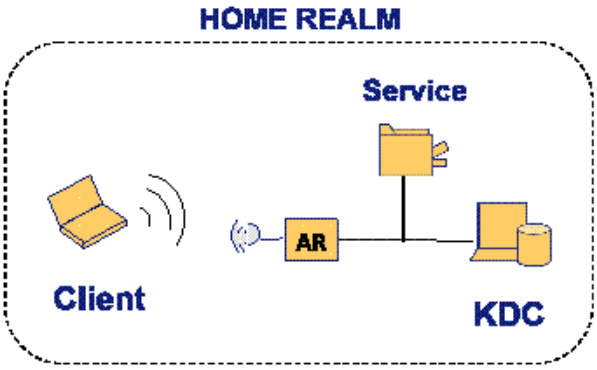
(API). Both programs interact with a KDC (already implemented by the MIT Kerberos distribution) in charge of distributing tickets to the client. The client implements all the Kerberos exchanges in order to be successfully authenticated by the service performing the next processes: (1) TGT acquisition through an AS exchange; (2) ST acquisition through one or several TGS exchanges, (3) service access through an AP exchange. As regards the service, we have implemented the privacy extensions that must implement real services in order to understand our new authorization data types (if present in the ST) containing privacy information of the user.

We have built a basic and generic network architecture that allows us to test different situations that may occur in a real mobile environment. The experimental testbed comprises the elements shown in Table 5.2 and two scenarios as depicted in Fig 5.8. Scenario I is to experiment with single-realm cases and the Scenario II for cross-realm, where the three KDCs are geographically distributed at different places (two of them in Spain and the third in Greece) by simulating three different administrative realms (average roundtrip times between realms are specified in section 5.5.2).

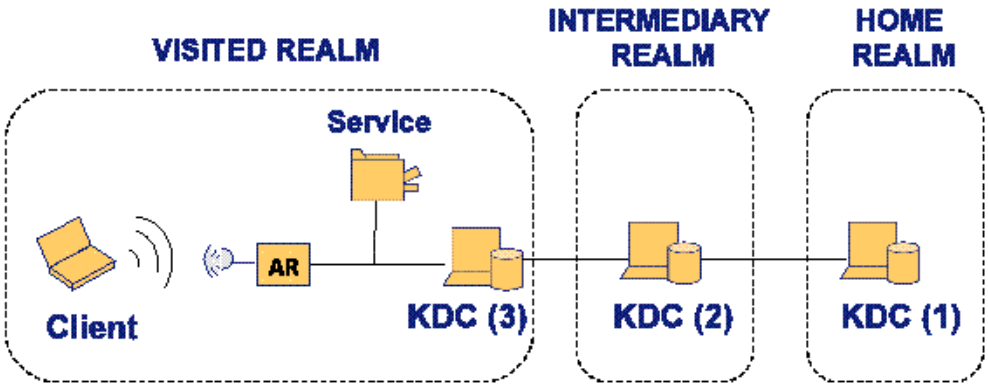
Role	**CPU Type**	**Freq(MHz)**	**RAM(MB)**
Kerberos client	VIA Nehemiah	1,200	488
Kerberos service	VIA Nehemiah	1,200	488
Wireless-G Broadband Router	Linksys WRT54GL	200	16
KDC	Pentium 4	3,200	1024
KDC	Pentium 4	3,000	512
KDC	Pentium 4	3,000	512

Table 5.2: Testbed machines

Finally, for the sake of completeness, we would like to highlight some important implementation aspects. First, we use the *KerberosString* format to generate pseudonyms (which act as principal names) following the instructions in [74]. This format is a *GeneralString* ASN.1 data type that is constrained to contain characters in *IA5String*. Since control characters should not be used in principal names, only the 95 printable characters of the IA5String [119] alphabet are available. To avoid collisions, we randomly produce 8 character length principal names which provides an enough number of pseudonyms. Second, in addition to the shared secret keys defined in [74], the KDC is also configured with a TSRK that is stored in the KDC database. The TSRK is a secret key only known by the TGS and independent of the user. Therefore, this key is not disclosed to any third parties and can only be recovered by the TGS module when it issues or receives self-renewed TGTs. Third, regarding the process followed to generate fake tickets in privacy level 3, we have implemented a fake ticket by filling the *enc-part* field of the ticket with random data. Only the *sname* and *realm* fields (sent in cleartext) are initialized with the true service's identity. Similarly, the *EncKDCRepPart* field of the KRB_AS_REP/KRB_TGS_REP associated with the fake ticket is randomly initialized, except for the *flags* field that is expected to indicate which flags (fake flag in this case)



(a) Scenario I



(b) Scenario II

Figure 5.8: Deployed Testbed

are set in the issued ticket. Fourth, with regard to the cryptographic operations, we have used the well-known *Advanced Encryption Standard* (AES) algorithm with a block cipher size of 128 bits and cipher-block chaining (CBC) operation mode [170, 172]. When integrity protection is required, we employ the HMAC-SHA1-96 function [172,173]. Finally, the configuration used to test the implementation disables enhancements to the protocol like pre-authentication and the communication between the different entities is performed through UDP sockets.

5.5.2 Performance Analysis

Using the aforementioned testbed we evaluate the penalty introduced by our privacy framework. We use the term *standard Kerberos* to refer the execution of the original version of the Kerberos protocol such as defined in [74] (that is, without our extensions); and the term *PrivaKERB* to denote our privacy-enhanced Kerberos implementation with privacy support. We carry out this comparison in two different scenarios depending on whether the client solicits access to a service controlled by its home KDC (single-realm scenario) or by a visited KDC (cross-realm scenario). Since we assume that the client needs to be authenticated against the KDC, the client performs the three Kerberos exchanges: AS, TGS and AP. Every service access is executed around 500 times with both the standard and the privacy-enhanced Kerberos. When testing the standard Kerberos configuration we collect the following information:

- *Message length.* This is the length of a specific message transmitted over the network including IP, data link and physical headers.

- *Network time.* This metric represents (as a 95% confidence interval) the time devoted to transmitting messages over the network.

- *Message processing time.* This measures the time devoted by a certain entity (represented as a 95% confidence interval) to process a message, since it is received on the network interface until a response is sent. Therefore, for example, IP routing time is also included.

- *Exchange time.* This collects a 95% confidence interval that contains the total time required by a client to complete a specific Kerberos exchange. This metric comprises both network and message processing times.

For the privacy-enhanced solution we employ the same metrics. Nevertheless, we specifically measure the *privacy processing time* to perform an accurate measurement of the additional latency. That is, the additional time required by each entity to perform the additional privacy-related tasks.

Scenario I: Single-realm Case

We assume a client attempting to access a service for the first time with any valid TGT. Thus, the client will need to perform the three different Kerberos exchanges in order to access the service. To test this situation we develop the network architecture shown in Fig. 5.8(a), where we simulate a client employing a wireless connection through a wireless access router (AR).

Table 5.3 collects results obtained for the standard Kerberos protocol. These values as used as base reference to contrast with the overhead introduced by our privacy extensions. As observed, we provide the specific values measured for the different metrics indicating confidence intervals when measuring times. However, to facilitate the analysis of our privacy solution and easily extract conclusions, results obtained for PrivaKERB are summarized in Fig. 5.9 to perform a graphical comparison among the different privacy levels. As observed, only mean values are graphically represented for the measured values. Nevertheless, as reference, in Tables 5.4, 5.5 and 5.6 we precise the specific values (indicating confidence intervals where appropriate) measured for PrivaKERB operating in the different privacy levels through different tables.

Figure 5.9(a) shows the mean time required by the client to complete each Kerberos exchange using both standard Kerberos and our privacy extensions. First, regarding the standard Kerberos, we notice that the AS exchange requires far more time (about 9 times) than the other exchanges. Through information obtained from the source code, we detected that most of this time is consumed to derive the client's secret key from the password. In fact, this heavy process is not required in other exchanges since the shared secret key to communicate with the TGS/KDC or service can be directly recovered from the client's database. Regarding the overhead introduced by PrivaKERB, in general we can observe that the use of self-renewed TGTs (introduced in level 2) produces a small increment in the TGS exchange, remaining both the AS and AP exchanges below similar values. Indeed, analyzing times related to level 1, we observe that a small latency of $\approx$0.2 ms is introduced in the AS exchange. As expected, since this level does not introduce any extension to the TGS and AP exchanges, these times remain below values similar to the standard Kerberos. Conversely, levels 2 and 3 require some additional time in every exchange. More precisely, level 2 increases the AS exchange in $\approx$0.35 ms. This time is higher in the TGS exchange, where about 1.4 ms are required to complete the privacy extensions. These values are higher for privacy level 3, where the additional latency is close to 0.89 ms and 2.01 ms, respectively. Instead, the latency introduced in the AP exchange is the same in both cases, close to 0.02 ms.

The impact of the privacy-enhanced solution over each entity in terms of computing time is depicted in Fig. 5.9(b). More specifically, we collect the latency spent in executing the privacy extensions. These values must be compared with the processing time devoted by each entity to complete every exchange (see Table 5.3) in the standard Kerberos case. Level 1 only produces a time penalty to the client of $\approx$0.15 ms and to KDC of $\approx$0.07 ms in the AS exchange. These values are slightly higher for privacy level 2, where the client and KDC require about 0.91 ms and 0.18 ms to handle anonymous tickets and self-renewed

	Standard Kerberos					
	Exchange Time (ms)	*Message Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	59.479 ± 0.041	*Client*	57.791 ± 0.103	*KRB_AS_REQ*	204	1.075 ± 0.023
		KDC	0.503 ± 0.031	*KRB_AS_REP*	584	
TGS Exchange	6.545 ± 0.023	*Client*	3.618 ± 0.052	*KRB_TGS_REQ*	624	1.295 ± 0.028
		KDC	1.740 ± 0.055	*KRB_TGS_REP*	611	
AP Exchange	6.987 ± 0.041	*Client*	2.583 ± 0.048	*KRB_AP_REQ*	451	0.915 ± 0.029
		Service	3.491 ± 0.042	*KRB_AP_REP*	127	

Table 5.3: Results for standard Kerberos in scenario I

	Privacy level 1 (Scenario I)					
	Exchange Time (ms)	*Privacy Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	59.685 ± 0.052	*Client*	0.148 ± 0.001	*KRB_AS_REQ*	204	1.115 ± 0.015
		KDC	0.067 ± 0.002	*KRB_AS_REP*	695	
TGS Exchange	6.594 ± 0.023	*Client*	0	*KRB_TGS_REQ*	624	1.307 ± 0.037
		KDC	0	*KRB_TGS_REP*	611	
AP Exchange	6.989 ± 0.035	*Client*	0	*KRB_AP_REQ*	451	0.911 ± 0.021
		Service	0	*KRB_AP_REP*	127	

Table 5.4: Results for privacy level 1 in scenario I

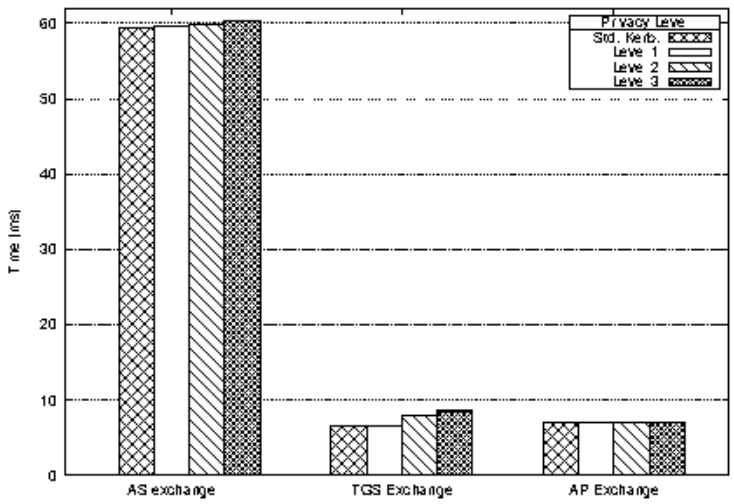
	Privacy level 2 (Scenario I)					
	Exchange Time (ms)	*Privacy Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	59.832 ± 0.046	*Client*	0.150 ± 0.002	*KRB_AS_REQ*	204	1.242 ± 0.014
		KDC	0.075 ± 0.001	*KRB_AS_REP*	762	
TGS Exchange	7.698 ± 0.039	*Client*	0.907 ± 0.001	*KRB_TGS_REQ*	692	1.585 ± 0.044
		KDC	0.183 ± 0.001	*KRB_TGS_REP*	1275	
AP Exchange	7.010 ± 0.045	*Client*	0	*KRB_AP_REQ*	515	0.921 ± 0.028
		Service	0.021 ± 0.001	*KRB_AP_REP*	127	

Table 5.5: Results for privacy level 2 in scenario I

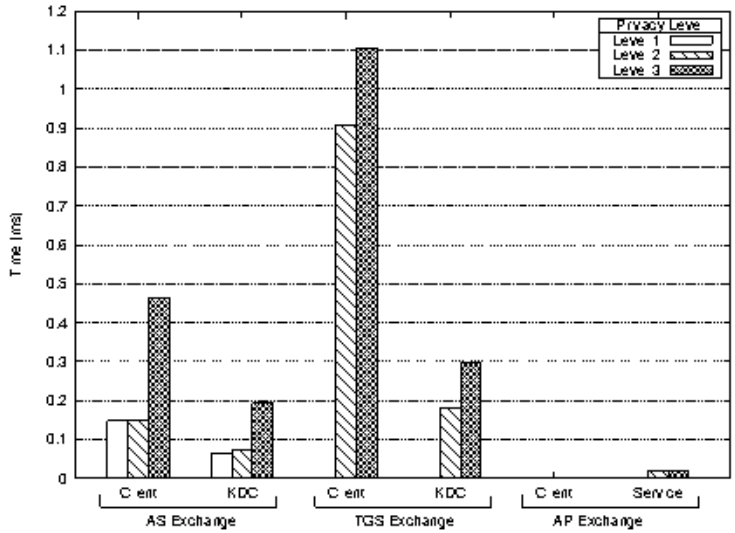
	Privacy level 3 (Scenario I)					
	Exchange Time (ms)	*Privacy Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	60.362 ± 0.041	*Client*	0.465 ± 0.002	*KRB_AS_REQ*	204	1.461 ± 0.017
		KDC	0.193 ± 0.001	*KRB_AS_REP*	1168	
TGS Exchange	8.563 ± 0.023	*Client*	1.106 ± 0.001	*KRB_TGS_REQ*	692	1.802 ± 0.039
		KDC	0.298 ± 0.001	*KRB_TGS_REP*	1514+181 *	
AP Exchange	7.008 ± 0.039	*Client*	0	*KRB_AP_REQ*	515	0.918 ± 0.019
		Service	0.020 ± 0.001	*KRB_AP_REP*	127	

