UNIVERSITAT POLITÈCNICA DE CATALUNYA

TESIS DOCTORAL

Seguridad en redes de computación ubicua: contribución a la validación de credenciales

Autor: **María Francisca Hinarejos Campos**

Ingeniera en Telecomunicaciones

Director: Dr. Jordi Forné Muñoz

Co-director: Dr. José L. Muñoz Tapia

DEPARTAMENT D'ENGINYERIA TELEMÀTICA

Tesis presentada para obtener el título de Doctora

por la Universitat Politècnica de Catalunya

Barcelona, 24 de mayo de 2010

Resumen

El avance tecnológico tanto de los dispositivos de usuario como de las redes permite que se puedan establecer comunicaciones en cualquier momento y en cualquier lugar. Si bien estos entornos ofrecen un gran abanico de posibilidades a los usuarios, también es cierto que generan nuevas amenazas. Por este motivo, son necesarias medidas que permitan saber con quién se está estableciendo la comunicación y qué acciones se pueden autorizar. Las soluciones propuestas en la literatura no se adaptan completamente a las nuevas características de movilidad, desconexión y limitaciones tanto de los dispositivos como de las redes. De hecho, muchas de las propuestas existentes se han centrado en ofrecer soluciones concretas a escenarios particulares, sin tener en cuenta que el usuario puede entrar a formar parte de entornos heterogéneos. Por lo tanto, se hace necesario diseñar mecanismos de seguridad que conviviendo con los estándares vigentes, se adapten a los nuevos escenarios. En este sentido, los certificados digitales son una solución estandarizada y ampliamente extendida. Los certificados digitales permiten llevar a cabo tanto la autenticación como la autorización de un usuario de forma distribuida. Sin embargo, las características de los entornos ubicuos complican el proceso de validación de certificados. Esta complejidad podría llevar a que no se pudiera acceder a los servicios.

El objetivo de esta tesis es contribuir a aumentar la seguridad en entornos ubicuos. Más concretamente, se proporcionan soluciones para reducir la carga en la validación de credenciales y aumentar la disponibilidad de los servicios de autenticación y autorización. En primer lugar se propone un sistema de verificación de credenciales que se adapta para funcionar tanto en entornos con conexión a servidores *on-line*, como en sistemas *off-line*. Por otra parte, el proceso de delegación en sistemas de autorización, aporta una gran flexibilidad a estos entornos, pero a su vez añade complejidad al sistema. Para reducir esta carga sobre el verificador se propone un sistema de revocación en cascada con delegación centralizada. Sin embargo, esta centralización del servicio limita la escalabilidad y flexibilidad de la solución. Para dar solución a ese inconveniente, se ha propuesto un sistema de revocación en cadenas de delegación basado en códigos prefijo. Esta solución permite mantener la reducción de la carga en la validación lograda en la propuesta centralizada, y además, hace posible la delegación

dinámica y la distribución de la información de revocación. Esta distribución puede realizarse a través de listas de revocación de credenciales. En redes con desconexión temporal esta información podría no estar accesible. Para solventarlo, se ha propuesto un sistema en el que los usuarios pueden realizar las funciones de servidores de revocación sin ser entidades de confianza. De esta forma se permite aumentar la disponibilidad del servicio de validación, y reducir el consumo de los recursos. Cada una de las propuestas realizadas se ha analizado para verificar las mejoras proporcionadas frente a las soluciones existentes. Para ello, se han evaluado de forma analítica, por simulación y/o implementación en función de cada caso. Los resultados del análisis verifican el funcionamiento esperado y muestran las mejoras de las propuestas frente a las soluciones existentes.

Abstract

Technology progress in both user devices and networks allows communications anytime and anywhere. New communication environments offer a wide range of possibilities to users, but also generate new threats. For this reason, it is necessary to establish measures to find out who is establishing a communication and what actions is authorized to do. Currently proposed solutions in the literature are not completely adapted to the new features such as user mobility, network disconnections and constraints of devices and networks. Many of the existing proposals have focused in providing specific solutions to particular scenarios, but they do not consider a global heterogeneous scenario. Therefore, it is necessary to design security mechanisms able to adapt themselves to new scenarios. In this sense, digital certificates are a standardized and widely used solution. Digital certificates enable performing user authentication and authorization in a distributed way. The problem is that ubiquitous environments complicate the process of digital certificates validation. This complexity could result in a service being not accessible.

The goal of this thesis is to contribute in making ubiquitous scenarios more secure. More specifically, the work proposes solutions for reducing the credential validation cost and for improving the availability of authentication and authorization services. In first place, we propose a solution for credential validation that works properly in environments with connection to on-line servers and also in environments where the connection to servers is sometimes not possible. In second place, we propose a cascade revocation system where the delegation is partially centralized. Delegation provides high flexibility to authorization systems, but adds complexity to the system. Our proposal reduces the burden on the verifier-side. In third place, we propose a revocation system for delegation chains based on prefix codes. This proposal deals with the problem of centralization of the previous proposal. In particular, the decentralized solution presented keeps the load reduction achieved in the partially centralized proposal, and also enables dynamic delegation and distribution of revocation data. While the user is connected, revocation data distribution can be done with a certificate revocation list. However, in scenarios where the connection can be lost temporally, this might not be possible. To address this issue, we have proposed a system in which users

can perform the functions of revocation servers without being trusted entities. This will allow increasing the availability of validation service, and reduce resource consumption. Each proposal has been analyzed and compared with existing solutions to verify the improvements achieved.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

AA Attribute Authority

AC Attribute Certificate

ACE Access Control Engine

ACRL Attribute Certificate Revocation List

ADOPT Ad-hoc Distributed OCSP for Trust

AES Advanced Encryption Standard

AODV Ad-hoc On Demand Distance Vector

ASN.1 Abstract Syntax Notation One

BFS Breath-First Search

CA Certification Authority

CARE CAscade REvocation

CERVANTES CERtificate VAlitatioN TESt-bed

CRL Certificate Revocation List

DER Distinguished Encoding Rules

DES Data Encryption Standard

DFS Depth First Search

DN Distinguished Name

DoS Denial of Service

DP Delegation Path

DSL Depth-limited Search

DSR Dynamic Source Routing

ECMA European Computer Manufacturer's Association

EE End Entity

eRL Extended Revocation List

ETSI European Telecommunications Standards Institute

GPRS General Packet Radio Service

GPL General Public License

IDS Iterative Deepening Search

ITU-T Telecommunication Standardization Sector

J2ME Java 2 Micro Edition

LDAP Lightweight Directory Access Protocol

MANET Mobile Ad-hoc NETwork

MD5 Message Digest 5

MHT Merkle Hash Tree

OCSP Online Certificate Status Protocol

OSI Open System Interconnection

OWHF One Way Hash Function

PAC Privilege Attribute Certificate

PAS Photo Album Service

PDA Personal Digital Assistant

PDP Pervasive Discovery Protocol

PGP Pretty Good Privacy

PKC Public Key Certificate

PKI Public Key Infrastructure

PMI Privilege Management Infrastructure

PREON Prefix REvocatiON

PTM Pervasive Trust Model

PV Privilege Verifier

RA Registration Authority

REPLIP REPLIcation Protocol

RETRIP RETrieval Revocation Information Protocol

RL Revocation List

RLE Run Length Encoding

RTO ReTransmissiOn

SD Service Discovery

SDSI WG Simple Distributed Security Infrastructure Working Group

SHA-1 Secure Hash Algorithm-1

SOA Source of Authority

SPKI Simple Public Key Infrastructure

TLS Transport Layer Security

TLV Type Length Value

TTP Trusted Third Party

UC3M Universidad Carlos III de Madrid

UML Unified Modeling Language

UMTS Universal Mobile Telecommunications System

URL Uniform Resource Locator

WiFi Wireless Fidelity

PREFACIO

La tecnología de la información está en constante evolución. Los inicios estuvieron dominados por computadoras *mainframe*, que vieron el inicio del fin a finales de los 80 y principios de los 90, cuando fueron reemplazados por los sistemas cliente-servidor. Esta evolución permitió disminuir significativamente el coste computacional utilizando una combinación de estaciones de trabajo y computadoras personales. Desde entonces, el paradigma de la computación ha sufrido una mayor transformación con la llegada de Internet y que se conoce como *network computing*. Pero la tecnología no se detuvo ahí. La llegada de las redes *wireless*, tanto GPRS-UMTS como Wi-Fi permitió que el acceso de los usuarios a la red pudiera realizarse a través de dispositivos móviles. Esta evolución se puede describir con el término *mobile computing*. Esta tecnología suele implicar la necesidad de un dispositivo móvil el cual puede tener limitaciones en cuanto a capacidad de cómputo, memoria y batería.

Después de los sistemas distribuidos y de la computación móvil cada vez se están escuchando más los términos computación y comunicación ubicua. La computación ubicua es el paradigma donde la tecnología llega a ser virtualmente invisible para el usuario dentro de los objetos. Pero para que esta tecnología realmente permita ofrecer un amplio abanico de posibilidades, sería necesario que los dispositivos no fueran meros objetos aislados. La comunicación ubicua permite que todos estos objetos puedan comunicarse entre sí y con el usuario por medio de redes que no requieren de la existencia de infraestructura, como sucedía hasta ahora. Este hecho proporciona mayor libertad para el usuario, a la vez que abre nuevas oportunidades de negocio.

Este nuevo enfoque implica mayor conectividad y acceso por parte de los usuarios, extendiéndose las posibilidades de establecimientos de comunicación entre ellos. Sin embargo, habitualmente los términos conectividad y seguridad parecen enfrentados. Este aumento de conectividad tiene sus implicaciones a la hora de ofrecer servicios de seguridad, abriendo nuevos retos a los que se debe dar solución.

En esta tesis se dan soluciones a la reducción del coste y el aumento de la disponibilidad del servicio de autenticación y autorización basado en credenciales para poder responder a la pregunta de *qué está autorizado a realizar una determinada entidad sobre los recursos en redes ubicuas*.

CAPÍTULO 1

INTRODUCCIÓN

1.1. Motivación y Objetivos

En los inicios de las comunicaciones y de la informática en general, el acceso a la información se realizaba de forma local. Si se necesitaba tener acceso a los recursos de un sistema, se requería la presencia física del usuario, lo que implicaba centralizar el control de acceso con la protección física del sistema. La evolución de los sistemas ha creado la necesidad de ampliar el acceso a los recursos a un ámbito distribuido, que permita que los usuarios puedan acceder a ellos de forma remota, ya bien sea a través de redes internas o de redes externas.

Este cambio de funcionamiento implica el aumento del riesgo de accesos fraudulentos, y por lo tanto, la necesidad de aumentar la seguridad de los sistemas. Cuando un usuario accede al sistema se deben llevar a cabo una serie de medidas de seguridad. En primer lugar, se debería verificar su identidad, de manera que se pueda comprobar que el usuario es quien dice ser. A este procedimiento se le conoce como autenticación. Una vez se conoce su identidad, se deberían obtener los privilegios asociados al usuario para evaluar si son suficientes para realizar la operación solicitada, lo que se conoce como autorización. Y si fuera necesario, la información intercambiada debería poder ser transmitida confidencialmente, de manera que entidades no participantes en la comunicación no pudieran interpretar la información.

La infraestructura de clave pública (PKI - *Public Key Infrastructure*) se ha erigido como la solución en la que se basa el servicio de autenticación en sistemas distribuidos, ofreciendo una solución escalable. Este esquema permite que los usuarios puedan obtener un certificado de clave pública (PKC - *Public Key Certificate*), que a

menudo se conoce como certificado de identidad. Este certificado es emitido por una entidad de confianza, la cual acredita mediante una firma, el enlace entre una entidad y una clave pública. Para poder comprobar que la entidad es quien dice ser, se utiliza un protocolo de autenticación basado en la demostración de que el usuario posee la clave privada asociada a la clave pública contenida en el certificado. A continuación, se ha de comprobar que el certificado presentado todavía es válido. Este procedimiento es necesario ya que los certificados son de carácter temporal. La validación de certificados implica la obtención de información que debe estar actualizada y disponible para cualquier entidad que quiera comprobar la validez de un certificado.

Una vez tenemos el mecanismo adecuado para lograr la autenticación, el segundo paso es verificar los privilegios del usuario. Tradicionalmente los privilegios asociados a una entidad no eran presentados por la propia entidad, sino que el sistema que controlaba los recursos gestionaba la información de forma local, asociando los privilegios a información de identificación. De la misma forma que se consigue ganar en escalabilidad, disponibilidad y seguridad mediante la autenticación de claves públicas a través de certificados, éstos también son una solución adecuada para autenticar privilegios. Este tipo de certificados se conocen como certificados de atributo.

La infraestructura de gestión de privilegios (PMI - *Privilege Management Infrastructure*) es al certificado de atributo (AC - *Attribute Certificate*) lo que la PKI a los certificados de clave pública. En ambas infraestructuras la validación de certificados es un aspecto crítico, ya que de él depende el éxito de la verificación de la identidad o de los privilegios de una determinada entidad, y por lo tanto, de permitir o no el acceso a los servicios. Esta validación es incluso más crítica con la evolución de la tecnología hacia sistemas ubicuos.

La computación ubicua permite establecer comunicaciones en cualquier momento y en cualquier lugar, lo que ha abierto tanto un abanico de posibilidades a los usuarios como la aparición de nuevas amenazas de seguridad. Las limitaciones tanto del canal como de los dispositivos de usuario dificultan, o hacen inviable la adopción de medidas de seguridad ya establecidas para otros entornos. La autenticación y autorización en entornos móviles se ha de realizar, en muchos casos, dinámicamente, por lo que dar soporte a estos servicios de una manera global no es una tarea sencilla. La movilidad de los usuarios a través de diferentes redes puede causar una pérdida de conectividad global, provocando una desconexión temporal de los servicios. Esta situación tiene un fuerte impacto en algunos de los servicios tanto de la PKI como de la PMI, tales como la verificación del estado de revocación de los certificados, ya que no es posible acceder a entidades *on-line*.

Por lo tanto, la movilidad tanto de los clientes como de las entidades que ofrecen servicios requiere de soluciones de seguridad de carácter global, que permitan una gestión distribuida y contemplen las limitaciones tanto de los dispositivos como de

las nuevas redes.

En esta tesis se proponen nuevas soluciones que permiten aumentar la disponibilidad de los servicios de autenticación/autorización en entornos de computación ubicua, y que permiten disminuir la carga de verificación en los procesos de validación de certificados. Para alcanzar estos objetivos se sigue la siguiente metodología:

- Estudio de los procesos involucrados y de las propuestas existentes para dar solución a los procedimientos críticos de la validación de certificados en entornos ubicuos.
- Propuesta de nuevas soluciones para reducir la carga de los procedimientos críticos en la validación de certificados que conlleven, o bien una carga elevada en la validación de certificados, o una probabilidad de denegación del servicio por falta de información disponible.
- Análisis de las nuevas soluciones propuestas. Para ello se han de identificar los parámetros a evaluar y comparar las soluciones propuestas frente a las soluciones existentes. Esta evaluación se puede realizar de forma analítica, por simulación, o mediante implementación, dependiendo de cada caso.

1.2. Contribuciones

De acuerdo con los objetivos y la metodología presentados en el apartado 1.1, se han realizado cuatro contribuciones principales. A continuación se presentan cada una de ellas junto a las contribuciones parciales realizadas dentro de cada una de ellas:

1. Diseño e implementación de un sistema de validación de certificados basado en confianza:
   - Análisis de los requerimientos de disponibilidad de información en el proceso de validación básico de cadenas de certificados.
   - Diseño de una arquitectura de validación de certificados para usuarios móviles. Esta arquitectura se ha diseñado de forma totalmente modular para simplificar la modificación o ampliación de funcionalidades, independizando la autorización de la autenticación. Esta arquitectura permite adaptar su funcionamiento en base a la información disponible en la red para llevar a cabo la autenticación y autorización de una entidad.
   - Implementación de la solución anterior sobre dispositivos móviles como las PDA (*Personal Digital Assistant*). Se ha verificado su viabilidad y correcto funcionamiento en un escenario de demostración en el marco del proyecto europeo UBISEC. El código de la implementación está accesible bajo una licencia

pública general (GPL - *General Public License*) desde la página web del proyecto.

2. Diseño y análisis de un sistema de revocación en cadenas de delegación:

   - Análisis de la revocación en cadenas de delegación. Durante el análisis se han identificado y definido formalmente dos parámetros principales para evaluar los mecanismos de validación del estado de revocación en cadenas de certificados.
   - Propuesta de una solución centralizada para llevar a cabo la revocación en cascada.
   - Análisis de la propuesta utilizando los parámetros identificados en el punto anterior.

3. Propuesta de una solución de revocación *ad-hoc* basada en códigos prefijo para cadenas de delegación:

   - Diseño de un sistema parcialmente descentralizado que permite mantener la reducción de la carga en el proceso de validación de la propuesta anterior. Así mismo, también se hace viable la distribución de la información de revocación mediante procedimientos *off-line*.
   - Extensión de funcionalidades en CERVANTES (*CERtificate VAlitatioN TESt-bed*), una plataforma de test para el estudio de los mecanismos de revocación. Entre las funcionalidades implementadas se encuentran las de permitir generar y analizar cadenas de delegación que dan lugar a topologías en árbol.
   - Análisis de la solución utilizando la plataforma anterior.

4. Propuesta de un sistema colaborativo de distribución de datos de revocación para redes híbridas:

   - Análisis de las soluciones para la distribución de información de revocación en redes donde la conexión a servidores centralizados no puede llevarse a cabo, o puede resultar muy costosa.
   - Propuesta de una solución para aumentar la disponibilidad de la información de revocación en el tipo de redes analizado en el punto anterior. Se ha utilizado una estructura estandarizada para ofrecer compatibilidad con las soluciones existentes. Además, la solución contempla la posibilidad de seleccionar el servicio de revocación en función de diferentes parámetros, como por ejemplo, servidor más cercano o con información más actualizada.

- Definición de un protocolo que permite a los nodos de la red colaborar para ofrecer el servicio de revocación. Este mecanismo permite distribuir la carga entre los dispositivos y aumentar la disponibilidad del servicio de revocación.
- Análisis de las prestaciones de la solución utilizando el simulador de red ns-2 (*network simulation-2*). La nueva aplicación desarrollada sobre el simulador, puede servir como base para otras aplicaciones que requieran distribuir información de forma periódica, ya que la aplicación desarrollada es independiente del contenido de los datos distribuidos.

El interés por el trabajo de investigación presentado en esta tesis está acreditado por los proyectos en los cuales se enmarca, como son:

- ARPA: Adquisición y Revocación de Privilegios en esquemas de Autorización. Proyecto CICYT - TIC2003-08184-C02-02.
- UBISEC: Ubiquitous Networks with a Secure Provision of Services, Access, and Content Delivery. Sixth Framework Programme - IST-2002-506926.
- ARES: Team for advanced research on information security and privacy. Proyecto Consolider del Ministerio de Educación y Ciencia - CSD2007-00004.

1.3. Publicaciones

En este apartado se enumeran las publicaciones que se han obtenido como resultado de la investigación de esta tesis, y que avalan el interés de investigación de la misma.

Publicaciones en revistas internacionales:

- (1) M.F. Hinarejos, J. L. Muñoz, J. Forné, O. Esparza. PREON: An Efficient Cascade Revocation Mechanism for Delegation Paths. Computers & Security. DOI: 10.1016/j.cose.2010.03.001.
- (2) J. Forné, F. Hinarejos, A. Marín, F. Almenárez, J. Lopez, J.A. Montenegro, M. Lacoste, D.Díaz. Pervasive Authentication and Authorization Infrastructures for Mobile Users. Computers & Security, 29(4):501-514, 2010.
- (3) J. Forné, J.L. Muñoz, F. Hinarejos, O. Esparza. Certificate Status Validation in Mobile Ad-Hoc Networks. IEEE Wireless Communications, 16(11):55-62, 2009.

Publicaciones en congresos internacionales:

- (4) M.F. Hinarejos, J. Forné. Revocation Scheme for PMI Based Upon the Tracing of Certificates Chains. Computational Science and Its Applications (ICCSA). Lecture Notes in Computer Science, pp. 1098-1106, ISSN 0302-9743 (ISBN 3-540-34077-7), 2006.

- (5) M.F. Hinarejos, J. Forné. A Lightweight Delegated Privileges Revocation Scheme Based on Coding. Proceeding of the 2005 conference on Applied Public Key Infrastructure (IWAP), IOS Press, pp. 207-221, ISSN 0922-6389 (ISBN 1-58603-550-9), 2005.

- (6) F. Almenarez, M. Carbonell, J. Forne, M.F. Hinarejos, M. Lacoste, A. Marin, J.A. Montenegro. Design of an Enhanced PKI for Ubiquitous Networks. Proceedings of sixteenth International Workshop on Database and Expert Systems Applications, pp. 262-266, ISSN 1529-4188, 2005.

- (7) J. Forné, M.F. Hinarejos. Web-based authorization based on X.509 privilege management infrastructure. IEEE Pacific Rim Conference on Communications, Computers and signal Processing (PACRIM), ISBN 0-7803-8179-3, 2003.

Publicaciones en congresos nacionales:

- (8) M.F. Hinarejos, J. Forné. L-DPR: un esquema ligero de revocación de privilegios delegados. V Jornadas de Ingeniería Telemática (JITEL), ISBN 84-8408-346-2, 2005.

- (9) M.F. Hinarejos, J. Forné, Florina Almenárez, Andrés Marín, Mildrey Carbonell, José A.Montenegro. Hacia una PKI mejorada para Comercio Electrónico Móvil. Simposio Español de Comercio Electrónico (SCE), ISBN 84-7632-929-6, 2005.

- (10) M.F. Hinarejos, J. Forné. Seguimiento de cadenas de certificados para un sistema de revocación. VIII Reunión Española Sobre Criptología y Seguridad de la Información (RECSI), ISBN 84-7978-650-7, 2004.

- (11) M.F. Hinarejos, J. Forné. Actualización de la Información de Revocación de Privilegios mediante el Seguimiento de Cadenas de Certificados. XII Jornadas Telecom I+D, 2004.

- (12) M.F. Hinarejos, J. Forné. Autorización en web basada en Infraestructuras de Gestión de Privilegios X.509. II Simposio Español de Comercio Electrónico (SCE), ISBN 84-932902-0-3, 2003.

1.4. Organización de la Tesis

La documentación de esta tesis se estructura en siete capítulos principales de la siguiente manera:

- El capítulo 2 recoge la información relativa al estado del arte referente a la validación de certificados. Se hace una introducción al concepto de computación ubicua centrándonos en sus características y peculiaridades, las cuales afectan al funcionamiento de las soluciones de autenticación y autorización existentes. Se presentan y analizan tanto los principales formatos de certificados para transportar la información de autenticación y autorización, como la infraestructura y los procedimientos necesarios para la validación de esos certificados.

- El capítulo 3 presenta una solución para poder llevar a cabo autenticación y autorización de usuarios en sistemas ubicuos, en los cuales la conexión a sistemas centralizados puede perderse. Esta solución es una integración de un sistema de verificación de privilegios adaptada para trabajar conjuntamente con un sistema basado en confianza.

- El capítulo 4 presenta un análisis de la revocación en cadenas de delegación. En este capítulo se identifican dos parámetros para evaluar los mecanismos de validación del estado de revocación en cadenas de certificados: el número medio de verificaciones del estado de revocación y la probabilidad de revocación de un certificado dentro de una cadena de certificados. También se presenta una solución centralizada para llevar a cabo la revocación en cascada, la cual se evalúa con los parámetros identificados.

- El capítulo 5 presenta un mecanismo de revocación en cascada basado en códigos prefijo. Esta solución nace como resultado del análisis realizado en el capítulo 4 para dar solución a la reducción del volumen de la información de revocación en la propuesta centralizada presentada en ese capítulo.

- El capítulo 6 estudia la aplicación de diferentes mecanismos para la validación de certificados en redes móviles inalámbricas y presenta un mecanismo cooperativo para la obtención de información de revocación. Este mecanismo se adapta a las limitaciones de la red y de los dispositivos, reduciendo tanto el ancho de banda de la red como el consumo de recursos de los dispositivos.

- El capítulo 7 recoge los resultados más relevantes obtenidos en esta tesis. También se perfilan las posibles líneas de investigación futuras.

CAPÍTULO 2

ESTADO DEL ARTE

2.1. Computación/Comunicación Ubicua

La tecnología de la información está constantemente evolucionando. Los inicios estuvieron dominados por computadoras *mainframe*, que vieron el inicio del fin a finales de los 80 y principios de los 90, cuando fueron reemplazados por los sistemas cliente-servidor. Esta evolución permitió disminuir significativamente su coste utilizando una combinación de estaciones de trabajo y computadoras personales. Desde entonces, el paradigma de la computación ha sufrido una mayor transformación con la llegada de Internet, la cual se conoce como *network computing*.

Pero la tecnología no se detuvo ahí, la llegada de las redes inalámbricas (*wireless networks*), tanto GPRS-UMTS (*General Packet Radio Service - Universal Mobile Telecommunications System*) como Wi-Fi (*Wireless Fidelity*) permitió que el acceso de los usuarios a la red pudiera realizarse a través de dispositivos móviles, cuya evolución se puede describir con el término *mobile computing*. Este término describe el proceso de utilizar un computador mientras el usuario se desplaza, lo que suele implicar la necesidad de un dispositivo móvil que tal vez pueda tener limitaciones en cuanto a capacidad de cómputo, memoria y batería. Sin embargo, todavía se requiere de una cierta infraestructura para poder acceder a los recursos ofrecidos por la red u otros dispositivos, y en la que la tecnología no es transparente para el usuario.

La computación ubicua es el paradigma donde la tecnología llega a ser virtualmente invisible para el usuario. En lugar de tener un computador de sobremesa o un portátil, la tecnología se incluirá en nuestro entorno. La Xerox PARC (*Palo Alto Research Center*) hace la siguiente descripción:

"Imaginar un mundo con cientos de dispositivos con tecnología wireless dentro de una misma habitación los cuales pueden comunicarse entre sí y cuya tecnología se hace invisible..."

La evolución de la tecnología no sólo ha sido de los dispositivos, lo que ha permitido la reducción del coste y tamaño de los dispositivos de usuario. Asociada a esta movilidad y a la posibilidad de establecer comunicaciones en cualquier momento y en cualquier lugar, se requiere que la tecnología de red lo permita.

El estándar IEEE 802.11 (13) define el protocolo para la construcción de dos tipos de redes inalámbricas principales, *ad-hoc* y cliente/servidor, aunque nosotros extenderemos esta clasificación con el concepto de red híbrida. A continuación se hace una clasificación a grandes rasgos de los tipos de redes que han permitido la evolución de una comunicación centralizada a una comunicación ubicua. En función de si existe o no cierta infraestructura que gestione la comunicación, podemos clasificarlas de la siguiente manera:

- Redes cliente/servidor con infraestructura (figura 2.1): este tipo de redes utilizan un punto de acceso que controla la asignación del tiempo de transmisión para todas las estaciones. Por lo tanto, la red deja de ser específicamente de contienda y cada estación pasa a tener el control del medio sólo cuando es autorizada por el punto de acceso centralizado. Es decir, el punto de acceso se utiliza para controlar el tráfico de esa red inalámbrica, ya sea entre los dispositivos de la propia red, o entre los dispositivos de esa red con otros de otra red (tanto cableada como inalámbrica).

  A pesar de la existencia de infraestructura, el dispositivo del usuario puede ser tanto fijo como móvil. El sistema de cableado telefónico y las telecomunicaciones móviles como GPRS y UMTS, son algunos ejemplos de redes con infraestructura fija.

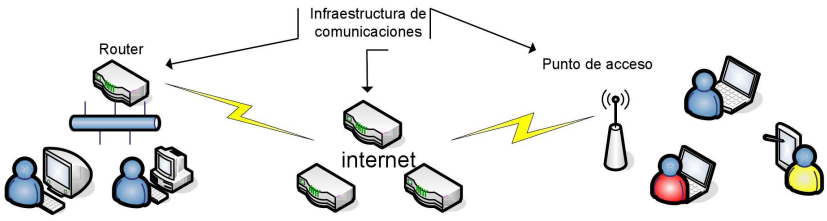

Figura 2.1: Redes con infraestructura. Cableadas e inalámbricas

- Redes *ad-hoc* (figura 2.2): una red *ad-hoc* es una red auto-organizada compuesta únicamente por estaciones independientes (nodos) que se comunican dentro

de una misma área a través del medio inalámbrico. En algunas redes los nodos son móviles y no existe una infraestructura fija de comunicaciones. A este tipo de redes se las conoce como *Mobile Ad-hoc NETwork (MANET)*. Un aspecto común de estas redes es que la topología suele ser multi-salto (*multihop*), es decir, si dos nodos no pueden comunicarse directamente, otros nodos podrían actuar como elementos de retransmisión o de *routers*, reenviando paquetes de otros nodos hacia su destino.

Como se ha comentado, una de sus principales características es que los nodos establecen la red cooperativamente con independencia de la existencia de cualquier infraestructura: estación base fija, elementos de almacenamiento o computacionales que sean comunes, o que tengan una administración centralizada. Por lo tanto, una de las ventajas de las redes sin infraestructura es su potencial para operar en cualquier lugar (*anywhere*) y en cualquier momento (*anytime*). Sin embargo, la falta de administración centralizada, hace que la asignación de direcciones, la gestión del espacio de nombres, la autenticación y la administración de claves, entre otras cosas, sea difícil de solucionar en términos generales. Dentro de este tipo de redes se pueden diferenciar entre redes en las que los nodos permanecen estáticos (*wireless ad-hoc*) y redes en las que los nodos son móviles (*mobile ad-hoc networks*).

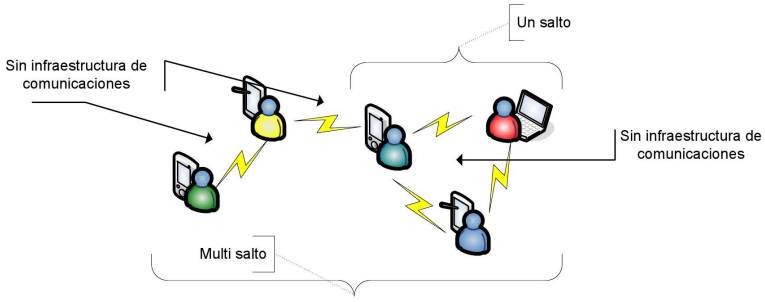

Figura 2.2: Redes sin infraestructura *ad-hoc*: un-salto y multi-salto

- Red híbrida (*Hybrid Network*): Una red híbrida está formada por dos o más subredes. Se pueden diferenciar principalmente dos tipos de subredes, una subred *ad-hoc* y una subred fija. En este caso, la subred *ad-hoc* está conectada a algún punto de la subred fija. Sin embargo, la subred *ad-hoc* puede seguir funcionando sin la subred fija, e incluso podría perder la conectividad con ella.

Para el desarrollo de esta tesis, se considera la posibilidad que un usuario puede estar conectado a una red con infraestructura en determinados periodos de tiempo, pero podría desplazarse y entrar en una red *ad-hoc* que ya está formada, o podría ini-

ciar una red *ad-hoc* con otros dispositivos, los cuales pueden ofrecer servicios, o demandarlos al resto de dispositivos de la red. Por lo tanto, los usuarios podrían moverse a través de diferentes redes en las que los servicios y sus requisitos pueden ir variando en el tiempo.

2.1.1. Servicios de seguridad

En (14) se define seguridad como las medidas tomadas para proteger un sistema. Para poder proporcionar seguridad a un sistema, en primer lugar se debe conocer qué es lo que se quiere permitir o impedir. El concepto de seguridad, como se refleja de la definición anterior, es muy amplio, por lo que se requiere proporcionar soluciones específicas a problemas concretos. En la arquitectura de seguridad presentada por el modelo de referencia de interconexión de sistemas abiertos (OSI - *Open System Interconnection*) (15), se establece que para proteger las comunicaciones de los usuarios en las redes, es necesario proporcionar los siguientes servicios de seguridad:

- Confidencialidad: proporciona protección para evitar que la información sea revelada, accidental o deliberadamente, a una entidad no autorizada. Es decir, los datos sólo podrán obtenerse en claro por las entidades autorizadas que intervienen en la comunicación. Este servicio permite proteger a las entidades contra ataques como el de escuchas pasivas (*wiretapping*), los cuales no pueden evitarse pero sí pueden contrarrestarse.

- Integridad: permite al receptor verificar que la información recibida no ha sido modificada. Los ataques de modificación de mensajes son mucho más fáciles de llevar a cabo en el mundo digital, ya que la modificación de cualquier mensaje se puede realizar cambiando cualquiera de los bits que lo componen, los cuales podrían no ser perceptibles por el receptor.

- Autenticación: debe permitir asegurar que una entidad es quien dice ser. Este servicio intenta evitar un ataque, que puede realizarse con cierta facilidad en sistemas donde no es necesaria la presencia física del usuario, como es el de la suplantación de la identidad (*masquerade*), mediante el cual un usuario remoto puede hacerse pasar por quien no es.

- Control de Acceso: evita la utilización no autorizada de los recursos de la red. De esta forma, permite que sólo quien esté autorizado pueda conectarse a un determinado sistema, y una vez conectado, cada entidad sólo pueda tener acceso a aquellas operaciones para las cuales tiene permiso. Es decir, el servicio permite controlar quién puede hacer qué. En la mayoría de los casos este servicio se implementa junto al servicio de autenticación, es decir, una vez se ha verificado

la identidad del usuario, se debe comprobar qué permisos o restricciones posee sobre el recurso al cual intenta acceder.

- No Repudio: permite evitar que alguna o todas los entidades de una comunicación, nieguen haber participado en ella. Este servicio proporciona la prueba ante una tercera parte que cada una de las entidades comunicantes han participado en una comunicación. Este servicio puede ser de dos tipos: 1) con prueba de origen, cuando el destinatario tiene prueba del origen de los datos, y 2) con prueba de entrega, cuando el origen tiene prueba de la entrega íntegra de los datos al destinatario.

Aún se podría mencionar otro servicio como es el *anonimato*. Se trata de lograr que la entidad que realiza una determinada operación, permanezca oculta frente a alguna de las partes involucradas. Una de estas situaciones puede ser el voto electrónico, en el cual es necesario verificar la identidad de quien realiza el voto, y proporcionar el anonimato de los datos que se van a depositar.

Otro servicio adicional que se puede considerar es el de la disponibilidad (*availability*). La disponibilidad hace referencia a la protección que es necesaria introducir en un sistema para que las distintas partes que lo componen, estén disponibles para ser utilizadas por entidades autorizadas. Por lo tanto, se podría decir que es un conjunto de facilidades y medidas de seguridad que sirve para contrarrestar, en parte, los ataques de denegación del servicio (DoS - *Denial of Service*). Este punto es de hecho uno de los principales aspectos que debe tenerse en cuenta a la hora de ofrecer servicios en entornos ubicuos, ya que la desconexión total o parcial a servidores *on-line* puede provocar que los procesos de autenticación y/o autorización no puedan llevarse a cabo.

2.1.2. Mecanismos de seguridad

Los servicios de seguridad se pueden proporcionar a través de la utilización de diferentes mecanismos de seguridad. Para poder conseguir los servicios de seguridad se utilizan tres técnicas criptográficas principales: funciones de *hash*, criptografía de clave simétrica y criptografía de clave pública.

Función de Hash.- Una función de hash unidireccional (OWHF - *One Way Hash Function*) es una función cuyo parámetro de entrada es una cadena de longitud variable, y cuya salida es una cadena de longitud fija, la cual se conoce como resumen (*digest*). Una función de *hash* (H) tiene las siguientes propiedades:

1. La función de *hash*, H, se puede aplicar a mensajes de cualquier tamaño (M).

2. H genera una salida de longitud fija.
3. Para cualquier M, $H(M)$ es fácil de computar.
4. Para cualquier resumen (m), es computacionalmente inviable encontrar un M tal que $H(M) = m$.
5. Dado cualquier mensaje M, es computacionalmente inviable encontrar otro $M' \neq M$ tal que $H(M') = H(M)$.
6. Encontrar cualquier par de mensajes (M, M') tal que $H(M') = H(M)$, es computacionalmente inviable.

Las funciones de hash más conocidas y utilizadas son MD5 (*Message Digest 5*) (16) y SHA-1 (*Secure Hash Algorithm-1*) (17). La primera genera una salida fija de 128 bits, mientras la segunda genera una salida de 160 bits. Ya existen nuevas versiones de SHA, a las cuales se las conoce como SHA-2 (SHA-224, SHA-256, SHA-384, y SHA-512), y que generan salidas que alcanzan los 512 bits.

Criptografía de clave simétrica.- La criptografía de clave simétrica también se conoce como criptografía de clave secreta, ya que la clave utilizada tanto para cifrar como para descifrar ha de ser conocida únicamente por emisor y receptor, y ha de permanecer secreta al resto de usuarios. Más formalmente, un mensaje M se cifra aplicando el algoritmo simétrico S utilizando la clave K:

$$C = S_K(M) \tag{2.1}$$

El mensaje secreto C se descifra aplicando el algoritmo inverso S^{-1} con la clave K al mensaje secreto C:

$$M = S_K^{-1}(C) \tag{2.2}$$

El algoritmo simétrico más conocido y utilizado hasta hace unos años era DES (*Data Encryption Standard*) (18). DES utiliza una clave de longitud pequeña, la cual no es suficiente para evitar un ataque de fuerza bruta. Para paliar este problema, se empezó a utilizar una versión extendida de DES, 3DES (19), el cual aumenta su fortaleza contra ataques de fuerza bruta. Un algoritmo que surgió posteriormente es AES (*Advanced Encryption Standard*) (20), el cual permite utilizar claves de longitud mayor que DES y 3DES.

En general, los sistemas de clave simétrica son simples y rápidos de computar, sin embargo, su mayor inconveniente es que tanto emisor como receptor deben compartir la clave, por lo que aparece un problema de distribución segura de la clave.

Criptografía de clave asimétrica.- La criptografía de clave asimétrica, también conocida como criptografía de clave pública, utiliza dos claves diferentes, a diferencia de la criptografía de clave simétrica. A pesar de utilizar dos claves diferentes, éstas están relacionadas matemáticamente. Una de estas claves es pública (KU) y la otra es privada (KR, sólo puede ser conocida por el poseedor). En función de la utilización de las claves, se pueden conseguir diferentes servicios de seguridad. A continuación se enumeran las condiciones que debe cumplir un sistema de clave pública:

1. La generación del par de claves pública (KU_A) y privada (KR_A) por una entidad A es computacionalmente sencillo.
2. Conocida la clave pública de la entidad A (KU_A) y el mensaje a ser cifrado M, es computacionalmente sencillo para un emisor B generar el mensaje cifrado C:
$$C = E_{KU_A}(M) \tag{2.3}$$
3. Dado un mensaje cifrado con la clave pública de una entidad A, es computacionalmente sencillo descifrar el mensaje con la clave privada asociada de A:
$$M = D_{KR_A}(C) = D_{KR_A}[E_{KU_A}(M)] \tag{2.4}$$
4. Conociendo la clave pública de una entidad A (KU_A) no es computacionalmente viable obtener la clave privada asociada KR_A.
5. Conociendo la clave pública (KU_A) y un texto cifrado con esa clave (C), no es computacionalmente viable recuperar el mensaje original M.

Estas técnicas se pueden combinar para proporcionar los siguientes servicios de seguridad:

- *Cifrado.*- El cifrado puede hacerse utilizando sistemas criptográficos simétricos o asimétricos, y se puede aplicar extremo a extremo o individualmente a cada enlace del sistema de comunicaciones. Básicamente se utiliza para proporcionar confidencialidad, sin embargo, también puede utilizarse para proporcionar autenticación y protección a la integridad de la información. Utilizando criptografía de clave asimétrica, se pueden conseguir diferentes servicios de seguridad en función de cómo se utilice el par de claves pública/privada:
  - Confidencialidad.- El emisor cifra el mensaje con la clave pública del receptor. De esa manera sólo el receptor puede descifrar el mensaje. Ni siquiera el emisor podría descifrar el mensaje.

- Autenticación.- Un emisor cifra el mensaje con su propia clave privada. Cualquier receptor que conozca la clave pública del emisor, puede descifrar el mensaje. La única entidad que ha sido capaz de cifrar el mensaje es la entidad que posee la clave privada.

- *Función de hash.*- La función de *hash* se utiliza para proporcionar protección a la integridad de la información y puede utilizarse para proporcionar autenticación si se usa conjuntamente con el cifrado de clave pública a través de la firma digital.

- *Firma digital.*- Se puede definir la firma digital como el conjunto de datos que se añaden a una unidad de datos para protegerlos contra la falsificación, permitiendo al receptor verificar la autenticidad (quién los generó) y la integridad (la no modificación) de los datos. Básicamente puede utilizarse para proporcionar no repudio.

2.1.3. Seguridad en entornos ubicuos

La computación y la comunicación ubicua tienen algunas restricciones peculiares en términos de conectividad y de potencia computacional de los dispositivos, que hace que proporcionar seguridad sea significativamente diferente frente a los sistemas distribuidos tradicionales. Es decir, requiere de servicios de seguridad que puedan soportar las características de la red ubicua.