* Message sent as two fragmented UDP messages

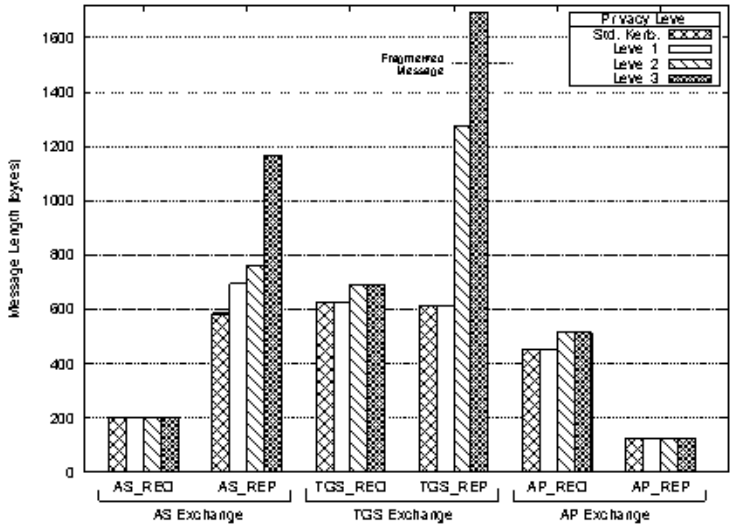
Table 5.6: Results for privacy level 3 in scenario I



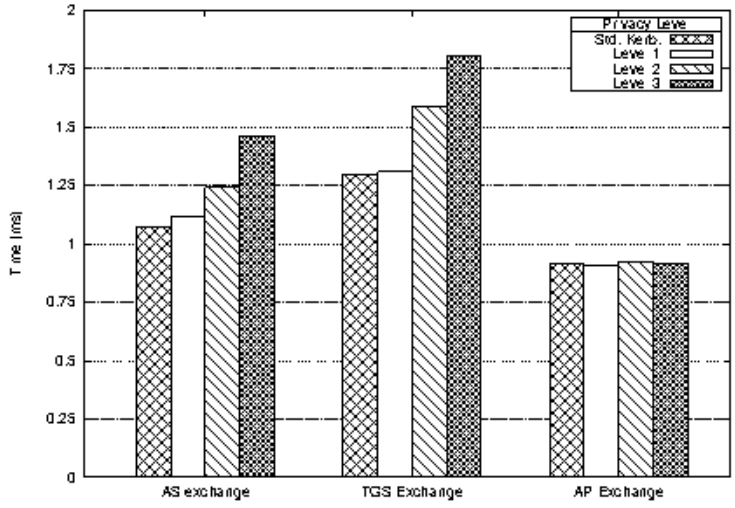
(a) Mean Exchange Time (ms)



(b) Mean Privacy Processing Time (ms)



(c) Message Length (bytes)



(d) Mean Network Time (ms)

Figure 5.9: Results for PrivaKERB in scenario I

TGTs. This time devoted to complete the privacy level 2 tasks increases in the TGS exchange where the client and KDC need ≈0.9 ms and ≈0.2 ms, respectively. Regarding the AP exchange, the service only needs some extra time (≈0.2 ms) to check the new authorization data types contained in the ST presented by the client. This situation also occurs in privacy level 3, where client and KDC require much time to complete the privacy tasks. Nevertheless, in the worst case (TGS exchange), this augment is ≈1.1 ms for the client and ≈0.3 ms for the KDC.

Finally, since the privacy-enhanced solution requires additional information to be exchanged between entities, another aspect we have measured is the message size and network times that may affect the bandwidth consumption. While Fig. 5.9(c) provides the size of those messages involved in the different Kerberos exchanges, Fig. 5.9(d) specifies the mean network time devoted to transmission and propagation over the network. In comparison with standard Kerberos, level 1 introduces the lowest network overload since only the KRB_AS_REP message increases in ≈110 bytes. Therefore, while we can appreciate an increment close to 0.04 ms in the AS exchange, network times in both TGS and AP exchanges remain under similar values. In contrast, levels 2 and 3 not only affects the size of the AS exchange messages but also the TGS and AP exchanges. In level 2, an appreciable increment is observed in the KRB_AS_REP (≈180 bytes) and KRB_TGS_REP (≈660 bytes) due to the presence of self-renewed TGTs. Additionally, we note that the use of anonymous tickets increases the message length ≈60 bytes (e.g., see KRB_AP_REQ). However, we observe that these sizes correspond to an insignificant increment of ≈0.30 ms (TGS exchange) in network time. On the other hand, as expected, level 3 represents the worst case. Compared with level 2, the use of the fake ticket feature significantly increases the size of those messages where the KDC delivers a ticket to the client. While the KRB_AS_REP is up to 1168 bytes, the KRB_TGS_REP is sent as two fragmented UDP packets of 1514 and 181 bytes, respectively. Note that the latter contains a fake ST, the real ST and the self-renewed TGT. Nevertheless, these increments do not provoke an appreciable penalization in the time required to transmit the messages over the network. As observed, compared with standard Kerberos, the network time required to complete the AS and TGS exchanges following the level 3 extensions only increases ≈0.39 ms and ≈0.5 ms, respectively.

Scenario II: Cross-realm Case

In this second scenario our intention is to analyze the behaviour of our privacy-enhanced solution during a cross-realm scenario. That is, when the client requests access to a service controlled by a foreign KDC. As in previous section 5.5.2, we suppose a user that initializes its device and access to a kerberized service for the first time. In the network architecture for this scenario (depicted in Fig. 5.8(b)) three different and geographically separated KDCs are deployed to simulate a cross-realm operation. The average roundtrip time between the home and visited network is ≈104 ms and ≈117 ms between the visited and intermediate networks, although these values may only be considered as an indication. The client will follow the authentication path from the home to the visited KDC through an intermediary

one. It is important to mention that all tests performed over this scenario, even in the privacy-enhanced configuration, the intermediary KDC deploys the standard MIT Kerberos implementation without our privacy extensions.

Table 5.7 shows the measurements obtained for the standard Kerberos protocol. It collects means values and confidence intervals of the exchange time, message processing time, message sizes and network times. Please note that, TGS Exchange 1, 2, 3 represents TGS exchanges for Home KDC, Intermediate KDC and Visited KDC respectively. For the PrivaKERB solution, results are summarized in Fig. 5.10 by graphs (detailed measured values are provided in Tables 5.8, 5.9 and 5.10).

Starting with the mean exchange times depicted in Fig. 5.10(a), we observe that level 1 introduces a small latency of ≈0.5 ms in the AS exchange. Since neither TGS nor AP exchange are extended at this level, times obtained for the other exchanges are similar to the standard Kerberos case. On the other hand, when analyzing measurements from levels 2 and 3, we observe that, except "TGS exchange 2" performed with the intermediary KDC, all times include an additional latency compared with standard Kerberos. Although these penalizations fluctuate around values between 1-2 ms, the impact in the final time is insignificant, especially in those messages exchanged with the home KDC. For example, the additional time to perform "TGS exchange 1" enabling level 3 extensions is ≈2.9 ms, which increases the total exchange time by only ≈2.5%.

Focusing on the processing time results (see Tab. 5.7 and Fig. 5.10(b)), we observe that the intermediary KDC does not perform any privacy task (columns of "TGS exchange 2" are set to 0). Recall that our privacy framework does not require intermediary KDCs to be privacy-enabled. Comparing the three privacy levels, we can conclude that level 1 is the most lightweight since it only overloads the AS exchange with 0.15 ms (client) and 0.11 ms (KDC) to execute the privacy extensions. On the contrary, level 2 introduces some extra computing time in every exchange with the home and visited KDC. The highest times are found for the client that devotes ≈0.9 ms to complete the "TGS Exchange 3" privacy tasks. On the other hand, KDCs attend to a privacy-enhanced request in less than 0.3 ms. Finally, level 3 follows the same behaviour but requires more time to cope with all privacy extensions. Nevertheless, penalization times continue fluctuating under inappreciable values. For example, in the worst case, the privacy process time for the client and KDC are ≈1.15 ms and ≈0.46 ms, respectively.

Regarding the message sizes (Fig. 5.10(c)) and network times (Fig. 5.10(d)), the same conclusions as extracted for the single-realm scenario can be drawn here. On examining the message sizes, we see that level 1 only requires the additional transmission of ≈110 bytes in the KRB_AS_REP message. Consequently, network times for level 1 are similar to that of standard Kerberos, detecting only an increment of ≈0.25 ms in the AS exchange. Conversely, the use of anonymous tickets in level 2 increases the size of every message transporting a ticket, even those exchanged with the intermediary KDC. This increment is even higher when the home or visited KDC sends the client a self-renewed TGT, where the KRB_TGS_REP is up to 1200-1300 bytes. However, that extra time to transmit this information over the network is ≈1.1 ms in the worst case (TGS exchange 1). Similar conclusions can be obtained for privacy level 3, with the particularity that more bytes are

	Standard Kerberos					
	Exchange Time (ms)	*Message Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	162.717 ± 0.061	*Client*	57.860 ± 0.087	*KRB_AS_REQ*	204	104.250 ± 0.067
		KDC	0.608 ± 0.037	*KRB_AS_REP*	584	
TGS Exchange 1	111.238 ± 0.097	*Client*	3.671 ± 0.111	*KRB_TGS_REQ*	602	104.723 ± 0.061
		KDC	2.844 ± 0.027	*KRB_TGS_REP*	564	
TGS Exchange 2	124.583 ± 0.128	*Client*	3.615 ± 0.104	*KRB_TGS_REQ*	606	117.653 ± 0.052
		KDC	2.810 ± 0.056	*KRB_TGS_REP*	609	
TGS Exchange 3	6.756 ± 0.110	*Client*	3.770 ± 0.034	*KRB_TGS_REQ*	652	1.291 ± 0.041
		KDC	1.712 ± 0.026	*KRB_TGS_REP*	622	
AP Exchange	6.964 ± 0.045	*Client*	2.535 ± 0.051	*KRB_AP_REQ*	460	0.917 ± 0.039
		Service	3.486 ± 0.043	*KRB_AP_REP*	127	

Table 5.7: Results for standard Kerberos in scenario II

	Privacy Level 1 (Scenario II)					
	Exchange Time (ms)	*Privacy Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	163.218 ± 0.169	*Client*	0.148 ± 0.021	*KRB_AS_REQ*	204	104.544 ± 0.054
		KDC	0.106 ± 0.003	*KRB_AS_REP*	695	
TGS Exchange 1	111.155 ± 0.073	*Client*	0	*KRB_TGS_REQ*	602	104.809 ± 0.087
		KDC	0	*KRB_TGS_REP*	564	
TGS Exchange 2	124.652 ± 0.094	*Client*	0	*KRB_TGS_REQ*	606	117.610 ± 0.057
		KDC	0	*KRB_TGS_REP*	609	
TGS Exchange 3	6.773 ± 0.094	*Client*	0	*KRB_TGS_REQ*	652	1.309 ± 0.064
		KDC	0	*KRB_TGS_REP*	622	
AP Exchange	6.961 ± 0.031	*Client*	0	*KRB_AP_REQ*	460	0.913 ± 0.024
		Service	0	*KRB_AP_REP*	127	

Table 5.8: Results for privacy level 1 in scenario II

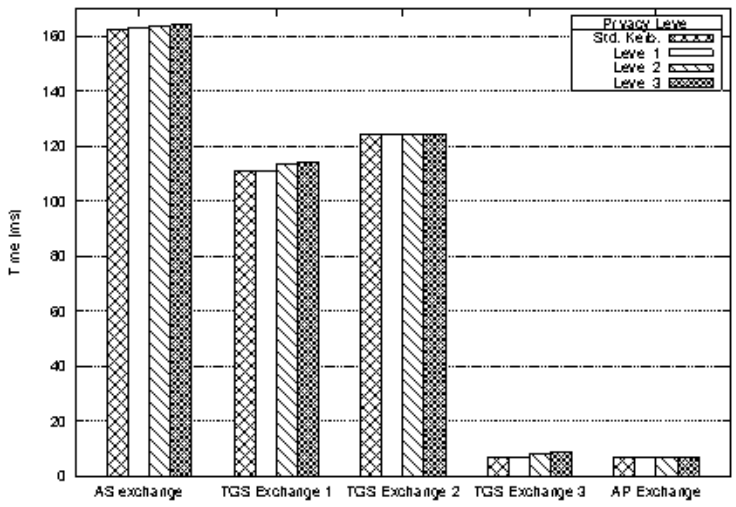
	Privacy level 2 (Scenario II)					
	Exchange Time (ms)	*Privacy Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	163.470 ± 0.115	*Client*	0.152 ± 0.001	*KRB_AS_REQ*	204	105.103 ± 0.064
		KDC	0.136 ± 0.003	*KRB_AS_REP*	762	
TGS Exchange 1	113.228 ± 0.073	*Client*	0.839 ± 0.001	*KRB_TGS_REQ*	670	105.877 ± 0.066
		KDC	0.272 ± 0.004	*KRB_TGS_REP*	1231	
TGS Exchange 2	124.301 ± 0.123	*Client*	0	*KRB_TGS_REQ*	674	117.680 ± 0.066
		KDC	0	*KRB_TGS_REP*	677	
TGS Exchange 3	8.201 ± 0.063	*Client*	0.895 ±0.002	*KRB_TGS_REQ*	720	1.592 ± 0.037
		KDC	0.182 ± 0.001	*KRB_TGS_REP*	1337	
AP Exchange	6.986 ± 0.024	*Client*	0	*KRB_AP_REQ*	547	0.922 ± 0.031
		Service	0.019 ±0.001	*KRB_AP_REP*	127	

Table 5.9: Results for privacy level 2 in scenario II

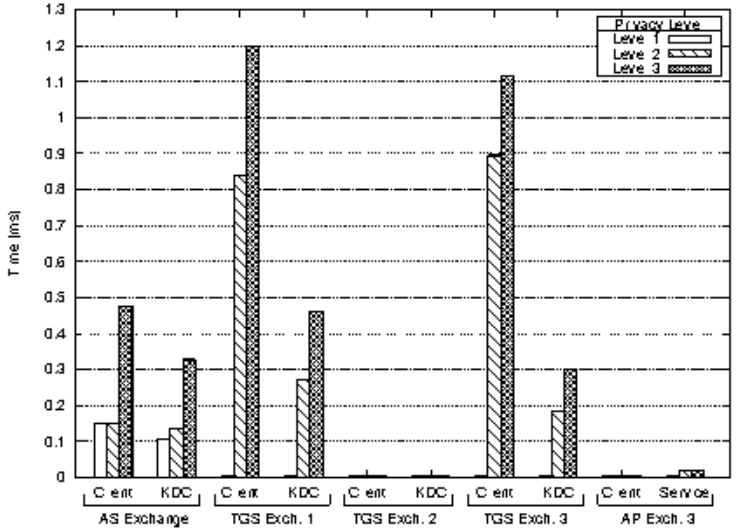
	Privacy level 3 (Scenario II)					
	Exchange Time (ms)	*Privacy Processing Time (ms)*		*Message Length (bytes)*		*Network Time (ms)*
AS Exchange	164.132 ± 0.084	*Client*	0.475 ± 0.002	*KRB_AS_REQ*	204	105.413 ± 0.038
		KDC	0.327 ± 0.004	*KRB_AS_REP*	1168	
TGS Exchange 1	114.120 ± 0.084	*Client*	1.198 ± 0.002	*KRB_TGS_REQ*	670	106.266 ± 0.075
		KDC	0.460 ± 0.004	*KRB_TGS_REP*	1506+145 *	
TGS Exchange 2	124.408 ± 0.142	*Client*	0	*KRB_TGS_REQ*	674	117.730 ± 0.090
		KDC	0	*KRB_TGS_REP*	677	
TGS Exchange 3	8.791 ± 0.078	*Client*	1.115 ±0.002	*KRB_TGS_REQ*	720	1.879 ± 0.068
		KDC	0.301 ± 0.001	*KRB_TGS_REP*	1514+247 *	
AP Exchange	6.983 ± 0.021	*Client*	0	*KRB_AP_REQ*	547	0.918 ± 0.028
		Service	0.020 ±0.001	*KRB_AP_REP*	127	

* Message sent as two fragmented UDP messages

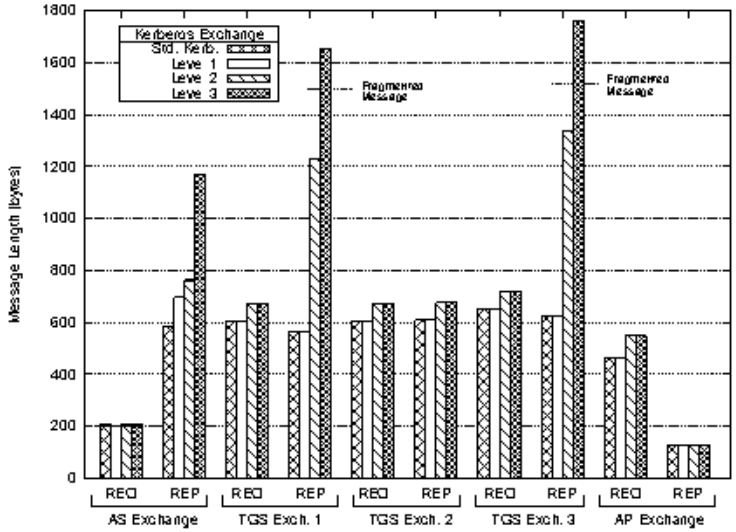
Table 5.10: Results for privacy level 3 in scenario II



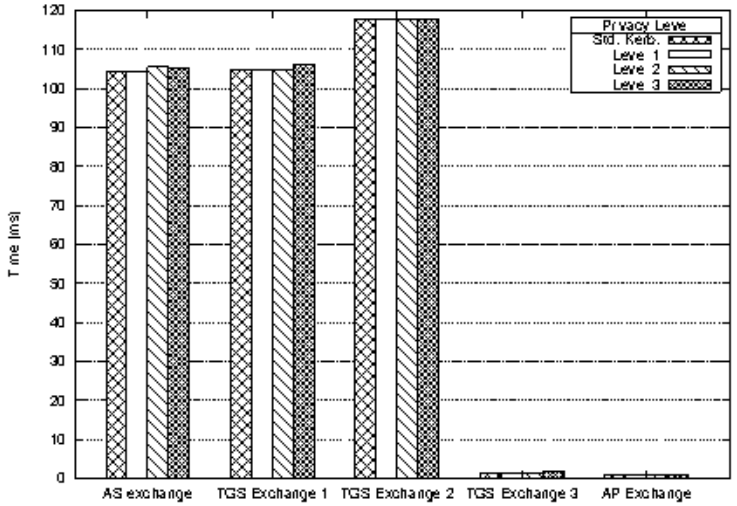
(a) Mean Exchange Time (ms)



(b) Mean Privacy Processing Time (ms)



(c) Message Length (bytes)



(d) Mean Network Time (ms)

Figure 5.10: Results for PrivaKERB in scenario II

required to implement the fake ticket feature in those message sent from the KDC to the client. While the KRB_AS_REP message reaches 1168 bytes, the KRB_TGS_REP messages generated by home and visited KDC exceed the size of 1500 bytes, being transmitted as two fragmented UDP packets. Nevertheless, this situation has a minimum impact over the network time. For example, in the worst case, "TGS exchange 1" requires an additional time of $\approx$1.54 ms compared with standard Kerberos.

5.6 Integrating PrivaKERB in the Kerberized EAP-FRM Architecture

Throughout this chapter, we have studied how to preserve user privacy in Kerberos, which is the secure three-party key distribution protocol we use to achieve a fast network access. Indeed, in previous section 5.5, we have verified that the privacy extensions introduce an almost negligible overload compared with the standard Kerberos protocol. Therefore, the fast re-authentication process provided by our *Kerberized EAP-FRM* architecture will not be affected when the privacy enhancements to Kerberos are enabled.

On the one hand, the use of PrivaKERB allows us to achieve an effective user privacy protection when performing a key distribution process and, in general, during the fast re-authentication. Let us recall that, during the *fast re-authentication phase*, our Kerberized EAP-FRM solution does not expose the user's identity at EAP-FRM level by omitting the user's name part in the *User-Id* TLV contained in the *EAP-FRM/Response* message. More precisely, as described in section 4.3.3, the *User-Id* TLV is strictly used to indicate the domain where the authentication data conveyed within EAP-FRM must be routed through the AAA infrastructure. Therefore, the client's real identity appears in the secure three-party key distribution protocol. In particular, this happens with Kerberos, which is the option we have chosen in chapter 4. Thus, to handle privacy aspects in our Kerberized EAP-FRM fast re-authentication solution, we need to handle with these aspects within Kerberos, as we have done in this chapter.

On the other hand, during the *bootstrapping phase*, the user is required to provide its identity in the *EAP-Response/Identity* (see Fig. 4.2). The realm in this identity is also used by the authenticator to route the information to the correct home AAA/EAP server which selects a bootstrapping EAP method. In order to not compromise the privacy protection during this phase, based on our work in [101], we allow the user to employ a different pseudonym each time the bootstrapping phase is executed. This pseudonym must be dynamically assigned and renewed each time the bootstrapping phase is executed. In particular, since PrivaKERB is based on these foundations, we propose to employ the same pseudonym used to carry out the initial authentication exchange (KRB_AS_REQ/KRB_AS_REP) in Kerberos. In section 4.3.1, we described two different alternatives to carry out the bootstrapping phase. On the one hand, when EAP-EXT is used as bootstrapping EAP method, the KRB_AS_REQ/KRB_AS_REP exchange is performed within EAP-EXT during the so-called *binding phase*. On the other

hand, when another different EAP method that generates key material is employed as bootstrapping EAP method, the KRB_AS_REQ/KRB_AS_REP exchange is performed over UDP once the EAP method has successfully finished and the mobile user has obtained network access. In either case, the same pseudonym (e.g., $CP_1@realm$) is used in the KRB_AS_REQ/KRB_AS_REP exchange during the Kerberos authentication process and the *EAP-Response/Identity*. Similarly, the new pseudonym delivered in the KRB_AS_REP (e.g., $CP_2@realm$) will be used in the next bootstrapping process, and so on. Applying this approximation, Fig. 5.11 shows a privacy-enhanced bootstrapping phase based on EAP-EXT.

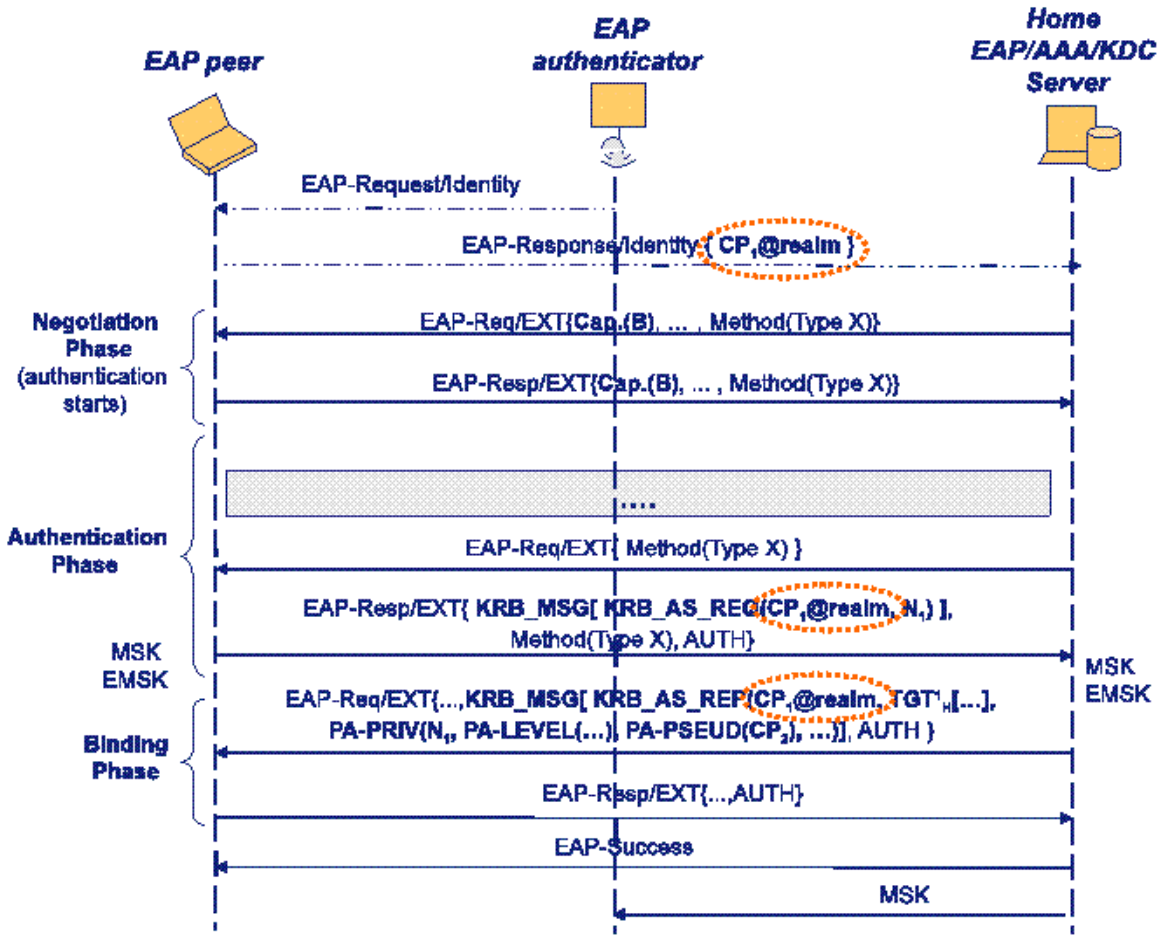


Figure 5.11: Privacy-Enhanced Bootstrapping Phase (based on EAP-EXT)

Finally, we would like to clarify two aspects related to the integration of PrivaKERB with our Kerberized EAP-FRM architecture for fast re-authentication. The first one has to do with the EAP protocol itself. As we described in section 2.2, EAP messages contain an *Identifier* field whose purpose is twofold: *(1)* to associate an *EAP-Request* with the corresponding *EAP-Response* and *(2)* to distinguish new messages from retransmissions of previous ones that were not correctly delivered. The standardized EAP specification [33]

mandates that the identifier value used for a new *EAP-Request* must be different from the value used for the previous *EAP-Request/EAP-Response* exchange. This behaviour has been typically achieved by initializing an initial value that is monotonically increased when a new identifier is needed for an *EAP-Request*. Nevertheless, as the reader may notice, this mechanism allows an eavesdropper to relate all the messages that integrate an EAP conversation. This may break the *untraceability* feature offered by privacy levels 2 and 3 when the reactive operation mode is used (let us recall that KRB_TGS_REQ/KRB_TGS_REP messages are transported in EAP-FRM packets between the peer and the authenticator). For this reason, we propose to pseudo-randomly generate new identifier values for each *EAP-Request/EAP-Response* exchange so that an eavesdropper is unable to predict the sequence of identifiers used within a specific EAP conversation. Second, another important remark concerning our solution is the fragmentation. From the results obtained in section 5.5.2, we can conclude that the size of some messages (specially KRB_TGS_REP) in privacy levels 2 and 3 may exceed in some cases the MTU of the wireless link. This situation may represent an inconvenient given that EAP-FRM does not support fragmentation of payloads larger than the MTU supported by the specific technology used in the wireless link. In this situation, we propose two alternatives. On the one hand, we recommend the use of the proactive operation mode of our Kerberized EAP-FRM fast re-authentication solution. In fact, regardless of whether a user employs the privacy extensions or not, the proactive operation is the preferred mode since it achieves the highest reduction in the time required to complete the re-authentication process. In this mode, since the user acquires a service ticket for the target authenticator before the handoff, the KRB_TGS_REQ/KRB_TGS_REP exchanges are performed by using TCP or UDP as transport. On the other hand, in case the reactive operation mode needs to be used, we recommend the use of privacy level 1 until the EAP-FRM method is improved with fragmentation support which, as described in the final chapter of this PhD thesis, is a future working item.

5.7 Conclusions

In this chapter, our concern has been related to the protection of user identification related data (e.g., user identity) in Kerberos. Typically, with the aim of establishing a context for the distributed key, secure key distribution protocols like Kerberos explicitly indicate in their messages the entities which are authorized to employ the distributed key. In particular, in the fast re-authentication scenario, this causes that the identity of the mobile user is exposed to unauthorized parties during the fast network access, thus creating some privacy issues. Since the fast re-authentication solution proposed in this PhD thesis relies on Kerberos, the privacy problems present in this key distribution protocol are inherited by our solution. Therefore, our intention is to avoid these disadvantages by enhancing our *Kerberized EAP-FRM* architecture in such a manner that user privacy is preserved during the fast re-authentication process.

Initially, from an exhaustive analysis of Kerberos, several privacy deficiencies are

detected in the protocol. To solve this problem, we propose a framework named *PrivaKERB*, which preserves user privacy in Kerberos. PrivaKERB comprises an opt-in multimode solution which provides anonymity, service access untraceability and exchange untraceability to Kerberos clients. This can be attained even in cross-realm transactions where the client roams away from his home network. Special care has been taken to not inflict any modifications on the standard Kerberos protocol and hence to be able to interact with existing implementations without privacy support. After analyzing the internal components of our framework, we demonstrate that it is also lightweight, imposing almost insignificant overheads in terms of service times, resource and network utilization. To reach this conclusion we compare the behaviour of PrivaKERB against standard Kerberos over a properly designed testbed. We argue that PrivaKERB provides a flexible mechanism to preserve privacy in Kerberos, so far as it guarantees anonymity, does not obstruct important network operations like accounting/charging service access and it does not harm fast re-authentication operations based on Kerberos.

The chapter ends by explaining how the privacy framework for Kerberos can be easily used within the Kerberized EAP-FRM architecture for fast re-authentication (presented in previous chapter 4). With this integration, we achieve an access control system that accomplishes the requirements defined in the introductory chapter (see Table 1.1): *support of any type of handoff* (thanks to the use of EAP which is independent of the underlying technology), *strong security* (thanks to the use of Kerberos which is a well-known and tested three-party key distribution protocol), *low deployment impact* and *avoid standard modifications* (since our proposal does not require modifications in current standardized protocols and employ the extensibility mechanisms available in EAP, AAA protocols and Kerberos) and *privacy support* (by using the PrivaKERB privacy extensions). Nevertheless, all these achievements are widely discussed in the following and final chapter of this PhD thesis.

Chapter 6

Conclusions and Future Work

This last chapter of the PhD thesis has two main objectives. First, we will highlight the most relevant conclusions extracted from the research activity, which have produced the contributions presented in previous chapters. More precisely, we will focus on those aspects that our fast re-authentication solution improves in comparison with existing proposals. Secondly, we will define the different future research lines, derived from our contributions within this thesis.