La heterogeneidad de las redes y de los dispositivos podría llevar a adoptar soluciones específicas de seguridad para cada entorno, sin embargo, desde nuestra visión del acceso global a los servicios, esta idea no haría más que dificultar la posibilidad de lograrlo. Nos centraremos en escenarios en los que un usuario puede pasar a formar parte de diferentes tipos de redes con características diferentes a lo largo del tiempo. Por lo tanto, la solución de seguridad que se adopte debe poder funcionar tanto en la red con mayores prestaciones como en redes con ciertas limitaciones.

El aumento de desarrollos de tecnologías como MANET o *Bluetooth* (21) están facilitando el desarrollo de redes móviles *ad-hoc*, las cuales poseen unas determinadas características que hacen que proporcionar seguridad sea más complejo que en las redes cableadas:

- La probabilidad de que algún nodo pueda fallar (*break-in*) en la red en un intervalo de tiempo elevado, puede ser alta, por lo que deben considerarse este tipo de situaciones para que la red pueda seguir estando disponible.

- Las redes *ad-hoc* no proporcionan soporte para infraestructura, por lo que son los propios nodos de la red los que deben gestionar las comunicaciones entre

los diferentes miembros. En este punto aparecen nodos que intentan reducir su consumo de batería a costa de los recursos de otros nodos. A este tipo de nodos se les conoce como *selfish*.

- Los errores propios del canal, la rotura de enlaces, entre otras cosas, hacen que las comunicaciones multi-salto no sean tan fiables como las comunicaciones cableadas.

- La topología de red es dinámica: los nodos que forman la red son móviles, haciendo que la topología de la red varíe a lo largo del tiempo, y en la cual, los nodos que la forman pueden abandonar o incorporarse a la red constantemente.

- Ancho de banda limitado: típicamente el ancho de banda de las redes inalámbricas es menor que el de las redes cableadas convencionales, por lo que podría limitar el tamaño de los mensajes intercambiados.

- Recursos limitados: los nodos móviles a menudo poseen como fuente de energía una batería limitada. Esto puede limitar los posibles mecanismos de seguridad a implantar, por lo que los nodos se enfrentan a una nueva amenaza. Por ejemplo, la obtención de grandes volúmenes de datos, o la ejecución de determinados algoritmos con elevada carga computacional, puede agotar rápidamente la batería de los nodos.

La mayor ventaja de este tipo de redes, es precisamente su peor inconveniente, la falta de infraestructura. Por ello, soluciones que están basadas en sistemas centralizados que requieren de cierta infraestructura, no son a priori la mejor elección para la disponibilidad de los servicios. En resumen, las características más importantes de los sistemas ubicuos que afectan para ofrecer servicios de seguridad, y más concretamente, para ofrecer autenticación y autorización, las podríamos agrupar en tres:

1. **Heterogeneidad**. El acceso ubicuo permite a los usuarios establecer comunicaciones en cualquier momento y en cualquier lugar. Se pueden utilizar diferentes tipos de redes, topologías y tecnologías: Wi-Fi, UMTS, Bluetooth, etc. Esta heterogeneidad implica la existencia de dominios de confianza que pueden encontrarse desconectados, cada uno aplicando sus propios mecanismos y políticas para llevar a cabo la autenticación y/o autorización de los usuarios.

2. **Desconexión temporal de los servicios**. Cuando los usuarios se mueven a través de diferentes redes, la conectividad global se puede perder y algunos servicios pueden estar no disponibles temporalmente. Esta situación tiene un fuerte

efecto en algunos de los servicios que requieren la obtención de información *on-line*, como son la autenticación y la autorización basada en credenciales (certificados digitales).

3. **Movilidad y limitaciones**. La movilidad puede hacer que los servicios disponibles en cada instante de tiempo cambien su ubicación. Esto hace necesario el descubrimiento de los servicios y/o la adaptación de los mecanismos de autenticación/autorización en función de la información y de los recursos de los dispositivos que forman la red.

2.2. Credenciales para Autenticación y Autorización

2.2.1. Introducción

En el apartado anterior se ha explicado la necesidad de ofrecer servicios de seguridad en entornos donde tanto los usuarios como los recursos están distribuidos, y en los que no es posible un control absoluto de los usuarios que acceden. Cualquiera puede acceder; la cuestión es si la entidad es quien dice ser y si esa entidad puede realizar las operaciones solicitadas en ese momento.

La primera cuestión se identifica con el servicio de autenticación. Si un usuario intenta acceder a los recursos, ya bien sean de la propia red o de un determinado sistema, tanto el usuario que accede al servicio como la entidad ofreciendo el servicio pueden querer verificar la identidad de la otra parte. Si ahora el usuario quiere realizar algún tipo de operación sobre el recurso, por ejemplo, imaginemos la situación en la que un usuario con una PDA (*Personal Digital Assistant*) quiere imprimir en una impresora, ésta deberá verificar en base a la identidad, los privilegios u otras características, si el usuario tiene permiso para realizar la impresión en ese instante. El problema es cómo ambas partes pueden verificar o demostrar quiénes son y qué permisos poseen. La comunicación puede ser realizada en redes abiertas en las que la información intercambiada puede ser eliminada, modificada o suplantada, por lo que se deben utilizar soluciones que puedan lograr su objetivo bajo estas condiciones.

En este apartado, se introducen las principales soluciones tecnológicas basadas en certificados digitales para transportar la información tanto de autenticación como de autorización en entornos ubicuos.

2.2.2. Criptografía y certificados digitales

Uno de los principales objetivos de los sistemas de clave pública es dar solución al problema de la distribución de claves. Sin embargo, el problema que se presenta es cómo comunicar la clave pública a los participantes de la comunicación de forma segura. En 1978 Kohnfelder introdujo en su tesis (22) la utilización de una estructura

de datos firmada para transportar la clave pública. Esta estructura de datos firmada se conoce como certificado digital, y es hoy día un método comúnmente utilizado para la publicación/distribución de claves públicas. El certificado digital está firmado por una tercera parte de confianza (TTP - *Trusted Third Party*) por lo que pueden ser publicados en repositorios sin que sea necesario ofrecer una seguridad extra para protegerlos (apartado 2.1.2). Hay que recordar que la firma proporciona integridad y autenticación a los datos firmados.

A pesar que la primera definición de certificado digital pudiera ir ligada al transporte de una clave pública, el concepto de certificado se ha ampliado y se puede considerar cualquier estructura de datos firmada que pretende certificar el enlace entre una determinada información (clave pública, rol, privilegio o atributo en general) y un usuario o entidad. Con lo cual, aunque inicialmente se utilizaron para ofrecer el servicio de autenticación, más tarde se han introducido para dar soporte al servicio de autorización.

Existen diversos tipos de certificados digitales, aunque su finalidad no tiene porqué ser la misma. Este apartado se centrará en aquellos que permiten certificar la identidad de una entidad, o los privilegios asociados a un determinado usuario. Dentro de este conjunto, se expone la estructura de tres grupos de certificados: los definidos por la ITU-T (*Telecommunication Standardization Sector*), los definidos por el SDSI WG (*Simple Distributed Security Infrastructure Working Group*) (23) y las definiciones propietarias realizadas dentro de algunos proyectos (24; 25).

2.2.3. Certificados X.509

En este apartado se consideran los certificados definidos por la ITU-T en su recomendación X.509 (26–28): el certificado de clave pública y el certificado de atributo. El primero de ellos se utiliza para proporcionar el servicio de autenticación, mientras el segundo se utiliza para dar soporte al servicio de autorización.

2.2.3.1. Certificados de clave pública X.509

Un certificado de clave pública (PKC - *Public Key Certificate*), también conocido como de identidad, es una estructura firmada por el emisor del mismo (CA - *Certification Authority*) y que garantiza que una determinada clave pública pertenece a una cierta entidad, la cual queda identificada de forma inequívoca (ver figura 2.3). Es una de las herramientas para poder ofrecer el servicio de autenticación.

A continuación se define la estructura de un certificado X.509 de clave pública, utilizando la notación ASN.1 (*Abstract Syntax Notation One*) (29) :

```
Certificate ::= SEQUENCE    {
   tbsCertificate           TBSCertificate,
```

```
   signatureAlgorithm      AlgorithmIdentifier,
   signature               BIT STRING
}
TBSCertificate ::= SEQUENCE {
   version              [0] Version DEFAULT v1,
   serialNumber         CertificateSerialNumber,
   signature            AlgorithmIdentifier,
   issuer               Name,
   validity             Validity,
   subject              Name,
   subjectPublicKeyInfo SubjectPublicKeyInfo,
   issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
   subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
   extensions      [3]  Extensions OPTIONAL
                        -- If present, version MUST be v3 --  }
```

- `Version`: indica la versión del certificado (actualmente versión 3).
- `SerialNumber`: número de serie asignado por la CA que expidió el certificado. Este valor es único dentro de cada CA.
- `Signature`: contiene el valor del identificador del algoritmo utilizado para firmar el certificado (`AlgorithmIdentifier`).
- `Issuer`: contiene una estructura con información de identificación de la CA que expidió el certificado.
- `Validity`: indica el periodo de validez del certificado. Está formado por dos subcampos:
  - `NotBefore`: especifica la fecha en la cual el certificado empieza a ser válido.
  - `NotAfter`: especifica la fecha a partir de la cual el certificado deja de ser válido.
- `Subject`: contiene una estructura con los datos del propietario de la clave.
- `SubjectPublicKeyInfo`: está formado por dos subcampos
  - `Algorithm`: especifica los algoritmos que pueden ser utilizados con la clave asociada al certificado.
  - `SubjectPublicKey`: contiene la clave pública.

- `IssuerUniqueID`: identificador único del emisor.
- `SubjectUniqueID`: identificador único del propietario de la clave.
- `Extensions`: permite ampliar la información contenida en los campos obligatorios.

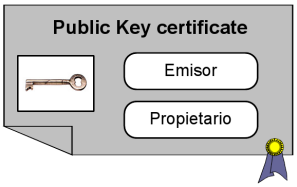

Figura 2.3: Visión general de la estructura de un certificado de clave pública (PKC)

Como podemos observar, lo que permite el contenido de esta estructura, es enlazar una clave pública con un usuario o entidad, dentro de un determinado periodo de validez. También nos permite incluir información adicional a través de las extensiones. Sin embargo, como las extensiones son campos opcionales, cada una puede ser definida de manera propietaria. El campo de extensiones es importante ya que permite ampliar la información contenida en un certificado. Por ejemplo, se puede utilizar una extensión para especificar el uso de la clave privada asociada al certificado (si puede utilizarse para firmar únicamente, para cifrar, etc.).

2.2.3.2. Certificados de atributo X.509

Un certificado de atributo (AC - *Attribute Certificate*) es una estructura de datos, firmada por una autoridad conocida como Autoridad de Atributo (AA - *Attribute Authority*), y que enlaza los valores de unos determinados atributos con la información de identificación de su propietario (ver figura 2.4). A diferencia de los certificados de clave pública, en este caso, es una herramienta para poder ofrecer el servicio de autorización, pero no se está autenticando al portador del mismo, aunque apoyándose en un servicio de autenticación, ayuda a probar quién tiene permiso para hacer qué.

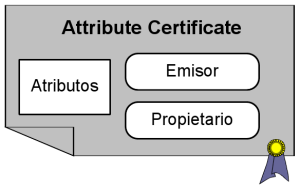

Figura 2.4: Visión general de la estructura de un certificado de atributo (AC)

El formato fue publicado en un primer momento en 1997 por la ITU-T, la cual lo incluyó en la Recomendación X.509 (26), sufriendo ésta una modificación en el 2000 (27). Las modificaciones se centraron básicamente en los siguientes campos:

- Se modificó el campo donde se especificaba el propietario del documento para que apuntara a la identidad del propietario o a un PKC específico.
- Se incluyó el campo de extensiones para poder incluir información adicional sobre el certificado, su propietario o su utilización.

La estructura del certificado de atributo es la siguiente:

```
AttributeCertificate ::= SEQUENCE {
   acinfo               AttributeCertificateInfo,
   signatureAlgorithm   AlgorithmIdentifier,
   signatureValue       BIT STRING
}

AttributeCertificateInfo ::= SEQUENCE {
   version              AttCertVersion -- version is v2,
   holder               Holder,
   issuer               AttCertIssuer,
   signature            AlgorithmIdentifier,
   serialNumber         CertificateSerialNumber,
   attrCertValidityPeriod   AttCertValidityPeriod,
   attributes           SEQUENCE OF Attribute,
   issuerUniqueID       UniqueIdentifier OPTIONAL,\label{ac-format}
   extensions           Extensions OPTIONAL
}
```

de la que se pueden destacar los siguientes campos:

- `Version`: indica la versión del formato del contenido del certificado (actualmente versión 2).
- `Holder`: identifica al titular del certificado de atributo. Puede ser una referencia al certificado de identidad del poseedor, utilizando alguno de los siguentes campos del PKC: `subject`, `Issuer&serialNumber`, un *hash* de la clave pública, etc.
- `Issuer`: identifica la autoridad (AA) que emite el certificado.
- `Signature`: identificador del algoritmo de firma digital utilizado por la AA para firmar el certificado.

- `SerialNumber`: número único que identifica este certificado entre otros emitidos por una misma AA.
- `attrCertValidityPeriod`: fecha y hora tanto de inicio como de finalización de la validez de la información contenida en el certificado. Marca el periodo durante el que el certificado es considerado válido.
- `Attributes`: contiene los atributos o privilegios asociados al usuario que está siendo certificado.
- `IssuerUniqueID`: campo opcional utilizado para evitar la ambigüedad del nombre de la autoridad emisora, en el caso de que pueda ocurrir.
- `Extensions`: permite añadir nuevos campos al certificado de atributo.

Los campos con mayor relevancia, si los comparamos con los del certificado de identidad, son: `Holder` y `Attributes` (ver figura 2.5).

El campo `Holder` permite identificar al propietario del certificado de diversas maneras. Para ello puede contener una referencia al certificado de identidad del propietario. Esta referencia debería identificar unívocamente al certificado enlazado, por lo que la mejor alternativa sería incluir la estructura que contuviera los campos del certificado de identidad `Issuer&serialNumber`.

El campo `Attributes` permite contener diferentes estructuras en función de los tipos de atributo que transporte. Alguno de los tipos que pudiera contener serían: `role`, `group`, `proxy`, etc. También se pueden definir nuevas estructuras en función de las necesidades del sistema.

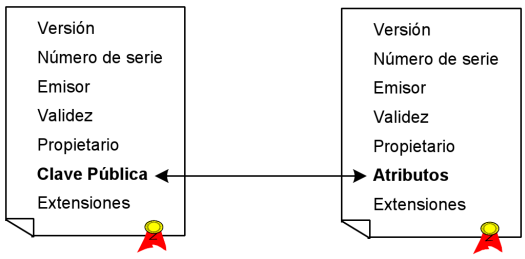

Figura 2.5: Comparación de la estructura general de los certificados de clave pública y de atributo

2.2.4. Certificados SDSI/SPKI

Los certificados vistos en el apartado 2.2.3 son estudiados por el PKIX WG (PKIX *Working Group*), el cual basa su trabajo en la recomendación X.509 de la ITU-T, mien-

tras que los certificados SPKI (*Simple Public Key Infrastructure*) (30; 31) han sido desarrollados por el SPKI WG (32). La principal diferencia entre el PKIX WG y el SPKI WG está en el hecho que el primero asume la existencia de un espacio de nombres global, mientras que el SPKI WG no realiza esta suposición, y en su lugar realiza los enlaces a un espacio de nombres local, como los propuestos en SDSI (*Simple Distributed Security Infrastructure*) por Ron Rivest y Butler Lampson (33). La unión de los esfuerzos del SPKI WG y del SDSI ha dado como resultado los certificados que son conocidos como certificados SDSI/SPKI.

El certificado SDSI/SPKI es un documento firmado que otorga poderes a su poseedor. Éste contiene información de, como mínimo, el propietario y la entidad emisora. Puede contener información del periodo de validez, condiciones de utilización e información de autorización y delegación de atributos. Básicamente, los certificados SDSI/SPKI se dividen en tres categorías: identificación (permite enlazar un nombre con una clave), atributo (enlaza una autorización con un nombre) y autorización (enlaza la autorización con la clave).

Los elementos que podría incorporar un certificado de este tipo son:

- `Issuer`: una clave pública (o su *hash*). Identifica a la entidad emisora del certificado que concede los permisos.
- `Subject`: una clave pública (o su *hash*). Identifica al poseedor del certificado.
- `Delegation`: es un valor booleano. Si es verdadero, el poseedor del certificado puede propagar o delegar la autorización que posee.
- `Authorization`: una estructura llamada *S-expression* que especifica el permiso que se le asigna.
- `Validity dates`: periodo de validez, incluyendo la fecha y la hora.

Hay dos tipos de certificados, que serían los equivalentes al de identidad y de atributo X.509, y cuya estructura se puede definir como sigue:

- identidad (*cert (issuer (name k n)) (subject p) (valid)*)
- autorización (*cert (issuer k) (subject p) (propagate) A (valid)*)

2.2.5. Otros tipos de certificados

Además de las estructuras de certificados vistos, podemos encontrar otros formatos, tanto para los certificados de identidad como para contener los privilegios de un usuario. En este apartado se presentan dos de estos tipos: 1) el PAC, el cual puede considerarse el precursor del AC presentado en 2.2.3.2, aunque puede utilizarse tanto para

autenticación como para autorización, y 2) los definidos para el proyecto Akenti, los cuales son una solución propietaria definida dentro de ese proyecto.

PAC.- El proyecto SESAME (24) iniciado en 1991, tenía como objetivo el desarrollo de una infraestructura segura para entornos distribuidos (34). Como base para asignar privilegios a los usuarios se definió el PAC (*Privilege Attribute Certificate*), un certificado de atributo cuya estructura está basada en la definida por el ECMA (*European Computer Manufacturer's Association*) en (35). Cada PAC tiene un contenido común para todos, y un contenido específico, además de la información asociada a la firma del certificado. El contenido específico contiene información relacionada con: la identidad del emisor, un número de serie, etc. En el contenido específico del certificado es donde se almacenan los privilegios e información referente a la delegación. Una descripción más detallada de los elementos que conforman el PAC se puede consultar en (36).

Se pueden diferenciar dos tipos de PAC:

- No delegables, los cuales enlazan una identidad y contienen información de seguridad sobre una sesión del usuario (*login session*).
- Delegables, los cuales se pueden utilizar para delegar algunos derechos de acceso de un usuario a un servidor de forma temporal para que éste actúe en nombre del usuario. Hay que señalar que no se delega la identidad, sino los derechos de un usuario.

Ambos tipos de certificados se expiden por un periodo de tiempo corto (alrededor de una horas), para limitar el tiempo en el cual una clave o un privilegio puede ser utilizado de forma fraudulenta.

El contenido del PAC tiene una gran similitud con el contenido del AC, con la salvedad de las extensiones, campo importante para ampliar las funcionalidades del certificado.

Certificado de Atributo en Akenti.- Dentro del proyecto Akenti (37), se definen dos tipos de certificados que se describen a continuación: el certificado de atributo y el de condiciones de uso.

Certificados de Atributo. En el proyecto Akenti se utilizan unos certificados de atributo propietarios, los cuales permiten certificar que un usuario, especificado por su DN (*Distinguished Name*) posee un determinado valor para un atributo dado. Al igual que el resto de certificados, se firma y almacena en un directorio accesible. Los principales campos del certificado de atributo son: nombre

del atributo, valor asociado, periodo de validez, usuario y su autoridad de certificación, expedidor y su autoridad de certificación, y firma del certificado de atributo expedido.

Condiciones de Uso. Un certificado de Condición de Uso es un documento firmado que especifica los atributos necesarios para poder operar sobre un recurso determinado. Todas las condiciones de uso definen el grupo de entidades a las cuales se les permite el acceso a un recurso. Por lo tanto, cada condición de uso es una pieza de una lista de control de acceso. Este tipo de certificados los crean y firman los poseedores de los recursos, almacenándolos en un directorio al cual pueda tener acceso el servidor de Akenti. La información que contienen se puede resumir en los siguientes puntos:

- Nombre del recurso: podría designar a un objeto o a todos los objetos por debajo de uno principal (dependencia jerárquica).
- Alcance: especifica si la condición de uso se debe aplicar únicamente al recurso al que se hace referencia o a todos los recursos por debajo de uno principal.
- Período de validez de la información que contiene.
- Valor booleano: si es verdadero, el usuario ha de cumplir la condición de uso o no podrá tener acceso al recurso.
- Condición de uso, compuesta por:
  - Una expresión booleana cuyo valor es una combinación de los atributos y valores que un usuario debe tener.
  - Una cadena de información de atributos para cada atributo-valor, la cual incluye las autoridades emisoras de los atributos, y opcionalmente, una lista con las localizaciones donde buscar los certificados de atributo.
- Lista de las acciones permitidas sobre el recurso.
- Firma del expedidor del certificado.

2.2.6. Ventajas e Inconvenientes

La interoperabilidad, adaptabilidad, flexibilidad y escalabilidad, entre otras propiedades, son deseables en entornos de ámbito distribuido, por lo que se deben tener en cuenta en el diseño de este tipo de sistemas. Por este motivo, es conveniente utilizar estructuras estandarizadas, y evitar soluciones propietarias en la medida de lo posible (ver apartado 2.2.5).

Se ha podido comprobar que existen diferentes estructuras para representar tanto la identidad como la información relativa a los privilegios de un usuario, sin embargo, como ya se ha mencionado, las soluciones propietarias no son la mejor alternativa. Si se tuviera que elegir entre las estructuras definidas por la ITU-T o el SPKI WG, es aconsejable utilizar la estructura definida por la ITU-T, ya que son las más utilizadas y soportadas por los sistemas basados en credenciales (certificados digitales). Este hecho permite que soluciones basadas en X.509 puedan ser integradas con sistemas ya implantados, de una manera más rápida y sencilla.

Ahora bien, si los privilegios están tan asociados a una identidad, ¿por qué no utilizar la misma estructura para certificar tanto la identidad como los atributos de un usuario? A pesar de que el certificado de identidad permite transportar atributos, en general, no es suficiente o al menos no recomendable que se utilicen para representar derechos o privilegios, ya que:

- A veces es deseable el anonimato del usuario que desea acceder al recurso.
- El periodo de validez de un privilegio o atributo puede ser bastante inferior (minutos, horas, semanas) si lo comparamos con el periodo de validez de un certificado de clave pública (meses, un año, etc.). Si el atributo deja de ser válido, el certificado que lo contiene también deja de ser válido. Este hecho implicaría la revocación del certificado y posterior emisión de uno nuevo, con el consiguiente aumento en la carga de gestión de las credenciales (ver apartados 2.3 y 2.4).
- La autoridad de certificación tiene la función de ser fuente de autoridad para identidades pero no tiene porqué ser la fuente de autoridad para privilegios, es decir, si se piensa en una autoridad como la administración que puede dar fe de la identidad de un usuario, no tiene porqué tener la autoridad de conceder privilegios de utilización de los recursos de una determinada entidad.
- La autorización posee ciertas características como pueda ser la delegación (traspasar ciertos privilegios o un subconjunto de éstos a otra persona durante un periodo de tiempo), que no proporcionan los certificados de identidad.

La utilización del certificado de clave pública para la representación tanto de la información de autenticación como de autorización, puede ser conveniente si:

- La misma entidad es responsable de ofrecer los dos servicios.
- La duración de los privilegios y de la clave pública del usuario son la misma, permitiendo que fuera necesario la expedición, y por tanto, la gestión de un único certificado.

La utilización únicamente del PKC podría ser, en los casos antes mencionados, beneficiosa en cuanto a la disminución de la posible carga en el sistema, ya que únicamente se debería verificar una única cadena de certificados. Sin embargo, para solucionar los inconvenientes comentados, se pueden utilizar los certificados de atributo. Pero si únicamente se utilizaran los certificados de atributo, al no tener éste ninguna clave asociada, no se podría verificar si la utilización del mismo se realiza por la entidad a la cual se concedió el certificado, por lo que se deberían utilizar otros mecanismos para detectar el uso ilegítimo.

Para un sistema genérico, la mejor alternativa sería la utilización de forma diferenciada de un sistema de gestión de certificados de identidad y de otro para la gestión de privilegios, aunque utilizados de forma conjunta para permitir autenticar y autorizar a una entidad.

2.3. Infraestructuras de Autenticación/Autorización

En el apartado 2.2 se han presentado los diferentes tipos de credenciales que permiten contener información necesaria para llevar a cabo los procesos de autenticación y/o autorización. Sin embargo, las credenciales requieren de procedimientos que permitan su utilización de forma segura. Es decir, la utilización de credenciales requiere de protocolos, entidades y políticas para una gestión segura de su ciclo de vida: generación, utilización y eliminación. Al conjunto de procedimientos que permiten gestionar el ciclo de vida de las credenciales, se le conoce como infraestructura de gestión de credenciales.

Como conclusión del apartado 2.2 se ha señalado que el formato que mejor se adapta a entornos distribuidos en los que el usuario puede moverse entre diferentes dominios, es el de estructuras estandarizadas como las definidas por la ITU-T, sobre las que trabaja el IETF (38; 39) para ofrecer una guía para su implementación y utilización. Como la tesis se centrará en este tipo de credenciales, en este apartado sólo especificaremos las infraestructuras necesarias para la gestión de credenciales X.509 PKC (para autenticación) y X.509 AC (para autorización).

2.3.1. Infraestructura de Autenticación: PKI X.509

Se puede definir infraestructura de clave pública (PKI - *Public Key Infrastructure*) como: el conjunto de *hardware*, *software*, personas, políticas y procedimientos necesarios para crear, gestionar, almacenar, distribuir y revocar certificados de clave pública (PKCs).

Una PKI está formada por cinco componentes principales (ver figura 2.6):

- Autoridad de Certificación (CA - *Certification Authority*): es una TTP encargada

de expedir, renovar, publicar y revocar certificados de clave pública. También podría soportar una variedad de funciones administrativas, aunque suelen ser delegadas a una o más Autoridades de Registro.

- Autoridad de registro (RA - *Registration Authority*): es un elemento opcional que puede asumir un número de funciones delegadas por la CA. A menudo, la RA se encarga del proceso de registro de la entidad final.
- Entidad final (EE - *End Entity*): usuario de los servicios de la PKI y/o entidad que posee un certificado de clave pública emitido por una CA.
- Directorio o repositorio: es un término general utilizado para denotar a cualquier método para almacenar certificados y/o información sobre el estado de revocación de los certificados (ver apartado 2.3.3), de forma que puedan ser recuperados por las diferentes entidades.
- Emisor de la CRL: sistema opcional al cual una CA delega la publicación de información sobre el estado de revocación de los certificados.

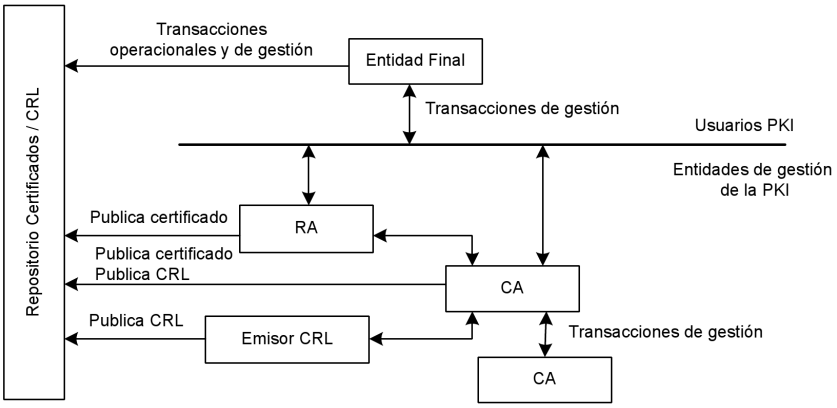

Figura 2.6: Modelo de Referencia. Entidades de la PKI

El certificado de clave pública tiene un periodo de validez limitado, durante el cual, la información contenida en él se considera válida. Sin embargo, es posible que durante este periodo alguna de la información contenida deje de ser válida: la clave privada del usuario se ha comprometido, la clave privada de la CA se ha comprometido, etc. Por lo tanto, una entidad debería poder conocer que el certificado ha dejado de ser válido antes de su fecha de expiración. Tanto el poseedor del certificado como otra entidad autorizada pueden solicitar la revocación de un certificado. Una vez validada la solicitud, la CA puede publicar de forma periódica una lista firmada, conteniendo el número de serie de los certificados revocados (ver apartado 2.3.3). De esta

forma, cualquier entidad que desee verificar la validez del certificado presentado por el usuario, puede conocer si ha sido revocado. Este es uno de los procesos más costosos y de mayor importancia dentro de la validación de certificados como veremos en el apartado 2.4.

El modelo de PKI presentado está pensado para entornos donde el acceso a determinados servidores es posible, es decir, las entidades pueden acceder (hay conectividad) a la RA, la CA y los repositorios. Sin embargo, en entornos ubicuos la posibilidad de desconexión a sistemas *on-line* podría provocar que los servicios no pudieran llevarse a cabo. Este hecho podría llevar a la denegación del servicio por falta de disponibilidad de información para realizar el proceso de autenticación. En el siguiente apartado se exponen los problemas que surgen cuando se intenta ofrecer los servicios de PKI en redes MANET.

2.3.1.1. Problemática de la PKI en redes ad-hoc

Como hemos visto en el apartado 2.1 uno de los mayores inconvenientes que puede encontrarse un usuario en entornos ubicuos, es estar en redes en las que pueda faltar cierta infraestructura. Esta posibilidad hace que las soluciones actuales de PKI no puedan ser trasladadas directamente a este tipo de redes. Por lo tanto, se hace necesario o bien la modificación/adaptación de dichas soluciones para que tengan en cuenta estas nuevas características, o bien la adopción de nuevas soluciones específicas para estos entornos.

Uno de los principales problemas en este tipo de redes es la autenticación de los nodos. Las soluciones habituales que utilizan terceras partes de confianza (TTPs) requieren de la existencia de una infraestructura organizativa o administrativa centralizada, que realice las funciones necesarias de autenticación. En redes híbridas o *ad-hoc* estas soluciones tienen ciertos inconvenientes:

- Los servidores que realizan la función de CA son considerados como un único punto de fallo o exposición a ataques. Simplemente alterando el canal inalámbrico próximo al servidor de autenticación, se puede lograr que el servicio no esté disponible.

- La elevada movilidad de los nodos (dispositivos móviles) hace que las rutas cambien frecuentemente, por lo que localizar y contactar con la CA no es una tarea trivial.

- Las comunicaciones multi-salto sobre canales inalámbricos (los cuales son propensos a errores) pueden provocar tasas de pérdida elevadas en la transmisión de datos, lo que eleva la probabilidad de que el servicio no se encuentre disponible.

La replicación de ciertas funcionalidades para aumentar la disponibilidad, no son una buena solución cuando esas funcionalidades implican introducir nuevas entidades de confianza. Es decir, el aumento de TTPs, aumenta la probabilidad de comprometer el servicio completo. Por todos estos motivos, es necesario definir nuevas soluciones que tengan en cuenta las características de estos nuevos entornos. Uno de los principales inconvenientes es el de la disponibilidad del servicio, que puede verse disminuida, tanto por la naturaleza del entorno (falta de rutas disponibles para acceder al servidor CA, fallos en los enlaces, modificación de la localización física de los servidores, etc), como por la aparición de nuevos tipos de ataques. A continuación se analizan las ventajas e inconvenientes de tres de estas soluciones y la dirección tomada por algunas nuevas propuestas.

2.3.1.2. Autoridad de Certificación en redes ad-hoc

El servicio de autenticación en redes *ad-hoc* no es directa, ya que las soluciones actuales, como X.509, PKIX o PGP (*Pretty Good Privacy*) (40), no se adaptan de forma adecuada. En la literatura existen dos tipos de enfoques para dar solución a estos inconvenientes. Unas soluciones se basan en compartir las funcionalidades de la CA (41; 42) entre todos los nodos de la red (CA completamente distribuida), o entre una serie de nodos seleccionados (43; 44) (CA parcialmente distribuida), las cuales utilizan criptografía umbral (45) para poder compartir la clave privada. Otra solución, presentada en (46; 47) se basa en la idea de PGP, es decir, los propios usuarios son los encargados de expedir los certificados al resto de nodos, creando rutas de confianza.

En el caso de la distribución de las funcionalidades de la CA, existen dos propuestas, compartir la clave privada de la CA entre todos los nodos de la red, o limitar el número de nodos que van a compartir la clave a aquellos que han sido previamente seleccionados. Cada una de las alternativas presenta sus ventajas e inconvenientes, como pueden ser:

- La disponibilidad del servicio viene determinada por la existencia de k nodos a un salto (no siempre posible), o la posibilidad de alcanzar a k nodos y recibir un número de respuestas suficiente para reconstruir el valor completo (los métodos utilizados pueden provocar un elevado *overhead* en la red).

- El problema de la revocación de certificados no está solucionado en todas las propuestas. En (43; 44) no se especifica un mecanismo de revocación que permita eliminar por completo la validez del enlace de una entidad con su clave pública, ya que el mecanismo sólo renueva el certificado, invalidando el certificado anterior. Esta renovación únicamente se lleva a cabo por el poseedor del certificado (ya que no se hace alusión a ninguna política que lo contemple).

- La utilización de criptografía umbral requiere de un sistema síncrono, lo cual no siempre puede garantizarse por la propia naturaleza de las redes *ad-hoc*.

- La necesidad de una entidad que inicialice la red y expida el primer certificado a cada uno de los nodos, no siempre será posible, y de hecho, puede constituir un problema de seguridad para la red completa.

- No se puede asumir que siempre exista una cooperación desinteresada por parte de los nodos que forman la red. Tal vez sí se adapte bien en entornos cerrados, como los militares o de emergencias, sin embargo, no son válidos para entornos comerciales.

Algunos de los inconvenientes anteriores los soluciona la propuesta basada en crear caminos de confianza, aunque su mayor problema es que sólo puede lograrse la autenticación entre nodos con una determinada probabilidad, debido a la posibilidad de no encontrar un camino de certificación entre dos nodos cualesquiera de la red.

Estas propuestas son soluciones concretas para entornos limitados. Por lo tanto, sólo son válidas dentro de esa red, y los mecanismos no son extrapolables a un entorno global.

2.3.2. Infraestructura de Autorización: PMI X.509

Se puede definir infraestructura de gestión de privilegios (PMI - *Privilege Management Infrastructure*) como el conjunto de *hardware*, *software*, personas, políticas y procedimientos necesarios para crear, gestionar, almacenar, distribuir y revocar certificados de atributo (39).

Una PMI está formada por cinco tipos de componentes (ver figura 2.7):

- Autoridad de Atributo (AA - *Attribute Authority*): se encarga de expedir y revocar certificados de atributo, es decir, es responsable de asignar o delegar privilegios a los usuarios finales o a otras AA. Una autoridad puede tener restringida la capacidad de delegar un privilegio a otra autoridad, y únicamente poder delegar privilegios a usuarios finales.

- Certificado de Atributo (AC - *Attribute Certificate*): contiene la información sobre los privilegios o roles que posee una determinada entidad.

- Directorio: lugar donde almacenar y del que poder obtener los certificados y listas de certificados de atributo revocados (ACRL - *Attribute Certificate Revocation List*). La revocación en PMI se explica con más detalle en el apartado 2.3.3.

- Clientes: entidades que solicitan un determinado servicio de la PMI (solicitud de un certificado, acceso a sus directorios, etc.).

- Verificador de privilegios: es la entidad encargada de verificar el AC de un usuario.

Una vez el cliente ha obtenido el certificado expedido por una AA, y se conecta al verificador de privilegios para poder tener acceso a los recursos a los que está autorizado, existen dos métodos para que el verificador de privilegios pueda obtener el certificado de atributo del usuario (ver figura 2.7):

- Modelo *Pull*: cuando un cliente realiza una petición de acceso, el servidor debe acceder a un directorio para obtener el certificado correspondiente del cliente.

- Modelo *Push*: el cliente presenta el certificado de atributo junto con la petición de acceso.

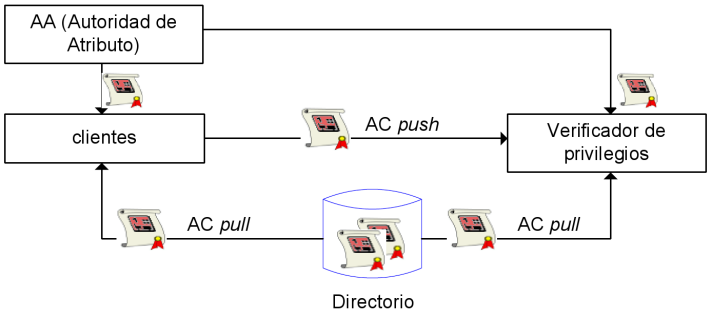

Figura 2.7: Modelo de intercambio de ACs

Dependiendo de la funcionalidad que se quiera ofrecer, se pueden diferenciar cuatro modelos dentro de la PMI: general, control, cometidos y delegación.

Modelo General.- El modelo general está formado por los componentes básicos que permiten autorizar a entidades para realizar ciertas operaciones sobre los recursos controlados por otra entidad. Estos componentes son:

- Recurso: es el objeto que se intenta proteger y sobre el que se pueden realizar diferentes operaciones, en función del tipo de privilegio que tenga el usuario sobre el recurso.

- Poseedor del privilegio: es la entidad a la cual se le ha asignado un determinado privilegio.

- Verificador del privilegio: es la entidad que determina si los privilegios asignados al poseedor son suficientes para realizar una determinada operación sobre el recurso.

La decisión para permitir el acceso a un recurso se basa en:

- Los privilegios del usuario: reflejan la confianza depositada por parte de la entidad emisora. Estos privilegios son presentados por el usuario en forma de certificado, ya sea de forma directa (modelo *push*), o bien indirectamente a través de un directorio (modelo *pull*).
- La política de privilegios establecida: especifica el grado de privilegio necesario para que se pueda realizar una determinada operación sobre un recurso.
- Sensibilidad de la operación sobre el recurso: refleja los atributos de un recurso al cual se quiere acceder.
- Las variables de entorno: son necesarias para determinar si se puede llevar a cabo un determinado acceso, pero cuyos valores únicamente se encuentran de forma local en el verificador, como pudiera ser la hora a la cual se realiza el acceso.

Modelo de Control.- El modelo de control es bastante similar al modelo general. Este modelo muestra cómo se realiza el control sobre el acceso al recurso, por parte del poseedor de privilegios, en base a la política establecida. Como podemos ver en la figura 2.8, el modelo consta de cinco componentes: el poseedor del privilegio, el verificador de privilegios, el objeto o recurso, la política de privilegios y las variables de entorno.

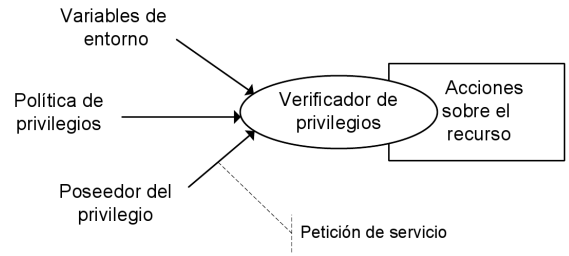

Figura 2.8: Componentes del modelo de control

Cuando el usuario o poseedor del privilegio desea acceder al recurso, realiza una petición de servicio que es procesada por el verificador de privilegios, para comprobar si el usuario puede acceder y qué operaciones puede realizar. Para ello,

analiza y compara el certificado de atributo con la política de privilegios establecida por el sistema, y en función del resultado y de las variables de entorno locales del servidor (por ejemplo, la hora en la que se ha realizado la petición, etc.), toma una decisión.

Modelo de Cometidos.- El rol es un método indirecto de asignar privilegios a los usuarios (ver figura 2.9). A cada rol se le asignan una serie de privilegios, y a los usuarios se les puede asignar uno o más roles (48). De esta forma la modificación o actualización de los privilegios asignados a un rol, no afecta (al menos en principio) a los certificados ya expedidos.

Para poder enlazar a los usuarios con el rol, y a éste con los privilegios necesarios, se utilizan dos tipos de certificados: el de asignación de roles (usuario-rol), y el de especificación del rol (privilegios-rol). En el certificado de asignación de roles, se incluye en el campo del poseedor una referencia al PKC del usuario, y en el campo de atributos, la estructura que representa el rol. Por otra parte, en el certificado de especificación de roles, el campo del poseedor incluye el nombre del rol, y el campo de atributos puede incluir los privilegios asociados al rol.

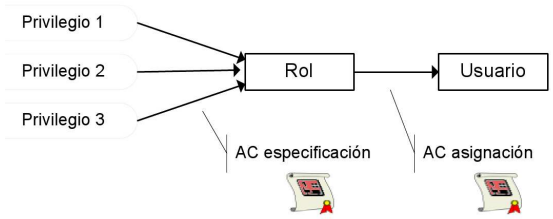

Figura 2.9: Enlace usuario-rol-privilegios

En el caso de que se utilicen certificados para ambos propósitos (un AC para la especificación de roles y otro para la asignación), el verificador de un certificado de atributo debería verificar dos cadenas de certificados. Esta validación puede ser compleja ya que ambos trayectos pueden ser generados por entidades independientes (ver apartado 2.4).