Initially, we will present the conclusions following the same order employed to present the contributions comprising this thesis. In contrast, when describing the future work activities, we will provide a general overview of the different research lines that arise from the work developed within this thesis.

6.1 Conclusions

Wireless telecommunication systems have experienced an enormous growth in the last decade, which has resulted in a wide variety of wireless technologies that use the air as a transmission medium. In addition, the proliferation of mobile devices such as smart phones, tablet PCs or netbooks, has caused users to show high interest in the *always-connected* experience. To support the combination of mobility and access to network services anywhere and any time, communication networks are moving towards an *all-IP* network configuration, integrated by an IP-based network core and a set of access networks based on different wireless technologies. This scenario, referred to as the *Next Generation of Heterogeneous Networks* (NGNs), enables the convergence of different heterogeneous wireless access networks so as to combine all the advantages offered by each wireless access technology per se.

The importance of achieving smooth and seamless movements each time a user changes its connection point to the network (*handoff*), has been recognized for the provision of high-quality multimedia services in NGNs. In fact, this is a critical process, where the connection to the network is interrupted, thus causing packet loss that may affect on-going communications. This problem represents a specially serious concern for emerging

applications for NGNs such as video conferencing or voice over IP (VoIP), which require high bandwidth and are seriously affected when access to the network service is interrupted.

The provision of *seamless mobility* has created an interesting research field within NGNs in order to find mechanisms which try to provide a continuous access to the network during the handoff. In particular, efforts are directed at reducing the time required to complete the different tasks performed during the handoff. In particular, network access control process has been demonstrated to be one of the most important factors that negatively affects handoff latency. This process is demanded by network operators in order to control that only legitimate users are able to employ the operator's resources.

We have shown that the integration of the *Extensible Authentication Protocol* (EAP) with *Authentication, Authorization and Accounting* (AAA) infrastructures is acquiring a relevant position as the access control framework in future NGNs. This success has been motivated by three important advantages offered by EAP. First, instead of proposing a specific authentication process, EAP offers a flexible framework that allows the definition of multiple authentication mechanisms (called *EAP methods*) which are executed between the mobile user (*EAP peer*) and the authentication server (*EAP server*). The process is performed through an intermediary entity (*EAP authenticator*) which acts as *Network Access Server* (NAS) controlling access to the network. Second, EAP is independent of the underlying wireless access technology, and is thus able to operate on any lower-layer. This feature is especially important, due to the heterogeneity of wireless access networks in NGNs. Third, EAP allows easy integration with existing AAA infrastructures.

However, EAP has shown some drawbacks when a mobile scenario is taken into account. Typically, an EAP authentication lasts considerable time and involves multiple message exchanges. Moreover, these authentication messages have to travel to the user's home domain, which could be situated far from the point of attachment to the network. This is especially problematic considering that, each time a mobile user performs a handoff to change its connection point to the network, the process must be completed before granting access to the network. Therefore, we observe that the introduction of an EAP-based network access control jeopardizes the seamless mobility requirement since the EAP authentication introduces a latency during the handoff that may provoke a substantial packet loss, thus affecting the quality of active network applications.

To solve the inefficiency of EAP, researchers agree that is necessary to define a fast re-authentication process to reduce the number of messages involved during the EAP authentication and the time devoted to network transmission by enabling a re-authentication server placed near the mobile user. Furthermore, considering that a successful EAP authentication generates key material valid for a certain period of time, it has been recognized [65] that a proper fast re-authentication process should consist of two phases. Initially, during the *bootstrapping phase*, the mobile user performs a full EAP authentication with the home EAP server. This initial authentication generates a cryptographic material that is used to enable an efficient key distribution process in a posterior *fast re-authentication phase*. In particular, it has been argued [71] that a secure three-party key distribution is the most appropriate model to perform this process securely.

In this (PhD) thesis, we do not only deal with the fast re-authentication problem of

EAP, but also with another challenging issue associated with NGNs: user privacy. The nature of wireless networks allows a malicious user to eavesdrop or capture messages from any active communication that takes place under its coverage area. As a consequence, among other implications, this situation enables the user's activity to be monitored. For this reason privacy represents a serious concern for both emerging applications and mobile users in future wireless networks.

Privacy is a complex problem that affects different network layers. Particularly interesting are the privacy issues that arise during the network access, where the user is expected to provide its identity to an authentication server. If the authentication mechanism does not have an adequate level of privacy to protect identification related data, private information regarding the user's activity can be revealed to unauthorized parties. Therefore, in NGN access control systems, a privacy-preserving mechanism needs to provide not only user *anonymity* during network access but also user *untraceability* so that eavesdroppers are unable to obtain anonymous profiles.

In conclusion, the research community faces the exciting challenge of developing a set of mechanisms that, operating together, will enable fast, secure network access in future EAP-based NGNs while preserving user privacy. This is precisely the problematic that has been addressed here by defining a novel access control system. While the contributions presented in chapters 3 and 4 are aimed at reducing the EAP authentication time during the handoff, chapter 5 deals with the aspect of user privacy.

The first contribution, presented in chapter 3, defines a transport-based architecture for fast re-authentication. The solution has been specially adapted to assist a fast network access based on a secure key distribution process to re-authenticate the user. More precisely, the architecture defines a new EAP method called *EAP Fast Re-Authentication Method* (EAP-FRM) which works on standalone authenticators. Despite standalone methods being executed between the peer and the authenticator, when required, the EAP-FRM method itself can contact a backend authentication server by using an AAA protocol such as RADIUS or Diameter. EAP-FRM offers a generic transport able to convey any key distribution protocol which, in the context of EAP-FRM, is referred to as *Fast Re-Authentication Protocol* (FRP). Using the key distributed by the key distribution protocol to both the peer and the authenticator, EAP-FRM generates the keying material (EMSK and MSK) that will be exported to lower-layers. Apart from its flexibility, EAP-FRM has been validated so as not to compromise the fast re-authentication process, whose efficiency depends on the specific FRP in use. Furthermore, given that we have used the extensibility mechanism available at EAP, which is the definition of new EAP methods, the EAP-FRM architecture does not impose any modification either on EAP or existing wireless technologies. The publications derived from this contribution are [94], [96], [97], [98] and [99].

Thus, the EAP-FRM architecture offers a vehicle to transport any key distribution protocol which, ultimately, is responsible for reducing the EAP authentication time. For this reason, in chapter 4 we integrate EAP-FRM with a specific key distribution protocol to complete the fast re-authentication solution. In particular, we have chosen the Kerberos protocol as the key distribution protocol. Kerberos is a secure three-party key distribution

protocol that has been widely used to control access to network resources. Strictly following the standard protocol specification, Kerberos is integrated with EAP-FRM to control access to the network service where the authenticator acts as the Kerberos application server offering the network access service. Compared with existing proposals, the resulting *Kerberized EAP-FRM architecture* offers important advantages. First, in the best case, the solution is able to achieve a fast re-authentication process consisting of only three messages between the peer and the authenticator, with no need of communication with an external authentication server. Second, unlike other proactive techniques such as key pre-distribution or pre-authentication, the proposed solution defines a proactive operation mode that does not require any pre-reservation of resources in candidate authenticators. Third, by using Kerberos as key distribution process, we avoid the definition of a new key distribution protocol, which is a complex and error-prone process, and we rely on a mature and well-known secure three-party key distribution protocol. The publications related to this chapter are [16] and [100].

Chapter 5 focuses on the need to protect user privacy during the Kerberos-based fast re-authentication process. Key distribution protocols in general, and Kerberos in particular, explicitly indicate in their messages the identities of the entities which are authorized to use the distributed key. When applied to enable fast network access, this means that the user's identity is exposed during the fast re-authentication process. This problem is solved by developing a novel privacy architecture called *Privacy Kerberos* (*PrivaKERB*) that preserves the privacy of the user during its activity with Kerberos. The solution provides a flexible framework offering three different levels of privacy that, respectively, preserve user anonymity, service untraceability and exchange untraceability. The privacy enhancements to Kerberos do not violate the standardized Kerberos protocol and allow operations (e.g., charging) that require some association with the specific user. All these features are achieved with an almost negligible overhead to the standard protocol, thus not jeopardizing the fast re-authentication process. Furthermore, PrivaKERB is a general purpose solution applicable not only to our fast re-authentication solution but also to any system based on Kerberos. Indeed, we show that our privacy framework is fully compatible with EAP and can easily be integrated with EAP authentication and Kerberized EAP-FRM fast re-authentication architecture. The publication related to this chapter is [101].

As the reader may recall, in the chapter 1, we defined some requisites that were required to be accomplished by the fast re-authentication solution designed in this PhD thesis. In particular, we distinguished three different categories: design goals (see section 1.4.2), security goals (see section 1.4.3) and privacy goals (see section 1.5.2). In the following sections, we detail the reasons why our privacy-enhanced Kerberized EAP-FRM fast re-authentication solution satisfies all these requirements. After that, we will conclude this section by summarizing the main results reached in this PhD thesis from a general perspective.

6.1.1 Design Goals

The fast re-authentication solution developed accomplishes all the design goals defined in section 1.4.2. Next, we outline which specific features of our proposal enable these requirements to be accomplished.

(D1) *Low latency operation.* The Kerberized EAP-FRM architecture achieves a reduced network access latency by avoiding contacting a backend authentication server (located in the AAA infrastructure) to re-authenticate the user. More precisely, as described in section 4.3.3, in the *proactive mode* the re-authentication process is completed in only three messages between the mobile user and the authenticator. On the other hand, it has been demonstrated that the application of PrivaKERB does not affect the fast re-authentication operation since, as verified in section 5.5, the privacy enhancements impose an almost negligible overhead.

(D2) *EAP lower-layer independence.* Thanks to EAP-FRM, the fast re-authentication solution is independent of the EAP lower-layer in use. In fact, EAP-FRM is a generic solution offering a transport to any fast re-authentication protocol without requiring any modification to either EAP or existing EAP lower-layers.

(D3) *Compatibility with existing EAP methods.* Our solution relies on the definition of a new EAP method (EAP-FRM) which is used to transport a key distribution protocol. Therefore, since EAP-FRM does not interact with any other EAP method, we can conclude that our fast re-authentication mechanism is fully compatible with existing EAP methods and no requirements are imposed on future EAP methods. In fact, during the bootstrapping phase, our solution can use any EAP method able to export keying material (EMSK and MSK), which is a recommendation for EAP methods used in wireless environments [111].

(D4) *AAA protocol compatibility and keying.* The fast re-authentication solution has been designed by using the extensibility mechanism available in EAP, which is the definition of new EAP methods. In particular, we design a standalone EAP method (EAP-FRM) following strictly both the EAP protocol specification [33] and the key distribution scheme defined in the *EAP Keying Management Framework* [38] (EAP KMF). Similarly, the extensibility mechanisms available in current AAA protocol such as RADIUS and Diameter are sufficient to carry out the communication between the authenticator and a backend authentication server. Therefore, we conclude that our solution does not impose any modification to existing AAA protocols or key management process for AAA environments.

(D5) *Compatibility with other optimizations.* As commented in chapter 2, existing optimizations to reduce the EAP authentication time require the execution of an EAP method able to export a MSK to both the peer and the authenticator. This key is used to enable the optimization proposed by the specific mechanism. Since EAP-FRM follows the EAP KMF [38] guidelines and exports keying material

(MSK and EMSK), our fast re-authentication solution is compatible with other optimizations. In particular, as an example, the pre-authentication concept can be applied to EAP-FRM. In this way, when the user performs the handoff, only a security association is established with the new authenticator. Moreover, the application of EAP-FRM to a pre-authentication process reports important advantages compared with existing pre-authentication solutions based on a full EAP authentication. For example, thanks to the low-latency of EAP-FRM, a minimal anticipation to the handoff is required due to the pre-authentication time being greatly reduced with Kerberized EAP-FRM.

(D6) *Backward compatibility.* The proposed solution is able to operate in situations where either the mobile user or the network does not support our Kerberized EAP-FRM architecture. This feature is achieved thanks to EAP-FRM. As described in section 3.2.2, an authenticator supporting our extensions is expected to start the fast re-authentication process by sending an *EAP-Request/FRM* message. Therefore, when an authenticator does not support our EAP-FRM architecture, it starts a traditional EAP authentication by sending an *EAP-Request/Id*. This situation is interpreted by the peer as an indication that our fast re-authentication solution is not supported. In the case that the peer receives an *EAP-Request/FRM* message but does not support our fast re-authentication solution, it sends an *EAP-Response/Nak* message. This causes the authenticator to send *EAP-Request/Id* in order to start a traditional EAP authentication process.

(D7) *Low deployment impact.* The deployment impact increases when the adoption of a solution requires modifications in current standardized protocols and technologies. The access control system developed within the PhD thesis has an extremely low deployment impact since the solution is fully compatible with current standards and uses the extensibility mechanisms offered by every technology. First, as explained in *D4*, the generic transport for fast re-authentication offered by EAP-FRM is designed by using the extensibility mechanisms available in EAP (definition of EAP methods) and AAA protocols like RADIUS (definition of new attributes) or Diameter (definition of new applications and commands). Second, the integration of Kerberos with EAP-FRM to achieve a fast network access is performed without requiring any change in the standardized Kerberos protocol operation or entities. Third, the privacy framework developed for Kerberos (PrivaKERB) employs the extensibility mechanisms available in Kerberos, such as definition of new flags in the messages, design of new pre-authentication data and authorization elements.

(D8) *Support of different types of handoffs.* Since our fast re-authentication architecture is designed on the basis of EAP, which has the media-independence property, a mobile user can engage in a fast re-authentication process with an authenticator using the same or a different technology (intra/inter-technology handoff), located in the same or in a different network (intra/inter-network handoff), managed by the same network

operator or by a different one (intra/inter-domain handoff) or requiring the same or a different security to protect the wireless access (intra/inter-security handoff).

6.1.2 Security Goals

Thanks to Kerberos, whose security has been demonstrated by both its extensive deployment and security studies [76 78], we can affirm that our solution achieves a fast re-authentication process with strong security properties. Let us recall that neither EAP-FRM nor PrivaKERB affect the security of the re-authentication process. On the one hand, the EAP-FRM method does not introduce any new vulnerability to EAP (see section 3.5.6) since the definition of new EAP methods is conceived as the extensibility mechanism available in EAP to design new authentication mechanisms without affecting the standardized protocol. On the other hand, PrivaKERB does not compromise the security of Kerberos (see section 5.4.2) since it is implemented by using the available mechanisms to extend the protocol functionality.

One of the reasons why Kerberos has been selected in this PhD thesis lies in its ability to satisfy all the security goals (defined in section 1.4.3) that a proper key distribution protocol must accomplish.

(S1) *Authentication.* Kerberos provides an authenticated key distribution process based on a trusted third-party called *Key Distribution Center* (KDC). By using shared secret keys, Kerberos allows a mutual authentication between the involved entities in every step of the key distribution process. In the initial KRB_AS_REQ/KRB_AS_REP exchange, a secret key derived from the user's long term credentials (e.g., a password or a shared key) is used by peer and AS/KDC to authenticate each other. Conversely, in the KRB_TGS_REQ/KRB_TGS_REP and KRB_AP_REQ/KRB_AP_REP exchanges, this mutual authentication is performed by using the *TGS session key* and *service session key*, which are keys dynamically established during the protocol execution.