Modelo de Delegación.- En algunos entornos existe la necesidad de delegar o traspasar privilegios de una entidad a otra. En estos entornos se pueden diferenciar cuatro componentes (representados en la figura 2.10): fuente de autoridad, autoridad de atributo, entidad final o poseedor del privilegio, y verificador de privilegios.

La fuente de autoridad (SOA - *Source of Authority*) es el expedidor inicial de privilegios a las entidades, las cuales pueden ser entidades finales (poseedor del

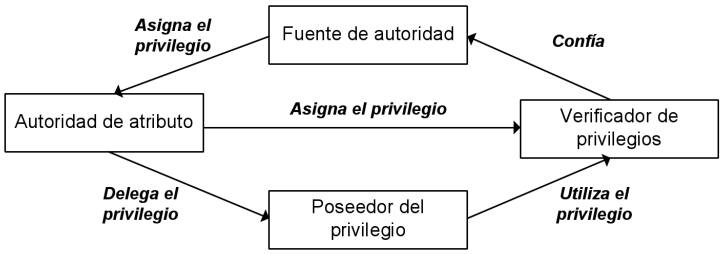

Figura 2.10: Componentes del modelo de delegación

privilegio) o autoridades de atributo (AA - *Attribute Authority*). Como una AA puede delegar privilegios, la SOA puede establecer una longitud máxima de delegación y una restricción sobre la capacidad de delegación de determinadas entidades.

Existe una extensión del certificado que permite indicar si es posible la delegación subsiguiente de los privilegios contenidos en el certificado de atributo que lo contiene. Por lo que permite especificar si la entidad que contiene la extensión puede actuar como una AA sobre los atributos contenidos, al mismo tiempo que se puede limitar la longitud del camino de delegación. Es decir, permite indicar el número de AAs intermedias que pueden existir hasta una entidad final.

En este esquema, cuando el poseedor de privilegios desea acceder a los recursos, presenta el certificado al verificador de privilegios, el cual deberá validar la cadena de certificados, desde la entidad que presenta la petición de acceso al recurso, hasta la SOA (ver apartado 2.4).

2.3.2.1. Servicios de la PMI

El ETSI (*European Telecommunications Standards Institute*) identifica cinco servicios relacionados con la gestión de certificados de atributo (49): registro de atributos, obtención, emisión, distribución y gestión de revocación de ACs. Sin embargo, para el desarrollo de la tesis agruparemos los diferentes servicios en los siguientes tres procesos:

- Expedición y almacenamiento de certificados: la publicación de certificados utilizando directorios con acceso público son adecuados cuando la memoria del dispositivo es limitada. Para acceder a los certificados almacenados, se puede utilizar LDAP (*Lightweight Directory Access Protocol*) (50).
- Revocación de certificados: el proceso de revocación de un AC es igual que el de un PKC. Es decir, el certificado deja de ser válido si alguno de los datos conteni-

dos en él deja de ser válido, o se ha realizado un uso fraudulento del mismo. La verificación del estado de revocación en cadenas de certificados es una característica importante para la escalabilidad del sistema. Por este motivo, se explica con más detalle en el apartado 2.3.3.

- Delegación de privilegios: la delegación es un mecanismo que afecta al coste que supone la validación de certificados. A continuación se explica con más detalle este servicio.

Proceso de delegación.- La complejidad del sistema aumenta cuando se utiliza el modelo de delegación. De hecho, el IETF (51) no recomienda, al menos por el momento, su utilización. Sin embargo, este modelo proporciona una gran flexibilidad al funcionamiento del sistema y es una característica bastante interesante en ciertos escenarios. Por ejemplo, supongamos el caso en el que un administrador del sistema necesita irse de vacaciones, éste podría delegar sus funciones a otro compañero que inicialmente no tuviera los mismos privilegios para administrar el sistema.

La delegación de privilegios puede llevarse a cabo sin la intervención de la SOA u otra AA intermedia. Este escenario permite reducir la carga en la SOA a costa de incrementarlo en el verificador de privilegios. Además, la validación del camino de delegación es más compleja que la validación del camino de certificación. De hecho, la validación de cada AC del camino de delegación puede requerir la validación de un camino de certificación asociado. El proceso de validación de certificados se explica con más detalle en el apartado 2.4.

2.3.3. Gestión de revocación de credenciales

La revocación de certificados se puede definir como "los mecanismos bajo los cuales una autoridad puede invalidar el enlace entre una identidad y una clave pública antes de la expiración del certificado que mantiene dicha relación". La revocación de credenciales X.509, tanto para PKI como para PMI, utiliza procedimientos parecidos, por lo que se ha preferido unificar su explicación en este apartado.

En general, un certificado puede dejar de ser válido antes de que expire el tiempo de validez para el cual fue expedido. Esto puede deberse a varios motivos: la clave privada del expedidor del certificado se ha visto comprometida, los datos que contiene deben ser modificados (clave pública, atributos), etc. Este hecho obliga que a la hora de verificar un certificado se deba comprobar su estado de revocación. Como los certificados de atributo son estructuras similares a los certificados de identidad, y dado que podrían ser identificados de forma similar, los procedimientos existentes para permitir verificar el estado de revocación de un certificado de clave pública, son

aplicables a los certificados de atributo. Por lo tanto, esta verificación puede realizarse mediante alguno de estos procedimientos: CRL (*Certificate Revocation List*), OCSP (*Online Certificate Status Protocol*) y MHT (*Merkle Hash Tree*).

Listas de Revocación de Certificados (CRL).- La aproximación más simple es la de crear una lista "negra"también llamada lista de revocación de certificados (CRL) (38; 52), que se distribuye usando entidades no confiables llamadas repositorios. La CRL contiene los identificadores de los certificados revocados. La integridad y autenticidad de la CRL la proporciona una firma digital del emisor de confianza de la CRL. Hay dos versiones estándar de CRLs X.509. X.509v1 es inherentemente defectuosa debido a limitaciones específicamente relacionadas con la imposibilidad de extender la CRL con información adicional a la ya especificada en la estructura. Además, las CRLs X.509v1 pueden ser objeto de ataques de sustitución, es decir, es posible sustituir maliciosamente una CRL por otra sin que pueda detectarse. La versión 2 de X.509 introduce la idea de las extensiones, las cuales permiten ampliar la información contenida en la CRL, tanto en su estructura general, como para cada entrada asociada a un certificado revocado.

La CRL fue inicialmente definida para certificados de clave pública, aunque una estructura similar se ha definido para contener los certificados de atributo revocados. Esta estructura se conoce como ACRL (*Attribute Certificate Revocation List*) (28).

Protocolo de Estado de Certificado On-line (OCSP).- El otro esquema estandarizado para la validación del estado de un certificado es OCSP, el cual es una propuesta del PKIX WG (53), en el que la distribución de los datos de estado de los certificados se hace utilizando autoridades intermedias de confianza llamadas *responder*. OCSP ofrece la posibilidad de conocer el estado de un certificado en particular sin tener que obtener la CRL completa.

En OCSP, el estado de un certificado está disponible mediante un mecanismo de petición/respuesta. Un cliente OCSP envía a un *responder* OCSP una petición de estado para un certificado en particular. Cuando el *responder* recibe la petición, verifica que es correcta, busca la información de estado en su base de datos local, crea la respuesta con los datos correspondientes, firma la respuesta y la envía de vuelta al cliente. El *responder* envía una respuesta indicando alguno de los siguientes resultados: "válido"(*good*), revocado"(*revoked*) o "desconocido"(*unknown*). Si el responder no puede procesar la petición, puede devolver un código de error. Hay que tener en cuenta que como el *responder* es una autoridad de confianza, el cliente ha de poder verificar su identidad. Finalmente, la distribución de la información de revocación entre el emisor de la informa-

ción de revocación y el *responder* tiene que ser realizada de manera segura. Esta distribución se realiza normalmente mediante una CRL.

Sistemas basados en árboles de Merkle.- Existen varios sistemas de revocación que se basan en árboles de Merkle (MHT - *Merkle Hash Tree*) (54), los cuales están basados a su vez, en las funciones de *hash* (apartado 2.1.2). MHT intenta sacar provecho del hecho que una función de *hash* es mucho más rápida de computar que una firma digital. En las hojas del árbol se almacena la información de revocación. En el nodo del árbol del nivel superior a una hoja dada se almacena el valor de hash resultante de la concatenación del valor contenido en las hojas. Esta operación se realiza de forma recursiva hasta llegar al nivel más alto del árbol el cual está formado por un único nodo, al que se conoce como *root*. Una TTP firma el valor del *root* para poder garantizar la integridad y autenticidad del árbol completo.

En el capítulo 6 se presenta un ejemplo completo de la utilización de un MHT.

2.3.3.1. Revocación en PKI

En PKI X.509, el proceso de revocación empieza con una petición para revocar un determinado certificado. En general, el dueño del certificado afectado o el emisor del certificado (CA) pueden crear peticiones de revocación. La petición de revocación se envía al gestor de revocaciones que es normalmente también la CA. En este caso, la CA es también responsable de emitir y distribuir de forma periódica, los datos de estado de revocación de los certificados que tiene bajo su control. Como se presentó en el apartado anterior, los dos mecanismos estandarizados para la validación del estado de revocación de un certificado son CRL y OCSP.

2.3.3.2. Revocación en PMI

PMI contempla dos casos en la gestión de revocación:

- no considerar la revocación, si el periodo de validación del AC es lo suficientemente corto (minutos, horas, etc).
- considerar la revocación, de forma similar a PKI.

En el primer caso, se elimina el coste debido a la verificación del estado de revocación. Sin embargo, en un escenario general, se ha de tener en cuenta el proceso de revocación. Si por alguna razón una AA revoca un AC, las entidades del sistema han de poder verificar que ese certificado ha sido revocado, de manera similar a la PKI. En PKI, la CRL es el sistema más utilizado para notificar la revocación de los certificados. La ITU-T señala que en PMI puede utilizarse una estructura casi idéntica a la CRL, a la

cual llaman ACRL, aunque también se menciona que podrían utilizarse otros protocolos de revocación. Por lo tanto, se puede asumir que cualquier protocolo de revocación definido para PKI podría utilizarse para PMI.

La recomendación X.509 define dos extensiones para poder localizar datos de revocación para el AC que las contiene:

- *CRL Distribution Points*: contiene el nombre o URL (*Uniform Resource Locator*) del directorio desde el que puede obtenerse la ACRL.
- *No revocation information*: si esta extensión se incluye en el certificado, indica que no existe información de revocación sobre ese AC. Este esquema es apropiado para ACs con un periodo de validez corto, y cuando el riesgo de seguridad debido al uso fraudulento de un AC es menor que el coste de gestión de la revocación.

2.4. Validación de credenciales

Como se ha presentado en el apartado 2.2, existen diversos formatos de certificados digitales que permiten transportar y certificar los privilegios, o más ampliamente los atributos de una determinada entidad. El certificado de atributo X.509 (AC) es el que mejor se adapta a las características necesarias: formato estandarizado, independencia con la identidad del usuario, gestión descentralizada y soporte de delegación.

El proceso de validación de certificados, en general, se puede dividir en dos procedimientos principales:

- Obtención tanto de los certificados que forman la cadena, como de los datos de revocación.
- Validación de todos los certificados que forman la cadena.

Esta idea es extensible a los pasos que se realizan en la verificación de otro tipo de certificados digitales, como puedan ser los certificados de clave pública. En este apartado se explican los procedimientos que requiere llevar a cabo una entidad para validar y obtener los privilegios de un usuario presentados a través de un AC. La validación de las cadenas de certificados de identidad no se presentará a parte, por ser un procedimiento que puede formar parte del propio proceso de validación de un certificado de atributo.

2.4.1. Construcción del camino de certificados

Uno de los procesos principales en el proceso de validación de cadenas de certificados es el de obtener los certificados que forman la propia cadena, desde el cer-

tificado presentado por el usuario, hasta el certificado raíz, de manera que se pueda certificar la veracidad de la información contenida en el certificado presentado. Como se ha comentado, a este procedimiento se le conoce como descubrimiento de cadenas.

El descubrimiento de cadenas (55) puede llegar a ser complejo cuando se permite la delegación, ya que el número de certificados a obtener para validar la cadena, puede llegar a ser elevado. A priori, es posible que no se conozca el número de certificados involucrados, por lo que debería descubrirse y recuperar uno por uno todos los certificados. Se pueden utilizar diferentes enfoques para llevar a cabo este procedimiento:

1. El solicitante presenta los certificados necesarios para realizar el procedimiento de validación (modelo *push*). En este caso, el verificador no necesita realizar accesos a sistemas externos, por lo que se disminuye la probabilidad de denegación del servicio por falta de información para la validación. Sin embargo, obliga a que el usuario almacene los certificados involucrados, aunque este hecho, a priori, no debería ser problemático.

2. El verificador de credenciales debe obtener los certificados que forman la cadena. En este caso, para facilitar las tareas al verificador de credenciales, se podría utilizar el campo extensiones del certificado, para incluir información relativa a:

   - Los identificadores de los certificados que forman la cadena. De esta manera, el verificador conoce el identificador de los certificados, lo que le permite conocer qué certificados localizar. Además debe conocer la localización de los mismos, ya bien sea por configuración o por otras vías.
   - La localización de directorios en los que se almacena, como mínimo el siguiente certificado en la cadena, o lo que es lo mismo, el certificado de la entidad emisora. Sin embargo, en este caso, la utilización de extensiones no es recomendable, ya que la modificación de esta información podría provocar que no se pudiera llevar a cabo la validación del certificado del usuario.

2.4.2. Verificación de la cadena de certificados

Una vez obtenida la cadena de certificados, se deben validar los diferentes campos de los certificados, así como el enlace entre ellos. A continuación se explican los procesos más relevantes dentro del proceso de verificación de cadenas de certificados, como son:

1. **Básico**: se realiza para todo certificado de atributo y es la base para el proceso de delegación.

2. **Delegación**: se realiza si existen autoridades de atributo intermedias entre la fuente de autoridad y la entidad final.

2.4.2.1. Validación Básica

Dentro del procedimiento básico (ver figura 2.11), se deben tener en cuenta los siguientes puntos:

- Determinar la intención explícita de un usuario para utilizar un determinado privilegio en el contexto en el que se encuentran los recursos. Este procedimiento no se encuentra contemplado dentro de la recomendación X.509.

- Verificar la información de revocación. Este procedimiento implica la obtención de información de algún servidor o repositorio. Sin embargo, existe la posibilidad de que debido a la corta duración del certificado se indicara la no existencia de información de revocación. En este caso, no es necesario realizar este procedimiento.

- Verificar que el tiempo de evaluación, o tiempo actual, en el cual se utiliza el certificado de atributo, está dentro del periodo de validez de cada uno de los certificados en la cadena de delegación. Esto debe ser cierto para el servicio de control de acceso, sin embargo, si se desea proporcionar un servicio de no repudio, la verificación se debe realizar para un tiempo pasado.

- Verificar si los privilegios son suficientes para realizar las acciones solicitadas por el usuario. Para ello, el verificador de privilegios utilizará la política junto con las variables de entorno necesarias.

Si se verifican positivamente cada uno de los puntos anteriores, el verificador permitirá al usuario, realizar las operaciones solicitadas.

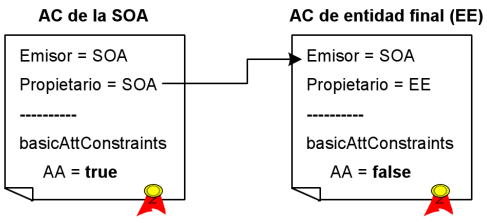

Figura 2.11: Cadena de delegación básica: AC de la SOA y AC de una entidad final

2.4.2.2. Validación en el modelo de delegación

En determinados entornos existe la necesidad de delegar alguno o todos los privilegios de una entidad a otra. El proceso de delegación genera cadenas de certificados, de manera que el verificador de privilegios deberá verificar toda la cadena de certificados (ver figura 2.12), comprobando que:

- Cada AA incluida en la cadena de delegación estaba autorizada a delegar los privilegios.
- Cada certificado de atributo en la cadena es válido con respecto a las restricciones establecidas para la cadena a la que pertenece. Por ejemplo, la longitud máxima del camino de delegación no se ha superado.
- El certificado de identidad asociado a cada certificado de atributo en la cadena es válido bajo las políticas concretas para la cadena a validar.
- Cada privilegio delegado por una AA no puede ser superior al privilegio que posee dicha AA.

Para poder llevar a cabo las verificaciones oportunas, el verificador de privilegios debe obtener la información necesaria, por ejemplo:

- La clave pública para poder verificar la firma realizada por la SOA. La confianza en la clave puede obtenerse a través de un certificado de clave pública expedido a la SOA.
- Los privilegios asociados al usuario o entidad final.
- La cadena de certificados de atributo desde la entidad que presenta el certificado hasta la SOA en la que se confía. En el caso de utilizar privilegios cuya definición puede encontrarse en un certificado, o a través de otros medios, se debe verificar que la sintaxis y valores de los privilegios presentados, cumplen con las normas establecidas.
- La política de privilegios podría ser publicada en un directorio accesible por el verificador de privilegios, o podría estar configurada de forma local.
- Las variables de entorno, como pudiera ser la hora en la cual tiene lugar el acceso.

Establecimiento de la cadena de delegación válida.- A diferencia de la cadena de delegación básica, en la verificación de la validez de una cadena genérica, en la que existen AAs intermedias, el proceso involucra los siguientes pasos (ver figura 2.12):

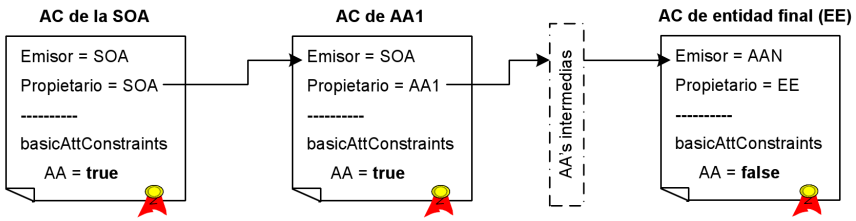

Figura 2.12: Cadena de delegación: SOA, AA intermedias y entidad final

- El expedidor de cada certificado ha de coincidir con el poseedor adyacente en la cadena de delegación.
- Cada AC intermedio, debe contener la extensión `basicAttConstraints` la cual indica que el poseedor del AC puede actuar como AA intermedia. En este caso se puede incluir un valor limitando el número de certificados intermedios (campo `pathLenConstraint`). Su valor indica el número máximo de autoridades de atributo intermedias que puede haber entre el certificado de atributo que contiene la extensión, y el certificado de una entidad final.

```
basicAttConstraints EXTENSION ::= {
    SYNTAX              BasicAttConstraintsSyntax
    IDENTIFIED BY       { id-ce-basicAttConstraints }
}
BasicAttConstraintsSyntax ::= SEQUENCE {
    authority           BOOLEAN DEFAULT FALSE,
    pathLenConstraint   INTEGER (0..MAX) OPTIONAL
}
```

- Puede utilizarse una extensión para poder localizar el certificado adecuado.
- Si se incluye la extensión referente a las políticas aceptables, se debe asegurar que el siguiente certificado en la cadena, contiene al menos una de las políticas especificadas en la extensión.

Verificación de la delegación.- En cuanto a la delegación o traspaso de los privilegios de una entidad a otra, se deben verificar, además de los pasos ya mencionados, los siguientes:

- Cada AA no puede delegar privilegios superiores a los privilegios que posee.
- El verificador de privilegios debe comprobar que cada AA intermedia estaba autorizada a delegar los privilegios, y que éstos cumplen con las reglas de sintaxis establecidas.

Una vez verificada la validez de la cadena, los privilegios obtenidos del certificado de atributo se utilizarán como entrada en el proceso de verificación básico descrito en el apartado 2.4.2.1.

2.4.3. Verificación del estado de revocación de un certificado

Como ya se ha explicado en los apartados anteriores, tanto la PKI como la PMI, necesitan proporcionar a sus usuarios la capacidad de verificar si un certificado es todavía válido (no revocado), en el instante de su utilización. Esta característica es comúnmente conocida en la PKI como verificación del estado de un certificado (*status checking*).

Como se ha explicado en el apartado 2.3.3, los mecanismos de revocación estandarizados son CRL y OCSP. Debido a que el tamaño de la CRL puede ser bastante elevado, el almacenamiento y la utilización de la CRL pueden ser problemáticos para dispositivos limitados. Por otra parte, el protocolo OCSP se basa en una evidencia criptográfica que refleja el estado de los certificados. Esta evidencia la genera *on-line* un OCSP *responder*, el cual debe ser una TTP. En este caso se reduce notablemente el consumo de ancho de banda, aunque el verificador ha de ser acceder al OCSP *responder* cada vez que realice una comprobación de revocación.

En situaciones donde el acceso a servidores *on-line* no es posible, no se puede garantizar la conexión a una TTP y, por lo tanto, ni OCSP ni CRL son, a priori, soluciones adecuadas para la verificación del estado de los certificados.

2.5. Conclusiones

Existen diversos procedimientos en la validación de certificados, que se pueden considerar complejos en entornos con ciertas limitaciones, tanto de los dispositivos como del canal. Pero sin duda, los más problemáticos se encuentran en aquellos que deben acceder a información que no se encuentra de forma local en el verificador. Se pueden considerar tres procedimientos básicos que son críticos a la hora de validar cadenas de certificados:

- El descubrimiento de cadenas (55) puede ser complejo en entornos donde la desconexión puede provocar la no localización de los certificados. Hay que tener en cuenta que si se parte del certificado a validar, y éste se encuentra dentro de una cadena, la localización de los certificados puede ser compleja. Esta situación puede llevar a la denegación del servicio por falta de información disponible.

- La verificación de la firma de los certificados es una operación costosa computacionalmente, sobre todo en dispositivos con capacidades limitadas. A mayor

longitud de la cadena, mayor número de validaciones necesarias. Esta operación podría ser delegada a una entidad con un mayor número de recursos, sin embargo, en entornos ubicuos no es posible garantizar este aspecto. Ahora bien, podría ser posible realizar un procedimiento conocido como descubrimiento de servicio mediante el cual las entidades que forman una red pueden anunciar los servicios que pueden ofrecer, y pueden averiguar los servicios ofrecidos por otras entidades. Sin embargo, se presenta de nuevo el problema de autenticar y autorizar a las entidades que solicitan y ofrecen los servicios.

- El problema de la falta de acceso a servidores *on-line* afecta directamente a la verificación del estado de revocación de un certificado. El hecho de que la firma de un certificado sea válida no es suficiente para garantizar que el propio certificado sea válido. De hecho, aunque un certificado se haya validado en un instante, es posible que su estado se haya modificado en un instante posterior.

CAPÍTULO 3

ePKI. UNA ARQUITECTURA ADAPTADA PARA LA VALIDACIÓN DE CREDENCIALES EN REDES UBICUAS

3.1. Introducción

Las infraestructuras de gestión de credenciales tradicionales adolecen de una visión limitada y estática de la confianza como ya se explicó en el capítulo 2. Las autoridades están organizadas en jerarquías cuya confianza se deriva de la raíz a las hojas. Los certificados denotan relaciones de confianza directa, mientras que las cadenas de certificación capturan relaciones de confianza indirectas. Esta definición hace que la infraestructura sea totalmente funcional en entornos conectados, en los cuales las autoridades son accesibles por los dispositivos que requieren acceder a los servicios ofrecidos por infraestructuras centralizadas. Por ejemplo, para obtener la información necesaria para la validación de certificados, como pueden ser los certificados de las entidades desconocidas que forman la cadena de certificados, o la información de revocación. Además, se requiere tener una confianza, bien sea implícita o explícita en la autoridad raíz, que es un punto crítico del sistema. Si se rompe la confianza con una autoridad raíz, la confianza en toda la infraestructura se ve comprometida. Por otra parte, los administradores son a menudo quienes, de forma manual, establecen las relaciones de confianza. De esta manera, el establecimiento dinámico de modelos de confianza entre dominios con diferentes entidades de certificación raíz podría llegar a ser complejo.

En este capítulo se presenta la propuesta de una arquitectura realizada en el marco

del proyecto UBISEC (56) para ofrecer servicios avanzados PKIX en entornos ubicuos, tanto cuando es posible el acceso a infraestructuras centralizadas como cuando el acceso no es posible.

3.1.1. Características del entorno

La validación de credenciales en entornos conectados ha sido un tema de estudio desde hace años y no presenta grandes retos. La introducción de nuevos servicios como la PMI ha hecho que surjan nuevos desafíos, como ya explicamos en el capítulo 2. Sin embargo, estos desafíos se agravan tanto para PKI como para PMI en entornos donde los recursos podrían no estar disponibles, y que a continuación se enumeran:

- **Configuración manual**. Para validar PKCs tanto de usuario como de SOA/AAs se ha de configurar el dispositivo de forma explícita con las autoridades raíz que se consideran de confianza. Algunas autoridades comerciales ya suelen estar previamente configuradas por los fabricantes, o sistemas operativos. Por ejemplo, los navegadores web (*Firefox* (57), *Chrome* (58), *Internet Explorer* (59), etc) ya tienen configuradas algunas autoridades raíz e intermedias, sin que el usuario haya podido decidir si confía o no en ellas. Esta visión sobre la confianza no siempre es adecuada, debido principalmente a que es estática (no cambia), y pesada en sus procedimientos (normalmente la gestionan servidores centrales o administradores).

- **Desconexiones temporales**. Cuando los usuarios se mueven a través de diferentes redes con sus dispositivos móviles, la conectividad global se puede perder y algunos servicios pueden estar temporalmente no disponibles. Por lo tanto, autoridades (CA, SOA, AA) y repositorios podrían no estar accesibles en el momento de la validación de credenciales. Esto implica que la información necesaria para la validación de credenciales no esté disponible, o no esté actualizada en un instante dado. Por lo tanto, el resultado de la validación podría ser erróneo, o podría llevar a denegar el acceso al servicio.

- **Gestión de claves**. No se puede presuponer la existencia de un administrador de confianza centralizado y disponible para mantener actualizadas las relaciones de confianza entre las entidades que forman la red.

- **Gestión de cadenas de certificados**. La validación de un AC implica la validación adicional de al menos dos PKCs, el PKC del usuario y el PKC utilizado para firmar el AC del usuario a validar (apartado 2.4). Por lo tanto, a medida que aumenta la complejidad en las cadenas de certificación, podría aumentar el consumo de recursos sobre el dispositivo móvil (carga computacional, memoria, batería). Este

hecho se ve agravado por la posibilidad de desconexiones temporales; la validación de cadenas de certificados requiere el acceso a la información que podría no estar disponible en el dispositivo del usuario en el momento de validación de un certificado (PKC o AC).

3.1.2. Requisitos

Para ilustrar mejor el entorno que estamos considerando, la figura 3.1 muestra un escenario típico en el que ha de trabajar la arquitectura que proponemos. Supongamos que Pedro y Alicia son dos usuarios móviles que tienen conexión temporal a los servicios centralizados de la PKIX (por ejemplo, cuando tienen una conexión estable a Internet). Durante el periodo de conectividad pueden trabajar en lo que hemos llamado «modo conectado», ya que tienen pleno acceso a todos los servicios de PKIX que ofrece la infraestructura: pueden registrarse, obtener certificados, descargar listas de revocación de certificados (CRL), solicitar el estado de revocación de certificados a servidores en línea como OCSP, etc. Como los usuarios son móviles, pueden perder la conectividad global. Sin embargo, todavía se pueden comunicar entre sí formando redes *ad-hoc*, pero no pueden acceder a servicios centralizados de PKIX. En esta situación, que llamamos «modo desconectado», los usuarios deben ser capaces de establecer comunicaciones seguras entre ellos. En particular, deben ser capaces de llevar a cabo la autenticación y/o autorización de otros usuarios.

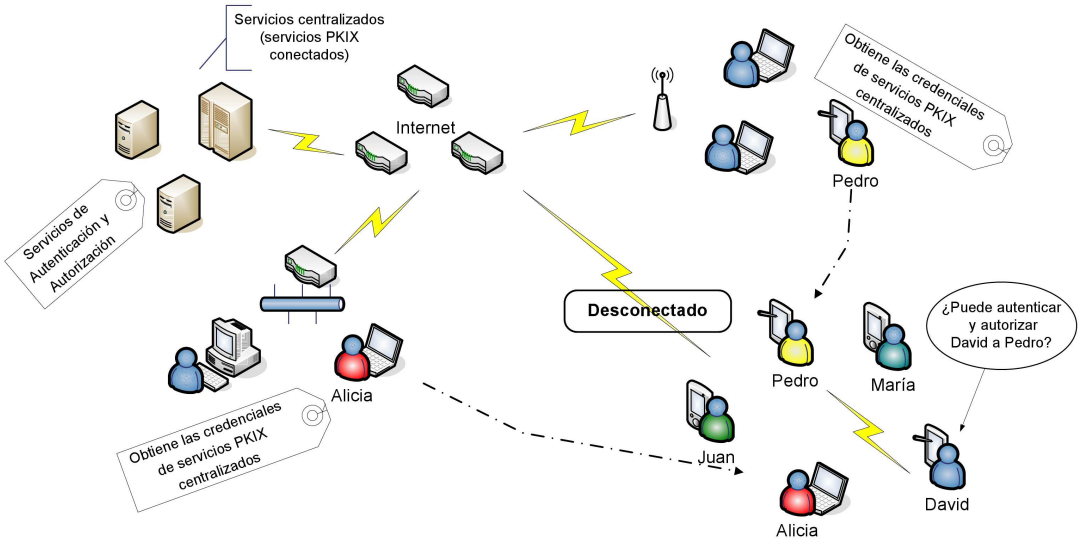

Figura 3.1: Escenario típico en el que trabaja la ePKI

Por lo tanto, es necesario proporcionar mecanismos que amplíen las funcionalidades de los servicios de la PKIX tanto en modo conectado como en modo desconectado. Para lograr este objetivo, proponemos una nueva arquitectura de una PKIX ubicua,

donde las funcionalidades tradicionalmente desempeñadas por una infraestructura centralizada se han trasladado a los dispositivos de usuario, haciendo también posible la validación de certificados en modo desconectado.

Las tecnologías móviles permiten que los usuarios puedan estar conectados tanto a una red centralizada como a otros usuarios formando su propia red (apartado 2.1). Esta movilidad hace que el diseño de sistemas de validación de credenciales deba tener en cuenta estas características. A continuación, se enumeran los principales requisitos que debe cumplir una solución para una infraestructura de autenticación y autorización (AAI) que permita trabajar tanto en modo conectado como en modo desconectado:

R1. **Validación de certificados independiente a una conectividad global**.- Los mecanismos basados en PKIX son adecuados para la autenticación en dominios de confianza heterogéneos, haciendo posible la movilidad del usuario y la desconexión temporal de los servicios, ya que los usuarios pueden llevar consigo sus credenciales y así pueden presentarlas para autenticarse, con independencia del momento y el lugar (modelo *push*, ver apartado 2.3.2). Además, la autorización basada en credenciales PMI permite la utilización de políticas más escalables para una autorización descentralizada.

R2. **Adaptación a entornos cambiantes**.- La decisión sobre el método a utilizar para llevar a cabo la validación de certificados podría variar en función de las condiciones del entorno. Por ejemplo, la disponibilidad de los recursos y servicios podría cambiar el método a utilizar en la validación de credenciales.

R3. **Validación de certificados ligera para dispositivos limitados**.- Los usuarios suelen usar dispositivos ligeros tales como PDAs o teléfonos móviles. Estos dispositivos son fáciles de transportar por lo que permiten una gran movilidad al usuario, sin embargo, tienen capacidades mucho más limitadas en comparación con ordenadores de sobremesa y portátiles. En particular, estas restricciones limitan su capacidad de cómputo, comunicación, almacenamiento y tiempo de vida sin conexión a la red eléctrica. Por lo tanto, la validación de PKCs y ACs debe requerir baja capacidad de cómputo y minimizar el tiempo necesario para completar el proceso. También hay que tener en cuenta que la validación de largas cadenas de certificados puede requerir el almacenamiento de la información necesaria para llevar a cabo la validación (certificados, datos de revocación, etc). Este volumen de información ha de ser el menor posible debido a las capacidades limitadas de los dispositivos.

R4. **Un modelo de confianza dinámico**.- La validación de las cadenas de certificados no se puede garantizar cuando se trabaja en modo desconectado. Más aún,

usuarios sin una relación previa y posiblemente con certificados emitidos por autoridades desconocidas, podrían querer interactuar entre ellos. Por lo tanto, es necesario introducir un nuevo modelo de confianza que permita trabajar en situaciones en las que, por ejemplo, un determinado certificado no pueda ser validado por los métodos tradicionales PKIX, o no permitan el establecimiento de nuevas relaciones de confianza. Se necesita un modelo que refleje la naturaleza dinámica de la confianza, con un bajo coste administrativo y el cual pueda explotar las capacidades de comunicación y colaboración de los nuevos escenarios en los que pueden encontrarse los usuarios.

R5. **Verificación del estado de revocación de los certificados**.- Los protocolos tradicionales de verificación del estado de los certificados no son adecuados para el modo desconectado o para dispositivos limitados. El sistema debería poder trabajar *off-line* (ver apartado 2.3.3) para poder comprobar el estado de un determinado certificado, ya que no se puede garantizar una conectividad constante. Así mismo, el sistema también debe proporcionar respuestas de longitud pequeña debido a las limitaciones tanto de la red como de los dispositivos. Además, debido al dinamismo de la delegación, la probabilidad de revocación puede aumentar, y por lo tanto, aumentar el volumen de información de revocación a distribuir. Este punto es crítico y su complejidad ha hecho que sea el foco de atención en el resto de capítulos de la tesis (capítulos 4, 5 y 6), y por ello, no se profundizará sobre este aspecto en este capítulo.

3.2. Análisis de la validación básica de ACs

El proceso básico de validación de ACs sólo involucra dos certificados, el AC de la entidad final y el AC de la SOA (apartado 2.3.2). En este caso, la SOA es la entidad encargada de expedir todos los certificados a entidades finales, es decir, no existen AAs intermedias. En este apartado se analizan los principales pasos que debe realizar un verificador de privilegios para validar el AC de una entidad final.

3.2.1. Clasificación de las operaciones

Para estudiar el proceso de validación de un AC, se va a dividir el proceso completo en dos tipos de operaciones:

- *Externas*: consideramos como operaciones externas a las operaciones que requieren acceso a servidores/repositorios externos, como por ejemplo, para obtener la información de revocación. Estas operaciones conllevan, básicamente, coste de comunicaciones.

- *Internas*: consideramos como operaciones internas a las operaciones que no requieren obtener información de servidores/repositorios externos para el proceso de validación. Es decir, todas estas operaciones pueden llevarse a cabo de forma interna al dispositivo, y de forma aislada a la red. Estas operaciones conllevan, básicamente, coste computacional.

3.2.2. Pasos de la validación

El IETF (51) define el siguiente conjunto de reglas que debe cumplir un AC para que pueda ser considerado válido (ver figura 3.2):

1. Si la identidad del usuario que posee un AC (`AC.Holder`) se ha enlazado a un PKC, se ha de validar dicho PKC. El RFC 3280 (38), y más recientemente el RFC 5280 (52), especifican los pasos que se han de llevar a cabo para validar un PKC. Esta validación se considera como un servicio externo al verificador de privilegios, por lo que no se detallará en este capítulo. Sin embargo, lo que sí es relevante es la información que se debe obtener y validar para llevar a cabo la validación de un PKC, como es:

   - el PKC del usuario.
   - la datos de revocación sobre el PKC.

   La validación del PKC del usuario implica la validación de la cadena de certificación asociada. Esta operación puede ser clasificada como externa por dos motivos:

   - puede ser delegada a un servidor de validación,
   - los certificados necesarios puede que se tengan que obtener previamente de un repositorio, o de otro dispositivo de red.

   A pesar de que la autenticación del usuario se realice utilizando un PKC, el campo `AC.Holder` ofrece una gran flexibilidad para permitir la autenticación del usuario mediante otros métodos. Por ejemplo, el campo `entityName` se podría utilizar para transportar un identificador del nombre del usuario en un sistema basado en Kerberos (51).

2. La validación de la firma de un AC está formada por los siguientes pasos (figura 3.2):

   - Obtener el PKC utilizado para la firma del AC.
   - Validar el PKC del emisor del AC como se especifica en (38).

- Validar la firma del AC con la clave pública contenida en el PKC del emisor del AC. En este proceso tienen lugar tres subprocesos:
  (a) Descifrar el campo *signature* del certificado con la clave pública del expedidor del certificado.
  (b) Generar el *hash* sobre la estructura *toBeSigned*.
  (c) Comparar el resultado de la primera operación (a) con el resultado de la segunda (b). Si ambos valores no coinciden, el certificado se considera no válido.

3. El emisor del AC ha de ser considerado como de confianza de forma directa, por ejemplo, por configuración.

4. Si el AC contiene una extensión crítica que el verificador no reconoce, el AC se ha de considerar como no válido. Para nuestro estudio, se consideran las siguientes extensiones principales (para otras extensiones consultar (28; 39)):

   - ***noRevAvail***: Esta extensión indica que no existe información de revocación para el AC a validar. Si el AC no contiene esta extensión, el verificador de privilegios debe verificar el estado de revocación del certificado.
   - ***basicAttConstraints***: Esta extensión permite establecer restricciones en la cadena de delegación a través de dos campos. El primer campo es el de `authority` el cual indica si el poseedor del AC está autorizado a delegar (`true`), o no (`false`), los atributos contenidos en el AC. El segundo campo (`pathLenConstraint`) indica el número máximo de autoridades intermedias en la cadena de delegación. El valor de este campo es significativo si el valor del campo `authority` es `true`. El poseedor de un AC en el cual `pathLenConstraint=0`, sólo puede expedir certificados a entidades finales.

5. Verificación de otros campos del certificado. Estas operaciones pueden considerase como comparativas. Por ejemplo, el instante de tiempo en el que se valida el AC ha de estar dentro del periodo de validez especificado en el AC. En algunas aplicaciones, el instante en el cual el AC se valida y el instante de tiempo actual, podría no coincidir. Estas operaciones se pueden considerar como incremento de coste.

En este capítulo nos centraremos en la información necesaria para la validación básica de un AC, como es la validación de la identidad del `AC Holder`, y en el establecimiento de la confianza de una AA.

La validación básica de un AC conlleva el acceso y validación de diversos PKCs, todos ellos procesos costosos. Al considerar redes ubicuas, el acceso a esta información no siempre es posible, por lo que la probabilidad de denegar el acceso a recursos

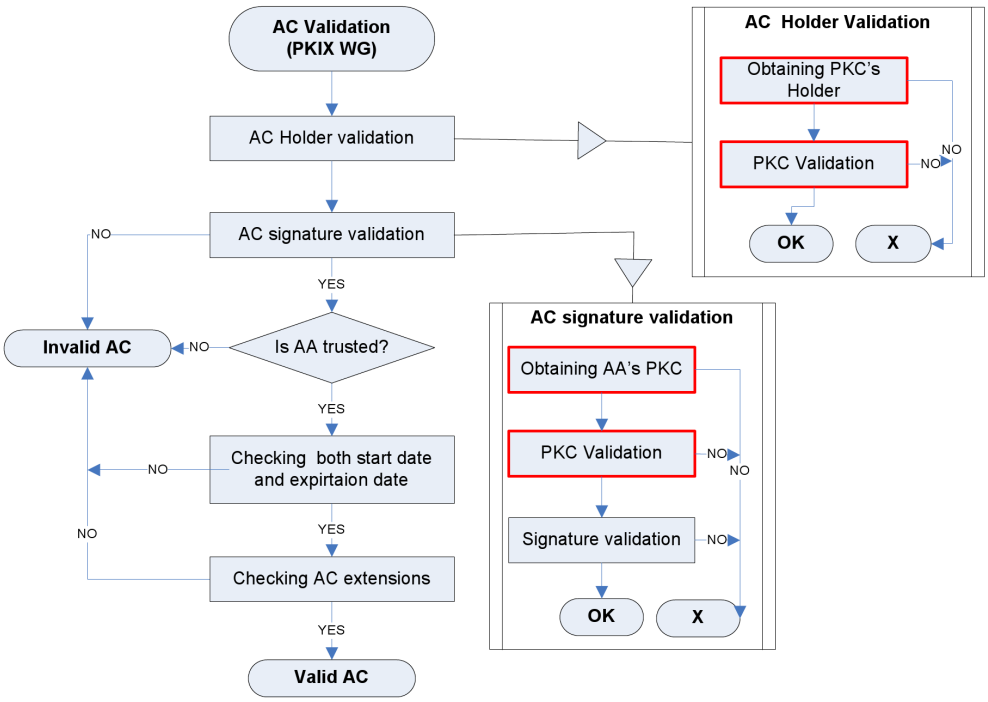

Figura 3.2: Proceso básico de validación de ACs

o el establecimiento de comunicaciones para compartir información entre usuarios puede verse incrementada.

La validación de la identidad del poseedor de un AC puede realizarse utilizando tanto un PKC como otro mecanismo, tal como especifica el IETF en (51). Por lo tanto, supondremos que el servicio de validación de identidad es externo al servicio de validación de ACs.

3.3. Una AAI para entornos ubicuos: *Enhanced-PKI*

Tal y como se presentó en el apartado 3.1, los servicios de autenticación y autorización de usuarios en Internet se pueden llevar a cabo a través de una infraestructura centralizada como la AAI. Para el modo conectado, la PKI y la PMI pueden soportar los servicios de autenticación y autorización, respectivamente, de forma eficiente (apartado 2.3). Sin embargo, como se ha comentado en el análisis (apartado 3.2), la validación básica de un AC puede requerir el acceso a información externa. Este hecho puede provocar que en el modo desconectado este tipo de servicios centralizados pueden no estar accesibles, por lo que se requieren nuevas soluciones que puedan dar soporte a estas situaciones. En este apartado se presenta una infraestructura que extiende la funcionalidad de varios servicios de la PKIX cumpliendo con los requisitos expuestos

en el apartado 3.1.2.

3.3.1. Visión general de la Infraestructura

El objetivo principal de la propuesta es la de proporcionar una AAI que dé soporte a entornos ubicuos, mediante la incorporación de: 1) una PMI para entornos conectados, y 2) un cliente móvil de PKI/PMI para entornos desconectados. A esta nueva infraestructura la hemos llamado *enhanced PKI* (ePKI), la cual se estableció dentro del marco del proyecto europeo UBISEC (60). La idea básica de la ePKI es adaptar los servicios tanto de la PKI como de la PMI, para ser utilizados en los escenarios de computación ubicua, desarrollando únicamente nuevos servicios de PKI-PMI cuando sea necesario.

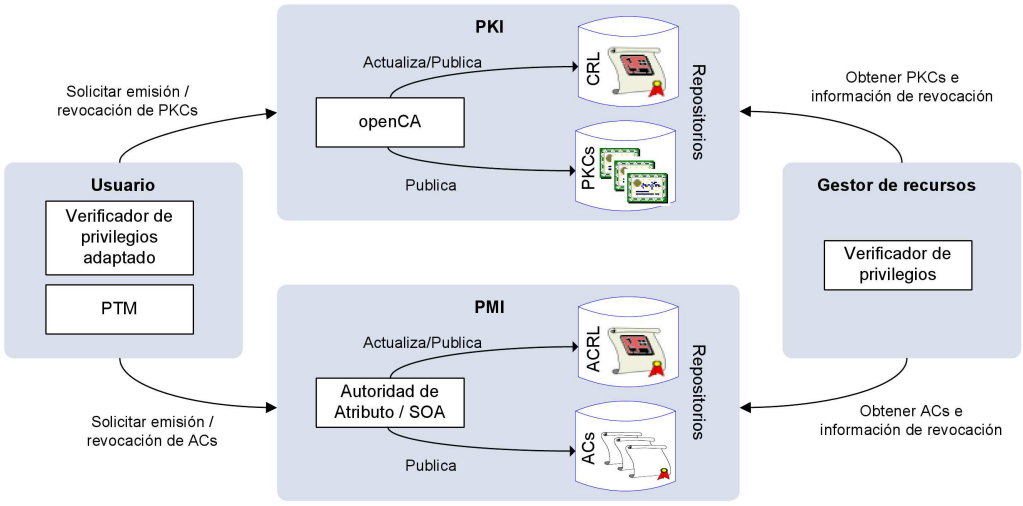

Figura 3.3: Infraestructura de la ePKI

La figura 3.3 representa la infraestructura ePKI, la cual puede dividirse en dos partes: 1) la infraestructura que ofrece los servicios de seguridad (bloques PKI y PMI); y 2) los clientes que utilizan los servicios de la infraestructura (bloques Usuario y Gestor de Recursos). Para la autenticación de las entidades en modo conectado se utiliza una PKI, mientras que para la autorización se utiliza una PMI. Los servicios básicos de la PMI para modo conectado se proporcionan fácilmente, ya que tanto los usuarios como el Gestor de Recursos pueden conectarse a la AAI y a los repositorios para acceder a los servicios ofrecidos. Sin embargo, el Gestor de Recursos debe ser adaptado para trabajar en modo desconectado, ya que el acceso a información (por ejemplo, datos de revocación), puede no estar disponible. El cliente de la ePKI puede ser cualquier entidad que requiera acceder a los servicios tanto de autenticación como de autorización. En la infraestructura ePKI se diferencian dos tipos de clientes:

1. Aquellos incorporados en los dispositivos de los usuarios que realizan, principalmente, peticiones de expedición y revocación de certificados a la ePKI;

2. El Gestor de Recursos, el cual se encarga de controlar el acceso de los usuarios a los recursos de la red, en base a los atributos contenidos en el AC del usuario, y que por lo tanto, utiliza los servicios de la ePKI.

Para la autenticación de las entidades en modo conectado se utiliza una PKI. Como es una infraestructura bastante extendida y uno de los objetivos es centrar el esfuerzo en el diseño de nuevas funcionalidades, se decidió utilizar un *software* de código abierto para proporcionar las funcionalidades básicas de la PKI. *OpenCA* (61) permite proporcionar, principalmente, los servicios de: expedición, revocación y publicación tanto de certificados de clave pública como de información de revocación de los mismos.

Para el servicio de autorización se utiliza una PMI basada en certificados de atributo X.509. A diferencia de la PKI, para PMI en el momento del desarrollo del proyecto, no se tenía constancia de plataformas desarrolladas y de acceso público. Por lo tanto, a continuación, se detalla el diseño de la PMI para modo conectado. La solución para llevar a cabo los servicios tanto de autorización como de autorización en modo desconectado se explican en el apartado 3.5.

3.3.2. Diseño de la PMI

En este apartado se muestra el diseño general de la PMI utilizando el lenguaje de modelado UML (62). Este lenguaje permite representar de una manera gráfica y estandarizada tanto los componentes de la arquitectura como el comportamiento de los usuarios o entidades que interactúan con los componentes del sistema PMI. A continuación se introducirán los elementos de UML necesarios para entender el diseño de la ePKI.

3.3.2.1. UML como lenguaje de modelado

UML (*Unified Modeling Language*) (62) es un lenguaje de modelado visual que se utiliza tanto para el diseño de sistemas *software* como de arquitecturas *hardware*. Se utiliza para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente, de diseño. Un diagrama en UML es la representación gráfica de un conjunto de elementos con sus relaciones, la cual ofrece una vista del sistema a modelar. Un modelo UML está compuesto por tres clases de bloques: 1) Elementos que son abstracciones de objetos y acciones, 2) Relaciones que enlazan los elementos entre sí, y 3) Diagramas que son colecciones de elementos con sus relaciones. Para poder representar de manera correcta un sistema, UML ofrece una amplia variedad

de diagramas que permiten visualizar el sistema desde varias perspectivas. En este capítulo se utilizarán tres de los múltiples diagramas que ofrece UML para el diseño e implementación de la PMI:

1. **Diagrama de Casos de Uso**.- representa gráficamente los casos de uso que tiene un sistema. Se define un caso de uso como cada posible interacción de las diferentes entidades con el sistema a desarrollar, donde se representan los requisitos funcionales. Es decir, se da una visión global de lo que tiene que hacer un sistema y cómo.

2. **Diagrama de Componentes**.- representa cómo un sistema software se divide en componentes, y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, módulos, ejecutables, etc.

3. **Diagrama de Secuencia**.- contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario. Un diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos que participan en la interacción y los mensajes que intercambian, ordenados según su secuencia en el tiempo.

En el documento técnico del proyecto UBISEC (63) se puede encontrar una especificación más amplia del diseño e implementación de la ePKI.

3.3.2.2. Diagramas de Casos de Uso

En el diseño de la PMI se han clasificado en tres grupos las personas o entidades administrativas que están involucradas de forma activa en el sistema. Estas tres entidades se identifican con tres casos de uso, en los que interactúan: el usuario con el dispositivo, la autoridad de atributo con el conjunto de posibles usuarios, y, finalmente, el verificador de privilegios con el usuario (este caso se verá con más detalle en el apartado 3.4.1).

La figura 3.4 representa el primer caso de uso en el que el usuario utiliza su dispositivo para gestionar sus certificados y claves privadas emitidas por la Autoridad de Certificación. Por otra parte, si se necesitan canales seguros basados en TLS (*Transport Layer Security*) (64; 65) con autenticación de cliente, el navegador debe permitir seleccionar el certificado de usuario adecuado. Es posible el uso de contraseñas y tarjetas inteligentes para el almacenamiento seguro de certificados de usuario y de claves privadas.

En el segundo caso (figura 3.5) se distinguen cuatro usuarios que interactúan con la infraestructura de la PMI, entre los que se encuentra el verificador de privilegios:

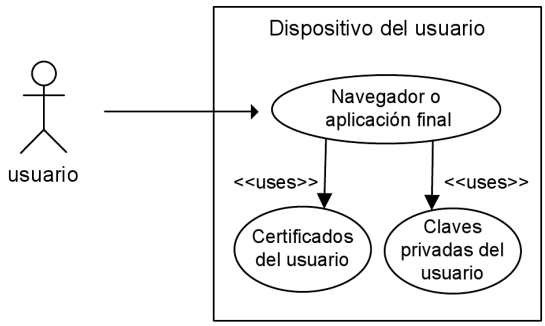

Figura 3.4: Caso de uso de la ePKI. Usuario-Dispositivo

1. **Usuario**.- Un usuario puede tener acceso a la AA para realizar solicitudes de los certificados de atributo, tales como: expedición, renovación, revocación y obtención del certificado previamente expedido.

2. **Entidad autorizada**.- Una entidad autorizada (definida en la política de revocación) podría solicitar la revocación del AC que pertenece a otra entidad.

3. **Administrador de atributos**.- Podría existir una autoridad de registro similar a la RA en la PKI. En este caso, un gestor de atributos puede revisar la solicitud del usuario para permitir o denegar la expedición de un AC al usuario solicitante.

4. **Verificador de privilegios**.- El verificador de privilegios (PV - *Privilege Verifier*) necesita obtener la información para validar las credenciales del usuario, tales como los certificados e información sobre el estado de revocación de los certificados.

En modo desconectado esta diferenciación podría no existir, ya que algunos usuarios podrían actuar tanto como entidades solicitando servicios como servidores proporcionando los servicios. Esta situación implica que un usuario en determinados momentos podría ofrecer determinados servicios, lo que podría requerir la validación de las credenciales (PKC y/o AC) de otro usuario, realizando funciones de *Gestor de Recursos*, como se explica en el apartado 3.5.

3.3.2.3. Diagrama de Componentes

La arquitectura del sistema general de la PMI se muestra en la figura 3.6. Ésta se divide en cuatro componentes principales:

Dispositivo de Usuario.- realiza las operaciones de seguridad necesarias, tales como la gestión de certificados, el almacenamiento seguro de claves de usuario y el establecimiento de un canal seguro con autenticación de cliente.

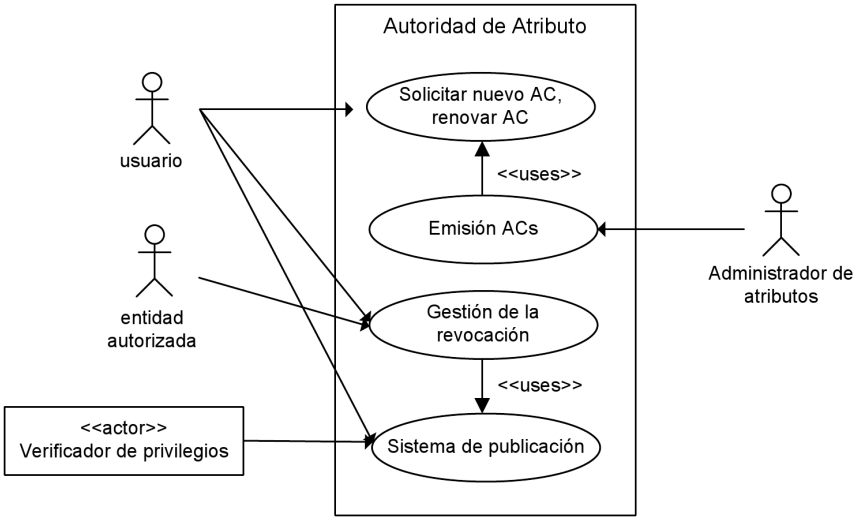

Figura 3.5: Caso de Uso. Autoridad de Atributos - Verificador de Privilegios

Autoridad de Certificación.- proporciona servicios de autenticación para los usuarios. Permite la expedición, renovación, revocación y publicación de los certificados de identidad. También proporciona información relacionada con los certificados, como puede ser: políticas de certificación y datos de estado de revocación.

Autoridad de Atributo.- es la encargada de asignar/delegar privilegios a usuarios a través de la expedición de certificados de atributo (AC). Al igual que la CA, la AA podría ser la encargada de gestionar el ciclo de vida de los ACs que ha expedido.

Verificador de Privilegios.- (o punto de acceso a los recursos) es el responsable de validar los privilegios del usuario. Permite a los usuarios realizar ciertas operaciones sobre los recursos en base a sus privilegios.

Cada componente necesita comunicarse con los otros componentes para obtener/proporcionar los servicios que ofrecen/requieren. Esta operación se realiza mediante las interfaces. Una interfaz en UML es una especificación para las operaciones externas visibles de un componente a otra entidad, pero siempre sin especificar la estructura interna. En el caso de la ePKI, se proporcionan cuatro interfaces para llevar a cabo esta comunicación, una por cada componente en la figura 3.6:

1. `I_PMI_ACA`: define la interfaz de comunicación entre un dispositivo de usuario y una AA. El usuario puede solicitar la expedición, revocación y obtención de sus certificados de atributo.

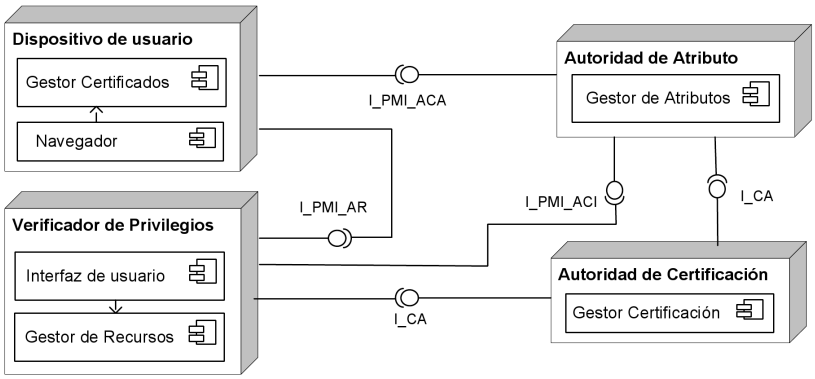

Figura 3.6: Diagrama de componentes de la PMI

2. `I_PMI_AR`: define la interfaz de comunicación entre un dispositivo de usuario y el PV. El usuario puede presentar un certificado, tanto para su autenticación como para su autorización. También puede solicitar y obtener acceso a los recursos controlados por el verificador de privilegios. Si la información solicitada por el usuario es confidencial, es necesario utilizar un protocolo para asegurar la comunicación, como puede ser TLS.

3. `I_PMI_ACI`: define la interfaz para utilizar los servicios de la AA. Permite obtener la información para validar las credenciales del usuario, tales como: certificados de atributo, información del estado de revocación, las políticas de los certificados, etc.

4. `I_CA`: define la interfaz con la CA. Permite obtener la información para validar los certificados de identidad: certificados de identidad involucrados en la cadena de certificación y datos del estado de revocación.

A continuación se define con más detalle el diseño del componente PV. El resto de componentes se pueden encontrar más detallados en la documentación técnica del proyecto (60).

3.4. Modo conectado. Diseño del Verificador de Privilegios

En modo conectado el verificador de privilegios (PV) puede acceder a los servicios ofrecidos tanto por la autoridad de certificación como por la autoridad de atributos. En este apartado se da una visión general del diseño de un verificador de privilegios para trabajar en modo conectado. Al igual que en el caso anterior, el diseño se especifica utilizando los diagramas UML de casos de uso y de componentes.

3.4.1. Diagrama de Casos de Uso

El verificador de privilegios es uno de los actores en los casos vistos en el apartado 3.3.2.2. En su diseño se identifica un caso de uso principal, el usuario con el verificador de privilegios. En este caso (figura 3.7), los usuarios interactúan con el verificador de privilegios para acceder a los recursos. El PV se encarga de gestionar el acceso a los recursos, verificando la identidad y los privilegios del usuario.

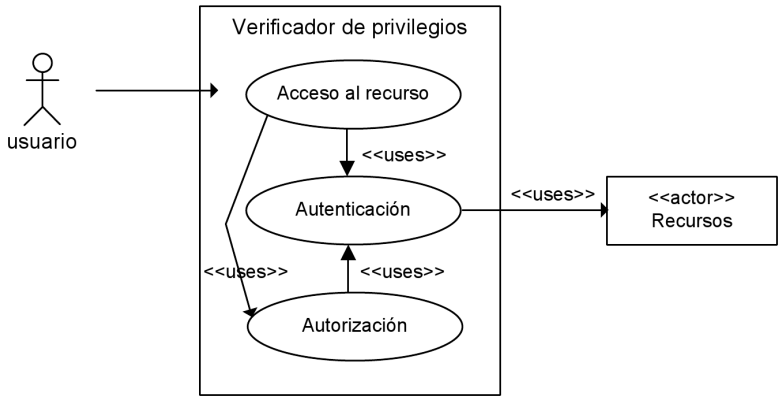

Figura 3.7: Caso de Uso. Verificador de Privilegios - Recursos

3.4.2. Diagrama de Componentes

El PV gestiona el acceso a los recursos. Permite utilizar los recursos que controla sólo a usuarios autorizados. Para llevarlo a cabo valida las credenciales del usuario, tanto de identidad como de atributos, y decide dar acceso a los recursos en base a: privilegios del usuario, políticas de acceso y variables de entorno tal como especifica el modelo de control (ver apartado 2.3.2).

El PV se divide en cuatro componentes principales (ver figura 3.8): Verificador de ACs, Verificador de Autenticación, Localizador de Información y Decisor; y en un interfaz accesible por sus usuarios, `I_PMI_AR`. Este interfaz permite el acceso a los servicios por el dispositivo del usuario. Si la información que se va a intercambiar es confidencial, es necesario utilizar un canal de comunicación seguro como TLS. El verificador de privilegios también puede almacenar información proporcionada por el usuario en la fase de acceso. Este hecho permite reducir tanto el tiempo de respuesta de acceso para el usuario en siguientes accesos, como la carga sobre el PV. Entre la información a almacenar podemos encontrar: la política aplicable, la especificación de las funciones de la CA, los certificados AC/PKC utilizados por los usuarios en accesos anteriores y la información sobre la sesión abierta del usuario.

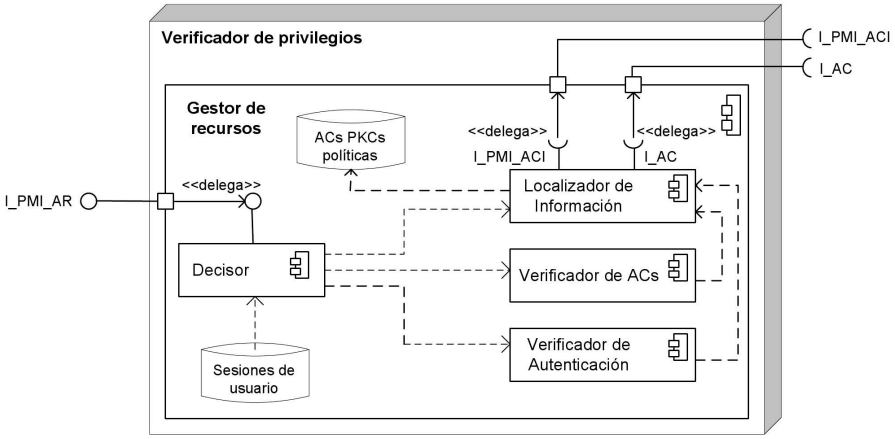

Figura 3.8: Diagrama de Componentes del Verificador de Privilegios

Como se ha comentado, el verificador de privilegios se puede dividir en cuatro subcomponentes principales cuyas funcionalidades principales se detallan a continuación:

Verificador de Autenticación.- proporciona servicios para validar la identidad de un usuario. En este caso, la identidad del usuario está vinculada a su clave pública a través de un PKC. En el proceso de validación de un AC de usuario, es necesario validar como mínimo, dos caminos de certificación: el que establece la identidad del usuario y el que permite validar la firma del AC (apartado 3.2.2). El proceso de autenticación implica la validación de otra información, como puede ser: la firma del certificado, el periodo de validez y el estado de revocación de cada certificado en el camino. Este componente utiliza la interfaz `I_CA` para obtener datos tales como, certificados de identidad o información de revocación (CRL, OCSP, etc.).

Localizador de Información.- proporciona a los diferentes módulos de verificación la información necesaria para la validación de los certificados: AC, ACRL, PKC, CRL, políticas, etc. Esta información la obtiene, o bien de su base de datos local, o a través de las interfaces de la PMI (`I_PMI_ACI`) o de la CA (`I_AC`).

Verificador de ACs.- es el encargado de validar el AC del usuario y de obtener los privilegios asignados al usuario. Este componente utiliza los servicios que ofrecen los componentes Verificador de Autenticación y Localizador de Información a fin de lograr esta tarea. El Verificador de Autenticación permite obtener y verificar la clave pública utilizada en la firma del AC presentado por el usuario, y la identidad del usuario a la cual está enlazado el AC presentado. Por su parte, el

Localizador de Información permite obtener la información para el proceso de validación.

Decisor.- es el encargado de tomar la decisión final sobre la solicitud de acceso a los recursos por parte del usuario. Esta decisión se toma basándose en la información proporcionada por el resto de componentes. Este componente lleva a cabo los siguientes pasos para tomar una decisión sobre la petición de acceso:

(a) Utiliza las funciones ofrecidas por los componentes Verificador de Autenticación y Verificador de ACs para validar las credenciales del usuario, PKC y AC respectivamente.

(b) Utiliza las funciones ofrecidas por el Verificador de ACs para obtener los privilegios asignados al usuario.

(c) Toma una decisión sobre si el usuario puede o no puede acceder a los recursos solicitados.

(d) Gestiona la sesión del usuario. Esta gestión permite reducir la carga en el PV ya que almacena en caché la información del usuario validado. En cada nuevo acceso, no es necesario llevar a cabo el proceso completo de validación de credenciales.

3.4.3. Diagrama de Secuencia

El diagrama de secuencia permite dar una visión del flujo de comunicación que existe entre los diferentes componentes. La figura 3.9 muestra el diagrama de secuencia que representa el acceso de un usuario a los recursos protegidos por el Verificador de Privilegios. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos.

En la figura 3.9 el dispositivo del usuario solicita el acceso a un recurso controlado por el PV enviando un mensaje de apertura de comunicación. Cuando el PV recibe la petición, obtiene la información necesaria para realizar el proceso de validación: 1) PKCs e información de revocación desde la autoridad de certificación utilizando dos mensajes diferentes, y 2) ACs e información de revocación asociada desde la autoridad de atributo. Con la información obtenida verifica las credenciales del usuario y permite/deniega el acceso al recurso. Si el usuario vuelve a solicitar acceso, el PV no necesita acceder a servidores externos, ya que posee toda la información en caché y puede hacer uso de la información de la sesión del usuario (ver apartado 3.4.2).

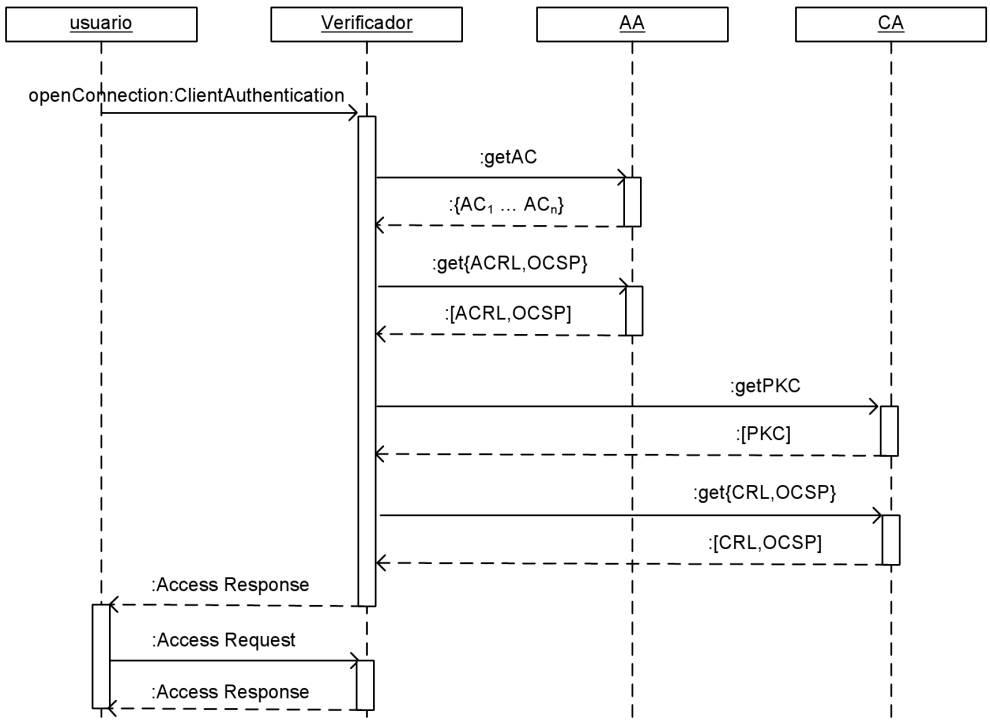

Figura 3.9: Diagrama de secuencia en la ePKI

3.5. Modo desconectado. Cliente móvil de la ePKI

En el apartado 3.4 se ha descrito cómo los usuarios móviles pueden obtener sus credenciales(PKCs/ACs) de una AAI centralizada y almacenarlas en sus dispositivos. De esta manera, la autenticación/autorización de un usuario podrá realizarse en base a las credenciales presentadas por él mismo (modelo *push*). Si el usuario no presenta las credenciales, el verificador de privilegios puede obtenerlas de los repositorios centralizados (model *pull*). En modo conectado, parte de la AAI se puede utilizar para ayudar a la validación de credenciales, mientras que en modo desconectado la infraestructura puede no estar accesible. En estas condiciones, varias funcionalidades como la validación de cadenas de certificados y de la información de revocación, podrían no poder llevarse a cabo. Además, los usuarios móviles pueden formar redes *ad-hoc* con el fin de compartir servicios. En este escenario, algunos dispositivos tienen que comportarse como servidores, tomar ciertas decisiones sobre el control de acceso, establecer nuevas relaciones de confianza, validar credenciales, etc. De esta manera, los dispositivos móviles se convierten en cierto modo en gestores de recursos, aunque con unas capacidades limitadas.

Para poder dar solución a los requisitos del sistema (apartado 3.1.2), en este apartado se presentan tres componentes que trabajan de forma colaborativa para dar so-

porte a la validación de credenciales. Estos componentes son los que forman el cliente móvil ePKI y que permiten llevar a cabo la autenticación/autorización del usuario, incluso en modo desconectado. La figura 3.10 representa los tres componentes principales que componen la arquitectura del cliente móvil ePKI, que son:

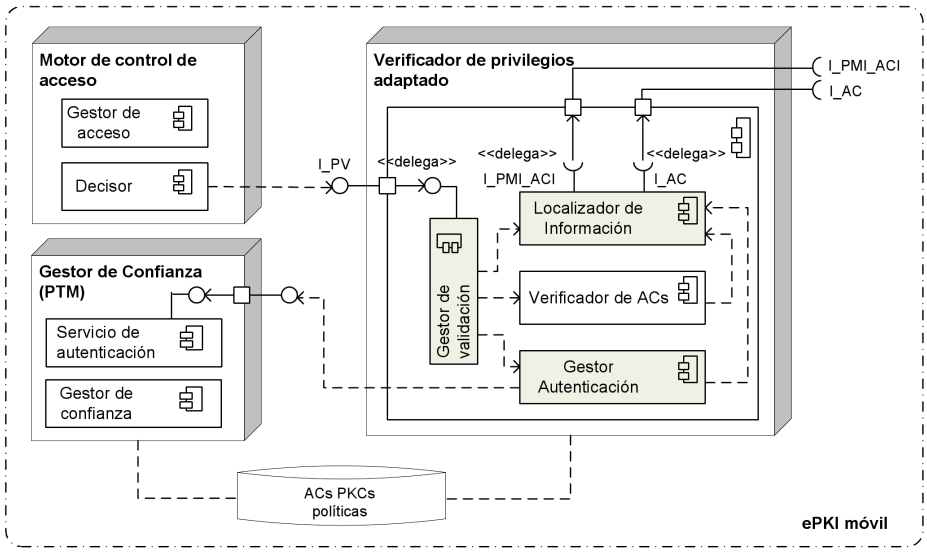

Figura 3.10: Arquitectura del dispositivo móvil del usuario en la ePKI

Gestor de confianza (PTM - *Pervasive Trust Model*) define un modelo de confianza distribuido similar a PGP, donde los usuarios pueden ser sus propias CAs, formando dominios de confianza con sus dispositivos. Las relaciones de confianza entre usuarios se establecen de igual-a-igual y son expresadas como una función continua entre 0 y 1, siendo los valores extremos de total desconfianza y total confianza, respectivamente. El valor de confianza se obtiene haciendo un promedio de todas las recomendaciones ponderadas con el valor de confianza que se tiene sobre el recomendador (66). Los dispositivos actúan en representación de los usuarios (entidades físicas) y cada uno posee su propia pareja de claves, una lista protegida de entidades fiables y no fiables, su grado de confianza, su comportamiento, y sus PKCs. Si un usuario posee certificados emitidos por CAs previamente configuradas por el fabricante, dicha relación será utilizada para autenticar al usuario.

PTM ha sido desarrollado por la UC3M (*Universidad Carlos III de Madrid*). Para más información, se puede consultar la documentación técnica del proyecto (60; 63; 67).

Verificador de privilegios adaptado es el componente que se encarga de la verifica-

ción de los certificados de atributo, los cuales son presentados por los usuarios cuando solicitan acceso a los recursos (modelo *push*). En modo conectado, el PV ofrece los mismos servicios que los presentados en el apartado 3.4. En modo desconectado, el verificador de privilegios modifica su funcionamiento, ya que no se puede acceder a los servidores centralizados de las autoridades de certificación. Los dos procesos críticos son: 1) la validación de la identidad de los usuarios y de las AAs; y 2) la validación del estado de revocación. El PV utiliza PTM para llevar a cabo las validaciones de los certificados de identidad necesarias cuando no es posible realizar la validación por el método PKIX estándar. Una solución para la obtención y validación del estado de revocación se presenta en el capítulo 6.

Motor de Control de Acceso (ACE - *Access Control Engine*) es el responsable de la toma de decisiones de control de acceso a los recursos en base a las políticas de control de acceso configuradas. El ACE invoca a PTM para autenticar a los usuarios que solicitan el acceso, e invoca a PV para obtener y validar los privilegios contenidos en el AC del usuario. Con esta información, el ACE toma una decisión:

- Si la autenticación y la autorización se realizan utilizando los certificados válidos (PKCs y ACs), la decisión de acceso se basa en certificados verificados utilizando los procedimientos establecidos por PKIX.
- Si la autenticación o la autorización no puede realizarse en base a certificados validados por los procedimientos establecidos por PKIX, la decisión de acceso se basa en información de confianza.

La colaboración entre PTM y el PV permite que la PMI pueda trabajar, en determinados casos, en modo desconectado. En estas situaciones PTM proporciona el servicio de verificación de la identidad del usuario, mientras PV proporciona el servicio de verificación de privilegios del usuario. El acceso a los recursos se permite o no en función de la política actual de autorización. A continuación se detalla el diseño del PV para que pueda trabajar en modo desconectado.

3.5.1. Adaptando el verificador de privilegios

El verificador de privilegios gestiona la validación de los certificados de atributo y consta de tres subcomponentes principales:

Gestor de Autenticación.- gestiona la validación de los PKC. Este componente fue adaptado para trabajar con el componente PTM en modo desconectado. Cuando es necesario validar cualquier PKC vinculado al AC del usuario y la informa-

ción necesaria para realizar el procedimiento PKIX estándar no puede llevarse a cabo, el Gestor de Autenticación delega esta funcionalidad al componente PTM.

Localizador de Información.- proporciona información a otros componentes del PV. Por ejemplo, el PKC de la AA para verificar la firma del AC del usuario. El Localizador de Información puede funcionar de dos maneras:

- proporciona el PKC necesario para la validación de una CA si está almacenado en su caché local, al igual que hacía en modo conectado.
- solicita esta información al componente PTM. Esta funcionalidad se amplió para funcionar en modo desconectado.

Verificador de ACs.- es el encargado de validar las credenciales de autorización del usuario y de obtener los privilegios asociados al usuario. Por lo tanto, este componente proporciona al ACE la información contenida en el AC.

Teniendo en cuenta las funcionalidades de los diferentes módulos, la validación de un certificado de atributo se lleva a cabo de la siguiente manera:

1. En primer lugar, el PV recibe un AC para ser validado. El PKC enlazado al AC puede ser proporcionado de la misma manera que el AC. Si el ACE no proporciona el PKC asociado, el PV puede obtenerlo utilizando el Localizador de Información.

2. La validación de PKCs la delega a PTM y es el Gestor de Autenticación quien se encarga de esa comunicación. Si el PKC es válido utilizando el método estandarizado, el resultado de la autenticación es un valor 1. De lo contrario, el resultado es un nivel de confianza (valor entre 0 y 1).

3. El Verificador de ACs utiliza la información proporcionada por el usuario y por PTM para validar los privilegios del usuario.

4. El ACE procesa el resultado de la validación realizada por el PV. En el apartado 3.5.2 se define el tipo y significado de la información que puede obtener el ACE como resultado de la validación por parte del PV.

Cabe señalar que el PV no verifica la información de autenticación. Esta función se delega al componente PTM. Sin embargo, la decisión final sobre la validación de un AC depende de la validación de, al menos, el PKC de la SOA utilizado para firmar el AC de usuario. Por lo tanto, el resultado de PTM sobre la validación de este PKC, junto con el resultado de la validación del AC realizada por el PV, son utilizados por el componente ACE para tomar una decisión sobre el acceso del usuario a los recursos solicitados. Un AC es considerado válido si tanto el PKC del usuario como el PKC de la SOA son válidas

o de confianza. El valor de confianza que se requiere como mínimo para ofrecer un determinado recurso puede ser definido por configuración, ya que diferentes recursos pueden requerir diferentes niveles de confianza. A este valor mínimo lo denotaremos como $TV_{threshold}$.

Generación de la respuesta sobre la validez de un AC

Cuando el acceso a las credenciales no es posible, o falta información para validar un AC siguiendo los pasos definidos por el PKIX (apartado 3.2), el PV utiliza PTM para obtener un valor de confianza sobre el poseedor de una determinada clave pública.

La validación básica de un AC, como se ha explicado en el apartado 3.2, requiere de la validación de básicamente dos PKCs:

1. El PKC de la SOA para verificar tanto la firma del AC como la confianza para emitir ACs.

2. El PKC del usuario para asegurar que el usuario que presenta el AC es quien dice ser y es el poseedor del AC presentado.

Considerando que el resto de pasos de validación se han podido llevar a cabo de forma correcta, la confianza resultante de la validación del AC puede ser estimada como:

$$TV_{AC\_Basic} = (1/2) \cdot (TV_{SOA} + TV_{user}) \tag{3.1}$$

donde TV (*Trust Value*) indica el valor de confianza sobre la clave pública de una entidad. Por lo tanto, TV_{AC_Basic} indica el valor de confianza resultante sobre el AC de un usuario. Un mismo valor de confianza puede permitir realizar una determinada operación sobre un recurso y puede denegar otra operación sobre el mismo recurso. Los valores se fijarán por configuración como ya se ha comentado en este apartado ($TV_{threshold}$).

Este caso es el más sencillo, ya que no se considera el proceso de delegación. Sin embargo, en el caso de considerarse, el valor de confianza puede ser calculado de forma similar. La ecuación 3.1 podría generalizarse de la siguiente manera:

$$TV_{AC\_path} = \frac{1}{L} \sum_{i=1}^{L} TV(i) \tag{3.2}$$

donde L representa la longitud de la cadena de delegación, y $TV(i)$ representa el valor de confianza asociado a cada certificado de atributo situado en la posición i de la cadena de delegación. La ecuación 3.2 no tiene porqué se la única manera de representar la confianza, ni la más adecuada. Podríamos tomar, de manera más genérica un valor de ponderación K para cada nivel en la cadena de delegación, quedando la

ecuación de la siguiente manera:

$$TV_{AC\_path} = \sum_{i=1}^{L} TV(i)k(i) \tag{3.3}$$

de manera que si $k(i) = 1/L$ para todo valor de $1 \leq i \leq L$, volveríamos a tener la ecuación 3.2.

3.5.2. Definición del interfaz del verificador de privilegios

El interfaz del PV (`IPV`) posee cinco operaciones principales que permiten validar y obtener la información contenida en un AC. Por ejemplo, los atributos del usuario, necesarios para la toma de decisión final sobre el acceso a un determinado recurso.

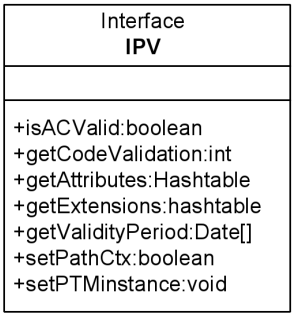

Figura 3.11: Interfaz adaptado del PV

La figura 3.11 muestra la especificación resumida del interfaz en UML. Las operaciones permitidas son:

- `isACValid`: permite verificar la validez de un certificado de atributo. Para ello se le ha de proporcionar, al menos, uno de los dos certificados del usuario, el AC o el PKC. Un AC se considera válido si verifica los pasos definidos en el apartado 3.2.2. El valor retornado por esta operación se puede interpretar de la siguiente manera:
  - Un valor entre [0, 1] indica que los campos del AC son válidos. Este valor es el resultado del nivel de confianza calculado utilizando la ecuación 3.1. Un valor 0 indica mínima confianza, y un valor 1 indica máxima confianza.
  - Un valor > 1 indica que el AC no es válido o no puede considerarse de confianza por algún motivo. Este motivo está codificado por el valor retornado.
- `getAttributes`: permite obtener los atributos que contiene el AC. Por ejemplo, el rol del usuario dentro de la empresa.
- `getExtensions`: obtiene las extensiones incluidas en el certificado.

- `getValidityPeriod`: devuelve una cadena con dos valores. El primero de ellos corresponde a la fecha y hora en la que el certificado empieza a ser válido. El segundo valor corresponde a la fecha y hora a la cual el certificado deja de ser válido.

- `setPathCtx`: permite inicializar los valores de los parámetros de funcionamiento del PV. Por ejemplo, la localización del directorio en el que se encuentran almacenados los certificados.

- `setPTMinstance`: permite pasar una instancia del módulo PTM. De esta manera, la validación de PKCs puede realizarse utilizando las funcionalidades de PTM cuando la validación de la identidad no puede realizarse utilizando los procedimientos definidos por PKIX.

3.6. Evaluación del funcionamiento

Para la demostración del funcionamiento de la ePKI en el dispositivo móvil se implementó un prototipo para el escenario que se describe a continuación. Supongamos tres usuarios (Kris, Alicia y Pedro) que no han tenido relación previa entre ellos, y por lo tanto, se consideran desconocidos. Cada usuario tiene una PDA en la que se ha instalado el siguiente *software*:

1. Una aplicación que ofrece un servicio de álbum de fotos (PAS - *Photo Album Service*)[1] la cual permite al usuario almacenar, visualizar y organizar sus fotos digitales.

2. El cliente ePKI para móvil el cual proporciona control de acceso a los usuarios de la red que lo solicitan, tanto a las fotos compartidas como a las fotos privadas.

En este escenario, Kris trata de romper la seguridad de los dispositivos que se encuentran a su alrededor. Alicia tiene algunas fotos privadas que no quiere compartir. Sin embargo, Alicia ha dado permiso a los aficionados del *Pervasive-Club* para poder acceder a algunas de sus fotos. Pedro pertenece al *Pervasive-Club* y tiene varias fotos divididas en dos categorías: acceso privado y acceso público. Los usuarios forman una red *ad-hoc* para poder compartir sus fotografías. Las imágenes disponibles se almacenan en el repositorio local del álbum de fotos, y se han de definir las políticas de acceso sobre las imágenes para que otros usuarios puedan acceder a ellas.