(S2) *Authorization.* Kerberos considers that, once a peer is successfully authenticated against an authenticator, some authorization process is necessary to determine if the user can access the network access service and the type of access allowed. For this reason, the protocol has been conceived so as to be integrated with an authorization system providing this functionality. Indeed, the protocol allows the transference of authorization data from the KDC to the authenticator (contained within a service ticket), from the peer to the KDC (contained in the KRB_AS_REQ and KRB_TGS_REQ messages) and from the peer to the authenticator (contained in the KRB_AP_REQ message). For example, this capability has been used within our Kerberized EAP-FRM architecture to deliver AAA-based authorization data to the authenticator.

(S3) *Key context.* Kerberos securely distributes the key to the peer and authenticator by using different mechanisms. On the one hand, by using the shared secret key between

client and KDC, the KDC sends the key to be shared with the authenticator. On the other, the same shared secret key is distributed to the authenticator within the ticket. In either case, Kerberos not only transmits the distributed key but also indicates additional information that defines a precise context for the distributed key. For example, this context indicates the entities which are authorized to use the distributed key or the period of time when the distributed key is valid.

(S4) *Key freshness.* In addition to confidentiality and integrity protection of the distributed key, Kerberos ensures that the key is fresh. Depending on whether the key is distributed to the peer or the authenticator, different mechanisms are used. On the one hand, the freshness of the key distributed to the client is achieved by means of the pseudo-random number (*nonces*) sent from the client to the KDC, since the client expects the same value in the received response from the KDC. On the other hand, the freshness of the key distributed to the authenticator (within the ticket) is ensured through timestamps indicating the validity time.

(S5) *Domino effect.* Kerberos is resilient to the domino effect since the keys distributed by the KDC are pseudo-randomly generated and cryptographically independent. Therefore, assuming a set of authenticators, a different and independent key will be distributed in each authenticator so that the compromise of one authenticator doest not affect other authenticators.

(S6) *Transport aspects.* Kerberos is independent of the protocol used to transport the messages between the different entities. Typically, the well-known IP protocols TCP or UDP have been used for this purpose. Nevertheless, these protocols assume the existence of network connectivity, a requisite which may not happen in traditional network access control scenarios (e.g., when link-layer authentication is required). For this reason, our fast re-authentication solution uses EAP-FRM as transport for Kerberos.

6.1.3 Privacy Goals

Compared to existing fast re-authentication mechanisms, another contribution of this PhD thesis has been to design a solution to preserve the user privacy during the network access control based on Kerberos. For this reason, in section 1.5.2 we defined some privacy requirements that have been satisfied in the following manner:

(P1) *User anonymity.* The basic operation mode in PrivaKERB (level 1) keeps the user anonymous during its activity with Kerberos. This requisite is accomplished by dynamically assigning pseudonyms to the user which are valid for a specific period. More precisely, pseudonyms are bound to the home TGT lifetime, so that when the home TGT expires, the current pseudonym employed by the user is renewed by a new one. Given that the KDC generates and distributes pseudonyms to the user, only this trusted entity knows the user's real identity.

(P2) *User untraceability.* The intermediate and advanced operation modes defined in PrivaKERB (levels 2 and 3) allows an anonymous user to remain untraceable. On the one hand, privacy level 2 achieves a *service access untraceability* by introducing both the *anonymous ticket* and the *self-renewed TGT* features that allow: (a) the user's pseudonym to be hidden from observers, and (b) the use of a different TGT each time the user contacts the KDC to request a service ticket. On the other hand, privacy level 3 goes a step further and improves privacy level 2 so as to provide *exchange untraceability* as well. This kind of untraceability is accomplished by hiding the ticket distributed to the user through the *fake ticket* feature.

(P3) *Adaptation to the EAP authentication model.* PrivaKERB is a privacy framework for Kerberos that does not violate the protocol operation and is implemented through the extensibility mechanisms defined in the Kerberos specification. Thus, it is a general purpose solution that can be applied to any system using Kerberos and, in particular, to our Kerberized EAP-FRM architecture for fast re-authentication (see section 5.6) in EAP-based networks.

(P4) *No jeopardizing of the fast re-authentication process.* PrivaKERB provides a flexible privacy preserving mechanism that, compared with the standard Kerberos protocol, imposes almost insignificant overheads in terms of service times, resource and network utilization. For this reason, we can conclude that our privacy solution does not affect the fast re-authentication procedure achieved by the Kerberized EAP-FRM architecture.

6.1.4 General Conclusions

On the basis that EAP and Kerberos are promising protocols to carry out the authentication processes in future NGNs, this PhD thesis has developed an efficient access control solution applicable in any kind of handoff, offering a low-latency re-authentication process with strong security properties, preserving the privacy of the user and full compatibility with existing standardized technologies which, in turn, minimizes the deployment cost of our solution.

One of the most important features of our fast re-authentication architecture is the reduction in the time required to re-authenticate the user. According to the performance results obtained in chapter 4 for a representative scenario in the best case (the user owns a valid ST for the target authenticator) the authentication time is about 25 ms, while in the most typical situation (the user does not have a ST for the target authenticator but a valid TGT for the visited AAA/KDC), the re-authentication process can be completed in about 75 ms. These values can be considered as references even when the PrivaKERB extensions are used, since we have verified that they introduce an almost negligible overhead to the standard Kerberos protocol. Nevertheless, our architecture can further reduce this small latency introduced during the handoff by applying the pre-authentication technique. In this situation, as with any pre-authentication solution, the handoff latency is reduced to the

time required to complete the security association protocol. Nevertheless, a noteworthy benefit of our solution is the minimal anticipation time required before the handoff to execute the pre-authentication process, which is useful when dealing with highly mobile users that suddenly change between points of attachment to the network.

Another important advantage of our solution relies on the implementation process. Given that our contributions do not violate either EAP or Kerberos specification and that they exploit the mechanisms offered by each protocol to extend its functionality, the implementation process of the privacy-enhanced Kerberized EAP-FRM architecture is not envisaged as extremely costly. In fact, the different prototypes used throughout this thesis to test our contributions have been implemented by developing software modules that extend the standard protocol functionality in existing open-source implementations (*wpa_supplicant* [135], *hostapd* [136], *FreeRadius* [137] and *MIT Kerberos* [142]).

The deployment aspect is another key feature that has been carefully taken into account when designing our architecture for fast re-authentication. Unlike existing fast re-authentication solutions, our architecture achieves a minimal deployment cost by not requiring existing EAP and Kerberos implementations to be modified. Instead, we use the extensibility mechanisms offered by the specific implementation (EAP, RADIUS, Diameter or Kerberos), so easing the future deployment of our solution.

Finally, thanks to the modularity of our contributions, our enhancements can be independently considered to experiment with other alternatives. For example, EAP-FRM is a transport able to convey any key distribution protocol. Thus, other key distribution protocols can be tested by using our proposed transport which is agnostic to the underlying wireless technology and compatible with current standards. Similarly, although the Kerberized architecture for fast re-authentication uses EAP-FRM to transport Kerberos messages, the solution is independent of the specific transport. Therefore, the proposed re-authentication solution can be an efficient candidate re-authentication solution in other scenarios not based on EAP as long as the corresponding wireless technology is adapted to carry Kerberos. In the same manner, the privacy framework for Kerberos is a general-purpose solution to preserve the privacy of the user, and therefore applicable to any system using Kerberos.

Given the complexity of the problem addressed in this PhD thesis, our approach leads to improvements of certain aspects, extensions or even new research lines due to the application of our architecture for fast re-authentication in other areas. The following section treats these aspects.

6.2 Future Work

This PhD thesis has presented an advanced access control system intended to reduce the network access time during the handoff, while preserving the user privacy during the process. The solution has been incrementally designed, addressing the following needs: (1) definition of an EAP-based transport for fast re-authentication; (2) definition of a fast re-authentication process based on a secure three-party key distribution protocol, and (3)

user privacy protection during the fast network access. Nevertheless, during the research process that has produced the different contributions, several issues have arisen that do not affect the applicability of the solution but open new research fields in this area. In the following we outline the main future activities derived from this PhD thesis, distinguishing between enhancements to the proposed architecture itself and new research areas.

6.2.1 Improvements and Extensions to the Architecture

In relation to the EAP-FRM transport, there are several aspects that can be considered in order to improve the functionality offered by the method. First, the proposed specification of EAP-FRM can be extended with negotiation capabilities like, for example, the FRP used. Indeed, the authenticator sets the specific FRP that is going to be used in the *EAP-Request/FRM* message. If the peer supports EAP-FRM but does not support this FRP, it must respond with an *EAP-Response/Nak message* which will provoke the initiation of a traditional EAP authentication. For this reason, the inclusion of an initial FRP negotiation phase in the architecture could be considered in a future version of the EAP-FRM method. Since this negotiation introduces additional exchanges between peer and authenticator, it is important to ensure that the inclusion of this phase does not jeopardize the fast re-authentication process. Second, another interesting enhancement to EAP-FRM is related to the transmission of large FRP payloads. Since EAP does not support fragmentation and reassembly, EAP authentication methods generating payloads larger than the minimum EAP MTU (1020 bytes) need to provide fragmentation support. In fact, this is a recommended requirement for EAP methods used in wireless networks [111]. For this reason, one significant improvement to EAP-FRM may be the inclusion of fragmentation and reassembly support.

As with EAP-FRM, the Kerberized EAP-FRM architecture for fast re-authentication can be improved by refining some aspects which have not been covered in this PhD thesis. For example, in the integration of Kerberos with EAP-FRM we have considered a simple but effective authorization process. As the reader may recall, we propose that the KDC includes some authorization information recovered from an AAA server in the service ticket. Thus, this authorization data is dependent on the specific AAA protocol in use (either RADIUS attributes or Diameter AVPs). Nevertheless, it would be beneficial to enrich our fast re-authentication solution by interfacing with authorization infrastructures in order to manage end user attributes and to achieve more advanced authorization and access control decisions. To do this, we can use technologies such as SAML 2.0 [174] and XACML 2.0 [175] which provide standardized languages for exchanging authorization data and representing access control policies, respectively.

As mentioned in section 4.3.4, an authenticator and KDC discovery mechanism is needed for the proactive operation modes. The reason is that the user needs to request STs for authenticators in neighboring networks before the handoff. The IEEE 802.21 standard is a candidate technology to implement such discovery mechanism, since it defines an *Information Service* that is independent of the underlying technology, which assists the handoff decision process. More precisely, this service provides mobile users with pieces of

information (called *Information Elements*) regarding both the neighboring access networks and the NASes. For this reason, the integration of our Kerberized EAP-FRM architecture with the IEEE 802.21 Information Service is an interesting improvement that favours both the single-realm and cross-realm proactive handoffs.

Another issue that can be addressed in future work is the optimization of the service ticket acquisition. Of the different operation modes defined for the Kerberized EAP-FRM architecture, the proactive mode achieves the highest reduction in the time required to complete the re-authentication process. Nevertheless, this mode requires the user to own a valid service ticket for the target authenticator which is acquired before the handoff. For this reason, it is interesting to explore mechanisms to optimize the acquisition of tickets for candidate authenticators to which the user may potentially roam in the future. In this regard, the Kerberos protocol shows certain inefficiency since it is necessary to contact the KDC per each requested ticket. That is, if a user desires to request service tickets for n different services managed by the same KDC, then n KRB_TGS_REQ/KRB_TGS_REP exchanges must be performed. In this sense, it would be useful to optimize the recovery process so that a user could recover a set of service tickets to access multiple services in one single contact with the KDC.

In the privacy framework developed for Kerberos (PrivaKERB) we can identify some aspects that are worthy of consideration. In the literature we can find other ticket-based protocols (e.g., [176 178]) which lack a mechanism to preserve the privacy of the user. We consider that this drawback can be solved by applying the concepts presented in PrivaKERB such as pseudonyms, anonymous tickets enriched with protected pseudonym, self-renewed tickets and fake tickets. On the other hand, the privacy protection achieved in PrivaKERB can be improved. For example, by applying the idea of self-renewed tickets also when a client accesses the same service with the same ST, we can increment the untraceability in Kerberos. The reason is that, potentially, a user can re-use an ST to access a specific service during the period of time the ticket is valid. Since the same ST is presented in every access, an eavesdropper can easily track all the accesses performed by the same anonymous user to a specific service. We foresee a solution based on applying the idea of self-renewed ticket to the STs. Moreover, it seems interesting to consider the identity protection not only of the user but also of the services, so that the identity of an accessed service is not revealed to unauthorized parties. This feature could be of interest for service providers that, for example, want to keep the most popular services secret.

6.2.2 New Research Areas

Apart from the improvements to the solution mentioned, the applicability of the Kerberized EAP-FRM architecture to other scenarios led us to identify excellent starting points for new research lines that we detail in the following:

- **Vehicular Networks**. *Vehicular networks* (*VANETs*) are a kind of ad-hoc networks which are composed of a set of mobile nodes (e.g., cars) that change their position in an unpredictable manner. Each node communicates with neighboring nodes

within its coverage area. In this scenario, two types of communication patterns are identified. While the *vehicle to vehicle* (V2V) communication takes place between two mobile nodes, the *vehicle to infrastructure* (V2I) communication is established between a mobile node and a point of attachment located in the infrastructure which offers access to Internet. The fast establishment of authenticated communications between the involved parties within this kind of networks has been a challenging issue. On the one hand, in V2I communications, solutions enabling a fast re-authentication when mobile nodes are sending traffic to the infrastructure and change of point of attachment to the infrastructure are necessary. On the other hand, V2V communications also require the establishment of authenticated sessions which enable protected communications. In fact, some initial works have been developed which address these aspects. For example, authors in [179, 180] evaluate the use of pre-authentication to enhance access control in V2I communications. Additionally, as we explained in chapter 4, other works like [148,150] try to optimize V2I communications by proposing a Kerberized access control solution. Conversely, Ref. [181] overviews aspects related with authentication and key establishment in V2V communications. Therefore, given this problematic, the contributions presented in this PhD thesis could be a starting point for finding solutions for achieving an efficient communication in VANETs which consider the particularities of these kind of networks, such as frequent changes in the network topology or mobile nodes high speed.

- **Federated Networks**. The impressive growth of telecommunications has promoted the establishment of business agreements between service providers within the so-called *federations* in order to increase the revenues of the deployed network services. Indeed, such network federations allow a service provider's subscriber to access the services of affiliated providers in the federation. Since most of the existing federation mechanisms are only for Web-based applications, the *Moonshot* project [182] and the IETF *Application Bridging for Federated Access Beyond web* [183] (ABFAB) working group, have been recently created to develop a federation solution valid for other network applications. In particular, although EAP has been traditionally used for network access, these initiatives have now selected this protocol for implementing access control to several non-web applications (e.g., SSH, IMAP, XMPP or NFS). This selection has been motivated by the ability of EAP to integrate both the authentication and authorization processes with AAA infrastructures that typically support existing federations. A fast re-authentication procedure has been discussed as a desirable feature to avoid the execution of lengthy full EAP authentications each time a user wants to access the service. Since EAP has been considered as an authentication protocol, our architecture for fast re-authentication could assist future works addressing this problematic. Nevertheless, the applicability of Kerberos to federated scenarios must consider the particularities of these environments. For example, current federated networks (such as *eduroam* [26]) and likely future initiatives do not assume the deployment of Kerberos cross-realm infrastructures.