Para verificar el funcionamiento se han llevado a cabo dos casos. En el primer caso Pedro realiza las funciones de cliente y Alicia las de servidor. La figura 3.12 muestra

[1] Esta aplicación fue desarrollada por Telefónica I+D dentro del proyecto UBISEC.

cómo Alicia permite el acceso de Pedro a las fotos clasificadas en el apartado de accesibles a los usuarios que pertenecen al *Pervasive-Club*. Los pasos que se siguen son:

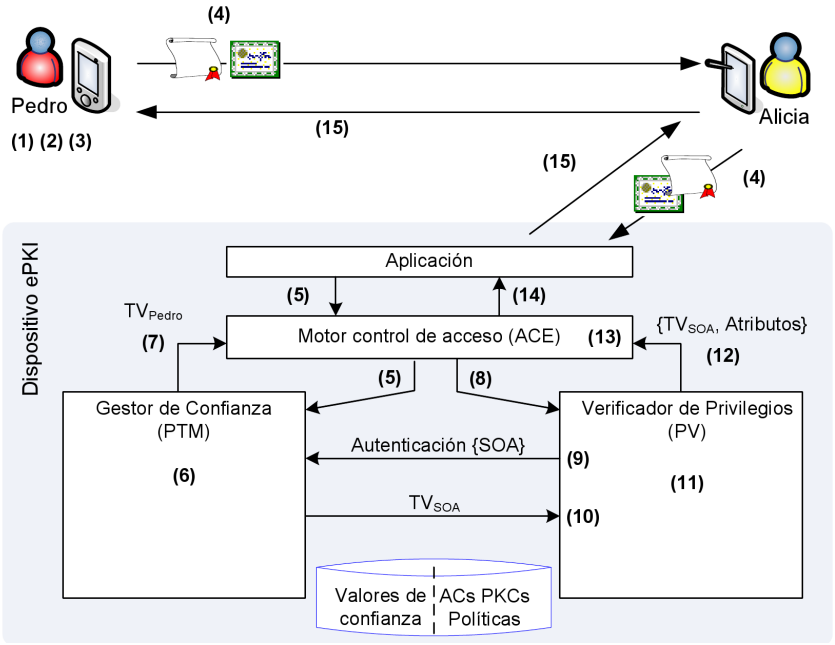

Figura 3.12: Escenario de test para ePKI móvil

1. Pedro inicia la aplicación PAS y se une a la red *ad-hoc*.

2. La PAS de Pedro inicia un proceso de descubrimiento de servicio para encontrar otras PAS accesibles en la red. Para el descubrimiento de servicio se ha utilizado PDP (*Pervasive Discovery Protocol*) (68) ya que una de sus principales ventajas es la de reducir el consumo de recursos en dispositivos limitados.

3. La PAS muestra a Pedro las otras PAS disponibles en su misma red, indicando que Alicia ofrece fotos sobre los juegos olímpicos de Beijin.

4. Pedro pide acceso a la imagen sobre el diseño de una de las mascotas de los juegos, la cual todavía no se ha presentado de forma oficial. En ese momento, se inicia el proceso de autenticación y autorización. Las credenciales de Pedro (PKC y AC) se envían al dispositivo de Alicia para demostrar que, por un lado, Pedro es quien dice ser, y por otro, tiene suficientes privilegios para acceder a esa foto.

5. El PAS entrega esta solicitud al cliente móvil de ePKI. El ACE solicita la validación del PKC del usuario a PTM.

6. PTM obtiene el valor de confianza asociado a la clave pública de Pedro. En este caso $TV_{Pedro} = 0{,}75$.

7. PTM envía al ACE el resultado de la validación del PKC de Pedro.

8. El ACE solicita la validación del AC al PV.

9. El PV solicita a PTM la validación del PKC de la SOA, a fin de validar la firma del AC.

10. PTM informa al PV que la SOA posee un valor de confianza $TV_{SOA} = 0{,}8$.

11. El PV obtiene los atributos contenidos en el AC de Pedro, y envía la respuesta al ACE.

12. El PV retorna un valor de 0,8 junto con los atributos del usuario.

13. El ACE toma una decisión sobre el acceso a los recursos por parte de Pedro. Pedro puede acceder ya que es un usuario de confianza para esta aplicación (por la ecuación 3.2 , $TV_{Pedro} = 0{,}75 > TV_{threshold} = 0{,}65$) y pertenece al *Pervasive-Club*.

14. El ACE envía el resultado al PAS para iniciar el envío de la imagen solicitada a Pedro.

15. La aplicación envía la foto a Pedro.

En el segundo escenario, Pedro actúa como servidor y Kris como cliente: Pedro también ofrece unas cuantas fotos, y Kris intenta conseguir una nueva imagen de la PDA de Pedro. La PDA de Pedro realiza los mismos pasos para validar las credenciales de Kris que en el primer caso (Pedro-Alicia). Sin embargo, Kris no es un usuario de confianza y además no tiene privilegios suficientes para obtener la imagen solicitada, ya que no es un miembro del *Pervasive-Club*. Por lo tanto, el ACE deniega el acceso y PTM actualiza el nivel de confianza de Kris. El PAS define 28 códigos de error resultantes del proceso de validación, es decir, la salida del PV/PTM. A estos códigos de error se les asigna unos determinados pesos, en base a unos patrones definidos, y son utilizados por el monitor de Acciones de PTM para actualizar los valores de confianza.

La aplicación cliente de la ePKI se ha desarrollado en J2ME (*Java 2 Micro Edition*) y ha sido probada en Linux, Windows y Windows CE. El software desarrollado está disponible bajo licencia de código abierto y puede descargarse desde la página web principal del proyecto UBISEC (56).

3.7. Conclusiones

En este capítulo se ha presentado el diseño e implementación de una nueva infraestructura de autenticación y autorización para entornos ubicuos. Esta arquitectura

permite la validación de las credenciales de usuario en redes heterogéneas, donde la conectividad global se puede perder y algunos servicios pueden estar temporalmente no disponibles. La autenticación y autorización se proporcionan a los usuarios y aplicaciones a través de la combinación de servicios PKI tradicionales y de los nuevos servicios de PMI.

Con esta solución un dispositivo móvil puede adaptarse al entorno. Cuando la conexión a sistemas centralizados de PKIX es posible, la validación puede llevarse a cabo de forma tradicional tanto para PKCs como para ACs. Por otro lado, cuando esa conexión no es posible, la validación de autenticación se realiza en base a la confianza establecida con PTM. De esta manera, la validación de autorización puede dar lugar a un mecanismo mixto que permite aumentar la disponibilidad del servicio tanto de autenticación como de autorización. PTM proporciona la validación de autenticación de las entidades involucradas, mientras el PV permite la validación de las credenciales de autorización tanto en modo conectado como en modo desconectado.

Como resultado, la implementación de la ePKI para móvil nos ha permitido verificar su viabilidad en dispositivos limitados.

CAPÍTULO 4

CARE. UN MECANISMO DE REVOCACIÓN EN CADENAS DE DELEGACIÓN

En el capítulo 3 se ha presentado una arquitectura que permite realizar el proceso básico de validación de certificados en entornos donde puede existir desconexión a los servicios PKIX centralizados. Sin embargo, no se ha tenido en cuenta la validación de cadenas de certificados generadas por el proceso de delegación. A pesar que la delegación proporciona gran flexibilidad al sistema, también aporta complejidad. En este capítulo se expone el problema de la carga que supone la validación de la revocación en cadenas de delegación, y se presenta una propuesta para reducir la carga sobre el verificador de privilegios tanto en modo conectado como en modo desconectado.

4.1. Introducción

El modelo de delegación de la PMI gestiona una situación en la que es necesaria la delegación de privilegios (ver apartado 2.3.2). En este modelo, el delegante (entidad que realiza la delegación) expide un nuevo certificado conteniendo los privilegios delegados a la entidad que recibe la delegación. Esta entidad se conoce con el nombre de delegado. De esta manera, los certificados facilitan la delegación distribuida. Sin embargo, largos caminos de delegación implican largas cadenas de certificados. Vamos a ilustrar este efecto con un sencillo ejemplo con cuatro usuarios: Alicia, Benito, Carla y David. Supongamos que Alicia tiene algunos privilegios para acceder a un recurso cuyo propietario es David. Estos privilegios están incluidos en un certificado expedido por David (D) a Alicia (A) el cual puede ser representado como $C_{D \rightarrowtail A}$. Imaginemos que Alicia quiere delegar todos o parte de sus privilegios a Benito (B). Benito, a su vez,

también quiere delegar algunos de estos privilegios a Carla (C). Para llevar a cabo esta delegación ($A \mapsto B \mapsto C$), Alicia expide un certificado a Benito para los privilegios delegados ($C_{A \mapsto B}$) y Benito expide un certificado similar a Carla ($C_{B \mapsto C}$). Por lo tanto, el proceso de delegación genera una cadena de certificados ($C_{D \mapsto A} \mapsto C_{A \mapsto B} \mapsto C_{B \mapsto C}$). Ahora, imaginemos que Carla está intentando acceder a un recurso propiedad de María. Para ello, Carla presenta su certificado ($C_{B \mapsto C}$) a María. Entonces, María debe comprobar si los privilegios presentados por Carla son o no son suficientes para permitir el acceso a los recursos solicitados.

En el caso en el que los privilegios son suficientes, María también debe verificar la cadena de certificados asociada. Este proceso implica que María tiene que construir el camino de delegación desde Carla hasta la fuente de la delegación (Alicia, en este caso) y comprobar el estado (válido/revocado) de cada certificado asociado. La construcción y validación de caminos de delegación son procesos que consumen recursos, los cuales son especialmente críticos para dispositivos limitados, tales como PDAs, teléfonos móviles, etc.

4.2. Atributos

Llamamos atributo a un privilegio o derecho que posee un usuario y que, de alguna manera, le permite acceder a un recurso determinado. Los atributos pueden ser delegados, es decir, un delegante puede conceder o transferir atributos a un delegado. Se considera que un atributo es delegable si éste puede ser cedido o transferido con éxito de una entidad a otra. Por último, el proveedor de servicios es la entidad que controla y proporciona los servicios/recursos a los delegados.

4.2.1. Delegación de Atributos

En la literatura existen diversas clasificaciones sobre la delegación (69–71). La tabla 4.1 resume los principales tipos de delegación de acuerdo con estas clasificaciones.

Tabla 4.1: Clasificaciones de la delegación

Tipo	*administrativa*	*ad-hoc*
Duración	*semi-permanente*	*temporal*
Transacción	*directa (single-step)*	*indirecta (multi-step)*
Categoría	*transferencia*	*concesión*

En (71) y (72) se definen dos tipos de delegación de atributos en los que se tiene en cuenta el tipo de entidades involucradas en el proceso de delegación. La delegación administrativa es la forma básica de delegación en la que una autoridad de autorización asigna atributos a los usuarios. La capacidad de asignar un atributo no significa necesariamente que una autoridad de autorización realmente posea la capacidad de

utilizar el atributo que delega. Por otro lado, la delegación *ad-hoc* se produce cuando los usuarios delegan atributos sin la intervención de ninguna autoridad administrativa. En la delegación *ad-hoc*, el delegante debe tener la capacidad de utilizar los atributos que va a delegar. Por lo general, este tipo de delegación se utiliza en sistemas altamente distribuidos, mientras que la delegación administrativa se utiliza en escenarios más centralizados.

De acuerdo a un criterio temporal, la duración de la delegación se clasifica normalmente como semi-permanente (larga duración) o temporal (corta duración). Otra de las características importantes para clasificar los sistemas de delegación es si se permiten delegaciones posteriores una vez se delega un atributo a un usuario. Esta característica define si un atributo se puede o no se puede delegar de nuevo por el usuario que recibió el atributo, así como la profundidad máxima de la ruta de la delegación (es decir, el número máximo permitido de delegaciones). En este sentido, la delegación directa no permite que un atributo delegado se pueda delegar de nuevo (es decir, la profundidad máxima de la delegación es uno). Por el contrario, la delegación indirecta permite que un atributo delegado se pueda delegar de nuevo por parte de la entidad que recibió el atributo.

Por último, de acuerdo a (73) y (71), la delegación también se puede clasificar en dos categorías: por concesión o asignación, y por transferencia. En la delegación por asignación, el delegante y el delegado comparten (poseen) el atributo delegado. Por consiguiente, tanto el delegante como el delegado pueden utilizar el atributo compartido, al mismo tiempo. En el caso de la delegación por transferencia, la capacidad de utilizar los atributos delegados se transfiere al delegado, por lo que a partir de ese momento los atributos dejan de estar disponibles para el delegante. Este requisito hace que la ejecución de la política de transferencia de delegación sea difícil de implementar, ya que se ha de introducir un mecanismo para prevenir el uso de los atributos delegados por el delegante.

4.2.2. Revocación de Atributos

Un usuario podría querer cancelar un atributo concedido/transferido anteriormente a un delegado. Este proceso se conoce como revocación de la delegación. La tabla 4.2 muestra un resumen de los principales tipos de revocación de atributos que existen en la literatura y que se definen de forma breve a continuación.

Tabla 4.2: Clasificaciones de la revocación de atributos

Concesión	*dependiente*	*independiente*
Recuperación	*privilegio eliminado*	*privilegio negativo*
Dominio	*revocación fuerte*	*revocación débil*
Propagación	*cascada*	*no-cascada*

Los autores en (73) proponen una clasificación para la revocación de la delegación basada en la legitimidad de un usuario a revocar un atributo delegado. La legitimidad se divide en dos categorías: revocación dependiente y revocación independiente. En la revocación dependiente, el delegante es el único que puede revocar el atributo delegado a la entidad a la cual se le delegó. En la revocación independiente cualquier entidad que sea origen de un determinado atributo puede revocar el atributo delegado a cualquier entidad que recibió ese atributo.

Otro aspecto importante a considerar es cómo afecta la revocación al resto de los atributos delegados. En (74) se realiza una clasificación de la revocación en tres tipos: recuperación, dominio y propagación. La primera de ellas (recuperación) hace referencia a la persistencia de la revocación en el tiempo. Según este parámetro, hay dos tipos de revocación: privilegio eliminado y privilegio negativo. El esquema de privilegio eliminado es el esquema más comúnmente utilizado. En este caso, los privilegios se eliminan, pero se pueden volver a conceder más tarde. La concesión de privilegios negativos significa que un privilegio negativo anula a un privilegio positivo hasta que el privilegio negativo se elimine.

Por otro lado, el dominio de la revocación hace referencia al efecto de la revocación sobre el conjunto de atributos del usuario. Si se lleva a cabo una revocación fuerte del privilegio de un usuario, entonces se elimina tanto el atributo a la entidad con delegación directa como el mismo privilegio concedido en el procedimiento de delegación indirecta. Una revocación débil sólo elimina el atributo delegado del usuario (miembro explícito) y deja intactos el resto de atributos. La revocación fuerte es teóricamente equivalente a una sucesión de revocaciones débiles.

Por último, la propagación se refiere al efecto que tiene la revocación sobre otros delegados. Una revocación en cascada, no sólo revoca la delegación realizada a un delegado, sino también el conjunto de delegaciones subsiguientes realizadas a partir de la delegación que se está revocando. Una revocación no en cascada sólo revoca la asignación realizada de forma directa sobre un delegado.

4.3. Infraestructuras para Gestión de Atributos

En la literatura se pueden encontrar distintas propuestas basadas en certificados para la gestión de atributos. Como ejemplos más relevantes se pueden citar SESAME (24), Akenti (25) y SPKI/SDSI (75), las cuales ya fueron introducidas en el capítulo 2. Estas propuestas utilizan formatos particulares para representar certificados de atributo y diferentes soluciones para llevar a cabo la delegación y revocación de atributos.

Como se muestra en la tabla 4.3, cada propuesta utiliza distintos formatos tanto para la representación como para la codificación de ACs. Por el contrario, la PMI utiliza un formato estándar tanto para la definición (ASN.1 - *Abstract Syntax Notation*

Tabla 4.3: Formato de los certificados de atributo y su infraestructura de gestión

	Formato AC	**Codificación AC**	**Delegación**	**Revocación**
PMI	**AC X.509**	**ASN.1**	**ad-hoc/DS**	**corta/larga duración**
SDSI/SPKI	S-expression	XML	parcialmente controlada	corta/larga duración
SESAME	PAC	ECMA	sólo delegación	corta duración
Akenti	Proprietario	XML	Política	corta duración
PERMIS	AC X.509	ASN.1	DIS	un uso

One) como para la codificación (DER - *Distinguished Encoding Rules*). PMI tiene dos ventajas principales respecto a los otros enfoques. En primer lugar, ACs X.509 son fácilmente integrables con otros protocolos o sistemas. En particular, la integración de los certificados de atributo X.509 con PKIs existentes, las cuales han sido ampliamente desarrolladas, requieren cambios mínimos para adaptarlos a los protocolos existentes de Internet, tales como S/MIME (76), IPSEC (77), SSL/TLS (78), etc. De hecho, también existe una versión extendida de OpenSSL que soporta AC X.509 y que permite que sean utilizados directamente en dispositivos limitados, tales como PDAs (79). En este sentido, los ACs X.509 son adecuados para entornos con dispositivos limitados. En segundo lugar, los ACs X.509, están codificados en ASN.1 DER (80). Esta codificación presenta mejores prestaciones que XML, ya que requieren menor espacio de almacenamiento, y por lo tanto, menor tiempo para su transmisión. Además, tanto la generación como la validación de la firma son también procesos considerablemente más rápidos que en XML (81).

En cuanto a la delegación y la revocación de atributos, SESAME (24) y Akenti (25) permiten que la delegación se realice para periodos cortos de tiempo, de manera que la revocación se hace innecesaria. Los certificados de corta duración se utilizan normalmente cuando el periodo de validez del AC es menor que el tiempo necesario para emitir y distribuir los datos de revocación. Por otra parte, SPKI/SDSI (75) considera la posibilidad de expedición de ACs de larga duración, para el que se propone un mecanismo de revocación explícita en el que los certificados revocables incluyen información para acceder a la entidad que se encarga de la gestión de revocación. En cuanto a la delegación, SPKI/SDSI utiliza un valor booleano para indicar si el poseedor del certificado puede o no propagar la autorización. Podemos considerar este tipo de delegación como parcialmente controlada (apartado 4.2.2).

Para concluir, puede observarse que la delegación basada en certificados de corta duración hace innecesaria la revocación de los mismos, a expensas de aumentar la carga sobre los emisores de los certificados; los certificados tienen que ser renovados con más frecuencia, ya que expiran con más rapidez. Por otra parte, la estimación del periodo de validez es un problema bien conocido para los certificados de corta duración (39; 82). Si se utilizan periodos de validez demasiado cortos, la frecuencia

de renovación del certificado puede ser innecesariamente elevada, mientras que si se utilizan periodos de validez demasiado largos, el riesgo en el sistema aumenta, ya que los certificados son válidos durante un largo periodo de tiempo sin la posibilidad de que puedan ser revocados.

4.4. Revocación en cadenas de delegación

La revocación de certificados es un proceso complejo que ha sido objeto de estudio desde hace tiempo. Sin embargo, la mayor atención de la revocación siempre se ha centrado bajo el dominio de una Autoridad de Certificación en el que la longitud del camino de certificados se ha limitado a 1 (83–86). Es decir, se han centrado en el estudio de la revocación sobre un conjunto de certificados que han sido expedidos por una misma entidad (delegación directa).

En los siguientes apartados se va a realizar un estudio sobre las implicaciones de la validación del estado de revocación de los certificados que forman una cadena de certificados y cómo afecta a las prestaciones de los sistemas tanto de validación como de gestión de la revocación de certificados.

4.4.1. Descripción del escenario

En el escenario en el cual se centra el estudio, las relaciones entre los ACs (delegaciones) pueden ser representadas como un grafo acíclico, es decir, utilizando un árbol como el de la figura 4.1. Por lo tanto, se considera un modelo jerárquico de delegación que se representa mediante un árbol de delegación, el cual denominamos como $\mathcal{T}_d$. Cada uno de los elementos y relaciones representadas en $\mathcal{T}_d$, se definen como:

- La SOA es la raíz del árbol.
- Los nodos intermedios representan entidades que pueden delegar sus privilegios a otras entidades. A estas entidades las denominamos como AA_x.
- Las hojas del árbol representan entidades finales las cuales no pueden delegar sus privilegios. A estas entidades las denominamos como EE_x.
- Las flechas entre entidades del árbol representan relaciones de confianza que se han establecido por el proceso de delegación. Por ejemplo, en la figura $AA_1 \rightarrow AA_4$ significa una relación de confianza directa en la cual AA_1 delega alguno de sus privilegios a AA_4.
- Una relación de confianza indirecta se define como una relación en la cual existe, al menos, una entidad intermedia. En la figura, AA_1 y AA_7 tienen una relación de confianza indirecta ($AA_1 \rightarrow AA_4 \rightarrow AA_7$).

- El dominio de delegación de una determinada entidad AA_x ($\mathscr{D}_{AA_x}$) es el conjunto de entidades que tienen una relación de confianza directa o indirecta con AA_x. Por ejemplo, en la figura, $\mathscr{D}_{AA_4}$ está formado por $\{AA_7, AA_8, AA_9, EE_1, EE_2\}$.
- Un camino de delegación (DP - *Delegation Path*) asociado a un determinado nodo, es la secuencia ordenada inversa de certificados $DP_n = \{AC_n, AC_{n-1}, ..., AC_1\}$, donde:
  - El último certificado de la secuencia (AC_1) corresponde a la SOA.
  - El primer certificado de la secuencia es el certificado del nodo objeto (AC_n).
  - Los certificados intermedios representan autoridades de atributo intermedias. Por lo tanto, el propietario en un certificado es el expedidor en el siguiente certificado de la secuencia (ver apartado 2.12).
  - La longitud del camino de delegación es n.

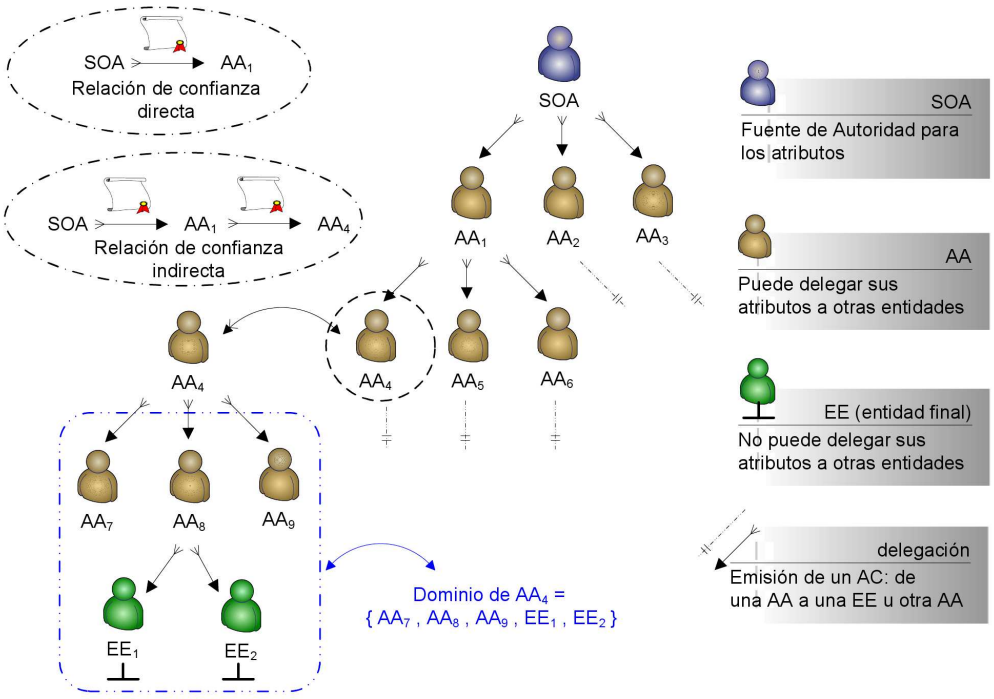

Figura 4.1: Representación del árbol de delegación ($\mathscr{T}_d$) bajo una fuente de autoridad

El hecho de que en nuestro estudio se necesite asegurar una topología en árbol con respecto a las relaciones de confianza, podría parecer a primera vista una fuerte restricción para el mecanismo de delegación. En realidad, es una hipótesis razonable que también ha sido adoptada por otros autores, como Chadwick[1].

[1] Como se indica en (87) los árboles de delegación simplifican significativamente el proceso de validación y revocación de credenciales, ya que cada credencial sólo tiene una autoridad que le delega.

Analicemos qué implicaciones tiene utilizar esta topología de delegación, y en particular, qué impacto puede tener esta restricción sobre la flexibilidad del modelo de delegación. En primer lugar, es necesario mencionar que cualquier grafo de delegación ha de ser un grafo acíclico. Esto significa que un delegante no debería poder delegarse a sí mismo o a un predecesor. Es decir, un delegado nunca debería necesitar delegar a una entidad que le ha delegado anteriormente, ya bien sea directa o indirectamente. De hecho, hay una razón de seguridad para esta restricción, ya que puede haber una brecha potencial de seguridad si un delegante, el cual puede delegar un privilegio pero no utilizarlo, se delega así mismo ese privilegio de manera que se le permita utilizar ese privilegio.

La pregunta aquí es qué significa restringir un grafo acíclico para que sea un árbol. La respuesta es que esta restricción significa evitar juntar atributos que provienen de delegantes diferentes en un único certificado. Obviamente, la siguiente pregunta que surge es qué impacto tiene esta restricción en la flexibilidad del mecanismo de delegación. Un árbol de delegación permite a los delegados recibir atributos de dos o más delegantes en procesos de delegación distintos, es decir, en diferentes certificados de atributo. Como certificados diferentes se representan como nodos y relaciones diferentes en el grafo de delegación, el grafo resultante todavía es un árbol.

Por lo tanto, para llevar a cabo una determinada tarea, el delegado ha de presentar al verificador de privilegios, el AC o conjunto de ACs adecuados. Estos ACs podrían haber sido expedidos por delegantes diferentes e incluso por delegantes que dependen de diferentes SOAs (diferentes árboles de delegación). El verificador de privilegios aplica la política de validación de credenciales, y obtiene como resultado, el conjunto de atributos del delegado. Por último, con el conjunto de atributos válidos, el verificador de privilegios concede/deniega el acceso para realizar las tareas necesarias.

La aplicación de una topología en árbol implica situar la mayor parte de la política de validación en el verificador de privilegios, lo que simplifica el comportamiento de los delegantes. Esto es debido a que en la topología en árbol, los delegantes (AAs) no tienen porqué trabajar de forma coordinada para cumplir con la política de validación. En su lugar, los delegantes pueden trabajar de forma independiente, y el verificador es quien aplica gran parte de la política de validación. Esto tiene sentido ya que el verificador de privilegios es la entidad última en decidir si permite realizar una determinada tarea a un usuario. Como conclusión, podemos decir que el árbol de delegación es una topología simple, flexible y adecuada para la mayoría de escenarios en los que se requiere delegación dinámica.

4.4.2. Validación de caminos de delegación

Cuando el usuario intenta acceder a un recurso, el verificador de privilegios ha de validar el correspondiente camino de delegación del AC del usuario. El verificador de privilegios puede obtener el AC de dos maneras: 1) el usuario ha presentado el AC (modelo *push*), o 2) el AC ha de ser obtenido de un repositorio (modelo *pull*). La validación del camino de delegación consta de dos procesos básicos:

- **Verificación del grafo de delegación**. La validación de la política de delegación está relacionada con la cadena de delegación e implica verificar la política de privilegios (teniendo en cuenta las variables de entorno), firmas digitales, relaciones jerárquicas entre los certificados, etc.
- **Verificación del estado de revocación**. La verificación del estado está relacionada con el estado de revocación de cada certificado en particular, e implica verificar la lista de certificados revocados (ACRL - *Attribute Certificate Revocation List*).

Esencialmente, el verificador de privilegios puede llevar a cabo la validación de un camino completo de delegación de dos formas diferentes:

(a) El verificador obtiene la cadena de certificados asociada al AC. A continuación, realiza la validación de la política de delegación para todos los certificados de la cadena. Si la delegación es correcta, se obtiene la ACRL y comprueba el estado de revocación del primer certificado de la cadena. Si el certificado no está revocado, el verificador comprueba el siguiente certificado en la cadena, y así sucesivamente. El verificador termina el proceso cuando se encuentra un certificado revocado o llega al final de la cadena.

(b) El verificador valida la política de delegación para el primer certificado de la cadena. Si este AC es correcto, entonces el verificador obtiene la ACRL y comprueba su estado de revocación. Si el certificado no está revocado, el verificador repite los siguientes pasos para cada certificado: obtener el siguiente certificado de la cadena, validar su política de delegación y comprobar su estado de revocación. El verificador sale del proceso de validación si algún paso falla, o cuando se llega al final de la cadena de delegación.

En general, (b) es más eficiente que (a) porque las cadenas de delegación ya establecidas tienen menor probabilidad de cambiar, en comparación al estado de revocación de los certificados que la forman. Si estudiamos este proceso nos damos cuenta de que el número de validaciones necesario para verificar el estado de revocación de la cadena completa dependerá de la posición en la que se encuentre el primer certificado revocado de la cadena. De hecho, cualquier cadena que contenga como mínimo

un certificado revocado, se considera no válida. Esta idea es similar a la revocación en cascada presentada en el apartado 4.2.2. Si en el momento de revocar un certificado X se pudieran revocar los certificados bajo su dominio $\mathscr{D}_X$, el verificador de privilegios sólo debería comprobar el estado de revocación del primer certificado de la cadena.

Por lo tanto, la revocación en cascada podría reducir la carga que supone la verificación del estado de revocación en cadenas de delegación, y por consiguiente, la carga total de validación de cadenas de certificados. A continuación se presenta una propuesta para poder llevar a cabo la revocación en cascada.

4.5. Un sistema administrativo de revocación en cascada

4.5.1. Introducción

Como se explicó en el apartado 4.2.2, la revocación en cascada permite que la revocación de un determinado privilegio a una entidad, conlleve la revocación del mismo privilegio delegado por esa entidad a las entidades que lo recibieron en el proceso de delegación, tanto de forma directa como indirecta. Si mapeamos este proceso al caso de certificados, tendríamos una situación en la que dada la revocación de un determinado atributo contenido en un certificado de atributo AC_X de una entidad X, provocaría que se revocaran los certificados de atributo bajo el dominio de esa entidad $\mathscr{D}_X$. Por lo tanto, si se revoca un certificado, todos los certificados expedidos a partir de él, dejan de ser válidos de forma explícita. La pregunta es, ¿cómo afecta esta característica al sistema?

Decimos que un certificado X está revocado de forma explícita cuando se ha recibido una petición de revocación de forma explícita para el certificado X. Por otra parte, decimos que un certificado X está revocado de forma implícita cuando no se ha recibido una petición de revocación para ese certificado, sin embargo, alguno de los certificados que forma la cadena de delegación asociada DP_X sí ha sido revocado mediante la solicitud expresa por la entidad correspondiente. En este apartado nos centraremos en el proceso de validación especificado para PKIX, intentando aprovechar la característica definida como revocación implícita para reducir la carga que supone esa validación sobre el verificador de privilegios.

4.5.2. Requisitos

La revocación en cascada requiere de un conocimiento del árbol de delegación. Es decir, se debería poder conocer las relaciones jerárquicas entre los diferentes certificados para poder revocar de forma explícita los certificados bajo el dominio de un AC que está siendo revocado. Por lo tanto, para poder llevarlo a cabo, se requiere:

R1 Conocer los certificados expedidos por cada autoridad de atributo.

R2 Introducir el menor número de cambios en los componentes de la PMI para compatibilizar la nueva solución con las soluciones existentes.

4.5.3. Arquitectura

En este apartado se presenta CARE (*CAscade REvocation*), un sistema que permite llevar a cabo la revocación en cascada a bajo coste para el verificador. CARE está basado en una entidad centralizada que tiene un conocimiento global de los certificados emitidos bajo el dominio de una SOA. Este conocimiento permitirá saber qué certificados están revocados de forma implícita (requisito R1). Por lo tanto, será suficiente con la verificación del estado de revocación de un certificado X para comprobar el estado de revocación del camino de delegación asociado DP_X. Este mecanismo ayudará a reducir la carga que supone la verificación de cadenas de certificados. Además, permitirá resolver varios problemas, tales como: evitar la retención de privilegios no autorizados y tener en cuenta la revocación transitiva y selectiva (88). Estas tareas se llevan a cabo a través del seguimiento de las cadenas de certificados, como se explicará más adelante.

A continuación, se presenta la arquitectura en la que se enmarca CARE. La figura 4.2 muestra los seis posibles elementos que integran una infraestructura de gestión de privilegios basada en certificados de atributo X.509. Esta infraestructura no ha de modificarse excesivamente con respecto al sistema introducido tanto en el capítulo 2 como en la ePKI presentada en el capítulo 3. El número de modificaciones dependerá del lugar en el cual se implementen las nuevas funcionalidades para lograr los requisitos definidos en el apartado anterior. Este punto se analizará con más detalle en el apartado de análisis 4.5.5.

A continuación, se explican las funcionalidades de cada módulo:

AA/SOA .- Entidad responsable de la gestión de certificados de atributo: expedición, renovación, revocación y publicación de información relacionada con los servicios anteriores.

Usuario .- Módulo que representa la entidad que intenta tener acceso a los recursos presentando un AC. Esta entidad puede revocar sus certificados.

RIS-AC (Servidor de Información de Revocación – AC) .- Módulo responsable de recibir las solicitudes de revocación, verificar la información y crear la estructura que contiene la información de estado de revocación de los certificados, como puede ser: ACRL, OCSP, etc.

RIS-PKC (Servidor de Información de Revocación – PKC) .- Las funcionalidades de este módulo son parecidas a las del módulo *RIS-AC*. Sin embargo, gestiona los certificados de clave pública en lugar de los certificados de atributo.

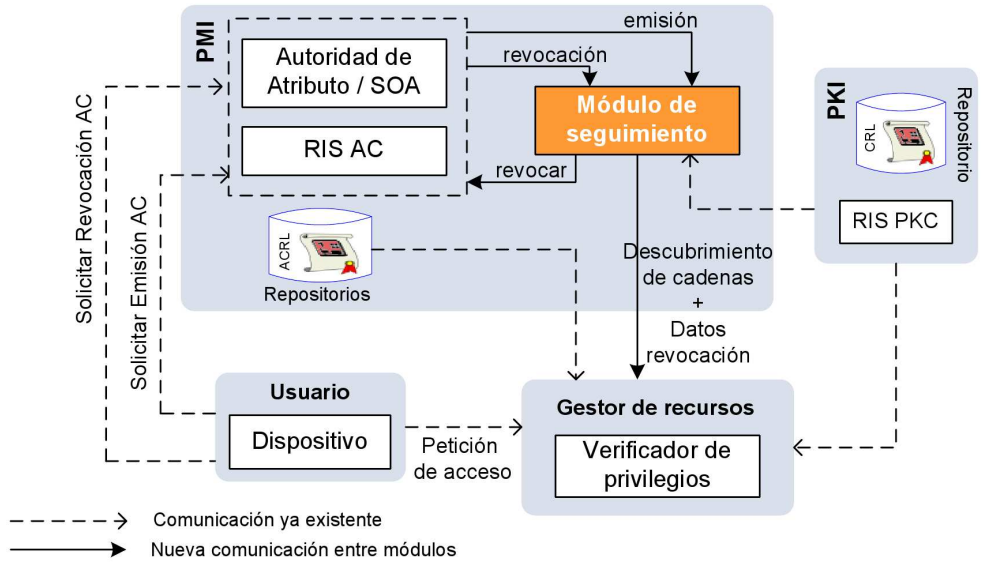

Figura 4.2: Integración del módulo de seguimiento dentro de la arquitectura completa para un sistema de revocación dentro de una infraestructura PMI

Módulo de seguimiento .- Entidad encargada del seguimiento del estado de revocación de los certificados de atributo. Su tarea principal es obtener información sobre el estado de los diferentes certificados: 1) los certificados de atributo que forman la cadena de delegación, y 2) los certificados incluidos en la cadena de certificación, necesarios para verificar la firma de cada AC en el camino de delegación. Para llevar a cabo esta tarea, el módulo de seguimiento realiza un seguimiento de las cadenas de certificados. Además, el módulo actualiza el efecto transitivo y selectivo sobre las cadenas de delegación. Este es el módulo principal del sistema CARE.

Verificador de privilegios .- Módulo encargado de controlar el acceso a los recursos. Para llevar a cabo esta tarea, el módulo verifica los privilegios del usuario contenidos en los certificados de atributo. El módulo obtiene la información necesaria de los diferentes repositorios para tomar una decisión. Además, el PV podría realizar tareas de apoyo al módulo de seguimiento. Su diseño e implementación no varía al presentado en la propuesta de la ePKI en el capítulo 3.

4.5.4. Funcionamiento

Para entender tanto los procedimientos como las entidades que se ven involucradas en cada uno de ellos, a continuación se explica con más detalle su funcionamiento (ver figura 4.2).

4.5.4.1. Expedición de ACs

El módulo de seguimiento interviene en el sistema en el momento en el que tiene lugar la expedición de un certificado, ya que ha de mantener un árbol de delegación $\mathcal{T}_d$. Una AA puede expedir un certificado de dos maneras: por propia iniciativa de la AA, o por solicitud directa de una entidad. En el primer caso, el procedimiento es interno, y el certificado generado se puede almacenar en un directorio público al cual puede acceder tanto la entidad a la cual se expidió el certificado como el verificador de privilegios. En el segundo caso, una entidad solicita la delegación de unos atributos a una determinada AA. La entidad puede enviar un mensaje de petición de expedición de un certificado de atributo generando una estructura como la especificada en (89). La AA verifica la información y expide el AC al delegado. El AC generado puede, o bien almacenarse en un directorio público como en el caso anterior, o enviarse al delegado. Este procedimiento no sufre ninguna modificación con respecto al estándar. Sin embargo, como se está contemplando revocación, la extensión `noRevAvail` no debería aparecer en el certificado (27; 51).

El módulo de seguimiento recibe información acerca de la delegación realizada por entidades autorizadas. Esta información se utiliza para construir el árbol de delegación $\mathcal{T}_d$ (ver figura 4.3). La información relativa a cada certificado de atributo expedido se almacena en un nodo. Por ejemplo, en la figura 4.3, los nodos 4, 5 y 6, representan los certificados expedidos a partir del AC representado por el nodo 3. El seguimiento se realiza sobre los nodos que representan el camino de delegación, desde el nodo final hasta el nodo que representa la SOA. Cuando una AA delega cualquier privilegio a un usuario, la AA debe verificar la identidad del usuario a través del PKC del usuario. Esta información puede ser almacenada para ayudar al procedimiento de seguimiento de las cadenas de certificados de identidad asociadas a los ACs emitidos.

4.5.4.2. Actualización del árbol de delegación

Una vez se ha incluido la información de un AC dentro de $\mathcal{T}_d$, el módulo de seguimiento ha de actualizar $\mathcal{T}_d$ cuando se producen otros dos eventos:

1. La revocación de un certificado previamente expedido. En este caso se han de podar las ramas de $\mathcal{T}_d$ siguiendo una política de revocación en cascada. Es decir, cuando se revoca un certificado, se busca en el árbol el nodo que representa el certificado revocado (supongamos que es el nodo 3). Tanto el certificado revocado (nodo 3) como los nodos dependientes (4, 5, 6 y 7) se eliminan del árbol. En este caso, los nodos dependientes son representados por las hojas cuya rama origen se encuentra en el nodo a eliminar (nodo 3). Los certificados revocados se pueden almacenar en un árbol de revocación ($\mathcal{T}_r$). Esta información es útil para la distribución de la información de revocación, como por ejemplo, a través de una ACRL.

2. La finalización del periodo de validez de un certificado válido. En este caso, se poda el sub-árbol cuya raíz es el certificado que ha expirado. En realidad esta poda no debería suponer la eliminación de más de un nodo cada vez, ya que el periodo de validez de un certificado expedido a partir de otro, no puede ser válido más allá del periodo de validez del certificado origen.

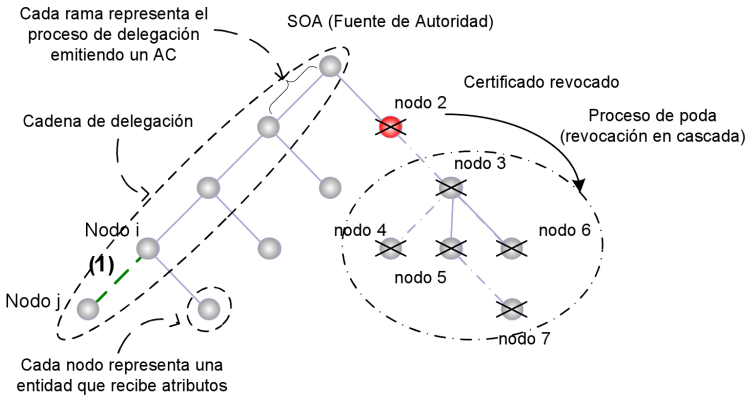

Figura 4.3: Árbol de delegación. Relaciones de confianza generadas en el proceso de delegación

4.5.4.3. Revocación de certificados

La revocación de un certificado conlleva la revocación de todos los certificados emitidos a partir de él. Para poder cumplir con el requisito R1, el módulo de seguimiento realiza tantas peticiones de revocación como certificados se vean afectados en una revocación en cascada, al módulo RIS-AC. Este hecho permite la coexistencia con entidades que no soporten el mecanismo propuesto. El proceso es directo y no requiere de conexiones a servidores externos si el nodo es interno a una AA. Si el módulo es utilizado como un mecanismo de revocación, no es necesario realizar procedimientos adicionales.

El módulo de seguimiento utiliza la información de $\mathcal{T}_d$, para conocer qué certificados se han de revocar, como se ha explicado en el apartado 4.5.4.2. Siguiendo el ejemplo de la figura 4.3, el módulo de seguimiento debe realizar las peticiones de revocación para los nodos (2,3,4,5,6,7).

4.5.4.4. Validación de ACs

Cuando un usuario necesita acceder a los recursos, envía una solicitud al verificador de privilegios. Éste verifica el certificado de atributo presentado por el usuario. El

verificador examina el AC del usuario buscando si contiene la extensión `noRevAvail` (ver apartado 2.3.3). Si no la contiene, busca la localización de la información de revocación utilizando la configuración local, o la extensión del certificado *CRL Distribution Points*. Si existe una ACRL disponible, el verificador obtiene la ACRL y verifica si el certificado está revocado. Este procedimiento se lleva a cabo para mantener la compatibilidad con los sistemas que no soporten revocación en cascada.

En el caso de que el AC se haya expedido en el sistema CARE, el AC contiene una extensión indicando que la revocación bajo la SOA se lleva a cabo en cascada, y por lo tanto, el verificador puede beneficiarse de este procedimiento. En este caso, el verificador solicita la información de revocación al módulo de seguimiento utilizando el siguiente procedimiento:

1. El módulo busca el certificado en el árbol de delegación intentando encontrar un camino de delegación para el certificado. Si el módulo no encuentra información sobre el certificado, el módulo sabe que se pueden presentar básicamente tres casos:

   - El certificado está revocado. En este caso el certificado ha de estar incluido en el árbol de revocación.
   - El certificado se ha emitido de forma fraudulenta.
   - La AA no ha informado de su delegación. En este caso el módulo devuelve un código de certificado desconocido que indica una situación inusual.

2. El certificado es válido si el módulo encuentra una entrada en el árbol. Si existe una entrada en el árbol para un determinado AC, implica que el certificado no está revocado, no ha expirado y es válido. Por lo tanto, el resto de los certificados implicados en la ruta no están revocados, todavía no han expirado y son válidos. Esto es así ya que el módulo de seguimiento comprueba la validez de un certificado antes de incluirlo en el árbol de delegación (no está revocado, no ha expirado y la firma es válida).

El sistema CARE puede proporcionar diferentes beneficios al verificador de privilegios, permitiendo un funcionamiento tanto en modo conectado como en modo desconectado para el proceso de validación. Es decir, en cada uno de los modos de funcionamiento, el sistema CARE permite:

- Modo conectado. CARE puede ser utilizado como servidor de validación, de forma similar a la especificación del IETF en el RFC 5055 (90). En este caso el verificador de privilegios puede saber con una única consulta, si el certificado y

toda la cadena asociada son o no válidos. Además, puede proporcionar la información necesaria para el descubrimiento de cadenas de certificados, el cual es complejo tanto en PMI como en PKI (91; 92).

- Modo desconectado. CARE puede distribuir la información de revocación utilizando una ACRL. De esta manera, el verificador puede trabajar en modo desconectado durante el periodo de validez de la ACRL. En este caso, la no existencia de un certificado en la ACRL, no implica que el AC sea válido. Por lo tanto, el verificador ha de validar la cadena de delegación.

4.5.5. Análisis de viabilidad

En este apartado se presenta un breve análisis sobre la viabilidad del sistema CARE, teniendo en cuenta aspectos de: coste de implementación, escalabilidad, existencia de entidades deshonestas y ubicación del módulo en la arquitectura. En el apartado 4.6 se realiza un análisis más orientado a las prestaciones del sistema en el verificador de privilegios.

Aspectos en el coste de implementación. Principalmente, se estudia el coste que supone la introducción de CARE en el sistema, desde dos puntos de vista:

- Volumen de información a ser almacenada por el módulo. Sólo es necesario almacenar una parte de la información del certificado, ya que el certificado completo se puede almacenar en un directorio LDAP. Básicamente, la información necesaria para cada nodo en el árbol es: el número de serie del certificado, el propietario del certificado y el campo *IssuerSerial* del PKC vinculado al AC.

- Actualización del árbol. El árbol se actualiza: 1) cuando una AA expide un nuevo AC, 2) una entidad autorizada revoca un AC, o 3) cuando el certificado ha finalizado su periodo de validez. La actualización del árbol depende del tamaño del árbol y el algoritmo de búsqueda de los árboles. Existen diferentes algoritmos de búsqueda en árbol, tales como: BFS (*Breath-First Search*), DFS (*Depth First Search*), DSL (*Depth-limited Search*), IDS (*Iterative Deepening Search*), etc. La complejidad en tiempo y en espacio es exponencial con la profundidad de un nodo. La complejidad en espacio de DFS es lineal con la longitud máxima de cualquier camino en el árbol, sin embargo, es exponencial en tiempo. Por otra parte, IDS es el algoritmo más utilizado cuando el espacio de búsqueda es grande y la profundidad de la solución no es conocida. Para reducir la complejidad, el sistema podría utilizar información adicional para buscar un determinado certificado. Por ejemplo, el AC podría incluir información sobre el camino de delegación. En este caso, cuando el módulo busca la información relacionada, sabe a priori el nivel por el cual empezar la búsqueda en el árbol. En este caso,

la complejidad tanto en tiempo como en espacio se reduce al máximo número de sucesores de cualquier nodo.

Escalabilidad. Hay que tener en cuenta que en muchas aplicaciones, los árboles serán razonablemente pequeños, ya que la PMI tiene un enfoque más local que la PKI. Sin embargo, cuando el número de nodos en el árbol aumenta, se han de considerar aspectos de escalabilidad. El problema puede reducirse mediante la división del árbol en diferentes dominios. Cada dominio está a cargo de los certificados de atributo bajo su autoridad. La figura 4.4 muestra la división del seguimiento de los certificados en diferentes dominios:

(1) El árbol principal está gestionado por la SOA o por una entidad autorizada.

(2) El nodo y las hojas representan las autoridades que gestionan su propio dominio. En la figura, el nodo (a) gestiona las delegaciones realizadas en el dominio 1.

(3) La revocación de certificados en un dominio sólo afecta a ese dominio. Es decir, la revocación no tiene efecto sobre los otros dominios.

(4) La revocación del certificado de una AA que administra un dominio, supone la revocación del árbol completo del dominio.

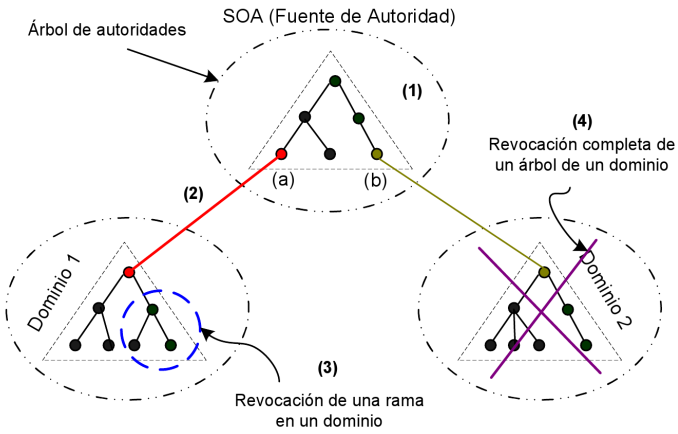

Figura 4.4: Gestión del árbol dividida en diferentes dominios. Cada dominio gestiona el conjunto de certificados de atributo bajo su dominio

Medidas contra entidades deshonestas. A la hora de construir el árbol con las cadenas de certificados, puede darse el caso que una entidad no comunique, intencionada o no intencionadamente, la expedición de un AC. En este caso, como la entrada no existe en el árbol, la revocación del certificado no puede llevarse a cabo. Sin embargo, el funcionamiento del sistema puede impedir que un usuario no válido entre en el sistema:

- El verificador solicita una cadena de certificados válida asociada al certificado presentado por el usuario. La entidad de seguimiento busca en el árbol una entrada para dicho certificado, o una entrada para el expedidor del mismo. Si existe una entrada para el expedidor, verifica la firma y actualiza el árbol con la nueva entrada, comprobando al mismo tiempo si el certificado debe ser revocado. Este caso se ajusta al escenario en el que una AA intermedia expide un certificado sin anunciarlo al sistema, como pueda ser por falta de conexión en este momento.

- Si no encuentra una cadena válida, el estado de revocación no se puede comprobar mediante la verificación del estado del último certificado. En este caso, se ha de denegar el acceso. Por lo tanto, un AC expedido de forma fraudulenta, no puede ser utilizado en el sistema sin ser detectado.

Ubicación del módulo de seguimiento dentro de la arquitectura. Otro de los factores a tener en cuenta es el de la localización del módulo de seguimiento dentro de la arquitectura. En función del escenario en el cual se desee implantar, pueden obtenerse diferentes beneficios. Para poder evaluar la conveniencia de ubicar el módulo en diferentes localizaciones, debemos tener en cuenta los parámetros que pueden verse afectados:

- Confianza necesaria sobre la nueva entidad. A mayor confianza sobre la información que proporciona, mayor riesgo de ser atacada. Por lo tanto, desde este punto de vista, es mejor situarlo dentro del módulo que gestione el mecanismo de revocación.

- Modificaciones a realizar en la arquitectura. Si el módulo de seguimiento es externo a la SOA, aumentan los nuevos establecimientos de comunicación, aunque no son necesarias las modificaciones sobre los mecanismos de revocación.

- Escalabilidad del sistema. A mayor número de certificados, mayor información a manejar por el módulo, por lo que sería mejor situarlo como elemento externo.

4.6. Análisis de prestaciones

En este apartado se proporciona un análisis estadístico para ilustrar las mejoras y/o inconvenientes que se producen cuando se utiliza CARE para llevar a cabo la revocación en cascada. El análisis compara CARE con un sistema de delegación en el que no se ha habilitado la revocación en cascada. Para realizar la comparación, se definen dos figuras de mérito: Δ_{CARE} y β_{CARE}. Estas figuras de mérito representan, respectivamente, el aumento en el número de entradas de una ACRL y la reducción del número

de certificados de la cadena de delegación que el verificador de privilegios ha de obtener y verificar. La tabla 4.4 muestra un resumen de las definiciones y parámetros que se utilizan en este apartado para el análisis.

Tabla 4.4: Definición de los parámetros para el análisis

Parámetro	Definición
β_{CARE}	Factor de reducción del número de certificados de la cadena a ser verificados
Δ_{CARE}	Incremento en el número de entradas de la ACRL
$\mathscr{C}_R$	Número medio de certificados revocados en cascada
C_R	Número medio de ACs revocados
N	Número de certificados actualmente expedidos y no expirados
p_c	Porcentaje de certificados revocados
$\mathscr{P}_c$	Porcentaje de certificados revocados en cascada
k	Longitud media de la cadena de delegación de un certificado en $\mathscr{T}_d$
L	Número máximo de niveles en un modelo de expedición jerárquico
M	Número medio de certificados expedidos por cada AA intermedia
C_{CARE}	Número medio de certificados de la cadena que se han de validar en CARE
C_{PKIX}	Número medio de certificados de la cadena que se han de validar en PKIX

4.6.1. Viabilidad de la distribución de la ACRL

Como ya se ha comentado en el apartado 4.5.3, a pesar de que la información de revocación se encuentre gestionada por el módulo de seguimiento, y las peticiones de revocación se realicen a él mismo, la información de revocación también se puede distribuir utilizando una ACRL. Para empezar el análisis, vamos a calcular el parámetro Δ_{CARE}, el cual proporciona una estimación del incremento en el número de entradas en la ACRL cuando se contempla la revocación en cascada con CARE, frente al número de entradas cuando no se utiliza revocación en cascada. Es decir, Δ_{CARE} representa el aumento en el número de certificados revocados cuando se considera revocación en cascada de forma explícita. Por lo tanto, Δ_{CARE} se define como:

$$\Delta_{\text{CARE}} = \mathscr{C}_R - C_R \tag{4.1}$$

donde $\mathscr{C}_R$ es el número medio de certificados revocados en cascada (revocados explícitamente y revocados implícitamente), y C_R representa el número medio de ACs revocados explícitamente. Debe notarse que siempre $C_R \leq \mathscr{C}_R$, y que Δ_{CARE} mide la diferencia entre estas dos magnitudes.

Un certificado revocado es un certificado para el que la SOA ha recibido y aprobado una solicitud de revocación. Supondremos que las solicitudes de revocación son independientes. Por otra parte, un camino de delegación se considerará revocado si cualquiera de sus certificados está revocado ya que asumimos revocación en cascada. También consideramos que el número medio de certificados revocados se puede

expresar como un porcentaje p_c de los certificados actualmente expedidos y no expirados N, es decir, $C_R = p_c \cdot N$. Además, el número medio de certificados revocados en cascada también se puede expresar como un porcentaje $\mathscr{P}_c$ de N, es decir, $\mathscr{C}_R = \mathscr{P}_c \cdot N$. Ambas expresiones están basadas en las mismas hipótesis que las utilizadas por otros autores (83–85). Por lo tanto, podemos reescribir la ecuación 4.1 de la siguiente manera:

$$\Delta_{\text{CARE}}(\%) = \mathscr{P}_c - p_c \tag{4.2}$$

la cual mide el incremento en tanto por ciento (%) de las entradas en la ACRL: $\Delta_{\text{CARE}} = 0$ significa un aumento del 0 % ($\mathscr{C}_R = C_R$) y $\Delta_{\text{CARE}} = 1$ significa un incremento del 100 %.

Ahora necesitamos una expresión para $\mathscr{P}_c$. Este parámetro representa el porcentaje de los certificados revocados en cascada que se incluirán en la ACRL. Un certificado está revocado de forma explícita con una probabilidad p_c. Por otra parte, la probabilidad de que un certificado esté revocado tanto de forma explícita como implícita, y por lo tanto tenga que incluirse en la ACRL, es la probabilidad de que alguno de sus predecesores o de que él mismo esté revocado. Si el número medio de certificados que forman la cadena de delegación de un AC seleccionado de forma aleatoria es k (longitud media de una cadena de delegación), la condición anterior puede expresarse como:

$$\mathscr{P}_c = 1 - (1 - p_c)^k \tag{4.3}$$

donde k dependerá de la longitud máxima del camino de delegación (L) y del número de certificados actualmente expedidos y todavía no expirados (N). Si consideramos un modelo de expedición jerárquico en el que cada AA expide un número medio de certificados M, k se puede expresar como:

$$k = \sum_{i=1}^{L} i \cdot p_i = \sum_{i=1}^{L} i \cdot (M^i / N) = \frac{1}{N} \cdot \sum_{i=1}^{L} i \cdot M^i = \frac{L \cdot M}{N-1} - \frac{1}{M-1} \tag{4.4}$$

donde p_i es la probabilidad de que un certificado elegido de forma aleatoria esté localizado en el nivel i-ésimo del árbol de delegación ($\mathscr{T}_d$). Es decir, la probabilidad de que la longitud del camino de delegación asociado a ese certificado sea i. La figura 4.5 muestra la evolución de la longitud media del camino de delegación (k) en función de la profundidad máxima del árbol y de la población de certificados. Como se puede observar, para valores bajos de L (hasta 4), la longitud media de la cadena a validar coincide con la longitud máxima. A medida que aumenta el valor de L, el valor de k no difiere demasiado del valor de L. Este hecho es debido a la distribución de los certificados dentro del árbol de delegación y a la suposición de que todos los AC tienen la misma probabilidad de ser validados por el verificador de privilegios[2].

[2] En el capítulo 5 se verán otro tipo de distribuciones utilizando una herramienta de validación de sistemas de revocación para el análisis.

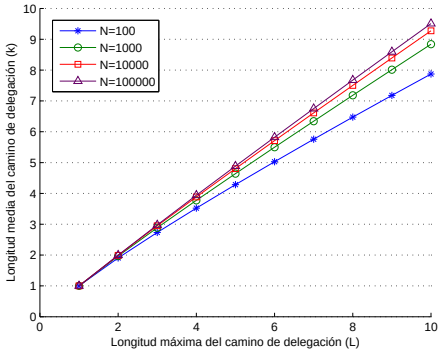

Figura 4.5: Longitud media del camino de delegación a validar por un verificador de privilegios

Finalmente, se obtiene la siguiente expresión para Δ_{CARE} que sólo depende de p_c, L y N:

$$\Delta_{\text{CARE}}(p_c, L, N) = (1 - p_c) - (1 - p_c)^{\,k} \quad \text{donde } k = f(L, N) \tag{4.5}$$

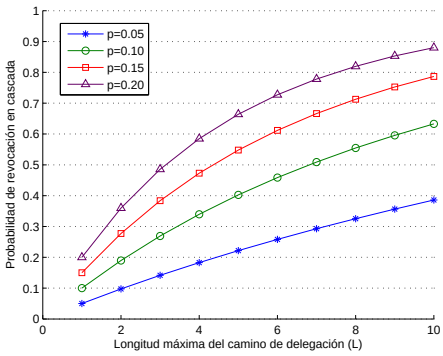

Figura 4.6: Probabilidad de revocación en cascada para diferentes valores de p_c

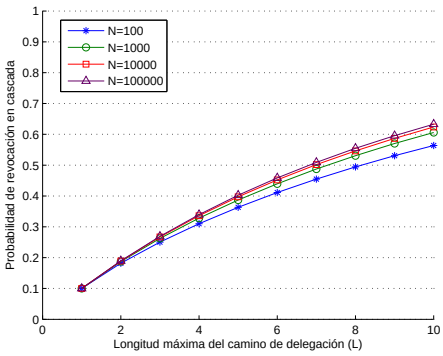

Figura 4.7: Probabilidad de revocación en cascada para diferentes valores de N

Como se puede observar en las figuras 4.6 y 4.7, la probabilidad de revocación en cascada aumenta con la longitud máxima del camino de delegación como era de esperar, mientras que la probabilidad de revocación p_c se mantiene constante. Esta diferencia hace que aumente el porcentaje de certificados revocados debido a CARE, y por lo tanto, el número de entradas en la ACRL (ver figuras 4.8 y 4.9). El valor máximo de Δ, y al cual tiende asintóticamente, es precisamente cuando se llega al 100 % de certificados revocados. Por lo tanto, el aumento del tamaño de la ACRL puede llegar a ser considerable. La distribución de la ACRL sería conveniente sólo en los casos en los que realmente sea necesario almacenar la información; por ejemplo, en el modo desconectado, o por un verificador con una elevada tasa de peticiones que compense el hecho de bajarse la ACRL una vez, frente al coste tanto de la comunicación constante

a un *responder* OCSP como de la validación de las respuestas. Por lo tanto, el aumento de la probabilidad de revocación explícita hace que en ciertos escenarios sea viable la distribución de la ACRL, pero en otros quizá es preferible, o bien reducir el periodo de validez de los certificados, o utilizar un mecanismo de consulta del estado de revocación como OCSP.

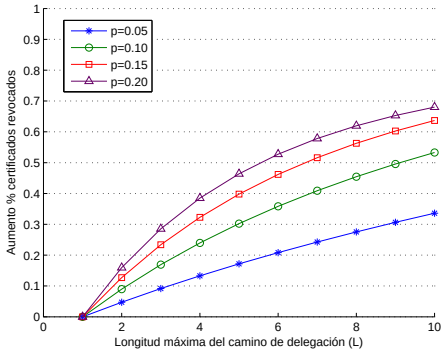

Figura 4.8: Aumento del % de certificados revocados en función de p_c

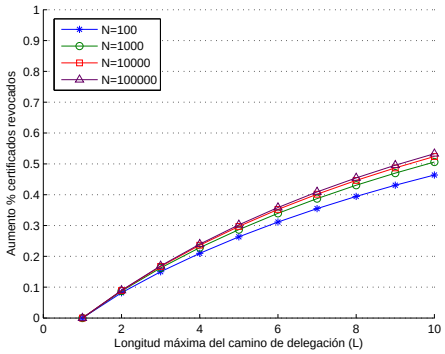

Figura 4.9: Aumento del % de certificados revocados en función de N

A la vista de los resultados, se puede pensar que el aumento elevado del número de certificados, hace que disminuya en la misma medida la población de certificados vigentes (no expirados y no revocados). Las figuras 4.10 y 4.11 muestran la evolución del porcentaje de certificados revocados ($\mathscr{P}_c$) y del porcentaje de certificados vigentes ($1-\mathscr{P}_c$). Por lo tanto, este estudio nos lleva a considerar los casos en los que es preferible la carga debido a emitir de nuevo los certificados, frente a la carga de revocación de los mismos. También podría utilizarse el estudio para estimar el periodo de validez adecuado para el escenario de implantación de la solución. Por ejemplo, en escenarios en los que se tenga una probabilidad de revocación explícita baja ($p_c = 0{,}05$) y el aumento en la probabilidad de revocación implícita debido a la cadena asociada no sea elevado ($\mathscr{P}_c = 0{,}2$), soportar un mecanismo de revocación puede ser viable.

La figura 4.12 muestra el tamaño de la ACRL que ha de distribuirse para diferentes valores de L y de p_c. El tamaño de la ACRL se ha estimado teniendo en cuenta los siguientes tamaños de cabecera (o_{crl}) y de tamaño de la información a representar por cada certificado revocado (l_{cr}): $l_{cr} = 48$ *bytes* por cada certificado revocado y $o_{crl} = 271$ *bytes* de la parte fija de la ACRL (emisor, número de ACRL, firma, etc) (93). Por lo tanto, el tamaño de la ACRL se puede expresar como:

$$ACRL_{size}(\mathscr{P}_c, N) = o_{crl} + l_{cr} \cdot N \cdot \mathscr{P}_c \tag{4.6}$$

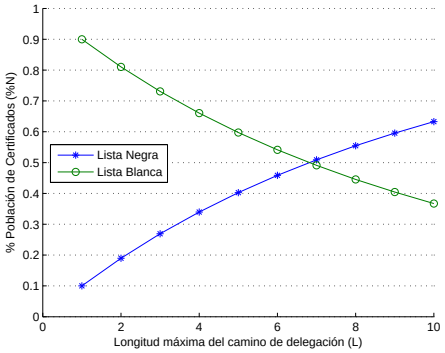

Figura 4.10: Evolución de certificados revocados ($\mathscr{P}_c$) y certificados vigentes ($1 - \mathscr{P}_c$) para $p_c = 0{,}1$

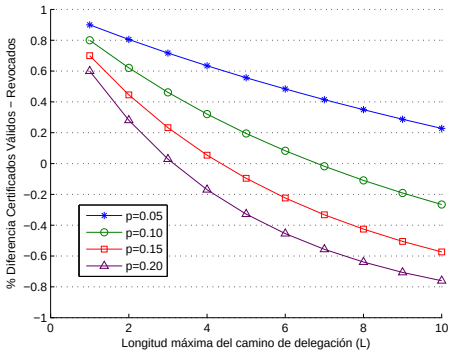

Figura 4.11: Diferencia entre certificados válidos ($\mathscr{P}_c$) y certificados vigentes ($1 - \mathscr{P}_c$)

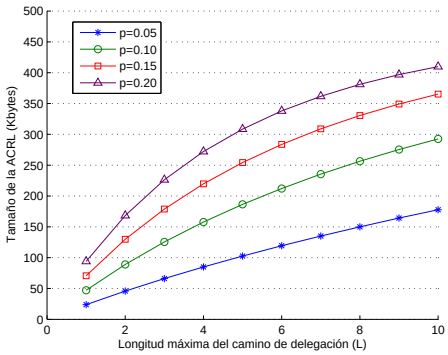

Figura 4.12: Tamaño de la ACRL para diferentes probabilidades de revocación ($N = 10000$)

4.6.2. Reducción del número de certificados necesarios

A continuación, se calcula la otra figura de mérito β_{CARE}. Este parámetro representa la reducción lograda por CARE sobre el número de certificados que se han de obtener (modo conectado) o validar (modo desconectado). Hay que recordar que lo mejor que puede hacer un verificador en el caso tradicional en el que no se utilice CARE, es obtener y verificar uno a uno los certificados de la cadena de delegación (ver apartado 4.4.2). Por el contrario, con un verificador con CARE se pueden dar dos casos:

- Modo conectado: se puede comprobar con una única operación el estado de validez de una cadena de delegación completa.
- Modo desconectado: se puede comprobar con una única operación el estado de revocación de una cadena de delegación completa.

En CARE, el único caso en que el verificador tiene que obtener los certificados de la cadena de delegación es cuando ninguno de los certificados está revocado. En

modo conectado, podría no necesitar la obtención de certificados, sin embargo, el verificador podría aplicar ciertas políticas de validación que necesitaran la validación de extensiones para escenarios particulares. Por lo tanto, se puede definir β_{CARE} como:

$$\beta_{\text{CARE}} = 1 - (C_{\text{CARE}} / C_{\text{PKIX}}) \tag{4.7}$$

donde C_{CARE} es el número medio de certificados de la cadena de delegación que se han de obtener cuando se utiliza CARE. C_{PKIX} es el número medio de certificados de la cadena de delegación que se han de validar cuando no se utiliza CARE. Suponemos que el propietario del privilegio ha presentado el AC al verificador durante el proceso de petición de acceso. Con CARE, el verificador sólo necesita obtener los $k-1$ certificados restantes si ninguno de los k certificados de la cadena está revocado, por lo tanto:

$$C_{\text{CARE}} = (k-1) \cdot (1-p_c)^k \tag{4.8}$$

Sin CARE, el verificador comprueba el primer certificado de la cadena, y si el certificado no está revocado, obtiene y comprueba el siguiente certificado, y así sucesivamente. El verificador finaliza el proceso si encuentra un certificado revocado, o se alcanza el final de la cadena. Es decir, se han verificado todos los certificados de la cadena, y no hay ninguno revocado. Así, el verificador tiene que obtener i certificados si los primeros i certificados de la cadena no están revocados y el certificado $i+1$ sí está revocado. En este último caso, el verificador tiene que obtener la cadena completa (los $k-1$ certificados restantes), si los primeros $k-1$ certificados de la cadena no están revocados. Estas condiciones pueden expresarse como:

$$C_{\text{PKIX}} = (k-1) \cdot (1-p_c)^{k-1} + \sum_{i=1}^{k-2} (i \cdot p_c \cdot (1-p_c)^i) = \frac{(1-p_c) - (1-p_c)^k}{p_c} \tag{4.9}$$

Que resulta en la siguiente expresión para β_{CARE}:

$$\beta_{\text{CARE}}(p_c, L, N) = 1 - [(k-1) \cdot p_c \cdot \frac{(1-p_c)^{k-1}}{1-(1-p_c)^{k-1}}] \quad \text{donde } k = f(L,N) \tag{4.10}$$

La figuras 4.13 y 4.14 presentan algunas gráficas de la expresión (4.10) para diferentes valores de la probabilidad de revocación, diferentes longitudes máximas del camino de delegación, y diferentes tamaños de población. Como se puede observar, CARE siempre tiene mejores prestaciones que los sistemas en los que no se utiliza revocación en cascada, llegando a alcanzar una mejora máxima en el comportamiento asintótico. El comportamiento asintótico se produce cuando $L \to \infty$ o $p_c \to 1$, es decir, CARE reduce el número de certificados de la cadena de delegación que se han de

obtener (o validar), a medida que aumenta la probabilidad de revocación de certificados y la longitud de la cadena de delegación. Como resultado, cuando la revocación en cascada se activa, se consigue nuestro objetivo, reducir el coste en la validación de cadenas de delegación, a costa de centralizar la información de delegación.

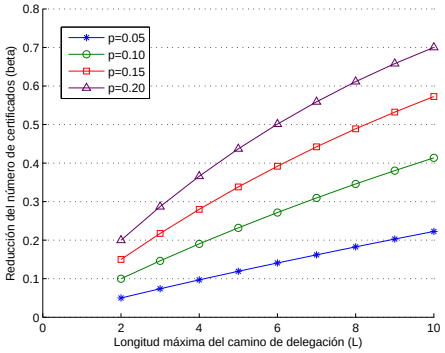

Figura 4.13: β_{CARE}- Reducción del número de certificados en función de p_c

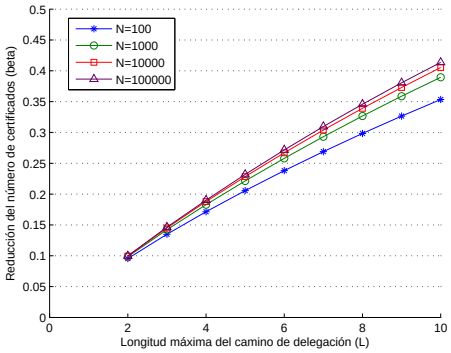

Figura 4.14: β_{CARE}- Reducción del número de certificados en función de N

Por ejemplo, si observamos la figura 4.13, podemos ver que para un escenario en el cual la longitud máxima del camino de delegación es 4 ($L = 4$) y la probabilidad de revocación es del 10 % ($p_c = 0{,}1$), CARE reduce en casi un 20 % el número de certificados de la cadena de delegación que se han de validar. A medida que aumenta L, las mejoras también aumentan, por ejemplo, cuando $L = 5$, la reducción es de un 23 % ($\beta_{\text{CARE}} = 0{,}23$).

4.6.3. Evaluación del tiempo de proceso

Como ya se ha comentado, en el caso de habilitar la revocación en cascada, si el último certificado no está revocado, hay que buscar la cadena completa, sino, no hay que obtener ningún otro certificado. Como hemos estimado en los apartados anteriores, el número medio de certificados a obtener en el caso PKIX viene determinado por la expresión 4.9, por lo que si consideramos que el tamaño medio de un certificado es de 1 KB (84), podemos estimar el tiempo necesario para validar los certificados en el caso de PKIX como:

$$t_{\text{PKIX}} = t_v \cdot C_{\text{PKIX}} \tag{4.11}$$

Por otra parte, en CARE, el número medio de certificados que se han de validar en el peor caso (modo desconectado) es 1 si el certificado está revocado, y k si ninguno de los certificados de la cadena de delegación está revocado. Por lo tanto, en este caso, el tiempo para validar los certificados se puede estimar como:

$$t_{\text{CARE}} = t_v \cdot C_{\text{CARE}} \tag{4.12}$$

Las figuras 4.15 y 4.16 representan la evolución del tiempo medio para la validación de diferentes cadenas de delegación en los que se ha fijado el valor de $N = 100000$, y se han variado los valores de L y p_c. Para el cálculo se ha implementado un *test-bed* sobre un teléfono móvil con procesador *Qualcomm* MSM7201A a 528 *MHz*. Con este dispositivo, se ha medido que el tiempo medio necesario para obtener un AC de un tamaño medio de $1KB$ utilizando una conexión *wireless* a un servidor externo, es de media $t_{AC} = 260ms$. Por otra parte, se ha obtenido que el tiempo medio de la validación básica de un certificado es de 200 *ms*, y se ha supuesto que la ACRL está en caché.

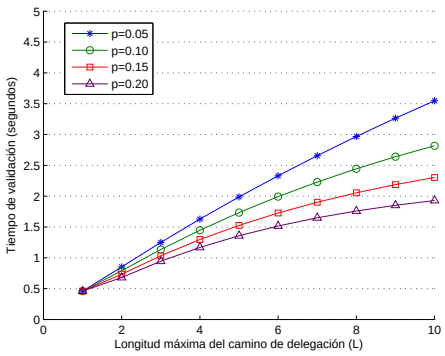

Figura 4.15: Tiempo medio de validación utilizando PKIX

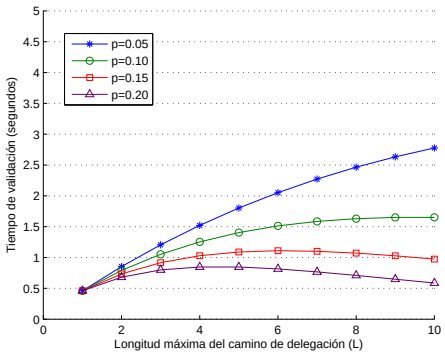

Figura 4.16: Tiempo medio de validación utilizando CARE

Como se puede observar el tiempo de validación en PKIX es mayor que el tiempo de validación cuando se permite la revocación en cascada, y se utiliza el sistema de verificación CARE.

4.7. Ventajas e inconvenientes de la revocación en cascada

Como se ha explicado en este capítulo, podemos hacer una clasificación del efecto que provoca la introducción de la revocación en cascada, en función de las entidades involucradas, desde dos puntos de vista. Por un lado está la SOA o la entidad que se encarga de la gestión de los certificados, y por otro lado, el verificador de privilegios que es la entidad que se encarga de validar los certificados. Desde un punto de vista de gestión, llevar a cabo la revocación en cascada conlleva mayor carga que en el caso general en el cual la revocación sólo se produce por petición explícita sobre un certificado. Sin embargo, desde el punto de vista de verificación, a diferencia del caso anterior, este tipo de revocación puede reducir la carga.

Por lo tanto, para poder llevar a cabo un procedimiento de revocación en cascada a partir de un determinado certificado AC_X, es necesario conocer los certificados delegados a partir de AC_X. A este conocimiento lo denominamos *up-down knowledge*.

La revocación en PKIX se lleva a cabo mediante la solicitud explícita de la revocación de un certificado AC_X. Por otra parte, cuando se revoca en cascada, se revoca el certificado para el cual se recibe una solicitud explícita, e internamente se revocan los certificados bajo el dominio del certificado AC_X. Como se ha comprobado en el apartado de análisis, esta solución supone un aumento en la carga sobre la entidad de gestión de revocación, por los siguientes motivos:

(1) Conocimiento del árbol de delegación ($\mathcal{T}_d$).- Para poder llevar a cabo la revocación en cascada es necesario que la entidad de gestión conozca el árbol de delegación. Para poder conocer el árbol de delegación, la entidad ha de conocer las delegaciones que se llevan a cabo en el sistema, lo que implica tener un mecanismo de delegación *semi ad-hoc*. Es decir, la expedición de certificados puede realizarse de forma *ad-hoc*, pero se ha de comunicar esa expedición a la entidad de gestión, lo que conlleva centralizar la solución.

(2) Revocación explícita.- Cuando se recibe la petición de revocación de un certificado, no sólo se revoca el certificado para el cual se realiza la petición, sino que también se revocan explícitamente los certificados expedidos a partir de él. Este proceso es posible si se tiene un conocimiento del árbol de delegación ($\mathcal{T}_d$).

(3) Aumento de los datos de revocación.- El hecho de revocar en cascada de forma explícita implica un aumento en el volumen de información de revocación, por lo que se pueden dar dos casos:

   a. Modo conectado.- Requiere de la existencia de una entidad a modo de OCSP que responda a las solicitudes de estado de validez de los certificados. Este modo conlleva mayor carga para el sistema de gestión.
   b. Modo desconectado.- Requiere distribuir la información de revocación. Si aumenta el número de certificados revocados de forma explícita, también aumenta el volumen de revocación a distribuir, y por lo tanto, el tamaño de la ACRL.

Desde el punto de vista de verificación, a la hora de validar un certificado, se necesitan los certificados que forman la cadena de delegación asociada a AC_X, es decir DP_X, el cual refleja los certificados por encima del certificado a validar. A este conocimiento lo denominamos *down-up knowledge*. En este caso, la revocación en cascada puede ofrecer algunas ventajas al módulo de verificación. En los dos modos de funcionamiento de CARE, el verificador puede hacer lo siguiente:

(1) Modo conectado .- La información de revocación está centralizada, es decir, el verificador de privilegios solicita a la entidad que gestiona la revocación, si un determinado certificado (identificado por su número de serie) está o no revocado. No

requiere obtener a priori los certificados que forman la cadena de delegación. Sin embargo, debe acceder a una entidad que ha de estar disponible en el momento de la verificación.

(2) Modo desconectado .- La validación es descentralizada, es decir, el verificador de privilegios obtiene una lista con los certificados revocados. Mira si el certificado está en la lista, y si no está en la lista, significa que ninguno de los certificados en el *DP* asociado, está revocado. En este caso, como ya se ha comentado, el verificador necesita obtener una ACRL de mayor tamaño, por lo que el consumo de recursos de red y de almacenamiento aumentan respecto a un sistema sin CARE.

Tabla 4.5: Comparación de CARE con PKIX desde el punto de vista de validación

	Modo	**ACs Necesarios**	**Datos revocación**	**Carga**
CARE	conectado	$1 \leq C_{\text{PKIX}}, C_{\text{CARE}} \leq C_{\text{PKIX}}$	ninguno	$\sim OCSP$
CARE	desconectado	$C_{\text{CARE}} \leq C_{\text{PKIX}}$	$ACRL_{\text{CARE}} \geq ACRL_{\text{PKIX}}$	$t_{\text{CARE}} \leq t_{\text{PKIX}}$

Por lo tanto (ver tabla 4.5), la implantación de un mecanismo de revocación en cascada presenta a priori una carga elevada en el sistema de gestión, e incluso puede trasladarse parte de esa carga al proceso de validación debido al aumento en el volumen de información de revocación. La carga en el sistema puede verse reducida a costa de centralizar el sistema de validación. Sin embargo, el sistema reduce la carga en el sistema de validación, el cual requiere sólo el certificado que se ha de validar para saber si la cadena de delegación asociada está o no revocada (modo desconectado) o es válida (modo conectado), sin que sea necesario realizar el proceso de descubrimiento de cadena, a menos que ninguno de los certificados de la cadena esté revocado.

4.8. Conclusiones

En este capítulo se ha presentado CARE, un mecanismo que permite reducir la carga en la validación de cadenas de certificados, tanto en modo conectado como en modo desconectado. Se han presentado los procedimientos y los elementos necesarios para aplicar una política de revocación de atributos en cascada, basada en un módulo de seguimiento de cadenas de certificados. Se ha podido comprobar que en función de los beneficios, o de las limitaciones del entorno en el que se quiera integrar, el módulo puede ser ubicado como un elemento independiente dentro de la arquitectura de la PMI, o integrado dentro de ella. En el primer caso, el hecho de situarlo de forma independiente hace que sea necesario una mayor confianza en el módulo y un mayor número de conexiones adicionales, sin embargo, proporciona al sistema una

mayor escalabilidad. Además, el módulo puede ser utilizado como proveedor de información (descubrimiento de cadenas). En el segundo caso, la integración requiere una menor confianza y un menor número de conexiones adicionales, aunque puede constituir un único punto de fallo y se hace necesario modificar los módulos ya implementados.

El análisis realizado ha permitido comprobar y establecer algunos de los parámetros útiles en la verificación del sistema de validación de credenciales, que no se han tenido en cuenta o no se han definido en la actualidad. La probabilidad de revocación en cascada y el número medio de verificaciones de estado son dos parámetros que nos han permitido evaluar la viabilidad de la revocación en cascada, tanto en la distribución de la información de revocación, como en la validación de las cadenas.

Como refleja el estudio, la carga del sistema debido al proceso de validación se reduce, a costa de centralizar la información de delegación. Por lo tanto, sería conveniente buscar mecanismos que nos permitan mantener la reducción lograda con CARE y que además permitan reducir la carga en el sistema de gestión.

CAPÍTULO 5

PREON. UN MECANISMO *ad-hoc* DE REVOCACIÓN EN CASCADA

5.1. Introducción

En este capítulo se presenta PREON (*Prefix REvocatiON*), un sistema que permite llevar a cabo la revocación en cascada para un escenario PMI en el que las delegaciones son *ad-hoc*, semi-permanentes e indirectas (ver apartado 4.2.1). A pesar de que este tipo de delegaciones proporcionan una alta flexibilidad en escenarios distribuidos, y de que la revocación explícita puede reducir la ventana de incertidumbre sobre el estado de revocación de los certificados, el IETF (39; 51) no recomienda que se utilicen estos mecanismos de forma extendida. Esto es debido a que estos mecanismos implican un coste elevado de validación, en especial, para cadenas largas de certificados.

Para reducir este coste, una posibilidad es introducir un servidor central que se encargue de gestionar las delegaciones tal como se presentó en el capítulo 4, o la solución de la ITU-T (28) y de otros autores (94), las cuales se basan en el mismo principio. La gestión de la revocación en cascada puede realizarse utilizando CARE (capítulo 4) u otras propuestas realizadas en la literatura (95). Sin embargo, estas soluciones no son *ad-hoc*, ya que implican un servidor centralizado para expedir nuevos ACs, el cual ha de estar siempre disponible. Esta centralización es necesaria para hacer el seguimiento de las cadenas de certificados y poder llevar a cabo la revocación en cascada. Nuestro objetivo es el de proponer una solución para reducir el coste de la validación de caminos de delegación sin tener que centralizar el servicio de delegación, y logrando la viabilidad de la distribución de información de revocación para la validación *off-*

line. Es decir, reducir la descentralización y el volumen de información de revocación, pero manteniendo las ventajas de CARE en el proceso de validación.