Without the support of a cross-realm infrastructure, Kerberos cannot satisfy one of the most important goals in network federations of allowing any user in the federation to access any application service regardless of the domain. For this reason, we envisage an important research effort to integrate Kerberos in federated networks without the need to deploy Kerberos cross-realm infrastructures.

- **Integral Cross-layer Privacy Protection**. An important innovation of the Kerberized architecture for fast re-authentication has been the achievement of a lightweight re-authentication process that preserves user privacy. Nevertheless, privacy is a complex problem that affects different network layers. For example, a user can be traced at network layer by the use of the same IP or at link-layer by observing interface identifier (e.g., MAC address in Ethernet networks). Similarly, the transmission of sensitive data such as the user's location or session identifiers is found at transport and application layers, which makes it easy for eavesdroppers to obtain private information by tracing the user's activity, thus violating their private sphere. Therefore, although this PhD thesis has dealt with the problem of privacy in a specific area (the network access control based on EAP and Kerberos), more work is necessary in order to offer the user an effective privacy protection in NGNs. For this reason, the work started with this thesis can be continued towards the definition of an integral cross-layer privacy solution addressing the privacy issues that arise at different network layers.

- **IEEE 802.21 Networks**. The definition of secure and fast handoffs is a relevant topic that is attracting attention from the research community. Proof of this increasing interest is that, within the IEEE 802.21 [115] standardization group, a new IEEE 802.21a [184] task group has recently been created to define mechanisms that reduce the latency introduced by the authentication and authorization processes during an IEEE 802.21-assisted handoff. Currently, members of this task group are using concepts such as pre-authentication, local re-authentication or key distribution in order to define a fast network access solution for IEEE 802.21-based networks. Nevertheless, current key distribution schemes proposed within the task group still follow the two party model used by EAP and require resource pre-reservation in the network side entities. Furthermore, preserving user privacy during the handoff has been an aspect not covered so far. Therefore, as we observe, there is room to optimize the IEEE 802.21-based handoff. In this sense, our Kerberized EAP-FRM architecture may constitute the beginning of future research efforts to overcome the aforementioned deficiencies and to address the new needs that may arise in this environment.

As observed, the fast re-authentication architecture presented in this PhD thesis, or the individual consideration of the different components that integrate our solution open the door to future research works that can be developed as final degree projects, new publications, PhD thesis or research projects in this field.

Appendix A

PrivaKERB: ASN.1 Specification for the Proposed Extensions to Kerberos

A.1 PA-DATAs

Table A.1 shows the new padata types defined in our solution, indicating the name and the associated content to be included in the *padata-value* field. Note that the *padata-type* field is omitted since this value is to be defined (TBD).

Name	Contents of padata-value
pa-level	Level (not ASN.1 encoded)
pa-pseud	DER encoding of PA-PSEUD
pa-ticket	DER encoding of PA-TICKET
pa-sr-TGT	DER encoding of PA-SR-TGT
pa-priv	DER encoding of PA-PRIV

Table A.1: Defined PA-DATA types

The *pa-level* padata contains an integer that refers to the privacy level assigned to the client. Initially, only the following values are allowed: 1,2,3. The pa-level can be only present in a KRB_AS_REP message. The *padata-value* for this pre-authentication type is defined as follows:

```
Level          ::= Int32
     -- Int32 is defined in RFC 4120
```

The *pa-pseud* padata aims to transport a new pseudonym generated by the KDC to the client. The *pa-pseud* can be only present in KRB_AS_REP messages. The *padata-value* for this pre-authentication type is defined as follows:

```
PA-PSEUD          ::= PrincipalName
     -- PrincipalName is defined in RFC 4120
```

The *pa-ticket* padata is designed to carry a newly-issued ticket and the associated encrypted part for the client. This padata can be only included in KRB_AS_REP and KRB_TGS_REP messages. The *padata-value* for this pre-authentication type is defined as follows:

```
PA-TICKET          ::= SEQUENCE {
    ticket               [0] Ticket,
    kdcrep-part          [1] EncRepPart
                             -- EncASRepPart or EncTGSRepPart as appropiate

       -- Ticket, EncASRepPart and EncTGSRepPart are defined in RFC 4120
}
```

The *pa-sr-TGT* padata contains a self-renewed TGT generated by the KDC (using the TSRK key) and information necessary by the client to process and use the TGT in the future. This padata can be only included in KRB_TGS_REP messages. Next, we show the specification of the *padata-value*:

```
PA-SR-TGT          ::= SEQUENCE {
    sr-tgt             [0] Ticket,
       -- Ticket is already defined in RFC 4120
    tgt-info           [1] TgtInfo
}

TgtInfo          ::= SEQUENCE {
    key              [0] EncryptionKey,
    flags                [1] TicketFlags,
    authtime             [2] KerberosTime,
    starttime        [3] KerberosTime OPTIONAL,
    endtime              [4] KerberosTime,
    renew-till           [5] KerberosTime OPTIONAL,
    srealm               [6] Realm,
    sname                [7] PrincipalName,
    caddr                [8] HostAddresses OPTIONAL
       -- All these types are defined in RFC 4120
}
```

Finally, the *pa-priv* padata has been conceived to provide a mechanism which protects (encapsulates) other padatas sent from the KDC to the client. This padata, allowed only in KRB_AS_REP and KRB_TGS_REP messages, is a container since it can contain other padatas that are encrypted and integrity protected. The key used to protect the *pa-priv* content is the same used to compute the enc-part of the message. In a KRB_AS_REP message, the client's long-term key or another key selected via pre-authentication mechanisms is selected. Otherwise, in a KRB_TGS_REP, the TGS session key is used or a subkey if a TGS authenticator is present. Apart from the sequence of padatas, the nonce sent by the client in the KRB_AS_REQ / KRB_TGS_REQ is also included. Therefore, the

client is able to verify that the received *pa-priv* padata is fresh and it is not a replay. The specification of the *padata-value* is shown below:

```
PA-PRIV         ::= EncryptedData -- EncPAPrivData

EncPAPrivData          ::= SEQUENCE {
    SeqPAs    [0] SEQUENCE OF PA-DATA,
    Nonce     [1] UInt32
      -- These types are defined in RFC 4120
}
```

A.2 Flags

The self-renewed TGT flag is a bit in the *KDCOptions* defined as:

```
KDCOptions          ::= KerberosFlags
     -- KerberosFlags is defined in RFC 4120
     -- self-renewed TGT (TBD)
```

The self-renewed KDC option is set by the client to inform the KDC that the TGT contained in the in KRB_TGS_REQ message is a self-renewed TGT. Therefore, the TGS module of the KDC must employ the TSRK key (instead of the secret key shared between the AS and TGS) to process and validate the TGT.

Both the anonymous ticket flag and the fake ticket flag are a bit in the *TicketFlags* defined as:

```
TicketFlags         ::= KerberosFlags
     -- TicketFlags is defined in RFC 4120
     -- anonymous ticket (TBD)
     -- fake ticket (TBD)
```

On the one hand, the anonymous ticket flag is used by the KDC to indicate that the current ticket (TGT or ST) has been generated following the process described in privacy level 2 to construct anonymous tickets. Therefore, the ticket is associated with the anonymous user. Additionally, the ticket is expected to include the *Client Privacy Level* and *Client Identity* authorization data elements (ADs) in the *authorization-data* field. These ADs inform about the specific privacy level assigned to the user and the real client identity. On the other hand, the fake ticket flag is used by the KDC to inform the client that the delivered ticket (TGT or ST) is a fake one. In this situation, the client must discard the fake ticket and search the original one in the *padata* field of the message contained in a PA-TICKET padata type.

A.3 Authorization Elements

Table A.2 summarizes the proposed authorization elements to be transported in the tickets. For each one we indicate a name and content to be transported in the *ad-data* field. Note that the *ad-type* values is omitted since they are to be defined (TBD).

Name	Content of ad-data
Client Privacy Level	Level (not ASN.1 encoded)
Client Identity	DER encoding of CLIENT-INFO

Table A.2: Defined authorization elements

The *Client Privacy Level* authorization data element contains the privacy level assigned to the client. Only values 2 and 3 are allowed. This element can be only present in the *authorization-data* field defined for the tickets. The content is specified as follows:

```
Level ::= Int32
      -- Int32 is defined in RFC 4120
```

The Client Identity authorization data element contains the real identity of the user (principal name and realm). This identity must be employed by the KDC to perform any operation related with the user (e.g., retrieve information from a database). Next, we provide the ASN.1 specification for the information transported in the *ad-data* field:

```
CLIENT-INFO          ::= SEQUENCE {
    cname    0 PrincipalName,
    crealm   1 Realm
       -- PrincipalName and Realm are defined in RFC 4120
}
```

Appendix B

List of Acronyms

3GPP	Third Generation Partnership Project
3PFH	3-Party for Fast Handoff Protocol
AAA	Authentication, Authorization and Accounting
AAAP	Authentication, Authorization and Accounting Protocol
AAK	Authenticated Anticipatory Keying
ABFAB	Application Bridging for Federated Access Beyond web
ACA	Accounting-Answer
ACR	Accounting-Request
AD	Authorization Data
AES	Advanced Encryption Standard
AKA	Authentication and Key Agreement
AN	Access Node
AP	Access Point
API	Application Programming Interface
AR	Access Router
AS	Authentication Server
ASI	Application-Specific Information
ASM	Application-Specific Module
ASN.1	Abstract Syntax Notation One
AVP	Attribute Value Pair
BSSID	Basic Service Set Identifier
CBC	Cipher-Block chaining
CEA	Capabilities-Exchange-Answer
CER	Capabilities-Exchange-Request
CM	Cryptographic Material
CNP	Configuration Network Protocol
CPU	Central Processing Unit
CxTP	Context Transfer Protocol
DNS	Domain Name Service

DPA	Disconnect-Peer-Answer
DPR	Disconnect-Peer-Request
DS	Distribution System
DSRK	Domain Specific Root Key
DS-rRK	Domain Specific re-authentication Root Key
DSUSRK	Domain Specific and Usage Specific Root Key
EAP	Extensible Authentication Protocol
EAP-FAMOS	EAP Fast Mobile
EAP-FRAP	EAP Fast Re-Authentication Protocol
EAP-FRM	Extensible Authentication Protocol Fast Re-authentication Method
EAP-GPSK	EAP General Packet Radio Service
EAP KMF	Extensible Authentication Protocol Keying Management Framework
EAPOL	Extensible Authentication Protocol Over LAN
EMSK	Extended Master Session Key
EP	Enforcement Point
ER	Efficient Re-authentication
ERP	Extensions for EAP Re-Authentication Protocol
FQDN	Fully Qualified Domain Name
FRM	Fast Re-authentication Method
FRP	Fast Re-authentication Protocol
GPRS	General Packet Radio Service
HOKEY WG	HandOver Keying Working Group
IANA	Internet Assigned Numbers Authority
IAPP	Inter-Access Point Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPSec	Internet Protocol Security
IRTF	Internet Research Task Force
IV	Initialization Vector
KDC	Key Distribution Center
KDE	Key Distribution Exchange
KDF	Key Derivation Function
KDS	Key Distribution Server
KHK	Kerberized Handover Keying
KM	Keying Material
KNAS	Kerberized Network Access Server
LAN	Local Area Network
LLP	Lower-Layer Protocol
LTE	Long Term Evolution
MAC	Media Access Control
MAP	Mobility-adjustment Authentication Protocol
MB	Mobility Broker

MIA	Media Independent Authenticator
MICS	Media Independent Command Service
MIES	Media Independent Event Service
MIH	Media Independent Handover
MIHF	Media Independent Handover Function
MIIS	Media Independent Information Service
MI-PMK	Media-Independent Pairwise Key
MN	Mobile Node
MPA	Media-Independent Pre-Authentication
MS-PMK	Media Specific Pairwaise Master Key
MSA	Media Specific Authenticator
MSK	Master Session Key
MTU	Maximum Transmission Unit
NAI	Network Address Identifier
NAS	Network Access Server
NAT	Network Address Translation
NFS	Network File System
NGN	Next Generation Network
PAA	PANA Authentication Agent
PaC	PANA Client
PAE	Port Access Entity
PANA	Protocol for Carrying Authentication for Network Access
PANA SA	PANA Security Association
PANA WG	PANA Working Group
PAR	PANA-Auth-Request
PCI	PANA-Client-Initiation
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PHT	Proactive Handover Tunnel
PK	Pre-shared Key
PKI	Public Key Infrastructure
PKINIT	Public Key Cryptography for Initial Authentication in Kerberos
PNR	PANA-Notification-Response
PoA	Point of Attachment
PoS	Point of Service
PrivaKERB	Privacy Kerberos
PTA	PANA-Termination-Response
PTR	PANA-Termination-Request
PRF	Pseudo-Random Function
QoS	Quality of Service
RAA	Re-Authentication-Answer
RAM	Random Access Memory
RADIUS	Remote Authentication Dial-In User Server

RAR	Re-Authentication-Request
RFC	Request for Comments
RGW	Residencial Gateway
rIK	Re-authentication Integrity Key
RK	Root Key
rMSK	re-authentication Master Session Key
rRK	re-authentication Root Key
SAML	Security Assertion Markup Language
SAP	Security Association Protocol
SCN	Security Context Node
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SSH	Secure Shell
ST	Service Ticket
STA	Session-Termination-Answer
STR	Session-Termination-Request
TCP	Transmission Control Protocol
TEK	Transient EAP Key
TG	Task Group
TGS	Ticket Granting Server
TGT	Ticket Granting Ticket
TLS	Transport Layer Security
TLV	Type-Length-Value
TSK	Transient Session Key
TSRK	TGT Self-Renewal Key
UDP	User Datagram Protocol
UMA	Unlicensed Mobile Access
UMTS	Universal Mobile Telecommunications System
USRK	Usage Specific Root Key
VANET	Vehicular Network
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
VLR	Visitor Location Register
VoIP	Voice over IP
XACML	Extensible Access Control Markup Language
XMPP	Extensible Messaging Presence Protocol
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WPAN	Wireless Personal Area Network
WWAN	Wireless Wide Area Network

Bibliography

[1] ABI Research. *Mobile Devices Annual Market Overview: Mobile Phones, UMDs, Cellular Modems, Consumer Electronics, and Emerging Mobile Services.* Technical Report RR-CEL, 2009.

[2] M.Simek, J.Kacalek, and R. Burget. *Bandwidth Efficiency of Wireless Networks of WPAN, WLAN, WMAN and WWAN. Electrotechnic magazine Elektrorevue*, Aug. 2007.

[3] M.S. Kuran and T. Tugcu. *A survey on emerging broadband wireless access technologies. Comput. Netw.*, 51:3013 3046, August 2007.

[4] Specifications of Bluetooth Systems, vol. 1, v.1.oB Core, December 1999. Bluetooth Special Interest Group.

[5] ISO 14223 - Radio Frequency Identification Advanced transponders Part 1: Air interface, July 2003. International Organization for Standardization (ISO).

[6] ZigBee specification: Technical Report Document 053474r06, April 2005. ZigBee Alliance.