Para lograr mejoras en el rendimiento en escenarios donde se habilita la revocación en cascada, se propone un mecanismo basado en la utilización de palabras código para identificar a cada certificado de forma unívoca bajo la administración de una SOA. Este hecho permite reducir dos parámetros: los datos de revocación que se han de gestionar y distribuir, y el número de certificados de la cadena de delegación que se han de validar. A través de un análisis estadístico, se demuestra que nuestra propuesta mejora el funcionamiento frente a los sistemas de delegación que no utilizan la codificación presentada. Esta mejora estimada mediante un análisis estadístico (apartado 5.4.2), nos ha llevado a su evaluación sobre la plataforma de test CERVANTES (96; 97) la cual se ha ampliado para tener en cuenta la validación de cadenas de certificados (apartado 5.5).

5.2. Delegación en PREON

El proceso de delegación en PREON funciona como sigue. La SOA tiene un identificador inicial o código. Cuando la SOA quiere llevar a cabo una delegación, se genera un código para el delegado (*child-code*), donde la palabra-código que tiene asignado el delegante se incluye como prefijo en el código del delegado. El *child-code* se utilizará como identificador o número de serie del AC del delegado. Este proceso se repite en cada nueva delegación. Como el escenario que se presenta contempla la revocación en cascada, un requisito es que los códigos generados deben ser únicos bajo el dominio de la SOA, y a su vez, también deben identificar inequívocamente a sus delegantes. Como se estableció en el capítulo 4, la estructura de delegación es la de un árbol. Existen diferentes enfoques para abordar la codificación de árboles, entre ellos, se ha decidido utilizar un código prefijo. En particular se ha utilizado la codificación conocida como *comma-coding* (98) para implementar PREON, ya que es fácil de implementar. De hecho, esta codificación se ha propuesto para ser utilizada en el descubrimiento de rutas de certificados de identidad (99). Con estos códigos, un *child-code* es simplemente la concatenación de varias palabras código que identifican al delegante, con otra palabra código que identifica a la delegación.

En aras de una mayor claridad, se ha dividido el proceso de delegación en dos apartados. En el apartado 5.2.1 se presentan los pasos generales que tienen lugar cuando un delegado solicita la delegación de ciertos privilegios a un delegante. Y en el apartado 5.2.2 se especifica con detalle cómo se ha de realizar la generación del identificador del certificado, es decir, la generación del número de serie.

5.2.1. Pasos en el proceso de delegación

El protocolo para llevar a cabo una delegación en PREON se presenta de forma resumida en la figura 5.1, donde un delegado solicita la delegación de algunos de los atributos a un delegante para acceder a un recurso determinado. El proceso de delegación es el siguiente:

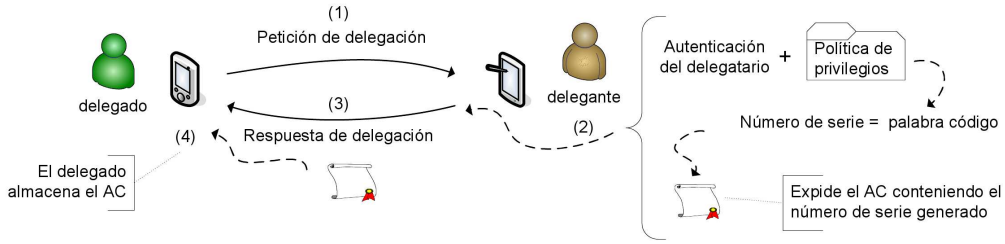

Figura 5.1: Proceso de Delegación. Pasos generales

(1) El delegado envía una petición de delegación al delegante. El delegado se autentica utilizando algún método, por ejemplo, a través de su certificado de clave pública (PKC). Un método bastante utilizado para llevar a cabo esta autenticación entre el cliente y el servidor en el que ambos se autentiquen, es utilizando el protocolo TLS (78). Por otra parte, a diferencia de la petición de un PKC, para los cuales existen mensajes estandarizados para su solicitud, para AC no existe tal estandarización. Lo que sí existe es un RFC (100) de carácter informativo que incorpora nuevos formatos de certificados a los protocolos y mensajes de gestión de certificados de identidad ya existentes (101–103). Sin embargo, creemos que la solución que se adapta mejor a las necesidades de un sistema distribuido, es la alternativa presentada en (104) ya que permite la solicitud de expedición de certificados de atributo de forma dinámica utilizando el protocolo TLS. De esta manera se puede autenticar tanto al delegante como al delegado en el mismo proceso.

(2) Si la delegación cumple con la política de privilegios, el delegante expide el AC correspondiente al usuario. La expedición de ACs implica la generación de un identificador del certificado que siga el esquema de codificación mediante código prefijo, en concreto *coma-code* (que se explica en detalle en el apartado 5.2.2). Otros de los campos relevantes en la expedición del AC en PREON son los siguientes:

   a. Enlace entre el AC y el PKC del usuario. Para evitar el problema de la retención de privilegios y poder llevar a cabo la delegación y revocación transitiva y selectiva (ver apartado 5.4.1), el enlace entre el certificado de atributo y la identidad del usuario puede llevarse a cabo utilizando la estructura `IssuerSerialNumber` del PKC del delegado, de la siguiente manera:

```
Holder ::= SEQUENCE {
     baseCertificateID   [0] IssuerSerial OPTIONAL,
     -- the issuer and serial number of
     -- the holder's Public Key Certificate
}
```

b. Extensiones con información de revocación. Como se está considerando que el AC se emite por un periodo de validez suficiente para tener en cuenta el proceso de revocación, el verificador debe poder verificar la información de revocación. Existen dos extensiones útiles para este caso: `CRLDistributionPoints` y `noRevAvail`. La primera de ellas se puede utilizar para indicar al verificador la ubicación de la lista de revocación, como se explicó en 2.3.3. La segunda de ellas no debería aparecer en el certificado ya que su no existencia le indica al verificador que debe comprobar el estado de revocación del certificado.

(3) El delegante envía la respuesta de delegación al delegado indicando si la delegación tuvo éxito o no. La distribución del certificado, en el caso en el que se haya expedido con éxito, puede llevarse a cabo de dos maneras:

a. El AC generado se almacena en un repositorio público al cual puede acceder, o bien cualquier entidad, o sólo entidades autorizadas (modelo *pull*).

b. El AC generado se envía al delegado como respuesta. De esta manera puede presentarlo al verificador de privilegios junto a la petición de acceso (modelo *push*). Por ejemplo, utilizando el siguiente mensaje:

```
struct {
     enum {
          success(0),
          success_with_changes(1),
          failure(2),
          denied(3),
          (255)
     } acstatus;
     opaque ASN1.C<1..2^16-1>
     Boolean encrypted
}ACResponse
```

El segundo caso es el más adecuado cuando se trabaja en entornos donde la conexión puede perderse, ya que en estos entornos puede aumentar la probabilidad de denegar el acceso al recurso por falta de información disponible.

(4) En el caso que la delegación se lleve a cabo con éxito, el delegado almacena su certificado de atributo de manera que pueda presentarlo al verificador de privilegios cuando intente acceder a los recursos autorizados (modelo *push*).

5.2.2. Generación del número de serie

La diferencia esencial de PREON sobre otras propuestas de delegación, es la generación de números de serie de los certificados en el proceso de delegación. En PREON, el número de serie identifica a los delegantes que participan en el camino de delegación. Para que el número de serie de un AC pueda contener los identificadores de sus predecesores en la cadena de delegación, se han de concatenar los siguientes valores para generar un nuevo número de serie:

- La palabra-código del delegante ($\mathcal{W}_X$). Si el delegante es la SOA, su palabra-código es un 0 ($\mathcal{W} = 0$).
- Una secuencia de Q 1's ($\{1\}^Q$), donde Q identifica el número de delegaciones realizadas por el delegante.
- Un 0. El valor 0 se utiliza como marca final de la palabra-código.

Por lo tanto, un delegante con palabra-código $\mathcal{W}_X$ y que expide su Q-ésimo AC deberá generar la siguiente palabra-código o número de serie:

$$\mathcal{W}_Q = \mathcal{W}_X||\{1\}^Q||0 \tag{5.1}$$

Donde "||"significa concatenación. Si aplicamos este mecanismo de codificación al árbol de delegación que se está utilizando como ejemplo, se obtienen los números de serie representados en la Figura 5.2.

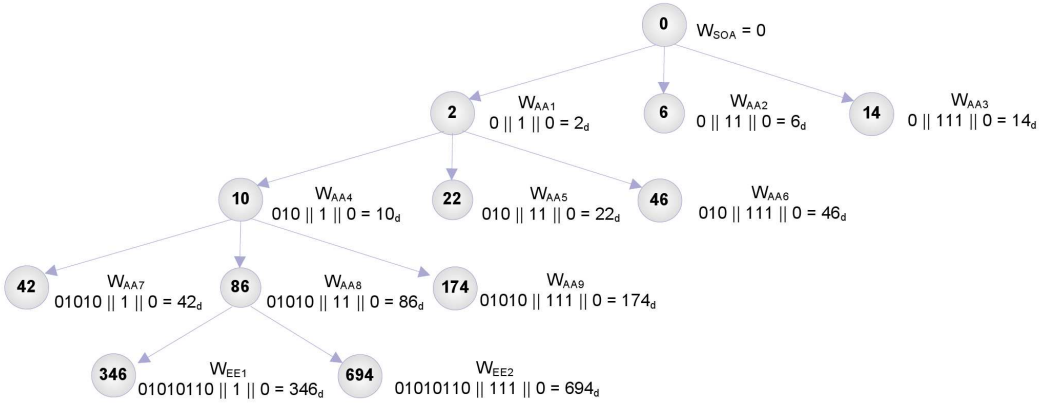

Figura 5.2: Codificación de las dependencias jerárquicas en PREON

Como muestra la Figura 5.2, la palabra-código asignada a la SOA es 0. La SOA ha realizado tres delegaciones, expidiendo los correspondientes certificados de atributo a las entidades: AC_{AA_1}, AC_{AA_2} y AC_{AA_3}. Siguiendo el esquema de codificación, los números de serie codificados en binario y en decimal, respectivamente, son:

- $\mathcal{W}_{AA_1} = 0||1||0 = 010_2 = 2_{10}$
- $\mathcal{W}_{AA_2} = 0||11||0 = 0110_2 = 6_{10}$
- $\mathcal{W}_{AA_3} = 0||111||0 = 01110_2 = 14_{10}$

Ahora supongamos que AA_1 ha delegado algunos atributos. En este ejemplo, AA_1 ha actuado como delegante tres veces, expidiendo tres nuevos certificados: AC_{AA_4}, AC_{AA_5} y AC_{AA_6}. Aplicando de nuevo el esquema de codificación, los números de serie generados para estos certificados son:

- $\mathcal{W}_{AA_4} = \mathcal{W}_{AA_1}||1||0 = 01010_2 = 10_{10}$
- $\mathcal{W}_{AA_5} = \mathcal{W}_{AA_1}||11||0 = 010110_2 = 22_{10}$
- $\mathcal{W}_{AA_6} = \mathcal{W}_{AA_1}||111||0 = 0101110_2 = 46_{10}$

La entidad AA_4 es también una AA y ha expedido tres certificados más (AC_{AA_7}, AC_{AA_8} y AC_{AA_9}) cuyos números de serie son:

- $\mathcal{W}_{AA_7} = \mathcal{W}_{AA_4}||1||0 = 0101010_2 = 42_{10}$
- $\mathcal{W}_{AA_8} = \mathcal{W}_{AA_4}||11||0 = 01010110_2 = 86_{10}$
- $\mathcal{W}_{AA_9} = \mathcal{W}_{AA_4}||111||0 = 010101110_2 = 174_{10}$

Por último, la entidad AA_8 también es una AA y ha expedido dos certificados más: AC_{EE_1} y AC_{EE_2}. Los números de serie de estos certificados son:

- $\mathcal{W}_{EE_1} = \mathcal{W}_{AA_8}||1||0 = 0101011010_2 = 346_{10}$
- $\mathcal{W}_{EE_2} = \mathcal{W}_{AA_8}||11||0 = 01010110110_2 = 694_{10}$

Cabe señalar que con este esquema de codificación, se logran algunas propiedades que son necesarias para alcanzar los objetivos de PREON:

(I) **Identificadores únicos.** Como la SOA va a distribuir la información de revocación utilizando ACRLs, los números de serie (identificadores de los certificados) han de ser únicos bajo el dominio de la SOA. Por lo tanto, los identificadores también son únicos bajo el dominio de las AA intermedias.

(II) **Delegación ad-hoc.** Cada AA intermedia puede expedir ACs de manera *ad-hoc* utilizando únicamente su propia palabra-código.

(III) **Caminos de delegación trazables.** Este objetivo se consigue gracias a que el identificador del certificado incluye las palabras-código de los delegantes que forman el camino de delegación.

Como ejemplo, si se quiere trazar el camino de delegación de AC_{AA_9}, sólo es necesario el número de serie o palabra-código de ese certificado para poder obtener la lista de palabras-código o números de serie de los delegantes. En este caso,

$$\mathcal{W}_{AA_9} = 174_{10} = 010101110_2 \tag{5.2}$$

Considerando que los 0's son marcas finales de las palabras-código, se puede extraer de forma sencilla la lista de palabras-código incluidas en el número de serie. Esta lista se denota como $\{\mathcal{W}_{AA_9}\}$:

$$\{\mathcal{W}_{AA_9}\} = \{0, 010, 01010, 010101110\} = \{0, 2, 10, 174\} = \{\mathcal{W}_{SOA}, \mathcal{W}_{AA_1}, \mathcal{W}_{AA_4}, \mathcal{W}_{AA_9}\} \tag{5.3}$$

Como se puede observar, $\{\mathcal{W}_{AA_9}\}$ contiene la lista de palabras-código de los delegantes más el número de serie considerado (última palabra-código). Esta palabra-código es el valor que ha de incluirse en el campo `serialNumber` del AC, cuyo tipo está definido como:

```
CertificateSerialNumber  ::=  INTEGER
```

Este campo no tiene restricciones en cuanto a qué tipo de información ha de incluir, aunque el IETF sí especifica algunas características que debe cumplir y que hemos agrupado en tres:

1. Se ha de asegurar que el número de serie sea único bajo el dominio directo de una AA.

2. El conjunto de los campos del certificado AC a generar `issuer/serialNumber` ha de ser único.

3. El valor del campo `CertificateSerialNumber` ha de ser un valor entero positivo para evitar ambigüedad al mapear una cadena de octetos en un valor entero.

Como se puede observar, la codificación del número de serie en PREON permite que se cumplan las tres características especificadas por el IETF.

Por otra parte, a pesar de que la definición del número de serie se haga como un entero, el cual tiene una longitud máxima en C y C++ de 64 bits, ASN.1 no tiene estas limitaciones, por lo que las entidades que requieran utilizar ACs han de poder manejar números de serie mayores de 4 octetos. Sin embargo, sí se establece una longitud máxima fijada en 20 octetos. En el apartado 5.4.2.1 se analiza la longitud de las palabras-código que se generan con PREON para diferentes poblaciones y distribuciones en la delegación de atributos.

5.3. Revocación en PREON

Se pueden diferenciar tres partes en el proceso de revocación: gestión de las peticiones de revocación, emisión de datos de revocación y verificación del estado de revocación de los certificados.

5.3.1. Gestión de las peticiones de revocación

Al igual que en el proceso de delegación, en el proceso de revocación se ven involucradas principalmente dos entidades, la que solicita la revocación y la que recibe y procesa la revocación. La SOA, u otra entidad autorizada, se encarga de recibir y procesar las solicitudes de revocación. La SOA también puede definir la política de autorización, la cual puede especificar qué entidades del árbol de delegación pueden enviar solicitudes de revocación y para qué certificados. Una política típica puede ser considerar que una entidad X puede revocar cualquier certificado que esté bajo su dominio de confianza $\mathscr{D}_X$. Esta política de revocación se conoce como revocación independiente (apartado 4.2.2).

Por otra parte, se necesita un protocolo para solicitar la revocación de un certificado. En PREON se puede utilizar cualquiera de los protocolos existentes de PKIX para las peticiones de revocación (102; 105), ya que estos protocolos, al igual que en PREON, utilizan los números de serie para identificar a los certificados. Con las peticiones de revocación de autorización, la SOA mantiene actualizado un árbol de revocación $\mathscr{T}_r$. En la figura 5.3 se ilustra el procedimiento para llevar a cabo la revocación de un certificado en PREON. Como se muestra en la figura, una entidad (delegado o delegante) envía una petición para revocar un certificado de atributo identificado por su palabra-código ($\mathscr{W}_r$). El proceso de revocación es el siguiente:

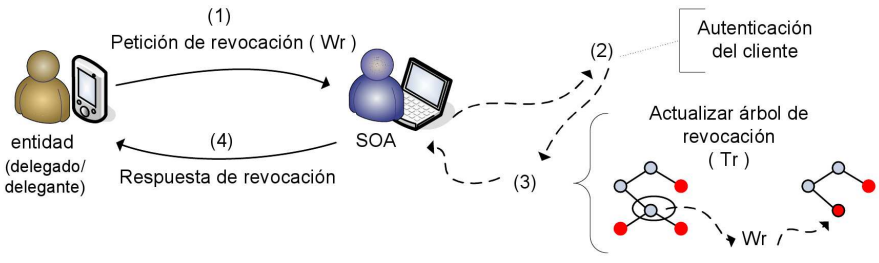

Figura 5.3: Proceso de Revocación. Pasos generales

(1) La entidad que solicita la revocación del certificado envía una petición a la SOA para revocar el certificado identificado por $\mathscr{W}_r$. Para ello, puede utilizar el siguiente mensaje de petición de revocación (106):

```
id-cmc-revokeRequest ::= { id-cmc 17 }
RevokeRequest ::= SEQUENCE {
     issuerName      Name,
     serialNumber    INTEGER,
     reason          CRLReason,
     invalidityDate  GeneralizedTime OPTIONAL,
     sharedSecret    OCTET STRING OPTIONAL,
     comment         UTF8string OPTIONAL }
```

Sin embargo, este mensaje está pensado para la revocación de PKCs donde la entidad que lo revoca es la entidad que posee el certificado y, o bien puede firmarlo (si el motivo no es el de pérdida/robo de la clave privada), o enviar un secreto generado en la fase de solicitud del certificado. En PREON se puede permitir una política en la que cualquier delegante pueda revocar el AC de un delegado bajo su dominio. Por lo tanto:

a. Si la revocación la realiza la entidad que posee el AC, éste puede demostrar que es el poseedor de ese certificado, autenticándose ante la SOA y enviando su AC. Como el AC está enlazado al PKC de la entidad, la SOA puede verificar que es el titular del AC y permitir la revocación del mismo (dependerá de la política de la SOA).

b. Si la revocación la realiza uno de los delegantes del AC, el delegante debe poder demostrar que es realmente uno de los delegantes del AC. En PREON este caso es muy similar al anterior, ya que si la entidad se autentica y envía su AC, la SOA puede verificar que: 1) es quien dice ser, 2) el AC presentado está enlazado a la identidad previamente certificada, 3) la cadena de delegación es válida y el AC no está revocado, y 4) el número de serie del AC de la entidad solicitante es un prefijo del número de serie del AC que se ha de revocar.

Este procedimiento puede llevarse a cabo utilizando el protocolo TLS con extensiones de autorización comentado en el apartado 5.2.1. De esta manera, se puede llevar a cabo el proceso anterior utilizando un único procedimiento.

(2) La SOA verifica si la entidad que solicita la revocación del certificado está o no autorizada a hacerlo. En el caso de revocación independiente, una entidad del árbol de delegación (X) puede revocar cualquier certificado que se encuentra bajo su dominio ($\mathscr{D}_X$).

(3) Si la petición de revocación es válida, la SOA actualiza el árbol de revocación $\mathscr{T}_r$ de acuerdo al algoritmo 1 que explicaremos posteriormente.

(4) La SOA envía una respuesta a la entidad que solicitó la revocación en la que indica si la revocación se ha podido llevar a cabo o no. No disponemos de un mensaje

estandarizado para el envío de una respuesta para ACs, pero podría utilizarse una estructura parecida a la utilizada para PKCs como la siguiente:

```
id-cmc-revokeResponse ::= { id-cmc XX }
RevokeResponse ::= SEQUENCE {
     issuerName      Name,
     serialNumber    INTEGER,
     reason          ResponseReason,
     invalidityDate  GeneralizedTime OPTIONAL,
     comment         UTF8string OPTIONAL }

 reasonCode ::= { ResponseReason }
 ResponseReason ::= ENUMERATED {
   succcessful        (0),  // Revocación realizada
   unauthorized       (1),  // No está autorizado
   alreadyrevoked     (2),  // Ya estaba previamente revocado
   unspecified        (3),  // Otros motivos
   badRequest         (4)}  // La petición no es correcta
                            // (pej. el número de serie no existe)}
```

Algorithm 1: Pseudo-código para la actualización del árbol de revocación $\mathcal{T}_r$

inputs: $\mathcal{T}_r$ y $\mathcal{W}_r$ los cuales son respectivamente, el árbol de revocación y el número de serie del certificado que debe ser revocado (palabra-código revocada).

begin

```
// Insertar la palabra-código revocada si el árbol de revocación
   está vacío.
if IsEmpty(T_r) then
    Insert(T_r , W_r) ;
    return T_r ;
end
{W_r} = ExtractDelegators(W_r) ; // Obtener la lista de
   palabras-código de los delegantes
// No hacer nada si ya existe un delegante revocado
forall W_i ∈ {W_r} with i ≠ r do
    if IsLeaf(T_r,W_i) then return T_r ; // ya existe un delegante
       revocado
end
// Insertar y podar si la palabra-código no está representada
Insert(T_r, W_r) ;
Prune(T_r,W_r) ; // Eliminar los delegados que contienen la
   palabra-código revocada
return T_r ;
```

end

El árbol de revocación $\mathcal{T}_r$ se actualiza de acuerdo con el algoritmo 1. Como muestra el algoritmo, se han de tener en cuenta tres situaciones. En el primer caso, el árbol de revocación está vacío. En esta situación, simplemente se ha de insertar la palabra-código o identificador del certificado que se ha de revocar en el árbol de revocación. La segunda situación es cuando el delegante del certificado a revocar ya está revocado. En este caso, no se tiene que hacer ningún procedimiento. Por último, en la tercera situación, no existe ningún delegante previamente revocado. En este último caso, se inserta la palabra-código revocada en el árbol y se podan (eliminan) las palabras-código de los delegados que dependen de la palabra-código que se inserta en el árbol de revocación. Las figuras 5.4, 5.5 y 5.6 muestran un ejemplo del algoritmo de ejecución.

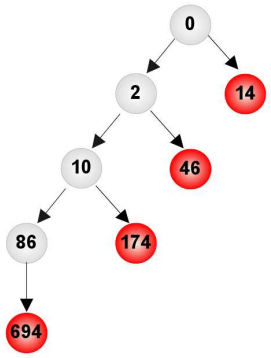

Figura 5.4: $\mathcal{T}_r \mapsto 4$ palabras-código revocadas

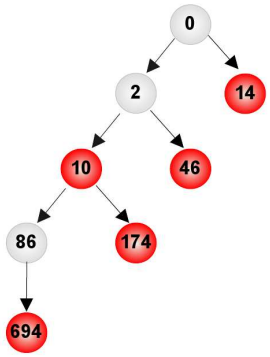

Figura 5.5: Insertando $\mathcal{W}_{AA_4} = 10$

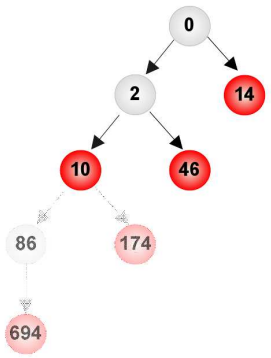

Figura 5.6: Podando todas las ramas bajo $\mathcal{W}_{AA_4} = 10$

Supongamos que el árbol de revocación $\mathcal{T}_r$ contiene cuatro palabras-código revocadas $\mathcal{T}_r = \{\mathcal{W}_{AA_3}, \mathcal{W}_{AA_6}, \mathcal{W}_{AA_9}, \mathcal{W}_{EE_2}\} = \{14, 46, 174, 694\}$ (figura 5.4). En este instante, se revoca un nuevo certificado (AC_{AA_4}) el cual se ha de introducir en $\mathcal{T}_r$. La palabra-código AC_{AA_4} es $\mathcal{W}_{AA_4} = 10$ y la lista de palabras-código es $\{\mathcal{W}_{AA_4}\} = \{0, 2, 10\}$. La SOA verifica si cualquiera de las palabras-código de la lista ya está incluida en $\mathcal{T}_r$. $\mathcal{W}_{AA_4} = 10$ es una palabra-código tanto de $\mathcal{W}_{AA_9} = 174$ como de $\mathcal{W}_{EE_2} = 694$ (figura 5.5). Por lo tanto, sólo es necesario representar la palabra-código revocada $\mathcal{W}_{AA_4} = 10$ en $\mathcal{T}_r$, ya que $\mathcal{W}_{AA_9} = 174$ y $\mathcal{W}_{EE_2} = 694$ están representadas de forma explícita por $\mathcal{W}_{AA_4} = 10$ (figura 5.6). El número de certificados revocados es cinco, sin embargo, el árbol de revocación sólo necesita almacenar tres palabras-código $\mathcal{T}_r = \{10, 14, 46\}$. De esta manera, se puede reducir el volumen de información que ha de contener la ACRL, como se verifica en el análisis realizado en el apartado 5.4.

5.3.2. Generación de los Datos de Revocación

El siguiente paso en el proceso de revocación es hacer que el árbol $\mathcal{T}_r$ sea accesible a los verificadores. PREON utiliza la estructura estándar ACRL para distribuir información de revocación. Una ACRL consta principalmente de cuatro partes ($ACRL = (i, id, vp, rc)$): 1) i representa el emisor de la ACRL, 2) id identifica de forma inequívoca a la ACRL, 3) vp es el periodo de tiempo durante el cual la ACRL es válida, y 4) rc contiene la lista de números de serie de los certificados que han sido revocados.

En PREON, la ACRL es en realidad una lista reducida de los números de serie, ya que sólo se incluyen en la ACRL las palabras-código necesarias para representar el conjunto de certificados revocados. La SOA emite de forma periódica la ACRL, la cual contiene el conjunto actual de palabras-código que están incluidas en $\mathcal{T}_r$. Hay que tener en cuenta que la ACRL se puede publicar en varios servidores o repositorios para proporcionar una mayor disponibilidad. Estos repositorios no requieren ser de confianza ya que la ACRL está firmada por la SOA, y por lo tanto, se puede verificar tanto su integridad como su autenticidad. De hecho, cualquier entidad del escenario de delegación con una copia no expirada de la ACRL puede llegar a ser un repositorio. Este enfoque alivia la carga en la SOA durante el proceso de verificación del estado de los certificados. Un usuario puede incluso obtener una ACRL sin disponer de una conexión con la SOA (este aspecto se tratará con detalle en el capítulo 6).

PREON puede utilizar la estructura ACRL para la distribución de la información de revocación. Sin embargo, se ha de informar al verificador que la ACRL contiene los prefijos revocados y no los números de serie de los certificados revocados de forma explícita. Para aportar esta información existen dos posibilidades: 1) reflejarlo en la política de autorización y 2) especificarlo en la ACRL. En el segundo caso, se puede utilizar el campo de extensiones para este propósito. La siguiente estructura podría ser una definición de la extensión a incluir en la ACRL[1]:

```
name           id-ce-revocationPolicy
OID            { id-ce XX }
syntax         RevocationTypeSyntax
criticality    MUST be FALSE
--------------------
RevocationType ::= SEQUENCE {
  type      ENUMERATED, // Identifica el tipo de revocación realizada
  data      SEQUENCE OF EXTENSIONS        OPTIONAL}

Type ::= ENUMERATED {
  cascade-imp       (0),    // Revocación en cascada implícita
```

[1] Esta estructura es la utilizada en la implementación de PREON en la plataforma de pruebas CERVANTES.

```
cascade-exp       (1),    // Revocación en cascada explícita
unspecified       (255)}  // No especificada
```

Por compatibilidad, la extensión puede ser no crítica, ya que en el caso de que un verificador no reconozca la extensión, podría seguir validando el certificado. Esto es posible ya que si en la cadena de delegación asociada existe un certificado revocado, la ACRL contendrá, como mínimo y como máximo, un certificado revocado, por lo que siguiendo el modelo de validación PKIX, el verificador acabará encontrando un certificado revocado en la cadena. Además, el verificador se beneficiará de la reducción en el tamaño de la ACRL (ver análisis en el apartado 5.4). Con la reducción del tamaño, se reducirá el coste de obtención y de almacenamiento de la ACRL.

5.3.3. Verificación del estado de revocación

En PREON, el verificador de privilegios puede comprobar el estado de revocación de un camino de delegación completo sin tener que descargar ningún certificado de la cadena de delegación. El único caso en el que el verificador ha de obtener los certificados de la cadena, es cuando ninguno de los certificados del camino de delegación está revocado (modo desconectado en CARE). Esto permite que el proceso de validación global sea más eficiente, ya que las cadenas de delegación establecidas son menos propensas a ser inválidas que el estado de revocación de certificados.

La verificación del estado de revocación en PREON consiste en obtener una ACRL no expirada de un servidor disponible, y verificar si alguna de las palabras-código que representan a los delegantes del certificado que se está validando, está revocado. A continuación, se proporciona un ejemplo para ilustrar las tareas que debe realizar un verificador de privilegios para conceder el acceso a la entidad que presenta el AC (figura 5.7):

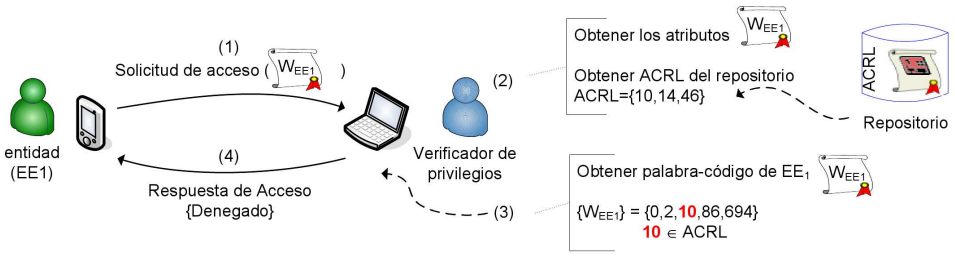

Figura 5.7: Ejemplo del proceso de validación

(1) Una entidad $EE1$ está intentando tener acceso a un recurso. Con este fin, envía una solicitud de acceso al verificador de privilegios en la que incluye su AC. Supondremos que la entidad $EE1$ se autentica de la misma manera que en el pro-

ceso de delegación. Para reducir la carga en la utilización de diferentes protocolos y permitir el intercambio de credenciales tanto para el proceso de autenticación como para el proceso de autorización, puede utilizarse el protocolo TLS.

(2) El verificador de privilegios comprueba si los privilegios contenidos en el AC son suficientes o no para acceder al recurso. Si los privilegios son suficientes, el verificador obtiene una ACRL de cualquier repositorio disponible. En este ejemplo, la ACRL contiene las palabras-código siguientes: $\{10, 14, 46\}$.

(3) El verificador valida la lista y analiza las extensiones que incluye la ACRL. Como se explicó en el apartado 5.3.2, la SOA ha incluido la extensión `RevocationType (0)`. Como el verificador reconoce esta extensión, puede obtener mayores beneficios en el proceso de validación.

(4) El verificador obtiene la lista de palabras-código de los delegantes del certificado que se está validando. En este caso tenemos $\{\mathscr{W}_{EE_2}\} = \{0, 2, 10, 86, 694\}$. Las palabras-código 0 y 2 no están incluidas en la ACRL, pero la palabra-código 10 sí está incluida en la ACRL. Por lo tanto, el camino de delegación no es válido. En este punto, el verificador finaliza el proceso de verificación.

(5) El verificador de privilegios envía una respuesta de acceso a la entidad solicitante $EE1$. En este ejemplo, el verificador deniega el acceso a los recursos ya que el camino de delegación está revocado.

Intuitivamente, el ejemplo anterior indica que PREON es más eficiente que los sistemas de delegación que no utilizan codificación prefijo cuando se habilita la revocación en cascada. En el apartado 5.4 se realiza un análisis tanto de seguridad como estadístico para evaluar cuantitativamente las mejoras logradas por PREON.

5.4. Análisis

5.4.1. Aspectos de Seguridad

En este apartado se clarifican algunos de los aspectos de seguridad relacionados con PREON, como son: el problema de la retención indebida de privilegios, la posibilidad de la falsificación del número de serie y los ataques de denegación del servicio.

Retención no autorizada de privilegios. Cada AC está asociado a un usuario, y cada usuario está identificado por un PKC. Los PKC enlazan la identidad de un usuario a una clave pública. Un problema de seguridad que se conoce como «retención no autorizada de privilegios» (107) podría aparecer cuando un usuario intenta utilizar un cierto AC pero su identidad certificada a través de un PKC ya

no es válida (por ejemplo, su PKC ha expirado o está revocado). Una solución a este problema es establecer un enlace criptográfico entre el AC y su PKC correspondiente. Este enlace se lleva a cabo utilizando el campo `Holder`, el cual contiene la información para identificar el PKC adecuado (ver apartado 5.2.1). Por lo tanto, como parte del servicio de validación de certificados, también es necesario validar el PKC asociado al AC que se está validando.

En el caso de la delegación dinámica, existen algunos aspectos que se han de considerar (4; 88):

1. Supongamos que se tienen dos ACs: AC_1 y AC_2 expedidos a una misma entidad con un único PKC:
   - Si el PKC deja de ser válido, ni AC_1 ni AC_2 serán válidos, ya que están enlazados a ese PKC.
   - AC_1 podría ser válido, mientras AC_2 podría ser no válido. Es decir, la revocación de AC_1 no implica la revocación de AC_2.
2. Supongamos que se tienen dos ACs: AC_1 y AC_2 expedidos a una entidad con dos PKCs: PKC_1 y PKC_2. Ahora consideremos que AC_1 está enlazado a PKC_1 y que AC_2 está enlazado a PKC_2. En este caso, la revocación de PKC_1 sólo afecta a AC_1, por lo que AC_2 no se ve afectado.

Falsificación del número de serie. Una de las propiedades de seguridad en PREON es que la generación y utilización de números de serie falsos puede ser detectada fácilmente. Es decir, no se pueden utilizar números de serie erróneos para evitar que se detecte que están revocados durante el proceso de validación (por ejemplo, números de serie que no siguen la codificación explicada en el apartado 5.2.2). En cualquier caso, el verificador detectará el error, tanto si el número de serie se ha generado de forma incorrecta deliberadamente (falsificación), como si se ha generado incorrectamente de forma involuntaria (error del programa). El motivo por el que los ataques no son viables, es debido a que los números de serie están organizados de forma jerárquica y están contenidos en el AC, el cual está firmado digitalmente por cada delegante. Lo anterior, unido a que cada AC está enlazado con el correspondiente PKC del delegado, hacen que no sea posible realizar con éxito un ataque de este tipo. Por último, para firmar un AC, el delegante utiliza su clave privada, la cual está asociada a su PKC.

A continuación se presentan algunos ejemplos de posibles ataques, y una explicación de cómo el verificador de privilegios puede detectar esa utilización deshonesta:

- Un delegado, cuyo certificado se ha revocado, quiere modificar el número de serie de su certificado para que el verificador no pueda detectar que

su certificado está revocado. Este caso se puede detectar de forma sencilla ya que si se altera cualquiera de los campos del AC, la firma del AC será inválida.

- Un delegante quiere suplantar a otra AA. Supongamos que un atacante $AA_{atacante}$ quiere suplantar a un delegante honesto $AA_{honesta}$. Ahora consideremos que el número de serie de un delegante honesto tiene los prefijos X_1, X_2 y X_3, y que el atacante tiene una lista de prefijos muy similar, como puede ser: X_1, X_2 y X_4. Si miramos ambas listas de prefijos, sabemos que $AA_{atacante}$ y $AA_{honesta}$ están situados al mismo nivel dentro del árbol de delegación. Sin embargo, $AA_{atacante}$ no podrá expedir ACs válidos cuyos códigos prefijo sean X_1, X_2 y X_3. Este caso sólo sería posible si el atacante pudiera tener acceso a la clave privada asociada al PKC al que están enlazados los ACs expedidos por $AA_{honesta}$. Sin embargo, si el PKC de $AA_{honesta}$ se ha revocado, el verificador podrá detectarlo. Del mismo modo, si el atacante utiliza su clave privada para firmar certificados que contengan la lista de prefijos X_1, X_2 y X_3, el verificador de privilegios detectará que falla la relación que ha de existir entre el $AA_{atacante}$ y su delegante, ya que el AC del delegante anterior no contiene la lista de prefijos X_1, X_2 y X_3.
- Un delegante quiere expedir ACs que no contengan su código prefijo para que las acciones realizadas por los delegados no puedan ser vinculadas a ese delegante. Siguiendo con el ejemplo del punto anterior, el atacante expide ACs en los que falta alguno de sus prefijos, por ejemplo X_4. El verificador de privilegios detectará que la cadena de delegación asociada no es válida.

Cuando una AA delega determinados privilegios a otra entidad, también está delegando cierto control sobre la utilización y la delegación de los privilegios delegados, a esa entidad. Sin embargo, la PMI proporciona mecanismos para llevar a cabo una delegación controlada, estableciendo restricciones sobre el uso de los atributos delegados (campo extensiones del AC). Por ejemplo, los autores de (108) presentan un escenario en el cual los recursos son limitados, por lo que se ha de restringir su utilización de forma adecuada en cada delegación. Para ello, la delegación se gestiona a través de un sistema de asignación de cuotas que permite que no se puedan conceder más privilegios que recursos disponibles. Por último, si un delegante no puede seguir confiando en un delegado, el delegante ha de revocar el AC correspondiente, o lo que es lo mismo, ha de revocar la palabra-código de ese AC.

Denegación del servicio. Tanto el IETF como la ITU-T definen a la SOA dentro de la PMI, como una entidad de confianza que administra un determinado atributo

o conjunto de atributos. Como la SOA es el responsable último de un atributo o un conjunto de atributos, ésta podría ser también un único punto de fallo. Un atacante puede intentar realizar ataques de denegación del servicio (DoS - *Denial of Service*) sobre la SOA que gestiona dominios más grandes. En este caso, se podría considerar la ejecución descentralizada de la SOA como contramedida a este tipo de ataques.

5.4.2. Análisis estadístico

En este apartado se proporciona un análisis estadístico para ilustrar las mejoras logradas por PREON cuando la revocación en cascada está habilitada. El análisis compara PREON con un sistema de delegación en el que no se ha habilitado la revocación en cascada y cuando se utiliza la revocación en cascada de forma explícita (CARE). Para realizar la comparación se definen dos figuras de mérito: δ_{PREON} y β_{PREON}. Estas figuras de mérito representan, respectivamente, la reducción en el número de entradas de la ACRL y la reducción del número de certificados de la cadena de delegación que el verificador de privilegios ha de obtener y verificar. La tabla 5.1 muestra un resumen de las definiciones y parámetros que se utilizan en el análisis. Aunque la mayoría de los parámetros ya han sido definidos en el capítulo 4 (7 últimos parámetros de la tabla), se incluyen de nuevo para facilitar la lectura del análisis.

Tabla 5.1: Definición de los parámetros para el análisis de PREON

Parámetro	**Definición**
δ_{PREON}	Factor de reducción del número de entradas de la ACRL
β_{PREON}	Factor de reducción del número de certificados de la cadena a ser verificados
$\mathcal{W}_R$	Número medio de palabras-código revocadas
p_p	Porcentaje de prefijos revocados
C_{PREON}	Número medio de certificados de la cadena a ser validados utilizando PREON
N	Número de certificados actualmente expedidos y no expirados
L	Número máximo de niveles en un modelo de expedición jerárquico
M	Número medio de certificados expedidos por cada AA intermedia
k	Longitud media de la cadena de delegación de un certificado en $\mathcal{T}_d$
p_c	Porcentaje de certificados revocados
C_R	Número medio de ACs revocados
C_{PKIX}	Número medio de certificados de la cadena a ser validados cuando no se utiliza PREON

La metodología del análisis es la misma que la utilizada en el análisis de CARE (apartado 4.6). Para empezar, se calculará el parámetro δ_{PREON}, el cual nos da una estimación de la reducción en el número de entradas en la ACRL que se consigue al utilizar PREON. Definimos δ_{PREON} como:

$$\delta_{\text{PREON}} = 1 - \mathcal{W}_R / C_R \tag{5.4}$$

donde $\mathcal{W}_R$ es el número medio de palabras-código revocadas (incluidas en la ACRL) y C_R es el número medio de ACs revocados. Notar que siempre $\mathcal{W}_R \leq C_R$ y que δ_{PREON} mide la relación entre estas dos magnitudes: $\delta_{\text{PREON}} = 0$ significa una reducción del 0 % ($\mathcal{W}_R = C_R$) y $\delta_{\text{PREON}} = 1$ significa una reducción del 100 %.