[7] IEEE 802.11 (2007) Std., Telecommunications and Information Exchange between Systems Local and Metropolitan Area Network Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 2007. IEEE Standards.

[8] IEEE standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems, May 2009. IEEE Standards.

[9] *WiMAX Forum.* http://www.wimaxforum.org/.

[10] 3GPP TS 23.060, General Packet Radio Service (GPRS), Service Description, Stage 2, Release 5, v5.0.0, September 2002. 3GPP Specifications.

[11] A. Jajszczyk. *UMTS networks: architecture, mobility, and services. IEEE Communications Magazine*, 39(9):18 20, Sept. 2001.

[12] M. Rinne and O. Tirkkonen. *LTE, the radio technology path towards 4G. Computer Communications*, 33(16):1894 1906, 2010.

[13] F. G. Marquez, M. G. Rodriguez, T. R. Valladares, T. de Miguel, and L. A. Galindo. *Interworking of IP Multimedia Core Networks between 3GPP and WLAN. IEEE Wireless Communications Magazine*, vol. 12(3):pp. 58 65, June 2005.

[14] N. Nasser, A. Hasswa, and H. Hassanein. *Handoffs in Fourth Generation Heterogenous Networks. IEEE Communications Magazine*, vol. 44(10):pp. 96 103, Oct. 2006.

[15] R. M. Lopez, A. Dutta, Y. Ohba, H. Schulzrinne, and A. F. Gomez Skarmeta. *Network-Layer Assisted Mechanism to Optimize Authentication Delay during Handoff in 802.11 Networks.* In *Proc. of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, ACM Mobiquitous 2007*, Philadelphia, USA, Aug. 2007. ACM.

[16] R. Marin-Lopez, F. Pereniguez, F. Bernal, and A.F. Gomez. *Secure three-party key distribution protocol for fast network access in EAP-based wireless networks. Computer Networks*, 54:2651 2673, October 2010.

[17] A. Dutta, D. Famolari, S. Das, Y. Ohba, V. Fajardo, K. Taniuchi, R. Lopez, and H. Schulzrinne. *Media-Independent Pre-Authentication Suppporting Secure Interdomain Handover Optimization. IEEE Wireless Communications*, vol. 15(2):55 64, April 2008.

[18] C. Politis, K. Chew, N. Akhtar, M. Georgiades, R. Tafazolli, and T. Dagiuklas. *Hybrid multilayer mobility management with AAA context transfer capabilities for all-IP networks. IEEE Wireless Communications 11 (2004) 76 88.*

[19] M. Badra, P. Urien, and I. Hajjeh. *Flexible and fast security solution for wireless LAN. Pervasive and Mobile Computing Journal*, 3:1 14, January 2007.

[20] B. Askwith, M. Merabti, Q. Shi, and K. Whiteley. *Achieving User Privacy in Mobile Networks.* In *Proc. of 13th Annual Computer Security Applications Conference, ACSAC*, pages 108 116, San diego, CA, USA, Dec. 1997. IEEE Computer Society.

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

[22] R. Marin-Lopez. Diseño de Mecanismos de Re-autenticación Rápida Basados en EAP para Entornos Móviles. PhD dissertation, University of Murcia, Department of Information and Communications Engineering, January 2008.

[23] International Organization for Standarization (ISO). *OSI Routeing Framework, ISO/TR 9575*, October 1989.

[24] S. Kent and K. Seo. *Security Architecture for the Internet Protocol.* IETF RFC 4301, Dec. 2005.

[25] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. *Generic AAA Architecture.* IETF RFC 2903, Aug. 2000.

[26] K. Wierenga and others. *DJ5.1.4: Inter-NREN Roaming Architecture.Description and Development Items*, September 2006. Project Deliverable.

[27] M. Nakhjiri. AAA and Network Security for Mobile Access: Radius, Diameter, EAP, PKI and IP Mobility . John Wiley & Sons, Sept. 2005.

[28] B. Aboba et al. *Criteria for Evaluating AAA Protocols for Network Access.* IETF RFC 2989, Nov. 2000.

[29] C. Perkins et al. *IP Mobility Support for IPv4.* IETF RFC 3344, Aug. 2002.

[30] J. Rosenberg et al. *SIP: Session Initiation Protocol.* IETF RFC 3261, June 2002.

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

[32] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. *Protocol for Carrying Authentication for Network Access (PANA).* IETF RFC 5191, May 2008.

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

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

[35] *IEEE 802.16e Standard: Air Interface for Fixed and Mobile Broandband Wireless Access System*, Feb. 2006.

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

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

[38] B. Aboba, D. Simon, and P. Eronen. *Extensible Authentication Protocol Key Management Framework.* RFC 5247, August 2008.

[39] J. Howlett, S. Hartmann, H. Tschofenig, and E. Lear. *Application Bridging for Federated Access Beyond Web (ABFAB) Architecture.* IETF Internet Draft, draft-lear-abfab-arch-01, December 2010.

[40] G. Køien and T. Haslestad. *Security Aspects of 3G-WLAN Interworking. IEEE Communications Magazine*, vol. 41(11):pp. 82 88, Nov. 2003.

[41] M.S. Bargh, R.J. Hulsebosch, E.H. Eertink, J. Laganier, A. Zugenmaier, and A.R. Prasad. *UMTS-AKA and EAP-AKA Inter-working for Fast Handovers in All-IP Networks.* In *Proceedings of the IEEE Globecom Workshops*, pages 1 6, Washington, DC, USA, Nov. 2007. IEEE Computer Society.

[42] M. Sher and T. Magedanz. *3G-WLAN Convergence: Vulnerability, Attacks Possibilities and Security Model.* In *Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 198 205, Washington, DC, USA, 2007. IEEE Computer Society.

[43] H. Mun, K. Han, and K. Kim. *3G-WLAN interworking: security analysis and new authentication and key agreement based on EAP-AKA.* In *Proceedings of the 2009 conference on Wireless Telecommunications Symposium*, WTS'09, pages 309 316, Piscataway, NJ, USA, 2009. IEEE Press.

[44] S. Park, N. Kim, and Y. Jee. *An authentication mechanism for the UMTS-WiFi networks.* In *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, Mobility '09, pages 49:1 49:4, New York, NY, USA, 2009. ACM.

[45] C. Ntantogian, C. Xenakis, and I. Stavrakakis. *A generic mechanism for efficient authentication in B3G networks. Computers & Security*, 29(4):460 475, 2010.

[46] 3GPP TS 33.234 V8.1.0, March 2008. 3rd Generation Partnership Project.

[47] Unlicensed Mobile Access (UMA) Protocols (Stage 3), May 2005.

[48] 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.

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

[50] A. Al Shidhani and V.C. Leung. *Enhancing the performance of secured handover protocols in UMTS-WiMAX interworking. Wireless Networks*, 16:1929 1943, October 2010.

[51] N. Psimogiannos, A. Sgora, and D. Vergados. *An IMS-based network architecture for WiMAX-UMTS and WiMAX-WLAN interworking. Computer Communications*, In Press, Corrected Proof: , 2010.

[52] R. Blom, K. Norr, M. Näslund, S. Rommer, and B. Sahlin. *Security in the Evolved Packet System.* Technical report, Ericsson Review, 2010.

[53] R. Dantu, G. Clothier, and Anuj Atri. *EAP methods for wireless networks. Elsevier Computer Standards & Interfaces*, vol. 29:pp. 289 301, 2007.

[54] D. Simon, B. Aboba, and R. Hurst. *The EAP-TLS Authentication Protocol.* IETF RFC 5216, March 2008.

[55] ITU-T General Characteristics of International Telephone Connections and International Telephone Circuits: One-Way Transmission Time, 1998. ITU-T Recommendation G.114.

[56] T. Aura and M. Roe. *Reducing Reauthentication Delay in Wireless Networks.* In *Proc. of 1st IEEE Security and Privacy for Emerging Areas in Communication Networks, SECURECOMM 2005*, pages 139 148, Athens, Greece, Sept. 2005. IEEE.

[57] H. Kim, K. G. Shin, and W. Dabbous. *Improving Cross-domain Authentication over Wireless Local Area Networks.* In *Proc. of 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks, SECURECOMM'05*, pages 103 109, Athens, Greece, Sept. 2005. IEEE Computer Society.

[58] J. Bournelle, M. Laurent-Maknavicius, H. Tschofenig, Y. El Mghazli, G. Giaretta, R. Lopez, and Y. Ohba. *Use of Context Transfer Protocol (CXTP) for PANA.* IETF Internet Draft, draft-ietf-pana-cxtp-01, March 2006.

[59] *IEEE Trial-Use Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation*, 2003. IEEE Standards.

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

[61] Y. Ohba and A. Yegin. *Pre-Authentication Support for the Protocol for Carrying Authentication for Network Access (PANA).* IETF RFC 5873, May 2010.

[62] S. Pack and Y. Choi. *Fast Inter-AP Handoff using Predictive-Authentication Scheme in a Public Wireless LAN.* In *Proc. of IEEE Networks 2002 (Joint ICN 2002 and ICWLHN 2002)*, Aug. 2002.

[63] Z. Cao, H. Deng, Y. Wang, Q. Wu, and G. Zorn. *EAP Re-authentication Protocol Extensions for Authenticated Anticipatory Keying (ERP/AAK).* IETF Internet Draft, raft-ietf-hokey-erp-aak-03, November 2010.

[64] F.Bernal, R. Marin-Lopez, and A.F. Gomez-Skarmeta. *Key Distribution Mechanisms For IEEE 802.21-Assisted Wireless Heterogeneous Networks.* In *ICWMC 2010: Proceedings of the Sixth International Conference on Wireless and Mobile Communications*, pages , 2010.

[65] T. Clancy, M. Nakhjiri, V. Narayanan, and L. Dondeti. *Handover Key Management and Re-authentication Problem Statement.* IETF RFC 5169, March 2008.

[66] R. Marin, J. Bournelle, M. Maknavicius-Laurent, J.M. Combes, and A. Gomez-Skarmeta. *Improved EAP keying framework for a secure mobility access service.* In *Proc. of International Wireless Communications & Mobile Computing Conference 2006, IWCMC 2006*, pages 183 188, Vancouver, British Columbia, Canada, March 2006.

[67] 3GPP TS 33.102 V7.1.0, Dec. 2006. 3rd Generation Partnership Project.

[68] V. Narayanan and L. Dondeti. *EAP Extensions for EAP Re-authentication Protocol (ERP).* IETF RFC 5296, Aug. 2008.

[69] R. Housley and B. Aboba. *Guidance for Authentication, Authorization, and Accounting (AAA) Key Management.* IETF RFC 4962, July 2007.

[70] A. Mishra, M. Shin, N. Petroni, C. Clancy, and W. Arbaugh. *Proactive Key Distribution Using Neighbor Graphs. IEEE Wireless Communication 11 (2004) 26 36.*

[71] D. Harskin, Y. Ohba, M. Nakhjiri, and R. Marin. *Problem Statement and Requirements on a 3-Party Key Distribution Protocol for Handover Keying.* IETF Internet Draft, draft-ohba-hokey-3party-keydist-ps-01, March 2007.

[72] S. Winter and K. Hoeper. *Threat Model for Networks Employing AAA Proxies.* Internet Draft, draft-hoeper-proxythreat-02, March 2009.

[73] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, Oct. 1996.

[74] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. *The Kerberos Network Authentication Service (V5).* IETF RFC 4120, July 2005.

[75] L. Tian, R. Dojen, and T. Coffey. *Extending Formal Security Protocol Specification Languages for use with New Verification Techniques. WSEAS Transactions on Information Science and Applications*, 3(2):372 378, February 2006.

[76] K. Fan, L. Kai, H. Li, and Y. Wang. *Security Analysis of the Kerberos Protocol Using BAN Logic.* In *Proceedings of the 2009 Fifth International Conference on Information Assurance and Security - Volume 02*, IAS '09, pages 467 470, Washington, DC, USA, 2009. IEEE Computer Society.

[77] F. Butlera, I. Cervesatob, A. D. Jaggardc, A. Scedrovd, and C. Walstadd. *Formal analysis of Kerberos 5. Theoretical Computer Science*, 367(1-2):57 87, 2006.

[78] Q. Li, F.Yang, Z. Fan, Z. Huibiao, and L. Zhu. *Formal Modeling and Analyzing Kerberos Protocol.* In *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering - Volume 07*, pages 813 819, Washington, DC, USA, 2009. IEEE Computer Society.

[79] G. Apostolopoulos, V. Peris, and D. Saha. *A Critical Review of 10 years of Privacy Technology.* In *Proc. of Proceedings of Surveillance Cultures: A Global Surveillance Society?*, UK, April 2010.

[80] A. Pfitzmann and M. Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management.* http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf, Aug. 2010. v0.34.

[81] H. Chen, Y. Xiao, X. Hong, F. Hu, and J. Xie. *A survey of anonymity in wireless communication systems. Security and Communication Networks*, 2(5):427 444, 2008.

[82] J. Krumm. *A survey of computational location privacy. Personal Ubiquitous Computing*, 13:391 399, August 2009.

[83] M. Bagnulo, A. Garcia-Martines, and A. Azcorra. *An Architecture for Network Layer Privacy.* In *ICCC 2007: International Conference on Communications 2007*, pages 1509 1514, Washington, DC, USA, 2007.

[84] D. Christin, M. Hollick, and M. Manulis. *Security and Privacy Objectives for Sensing Applications in Wireless Community Networks.* In *ICCCN 2010: Proceedings of 19th International Conference on Computer Communications and Networks*, pages 1095 2055, Washington, DC, USA, 2010. IEEE Computer Society.

[85] R.S. Cardoso, R. Speicys, and I. Valerie. *Architecting Pervasive Computing Systems for Privacy: A Survey.* In *WICSA 2007: Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture*, page 26, Washington, DC, USA, 2007. IEEE Computer Society.

[86] G. Karopoulos, G. Kambourakis, S. Gritzalis, and E. Konstantinou. *A framework for identity privacy in SIP. Journal of Network and Computer Applications*, vol. 33(1):pp. 16 28, Jan. 2010.

[87] J.Ghosh, M.J. Beal, H.Q. Ngo, and C. Qiao. On profiling mobility and predicting locations of wireless users. In *REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 55 62, New York, NY, USA, 2006. ACM.

[88] J. Zhu and J. Ma. A new authentication scheme with anonymity for wireless environments. *IEEE Transactions on Consumer Electronics*, 50(1):231 235, 2004.

[89] C-C. Lee, M-S. Hwang, and I-E Liao. Security Enhancement on a New Authentication Scheme With Anonymity for Wireless Environments. *IEEE Transactions on Industrial Electronics*, 53(5):1683 1687, 2006.

[90] J. Go and K. Kim. Wireless authentication protocols preserving user anonymity. In *Proc. of the 2001 Symposium on Cryptography and Information Security (SCIS 2001)*, pages 159 164, Oiso, Japan, January 2001. IEEE Computer Society Press.

[91] C-S Park. Authentication protocol providing user anonymity and untraceability in wireless mobile communication systems. *Computer Networks*, 44(2):267 273, 2004.

[92] Y. Jiang, C. Lin, and M. Shi. Mutual Authentication and Key Exchange Protocols for Roaming Services in Wireless Mobile Networks. *IEEE Transactions on Wireless Communications*, 5(9):2569 2577, 2006.

[93] C. Boyd and A. Mathuria. Protocols for Authentication and Key Establishment. Ed. Springer, 1st edition, Sept. 2003.

[94] F. Pereniguez, R. Marin, and A.F.G. Skarmeta. *Analisis de Propuestas de Re-autenticación Rapida en Entornos Moviles*. In *JITEL 2008: VII Jornadas de Ingenieria Telematica*, pages 233 240, Alcala de Henares, Madrid, 2008.

[95] F. Pereniguez. Pre-autenticación y Securización del Tráfico en Entornos de Movilidad sobre Redes 802.11. Editorial Euroeditions, 1st edition, Mayo 2008.

[96] R. Marin-Lopez, Y. Ohba, F. Pereniguez, and A.F. Gomez. *Analysis of Handover Key Management schemes under IETF perspective. Computer Standards & Interfaces*, 32(5-6):266 273, 2010. Information and communications security, privacy and trust: Standards and Regulations.

[97] R. Marin-Lopez, F. Pereniguez, Y. Ohba, F.Bernal, and A.F. Gomez-Skarmeta. *A transport-based architecture for fast re-authentication in wireless networks.* In *SARNOFF'09: Proceedings of the 32nd international conference on Sarnoff symposium*, pages 40 44, Piscataway, NJ, USA, 2009. IEEE Press.

[98] F.Pereniguez-Garcia, R. Marin-Lopez, F. Bernal-Hidalgo, and A. Gomez-Skarmeta. *Architecture for Fast EAP Re-authentication based on a new EAP method (EAP-FRM) working on standalone mode.* IETF Internet Draft, IETF draft-marin-eap-frm-fastreauth-03.txt, March 2011.

[99] R. Marin Lopez, F. Pereniguez Garcia, F. Bernal Hidalgo, and Antonio F. Gomez Skarmeta. *Procedimiento de Re-autentication.* European Patent, Publication N° PCT/ES2010/070069, September 2010.

[100] R. Marin Lopez, F. Pereniguez Garcia, Y. Ohba, F. Bernal Hidalgo, and A.F. Gomez-Skarmeta. *A Kerberized Architecture for Fast Re-authentication in Heterogeneous Wireless Networks. MONET*, 15(3):392 412, 2010.

[101] F. Pereniguez, G. Kambourakis, R. Marin-Lopez, S. Gritzalis, and A.F. Gomez. *Privacy-enhanced fast re-authentication for EAP-based next generation network. Computer Communications*, 33(14):1682 1694, 2010.

[102] J. Vollbrecht et al. *AAA Authorization Application Examples.* IETF RFC 2905, Aug. 2000.

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

[104] T. Dierks and C. Allen. *The TLS Protocol Version 1.0.* IETF RFC 2246, Jan. 1999.

[105] S. Winter, M. McCauley, S. Venaas, and K. Wierenga. *TLS encryption for RADIUS.* IETF Internet-Draft, July 2010.

[106] P. Calhoun, G. Zorn, D. Spence, and D. Mitton. *Diameter Network Access Server Application.* IETF RFC 4005, Aug. 2005.

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

[108] P. Eronen, T. Hiller, and G. Zorn. *Authentication, Authorization, And Accounting: Protocol evaluation.* IETF RFC 3127, June 2001.

[109] R. Dantu, G. Clothier, and A. Atri. EAP Methods for Wireless Networks. *Computer Standards Interfaces*, 29(3):289 301, 2007.

[110] J. Salowey, L. Dondeti, V. Narayanan, and M. Nakhjiri (2008). *Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK).* RFC 5295, August 2008.

[111] D. Stanley, B. Aboba, and J. Walker. *Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs.* IETF RFC 4017, March 2005.

[112] W. A. Arbaugh, N. Shankar, and Y. Wan. *Your 802.11 Wireless Network has No Clothes. IEEE Wireless Communications*, vol. 9(1):pp. 44 51, Nov. 2006.

[113] *Protocol for carrying Authentication for Network Access (PANA).* http://www.ietf.org/html.charters/pana-charter.html.

[114] W. Stallings. *SNMPv3: A Security Enhancement for SNMP. IEEE Comunication Surveys*, 1998. http://www.comsoc.org/livepubs/surveys/public/4q98issue/stallings.html.

[115] Institute of Electrical and Electronics Engineers, Draft IEEE Standard for Local and Metropolitan Area Networks: Media Independent Handover Services, 2008. IEEE Standards for Information Technology.

[116] K. Taniuchi, Y. Ohba, V. Fajardo, S. Das, M. Tauil C. Yuu-Heng, A. Dutta, D. Baker, M. Yajnik, and D. Famolari. *IEEE 802.21: Media independent handover: Features, applicability, and realization.* *IEEE Communications Magazine*, 47(1):112 120, January 2009.

[117] *Option III: EAP to conduct service authentication and MIH packet protection.* 21-10-0078-08-0sec-option-iii-eap-over-mih-service-authentication, November 2010.

[118] K. Raeburn. *Encryption and Checksum Specifications for Kerberos 5.* IETF RFC 3961, Feb. 2005.

[119] O. Dubuisson. *ASN.1 - Communication Between Heterogeneous Networks.* Morgan Kaufmann Publishers, 1st edition, Aug. 2001.

[120] J. Loughney, M. Nakhjiri, C. Perkins, and R. Koodli. *Context Transfer Protocol (CXTP).* IETF RFC 4067, July 2005.

[121] Y. Ohba, Q. Wu, and G. Zorn. *Extensible Authentication Protocol (EAP) Early Authentication Problem Statement.* IETF RFC 5836, April 2010.

[122] IEEE 802.11r Std., Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 8: Fast BSS Transition, Dec. 2005.

[123] *IEEE 802.21a. Proactive Authentication and MIH Security.* 21-09-0102-02-0sec-proactive-authentication-and-mih-security.doc, September 2009.

[124] *IEEE 802.21 Security Study Group Technical Report.* 21-08-0172-02-0sec-21-08-0012-02-0sec-mih-security-technical-report.doc, December 2008.

[125] T. Clancy and H. Tschofenig. *EAP Generalized Pre-Shared Key (EAP-GPSK).* IETF RFC 5433, Feb. 2009.

[126] *Handover Keying (hokey) IETF Working Group.* http://www.ietf.org/html.charters/hokey-charter.html.

[127] K. Hoeper, M. Nakhjiri, and Y. Ohba. *Distribution of EAP based keys for handover and re-authentication.* IETF RFC 5749, March 2010.

[128] J. Vollbrecht, P. Eronen, N. Petroni, and Y. Ohba. *State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator.* IETF RFC 4137, Aug. 2005.

[129] 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 2007.

[130] 3GPP TS 29.230 Diameter applications; 3GPP specific codes and identifiers, Release 7, Sept. 2007. 3rd Generation Partnership Project.

[131] C. Tang and D. Oliver Wu. Mobile Privacy in Wireless Networks - Revisited. *IEEE Transactions on Wireless Communications*, 7(3):1035 1042, 2008.

[132] W. Juang and J. Wu. *Efficient 3GPP Authentication and Key Agreement with Robust User Privacy Protection.* In *Wireless Communications and Networking Conference, WCNC 2007*, pages 2720 2725, Kowloon, China, March 2007.

[133] Y. Ohba, S. Das, and R. Marin. *An EAP Method for EAP Extension (EAP-EXT).* IETF Internet Draft, draft-ohba-hokey-emu-eap-ext-02, July 2007.

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

[135] *Linux WPA/WPA2/IEEE 802.1X Supplicant Software.* http://hostap.epitest.fi/wpa_supplicant.

[136] *IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator Software.* http://hostap.epitest.fi/hostapd.

[137] *Free Radius.* http://www.freeradius.org.

[138] *WIRESHARK.* http://www.wireshark.org.

[139] T.Coffey, R. Dojen, and T. Flanagan. *Formal verification: an imperative step in the design of security protocols. Elsevier Computer Networks*, 43:601 618, December 2003.

[140] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5).* IETF RFC 1510, Septembre 1993.

[141] *Frequently Asked Question About the MIT Kerberos Consortium.* http://www.kerberos.org/about/FAQ.html.

[142] *MIT Kerberos Distribution.* http://web.mit.edu/Kerberos/.

[143] *Heimdal Kerberos Distribution.* http://www.h5l.org/.

[144] *GNU Shishi Kerberos Distribution.* http://www.h5l.org/.

[145] M. A. Kaafar, L. Benazzouz, F. Kamoun, and D. Males. *Kerberos-Based Authentication Architecture for Wireless LANs.* In *Proc. of IFIP Networking'04*, volume LNCS 3042, pages 1344 1353, Athens, Greece, May 2004.

[146] S. Zrelli and Y. Shinoda, "Specifying Kerberos over EAP: Towards an integrated network access and Kerberos single sign-on process," *Proc. 21st International Conference on Advanced Networking and Applications* (AINA'07), 2007, pp. 490-497.

[147] S. Zrelli and Y. Shinoda. *EAP Fast Re-Authentication Protocol (EAP-FRAP)*. IETF Internet Draft, draft-zrelli-eap-frap-04, June 2008.

[148] S. Zrelli, A. Miyaji, Y. Shinoda, and T. Ernst. *Security and Access Control for Vehicular Communications. IEEE International Conference on Wireless and Mobile Computing, Networking and Communication*, pages 561 566, 2008.

[149] Y. Ohba, S. Das, and A. Dutta. *Kerberized Handover Keying: A Media-Independent Handover Key Management Architecture.* In *Proc. of The 2nd ACM International Workshop on Mobility in the Evolving Internet Architecture , ACM MobiArch 2007*, Kyoto, Japan, Aug. 2007.

[150] H. Moustafa, G. Bourdon, and Y. Gourhant. *AAA in Vehicular Communication on Highways with Ad hoc Networking Support: A Proposed Architecture.* In *Proc. of the 2nd ACM International Workshop on Vehicular ad hoc Networks*, pages 79 80, Cologne, Germany, Sept. 2005.

[151] H. Almus, E. Brose, and K. Rebensburg. *A Kerberos-based EAP Method for re-authentication with integrated support for fast handover and IP mobility in Wireless LANs.* In *Proc. of The 2nd International Conference on Communications and Electronics, ICCE 2008*, pages 61 66, Las Vegas, USA, June 2008.

[152] B. Aboba, M. Beadles, J. Arkko, and P. Eronen. *The Network Access Identifier.* IETF RFC 4282, Dec. 2005.

[153] 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.

[154] Assertions and protocol for the OASIS Security Assertion Markup Language (SAML) V1.1, Sept. 2003. OASIS standard.

[155] *Host AP software.* http://hostap.epitest.fi.

[156] MIT Information Systems. *Kerberos V5 Application Programming Library*, Sept. 2008.

[157] KERNAC: Kerberized Network Access Control. http://kernac.codealias.info.

[158] *The Network Simulator - NS-2.* http://www.isi.edu/nsnam/ns.

[159] *Seamless and Secure Mobility.* http://www.antd.nist.gov/seamlessandsecure.shtml.

[160] *BonnMotion - A Mobility Scenario Generation and Analysis Tool.* http://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion/.

[161] O. Tene. *Privacy: The New Generations. Oxford Journal International Data Privacy Law*, pages 1 13, Nov. 2010.

[162] N. J. King and P. W. Jessen. *Profiling the mobile customer ? Privacy concerns when behavioural advertisers target mobile phones. Computer Law & Security Review*, 26(5):455 478, 2010.

[163] A. Medvinsky, J. Cargille, and M. Hur. *Anonymous Credentials in Kerberos.* IETF Internet Draft, IETF draft-ietf-cat-kerberos-anoncred-00.txt, March 1998.

[164] L. Zhu and B. Tung. *Public Key Cryptography for Initial Authentication in Kerberos (PKINIT).* IETF RFC 4556, June 2006.

[165] L. Zhu, P. Leach, and S. Hartman. *Anonymity Support for Kerberos.* IETF Internet Draft, IETF draft-ietf-krb-wg-anon-12.txt, August 2010.

[166] S. Josefsson. *Using Kerberos V5 over the Transport Layer Security (TLS) protocol.* IETF Internet Draft, IETF draft-josefsson-kerberos5-starttls-09.txt, August 2010.

[167] G. Apostolopoulos, V. Peris, and D. Saha. *Transport Layer Security: How much does it really cost?* In *Proc. of IEEE INFOCOM 1999 Vol. 2*, pages 717 725, New York, NY, USA, March 1999. IEEE Computer Society.

[168] M. Shimaoka, N. Hastings, and R. Nielsen. *Memorandum for Multi-Domain Public Key Infrastructure Interoperability.* IETF RFC 5217, July 2008.

[169] S. Hartman and L. Zhu. *A Generalized Framework for Kerberos Pre-Authentication.* IETF Internet Draft, draft-ietf-krb-wg-preauth-framework-17, June 2010.

[170] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. *Encryption and Checksum Specifications for Kerberos 5.* IETF RFC 3961, February 2005.

[171] A. Medvinsky, M. Hur, D. Brezinski, G. Tsudik, and B. Tung. *Integrity Protection for the Kerberos Error Message.* IETF Internet Draft, raft-ietf-cat-kerberos-err-msg-00, September 1997.

[172] K. Raeburn. *Advanced Encryption Standard (AES) Encryption for Kerberos 5.* IETF RFC 3962, Feb. 2005.

[173] *C. Madson* and R. Glenn. *The Use of HMAC-SHA-1-96 within ESP and AH.* IETF RFC 2404, Nov. 1998.

[174] S. Cantor, J. Kemp, R. Philpott, and E. Maler (Eds.). *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0*, March 2005.

[175] *eXtensible Access Control Markup Language (XACML) Version 2.0*, February 2005. OASIS Standard.

[176] P. Syverson. *On key distribution protocols for repeated authentication. ACM SIGOPS Operating Systems Review*, 27(4), 1993.

[177] W. Lootah, W. Enck, and P. McDaniel. *TARP: Ticket-based address resolution protocol. Elsevier Computer Networks*, 51(4):4322 4337, 2007.

[178] J. Lee I. You and B. Kim. *caTBUA: Context-aware ticket-based binding update authentication protocol for trust-enabled mobile networks. Wiley International Journal of Communication Systems*, 23:1382 1404, 2010.

[179] J.A. Martinez, P.M. Ruiz, and R. Marin. *Impact of the Pre-Authentication Performance in Vehicular Networks.* In *2010 IEEE 72nd Vehicular Technology Conference Fall (VTC 2010-Fall)*, pages 1 5, Sept. 2010.

[180] J.A Martinez, P.M. Miguel Ruiz, R. Marin-Lopez, and F. J. Ros. *Enhanced access control in hybrid MANETs through utility-based pre-authentication control. Wireless Communications and Mobile Computing*, 10:688 703, May 2010.

[181] L. Buttyán and J. Hubaux. Security and Cooperation in Wireless Networks. Cambridge University Press, 1st edition, Nov. 2007.

[182] J. Howlett and S. Hartman. *Project Moonshot.* February 2010.

[183] *Application Bridging for Federated Access Beyond web (abfab) IETF Working Group.* http://datatracker.ietf.org/wg/abfab/charter/.

[184] *PAR for an Amendment to an existing IEEE Standard 802.21-2008.* https://mentor.ieee.org/802.21/dcn/09/21-09-0010-00-0sec-p802-21a-par.pdf.