Al igual que en caso de CARE, suponemos que las solicitudes de revocación son independientes, y que un camino de delegación se considerará revocado si cualquiera de sus certificados está revocado. También consideramos que el número medio de certificados revocados puede ser expresado como un porcentaje p_c de los certificados actualmente expedidos y no expirados N, es decir, $C_R = p_c \cdot N$. Además, el número medio de prefijos revocados también puede ser expresado como un porcentaje p_p de los certificados revocados, es decir, $\mathcal{W}_R = p_p \cdot C_R$. Como resultado, la ecuación (5.4) puede ser reescrita como:

$$\delta_{\text{PREON}} = 1 - p_p \tag{5.5}$$

Ahora tenemos que dar una expresión para p_p. Este parámetro representa el porcentaje de los identificadores revocados que se incluirán en la ACRL. Hay que recordar que la palabra-código de un certificado revocado se incluye en la ACRL sólo en caso de que ninguna de las palabras-código de sus delegantes ya esté revocada. Si el número medio de palabras-código de un número de serie seleccionado de forma aleatoria es k, la condición anterior puede expresarse como:

$$p_p = (1 - p_c)^{k-1} \tag{5.6}$$

donde k dependerá de la longitud máxima del camino de delegación (L) y del número de certificados actualmente expedidos y todavía no expirados (N). Si consideramos el mismo modelo de expedición jerárquico que en CARE, k coincide con el número medio de niveles del árbol de delegación $\mathcal{T}_d$, ya que el número medio de palabras-código para un AC, coincide con el número medio de niveles en $\mathcal{T}_d$, y por lo tanto, con la longitud media del camino de delegación. De esta manera, k se corresponde con la ecuación 4.4 calculada en el apartado 4.6.

En este punto podemos evaluar cuál es la probabilidad de revocación en PREON ($\mathcal{P}_p$) y compararla con la probabilidad de revocación en PKIX (p_c), y con la probabilidad de revocación en cascada ($\mathcal{P}_c$) evaluada en el apartado 4.6. Como sabemos $C_R = p_c \cdot N$ y $\mathcal{W}_R = p_p \cdot C_R$, por lo que se puede escribir que $\mathcal{W}_R / p_p = p_c \cdot N$, con lo cual $\mathcal{W}_R = (p_p \cdot p_c) \cdot N$, de manera que la probabilidad de revocación en PREON puede expresarse como:

$$\mathcal{P}_p = p_p \cdot p_c \tag{5.7}$$

La figura 5.8 muestra cómo se reduce de forma exponencial la probabilidad de revocación en PREON con la longitud de la cadena de delegación máxima permitida.

Por lo tanto, también se reduce el número de certificados revocados que ha de mantenerse, como muestra la figura 5.9. Por ejemplo, tomando $L = 5$ y $p_c = 0{,}15$ se reduce en casi un 50 % el número de certificados revocados que ha de mantenerse como información de revocación, si lo comparamos con PKIX.

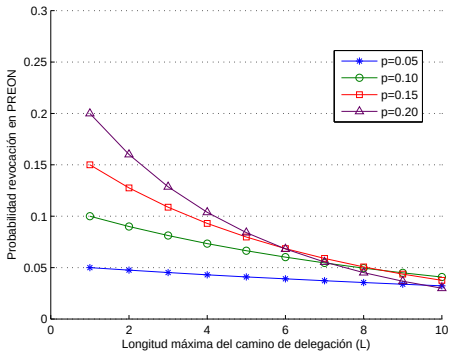

Figura 5.8: Probabilidad de revocación en PREON (%N)

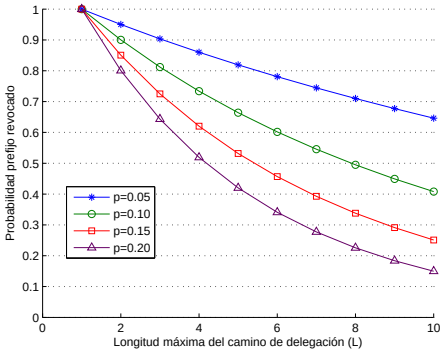

Figura 5.9: Probabilidad de revocación de prefijos (%C_R)

Finalmente, se obtiene la siguiente expresión para δ_{PREON} que sólo depende de p_c y k:

$$\delta_{\text{PREON}}(p_c, k) = 1 - (1 - p_c)^{k-1} \quad \text{donde } k = f(L, N) \tag{5.8}$$

La otra figura de mérito (β_{PREON}) representa la reducción lograda por PREON sobre el número de certificados que se han de obtener para ser validados. En este caso, β_{PREON} coincide con β_{CARE} (modo desconectado), ya que tanto CARE como PREON logran la misma mejora respecto a un sistema PKIX (ver apartado 4.6.2).

La figuras 5.10 y 5.11 presentan algunas gráficas para la ecuación 5.8 para diferentes valores de la probabilidad de revocación, diferentes longitudes máximas del camino de delegación y diferentes poblaciones de certificados. Por ejemplo, para un escenario en el cual $L = 3$, $N = 100000$ y la probabilidad de revocación de certificados es del 20 % ($p_c = 0{,}2$), PREON reduce aproximadamente alrededor de un 32 % el número de entradas de la ACRL. Obtenemos mejoras similares considerando una profundidad mayor ($L = 5$), pero una probabilidad menor de revocación ($p_c = 0{,}1$): $\delta_{\text{PREON}} = 0{,}32$.

Como se puede observar, PREON es siempre más eficiente que los sistemas en los que no se utiliza revocación en cascada. Es decir, PREON reduce tanto el número de entradas en la ACRL como el número de certificados de la cadena de delegación que han de obtenerse para ser validados, a medida que aumenta la probabilidad de revocación de certificados y la longitud de la cadena delegación. Como resultado, cuando la revocación en cascada se activa, se consigue nuestro objetivo: reducir el coste en la validación de cadenas de delegación.

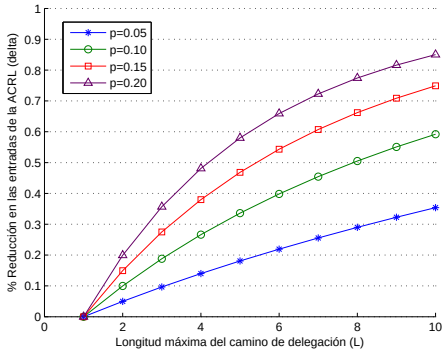

Figura 5.10: Reducción del % de certificados revocados en función de p_c

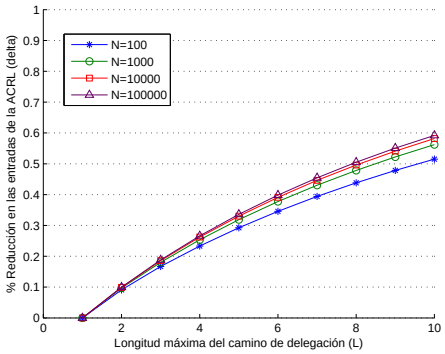

Figura 5.11: Reducción del % de certificados revocados en función de N

5.4.2.1. Longitud máxima del código

En este apartado se calcula la longitud máxima del número de serie en PREON (l_{max}). Para hacerlo de forma sencilla, tomemos el ejemplo del apartado 5.2.2 en el que la palabra código $\{\mathcal{W}_{AA_9}\}$ se descompone de la siguiente manera:

$$\{\mathcal{W}_{AA_9}\} = \{0, 010, 01010, 010101110\} = \{0, 2, 10, 174\}$$

Podemos observar que $\{\mathcal{W}_{AA_9}\}$ contiene la lista de palabras-código de los delegantes, más la palabra-código que se corresponde con el número de serie considerado (última palabra-código). Por otra parte, la longitud de los identificadores depende del número máximo de delegaciones que puede llevar a cabo cada AA (m), y la profundidad máxima de la delegación (L). Por lo tanto, l_{max} se puede calcular como la suma de:

- 1 bit: debido a la palabra-código de la SOA.
- $m \times L$ bits: debido al número máximo de ACs expedidos por una misma AA, multiplicado por la profundidad máxima del árbol de delegación L.
- L bits: debido al máximo número de delegantes de un AC.

La figura 5.12 representa la longitud máxima de un número de serie para diferentes tamaños de población de certificados (N), y diferentes profundidades máximas del árbol de delegación (L).

Con el cálculo de la longitud del número de serie, se puede obtener la población máxima que se puede representar utilizando un código como el considerado para PREON. La tabla 5.2 presenta el número máximo de certificados que se pueden representar tomando como longitud máxima los 160 bits que establece el IETF ($l_{max} = 160$). Si observamos la tabla, el código no es muy adecuado para valores de $L \leq 2$, aunque

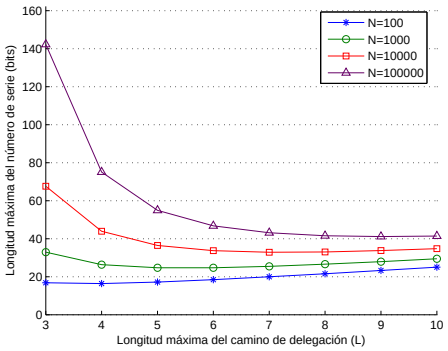

Figura 5.12: Evolución de la longitud máxima del número de serie en función de L

hay que tener en cuenta que PREON está pensado para cadenas de delegación de longitud mayor. Sin embargo, en los escenarios en los que $L = 2$ o similares, se podría reducir la longitud de las palabras-código debido a las cadenas de 1's consecutivos generadas en cada nivel, lo que conllevaría una reducción de la longitud máxima del código.

Tabla 5.2: Número máximo de certificados para diferentes profundidades de $\mathcal{T}_d$

L	N máxima	L	N máxima	L	N máxima	L	N máxima
1	$1,58 \cdot 10^2$	4	$2,3144 \cdot 10^6$	7	$2,3861 \cdot 10^9$	10	$5,7814 \cdot 10^{11}$
2	$2,408 \cdot 10^3$	5	$2,8648 \cdot 10^7$	8	$1,7011 \cdot 10^{10}$		
3	$1,43364 \cdot 10^5$	6	$2,8616 \cdot 10^8$	9	$1,0556 \cdot 10^{11}$		

Consideremos un ejemplo en el que un certificado expedido por la AA_{xx}, cuyo número de serie asociado generado puede expresarse como $011\ldots101111\ldots11110$. Las cadenas de unos seguidos ocupan muchos bits, pero pueden reducirse si se utiliza alguna codificación, por ejemplo, sustituyendo la cadena de 1s por un valor que refleje el número de 1s seguidos. Una posible técnica es RLE (*Run Length Encoding*) (109), sin embargo, esta técnica utiliza 1 *byte* para codificar el símbolo, y otro *byte* para indicar las veces que aparece ese símbolo (1 *byte* puede representar como máximo 255 valores). En PREON para L=1 y L=2 pueden aparecer más de 255 unos seguidos, por lo que RLE no es del todo apropiado. Una técnica más apropiada podría ser TLV (*Type Length Value*), utilizada en la codificación de estructuras ASN.1, y por lo tanto, en la del propio certificado X.509. Esta es la codificación que se ha utilizado para su implementación en CERVANTES (apartado 5.5).

Cabe señalar que aunque puede parecer que la utilización de números de serie elevados es ineficiente, las autoridades PKI en sistemas reales que expiden certificados a diferentes niveles (educativos, comerciales, institucionales,etc) están utilizando números de serie entre 11 y 16 octetos, en su mayoría, como refleja la tabla 5.3, obte-

nida a partir de los certificados incluidos por defecto en el navegador *Firefox* (57).

Tabla 5.3: Longitud del número de serie utilizado en certificados incluidos en el navegador Firefox

Empresa de certificación	**Longitud número de serie**
FNMT Clase 2 CA	4 octetos
Entrust.net	4 octetos
Cybertrust Educational CA	11 octetos
Cybertrust Educational CA	11 octetos
RSA Security	16 octetos
Thawte	16 octetos
Visa eCommerce	16 octetos
Verisign	16 octetos

5.5. Prestaciones: estudio de PREON utilizando CERVANTES

Los resultados analíticos obtenidos sobre las mejoras que se proporcionan utilizando PREON, nos han llevado a realizar su implementación sobre una plataforma de *test*. La plataforma utilizada es CERVANTES (96; 97), la cual permite tomar medidas de prestaciones para diferentes políticas de revocación. CERVANTES incluye los mecanismos necesarios para la validación de nuevas políticas, sin embargo, el sistema estaba pensado para mecanismos de revocación en los que todos los certificados estaban expedidos por una misma autoridad. Es decir, no contemplaba cadenas de certificados, ni en su generación, ni en su validación. Por lo tanto, se ha tenido que modificar CERVANTES para que permita la evaluación de PREON[2].

5.5.1. Parámetros del análisis

Las modificaciones realizadas en CERVANTES permiten que podamos realizar simulaciones en las que se pueden modificar los siguientes parámetros:

- Población de certificados (N): permite modificar la población de certificados bajo una SOA.
- Longitud máxima del camino de delegación (L): establece el valor máximo de la longitud de la cadena de delegación permitida bajo la SOA.
- Concentración de las delegaciones (ϕ_d): este parámetro permite modificar la distribución de las delegaciones sobre la población de certificados. Se puede

[2] Para una mayor claridad, en este apartado no se describen las modificaciones realizadas en CERVANTES.

modelar qué entidades tienen mayor probabilidad de delegar, permitiendo de esta manera modelar la topología del árbol, junto a los parámetros N y L.

- Probabilidad de revocación (p): establece la probabilidad de revocación dentro del dominio de la SOA.
- Concentración de las revocaciones (ϕ_r): este parámetro permite modificar la distribución de las revocaciones sobre la población de certificados. Se puede modelar qué certificados tienen mayor probabilidad de ser revocados.
- Concentración de las peticiones (ϕ_p): este parámetro permite modelar sobre qué niveles del árbol de delegación se van a concentrar las peticiones de validación de certificados. De esta manera, no todos los certificados tienen la misma probabilidad de ser validados, o lo que es lo mismo, se puede modelar qué entidades tienen mayor probabilidad de solicitar un recurso.

Tabla 5.4: Definición de los parámetros del test en CERVANTES

Parámetro	**Definición**
N	Número de certificados actualmente expedidos y no expirados
L	Número máximo de niveles en un modelo de expedición en árbol
p	Porcentaje de certificados revocados
ϕ_d	Nivel en el que se concentran las delegaciones
ϕ_p	Nivel en el que se concentran las peticiones de validación
ϕ_r	Nivel en el que se concentran las revocaciones

La tabla 5.4 muestra un resumen de los parámetros que se van a modificar para comparar el funcionamiento de PREON frente al funcionamiento PKIX. El análisis se ha centrado en la carga sobre el canal de comunicaciones, ya que ésta repercute directamente en la carga en el verificador.

5.5.2. Carga sobre el sistema

La carga en el sistema viene dada por el volumen de información que ha de obtener el verificador de privilegios. Para verificar cómo afectan los diferentes parámetros al volumen de información de revocación que se ha de obtener para validar un certificado, se han realizado cuatro gráficas en las que se compara el *throughput* en el canal cuando se utiliza PREON y cuando no se tiene en cuenta la revocación en cascada.

La figura 5.13 muestra cómo aumenta de forma lineal *throughput* en función de la probabilidad de revocación. Si el parámetro que se modifica es el nivel en el que se concentran las revocaciones, PREON aumenta la carga en el canal, mientras que en PKIX permanece constante (figura 5.14). Este comportamiento es el esperado, ya

que en PKIX el número de certificados revocados viene determinado por la probabilidad de revocación y por el tamaño de la población, como se puede comprobar en las figuras 5.13 y 5.16.

Al igual que sucede en PKIX, en PREON, el *throughput* aumenta para los diferentes parámetros consideramos, excepto cuando aumenta la longitud de la cadena de delegación, ya que en PREON se reduce exponencialmente (figura 5.15) como ya se había comprobado en el apartado 5.4.2. Por lo tanto, PREON siempre reduce la información de revocación que se necesita distribuir, y por lo tanto, la carga en el sistema.

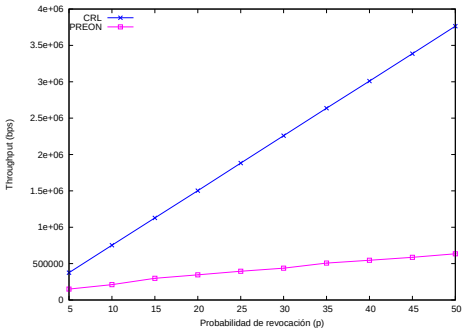

Figura 5.13: *Throughput* variando p_c

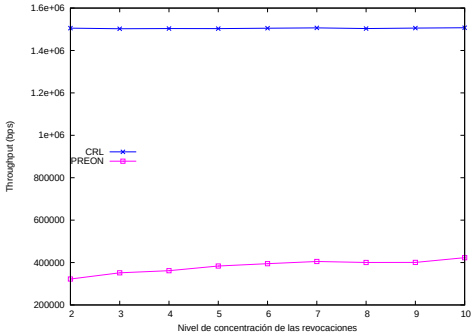

Figura 5.14: *Throughput* variando ϕ_r

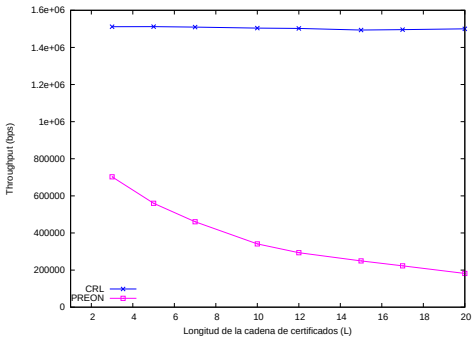

Figura 5.15: *Throughput* variando L

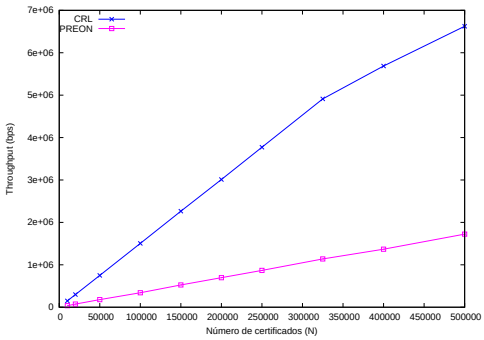

Figura 5.16: *Throughput* variando N

5.6. Conclusiones

En este capítulo se ha presentado PREON, un mecanismo que genera el número de serie de los certificados de una manera que permite reducir el coste de validación de la revocación en cadenas de delegación, y por lo tanto, el coste global del proceso de validación. Como se ha podido comprobar, PREON es un mecanismo sencillo por lo que respecta a la implementación, y de hecho ha sido implementado para su estudio en la plataforma CERVANTES.

El principal objetivo de PREON es el de reducir el coste de validación de credenciales, evitando, en lo posible, la centralización que requería la propuesta CARE presentada en el capítulo 4. Por lo tanto, PREON puede utilizarse en modo desconectado. Este objetivo se logra utilizando una codificación prefijo para los números de serie, la cual permite trazar el camino de delegación para un certificado dado. Este hecho permite verificar con una única operación si la cadena asociada a un certificado está o no revocada. Por lo tanto, los certificados que forman la cadena sólo se han de obtener en el caso de que ninguno de los certificados de la cadena esté revocado. Esta característica permite reducir la carga en el proceso de validación, tanto en la obtención de la información de revocación (el número de entradas en la ACRL se reduce), como el número de certificados que se han de obtener y validar.

La reducción de la carga en el sistema se ha podido comprobar utilizando dos procedimientos diferentes, uno analítico y otro empírico (mediante la plataforma de *test* CERVANTES). El estudio analítico nos ha dado una primera aproximación de la reducción en la carga en el sistema, la cual se corrobora con los resultados obtenidos mediante la plataforma de test. Tanto el volumen de información de revocación a distribuir, como el número de certificados a validar se reducen. El análisis muestra que PREON es más eficiente que los sistemas que no utilizan codificación prefijo. Por lo tanto, PREON es especialmente adecuado en escenarios en los que la entidad final tiene unos recursos limitados (memoria, capacidad computacional y canal de comunicación limitado), como puedan ser los terminales móviles.

CAPÍTULO 6

eRL. UN SISTEMA DE DISTRIBUCIÓN DE DATOS DE REVOCACIÓN EN REDES HÍBRIDAS

La validación de certificados requiere de información que puede estar almacenada en servidores que han de ser accesibles a los dispositivos (modo conectado), o de estructuras seguras que permitan obtener la información necesaria para realizar las validaciones sin que sea necesaria la conexión a los repositorios o servidores PKIX (modo desconectado). Las soluciones presentadas en el capítulo 4 (CARE) y el capítulo 5 (PREON) permiten reducir la carga en el proceso de validación en modo conectado, y reducir el volumen de información que se ha de obtener para poder llevar a cabo la validación en modo desconectado. Sin embargo, pueden existir dispositivos que no hayan obtenido la información de revocación necesaria para la validación antes de entrar en modo desconectado. En este caso se podrían utilizar mecanismos para compartir de forma dinámica la información de revocación, pero aún en ese caso, el tamaño de los datos de revocación podría llegar a cargar bastante la red. PKIX es una solución conveniente pero limitada en entornos MANET, donde puede perderse la conexión a los repositorios centralizados y en el que los dispositivos pueden tener recursos limitados. Por lo tanto, se necesita una solución que permita una mayor disponibilidad del servicio de validación de certificados.

6.1. Introducción

Las MANET son redes móviles que permiten al usuario establecer comunicaciones en cualquier lugar y en cualquier momento, de manera "espontánea"(apartado 2.1). Como se indica en (110) este tipo de redes puede cambiar de forma dinámica y

su arquitectura multisalto puede tener una topología aleatoria debido a la dinamicidad de los elementos que la constituyen. Las MANETs podrían operar de forma aislada (autónoma) o contar con una pasarela (*gateway*) hacia una red fija. El comportamiento híbrido de estas redes es debido a que de forma temporal podrían estar trabajando de forma aislada y a veces podrían estar conectadas a Internet. Es decir, estamos considerando el modo conectado y modo desconectado definidos en el capítulo 3.

Como ya se explicó en los capítulos anteriores, las soluciones PKIX son una elección adecuada para estos escenarios, sin embargo, se han de tener en cuenta las limitaciones tanto de la disponibilidad de los recursos y servicios, como de los elementos que forman la red. En la actualidad, la mayoría de dispositivos móviles (teléfonos, PDAs, portátiles, etc) soportan operaciones criptográficas (aunque eso no significa que no tengan un consumo elevado). Sin embargo, los servicios PKIX están sujetos a la existencia de mecanismos para gestionar certificados digitales.

El resto del capítulo se organiza de la siguiente manera. En el apartado 6.2 se analizan las soluciones existentes para la gestión de la revocación de certificados en MANET. En el apartado 6.3 se presenta un mecanismo colaborativo para la validación de estado de certificados en MANETs, que tiene en cuenta las características especiales de este tipo de redes. Finalmente, se hace una primera aproximación de la evaluación de la propuesta utilizando el simulador de red ns-2 en el apartado 6.4.

6.2. Estudio de la gestión de certificados en MANET

En este apartado se presentan las soluciones para la gestión de certificados en PKI, ya que las soluciones para la gestión de revocación de certificados pueden ser válidas tanto para PKI como para PMI, tal y como se comentó en el capítulo 2. Por ello, de ahora en adelante se hablará de revocación de certificados en general.

6.2.1. Revocación en MANET

En la literatura hay varias propuestas basadas en criptografía de clave pública para gestionar la seguridad en MANET's. Difieren básicamente en el grado de descentralización de los mecanismos empleados para emitir, publicar y revocar certificados. En la mayoría de los esquemas de PKI descentralizados (46; 47), los nodos de la MANET emiten, publican y revocan ellos mismos los certificados. La gestión de certificados es completamente autónoma y autoorganizada porque no hay necesidad de ninguna autoridad de confianza o servidor fijo, ya que todos los nodos tienen el mismo rol. En este sistema, como en PGP (40), cada usuario es su propio emisor. Los certificados son almacenados y distribuidos por los nodos de manera completamente auto-organizada. Cada certificado se emite con un periodo de validez limitado y contiene sus fechas de emisión y expiración. Antes de que un certificado expire, el dueño puede emitir una

versión actualizada del mismo, que contiene un tiempo de expiración extendido. Al igual que en PGP, los nodos construyen caminos de confianza, desde un nodo a otro, como en un círculo de "amistad", formando un anillo de autenticación para conseguir las relaciones de confianza con otros nodos de la MANET.

Otro gran grupo de esquemas de PKI propuestos para MANET's están basados en la criptografía umbral (45). Este tipo de esquemas fue propuesto en primer lugar por Zhou y Haas en (43). La idea subyacente en estos esquemas es la de distribuir las tareas de certificación entre los nodos de la red. Un esquema umbral (k, n) permite que se divida la clave privada entre n partes, de forma que para un cierto umbral $k < n$, cualesquiera k nodos pueden combinar y obtener la clave de firma, mientras que para $k-1$ o menos nodos, esto es imposible. Usando esta técnica criptográfica, la clave de firma puede ser dividida en n partes y distribuida a n nodos. Por ejemplo, cualquier conjunto de k nodos puede colaborar para firmar y emitir certificados digitales válidos, o emitir datos de estado, mientras que una coalición de $k-1$ nodos, o menos, no podría. Este esquema requiere de una fase de inicialización en la cual una autoridad central asigna el rol a los n nodos que actuarán como servidores para la gestión de certificados. El resto de la gestión de certificados es completamente descentralizada.

Finalmente, la solución más centralizada es el uso de PKI X.509 (28). En esta solución, las autoridades centrales de confianza emiten, publican y distribuyen el estado (válido/revocado) de los certificados, de acuerdo con una metodología estándar bien definida. La PKI X.509 es la infraestructura estándar de seguridad que actualmente funciona en Internet y esto se considera lo suficientemente importante como para tener en cuenta la solución PKI también para escenarios MANET. Sin embargo, la PKI está diseñada principalmente para redes con conexión a los servidores PKIX centralizados, así que adaptar una PKI a un escenario de red MANET no es una tarea sencilla.

En la tabla 6.1 se resumen las características principales de las diferentes soluciones basadas en clave pública para MANET. Como se puede observar, la utilización de los certificados X.509 emitidos por una PKI tradicional tiene importantes ventajas, como la validez global de los credenciales y el soporte para la movilidad de usuarios. En muchos escenarios, podemos asumir que los usuarios pueden acceder a la PKI tradicional para obtener sus certificados. No obstante, la información actualizada del estado de revocación de los certificados, no está garantizada en las MANET's, ya que la conectividad a los servicios PKI de Internet pueden no estar disponibles.

El escenario que se está considerando ya se ilustró en el capítulo 3 (ver figura 3.1). Tal como mostraba la figura 3.1, se espera que los usuarios móviles se muevan a través de las diferentes redes, a veces con conexiones a los servicios de la PKIX y otras veces sin conexión. Cuando un usuario está en una red con conexión a la PKIX, siempre puede usar los servicios de la misma para conseguir un certificado, realizar una consulta de estado, etc. Por lo tanto, podemos asumir que los usuarios pueden obtener

o renovar sus certificados de una visita previa a una red conectada, o de una MANET durante el modo conectado. Sin embargo, los usuarios pueden estar desconectados de la PKIX cuando realmente necesitan un servicio. En este sentido, la comprobación del estado del certificado es un servicio de PKI crítico porque las aplicaciones deben comprobar, en el momento del uso, si un certificado es válido o si ha sido revocado antes de finalizar su periodo de validez. En los siguientes apartados se trata el problema de la obtención de la información de revocación en modo desconectado, y las soluciones existentes para aumentar la disponibilidad del servicio.

Tabla 6.1: Comparativa de los esquemas de gestión de clave pública para MANET

	Auto-certificación	**Criptografía Umbral**	**PKIX**
Grado de descentralización	Alta	Media	Baja
Infraestructura	No requerida	Se requiere una autoridad administrativa para inicializar los n servidores	PKI estándar funcionando actualmente en Internet
Disponibilidad	Recae en la posibilidad de crear un camino de confianza de los datos locales	Se requieren k nodos de los n funcionando correctamente	Sujeto a la disponibilidad de la conexión a los servidores centralizados que ofrecen servicios PKI
Aceptación de certificados en Internet	Baja	Baja	Alta
Revocación	Cada usuario revoca su certificado	Un certificado se revoca si existen k "acusaciones"de otros nodos	El propietario o emisor del certificado puede revocarlo
Validación de estado	Local El usuario considera válido el certificado si puede encontrar un camino de confianza actualizado en su caché local	Local Una credencial es válida si acumula menos de k "acusaciones"en la lista local del usuario	Global Depende del estado que publica la PKI
Validez de los certificados	Local	Local	Global
Soporte a la movilidad de usuarios	Baja Cada vez que un usuario entra en una nueva MANET ha de crear una nueva PKI propia	Baja Un usuario requiere que k servidores de la nueva MANET le emitan un nuevo certificado	Alta El usuario sólo necesita actualizar de forma periódica su certificado (emitido por un largo periodo de tiempo)

6.2.2. Adaptación de la revocación PKIX a MANET

CRL y OCSP están diseñados para redes con alta conectividad en las que los nodos que hacen las funcionalidades tanto de repositorios como de servidores *on-line* tienen direcciones de red conocidas y están siempre disponibles para los usuarios. Sin embargo, la MANET es una red dinámica en la que la topología de red cambia de forma aleatoria y en la que los usuarios móviles entran y salen de la red a lo largo del tiempo. Por lo tanto, son necesarios nuevos mecanismos para distribuir la información de revocación. Una solución puede ser la presentada en (111), la cual se basa en un esquema de caché. Los sistemas de caché *ad-hoc* tienen tres tipos diferentes de nodo: los nodos servidores, los nodos caché y los nodos clientes. Los nodos servidores son nodos que anuncian datos y pueden, o no, ser parte de la red, por lo que pueden estar no disponibles de forma temporal. Cuando los datos se anuncian, se almacenan en los nodos caché. Los nodos caché son nodos que almacenan y reenvían información. No todos los nodos de la red pueden ser nodos caché ya que conlleva un cierto consumo de recursos, por lo que se han de seleccionar en base a algún criterio, como puede ser: capacidad de almacenamiento, consumo de energía y movilidad de los mismos. Cuando un nodo cliente necesita cierta información emite una solicitud en la red. Si un nodo caché recibe esta solicitud busca la información solicitada en su caché, y si la encuentra, la devuelve al nodo cliente. Los nodos caché necesitan mantener un camino a los nodos servidores para recibir la información actualizada. Por otro lado, debido a los cambios frecuentes en la topología de la red, la MANET también necesita un mecanismo para descubrir nodos caché y nodos servidores. Por ejemplo, puede seguirse un protocolo de inundación parecido a DSR (*Dynamic Source Routing*) (112; 113) para realizar el descubrimiento. El protocolo podría funcionar del siguiente modo:

- Los nodos cliente envían solicitudes de información a la red.
- Los nodos intermedios que reciben estas solicitudes, y que no son nodos caché, las retransmiten hasta que se encuentra un nodo caché.
- Si el nodo caché tiene una respuesta que coincide con la solicitud realizada, envía la respuesta al nodo cliente utilizando el camino inverso. En otro caso, intenta obtener datos nuevos de un nodo servidor.
- Si el enlace al nodo servidor no está disponible en ese momento, la petición se difunde de nuevo hasta que se encuentra un nodo caché o un nodo servidor con la correspondiente respuesta, o hasta que se alcanza el número máximo de saltos para el servicio de caché.

Tal como se especifica en (111), el nodo cliente también debe poder especificar en su petición la antigüedad esperada de los datos de la respuesta. En este caso, sólo un nodo caché que haya actualizado sus datos recientemente podrá ser capaz de responder a la petición. Sino, la petición debería ser difundida hasta que se encuentre un nodo con información suficientemente reciente, o hasta que se alcance el número máximo de saltos. Otro aspecto interesante para el nodo cliente sería la posibilidad de especificar el tamaño máximo de la respuesta en la petición. Un nodo cliente puede usar esta característica para evitar quedarse sin capacidad de almacenamiento.

CRL en MANET .- En el caso de la CRL, los nodos servidores son los repositorios. Los repositorios tienen una conexión estable con la CA y pueden tener acceso a la versión más reciente de la CRL. Un nodo caché es un nodo que quiere colaborar en el servicio de distribución de la información de revocación y que tiene suficiente capacidad para almacenar una copia de la CRL. El nodo caché responde a la petición de estado de nodos clientes de la MANET. Notar que el nodo cliente que consigue una copia válida de la CRL puede convertirse en un nuevo nodo caché. Es más, un nodo caché que se mueve a otra MANET puede colaborar en la nueva red dando el servicio. En este sentido, la movilidad del usuario ayuda al servicio de revocación. En (111), los autores estudian la viabilidad de usar el método de inundación para distribuir la información de la CRL, y concluyen que los dos factores más importantes para que la inundación funcione correctamente son el número de nodos y la cobertura radio de un nodo. Este último parámetro se define como el rango de radio que la señal puede ser transmitida a los vecinos del nodo y viceversa. Los autores enfocan su trabajo en temas de cobertura pero no hacen simulaciones para otros aspectos importantes como el *throughput*, *overhead* o el consumo de memoria de almacenamiento teniendo en cuenta el tamaño de la CRL. En este sentido, es importante remarcar que la CRL crece con el tiempo, pueden hacerse grandes, y los nodos de la MANET tienen recursos de red, procesamiento y memoria limitados.

OCSP en MANET .- En OCSP los nodos servidores actúan como nodos servidores *on-line* dentro de la propia red. A pesar de esta posibilidad, se desaconseja el uso de nodos servidores OCSP ya que se supone que deben tener datos de estado actualizados. Para conseguir que los datos estén actualizados, los nodos servidores han de tener conexión a repositorios o servidores *on-line* fijos, pero esta conectividad no siempre está garantizada en una MANET. Por otro lado, un servidor OCSP es una autoridad de confianza que tiene una clave privada que ha de proteger frente a nodos maliciosos. No es apropiado tener un nodo servidor que esté expuesto a ataques y que puede no tener datos actualizados. En general, no es deseable incrementar el número de autoridades de confianza; cuantas me-

nos autoridades de confianza, menor es la posibilidad de que se comprometa la clave privada. Por otro lado, si se utilizan servidores OCSP móviles, es necesario definir un mecanismo para confiar en ellos, lo cual no es trivial. Con respecto a los nodos caché, se encargan de almacenar respuestas OCSP emitidas por nodos servidores y las distribuyen a nodos clientes cuando detectan una petición que coincide con los requisitos de antigüedad de los datos. En (113) se realiza una propuesta completa sobre un esquema de caché para OCSP en MANET, la cual se conoce como ADOPT (*Ad-hoc Distributed OCSP for Trust*).

6.2.3. Análisis de CRL y OCSP en MANET

En este apartado se analizan las principales ventajas e inconvenientes de los enfoques anteriores, concluyendo que ni CRL ni OCSP son completamente adecuados para la comprobación de estado de certificados en MANET.

La principal desventaja de CRL es que pueden crecer con el tiempo y llegar a tener un tamaño considerable, de hasta varios *megabytes* (83; 85; 96). Por lo tanto, distribuir una CRL a cada nodo que requiera comprobar la validez de los certificados, implica un gran gasto de recursos de la red, y requiere que los nodos tengan suficiente memoria. Estos problemas son muy importantes en el entorno MANET ya que los nodos tienen recursos de red, almacenamiento, procesamiento y memoria limitados. Por lo tanto, es poco eficiente difundir CRLs. Sin embargo, el enfoque de CRL es sin conexión lo cual es una característica muy útil para MANETs. La CRL contiene toda la información necesaria para saber el estado de todo el conjunto de certificados controlados por la CA durante un periodo de tiempo. También es muy fácil replicar CRLs, de manera que un nodo que tiene una CRL válida podría compartir esa información con otros nodos de la red durante el periodo de validez de la CRL, sin que sea necesaria que las autoridades de confianza conectadas estén accesibles.

Por otro lado, OCSP tiene la ventaja de propagar respuestas pequeñas, pero tiene algunas desventajas. En primer lugar, cuando un usuario necesita comprobar el estado de un certificado, el *responder* tiene que estar accesible o, en su defecto, se ha de poder encontrar una respuesta válida en un nodo caché. Además, los nodos caché pueden tener sólo información parcial del estado de los certificados. Por ejemplo, si nadie ha solicitado el estado de un determinado certificado anteriormente, es imposible encontrar la correpondiente respuesta en la caché de alguno de los nodos que forman la red. En segundo lugar, dependiendo del escenario, los requisitos de almacenamiento de los nodos caché pueden ser más restrictivos que para CRL. En OCSP, cada certificado puede tener una respuesta OCSP asociada, por lo que el número de respuestas en la caché de la red puede ser alto (dependerá de parámetros como el número de certificados, etc). Y por último, pero no menos importante, tal como hemos

explicado previamente, es deseable tener el menor número de autoridades de confianza posible.

6.2.4. Árboles de *Merkle*

Los sistemas basados en árboles de *Merkle* (MHT - *Merkle Hash Tree*) presentan características intermedias a las ofrecidas por CRL y OCSP (ver apartado 2.1.2). Son una buena opción cuando se requieren, por un lado, un ancho de banda inferior al requerido por CRL, y por otro lado, una carga computacional en el servidor menor a la requerida por OCSP debido a la validación de las respuestas.

La figura 6.1 representa un ejemplo de MHT. Los nodos del nivel inferior del árbol son llamados hojas. Las hojas representan los datos almacenados en el árbol. Las hojas adyacentes se concatenan y al resultado se le aplica una función de *hash* para formar los nodos del nivel inmediatamente superior. El proceso se repite de forma recursiva hasta alcanzar el nivel más alto, en el cual sólo existe un nodo llamado raíz y el cual es un resumen de todos los datos almacenados en el árbol.

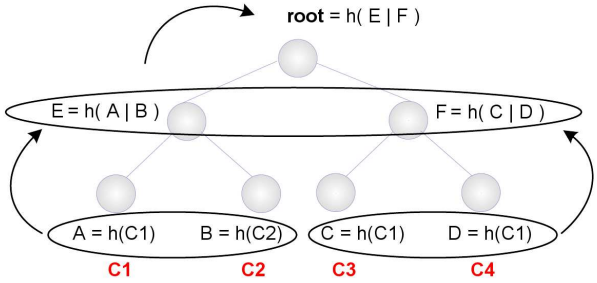

Figura 6.1: Generación de un MHT

En la figura 6.1, $C1$, $C2$, $C3$ y $C4$ representan las entradas de los cuatro certificados revocados, $h()$ es la función de *hash* y | indica concatenación. Ahora, asumiendo que la raíz está firmada por una autoridad de confianza, la integridad de una entrada obtenida de una autoridad que no es considerada de confianza, se puede comprobar utilizando una pequeña cantidad de datos. Por ejemplo, si se recibe C1 de un nodo caché que no se tiene configurado como de confianza, la integridad de $C1$ se puede verificar con tres valores: B, F y la raíz firmada. $C1$ se puede combinar con esos valores hasta conseguir la raíz firmada. Para conseguirlo, a $C1$ se le aplica una función de hash, el resultado se concatena con B, se aplica de nuevo la función de hash, el resultado se concatena con F y se aplica de nuevo la función *hash*. El resultado ha de coincidir con el valor de la raíz firmada. Como la raíz está firmada, se puede verificar su integridad y autenticidad, por lo que no es viable falsificar los valores de B y F, ni manipular alguna de las funciones de *hash*. Por lo tanto, una entidad que no es de

confianza puede proporcionar los valores B y F.

Como conclusión, un sistema que funcione sobre redes MANET, debería ser un sistema en el que: no existan autoridades de confianza conectadas, los nodos caché tengan todos los datos del estado de revocación, y el tamaño de las respuestas sea el menor posible. Un sistema basado en la combinación de una CRL y un árbol de *hash* puede conseguir estos objetivos.

6.3. Un sistema colaborativo de distribución de información de revocación

En este apartado se presenta un esquema colaborativo para la comprobación del estado de revocación en un escenario como el presentado en el capítulo 3. Para entender el sistema, en el apartado 6.3.1 se especifican los requisitos y la notación del escenario. A continuación se describe cómo la autoridad de confianza construye la estructura con los datos de revocación (apartado 6.3.2). A través de diferentes protocolos y procedimientos, un nodo que necesita verificar el estado de un certificado puede descubrir dónde se encuentra la información de revocación (apartado 6.3.3), obtener los datos de revocación (apartado 6.3.4) y convertirse en un nodo que proporciona información de revocación (apartado 6.3.5).

6.3.1. Requisitos del escenario

La solución propuesta está pensada para un entorno híbrido en el que un dispositivo móvil puede estar en ciertos momentos en modo conectado, y en otros puede encontrarse en modo desconectado.

Cuando un dispositivo móvil se encuentra dentro de la red MANET podría necesitar comunicarse con otro usuario, como puede ser el caso de acceder a un determinado servicio ofrecido por otro dispositivo. Como ya se ha comentado, en este tipo de entornos es importante saber si el dispositivo con el que se está comunicando es realmente quien dice ser. En este escenario, los dispositivos de usuario pueden ser teléfonos móviles, PDAs o portátiles. Este tipo de dispositivos se caracteriza por tener capacidades limitadas, sobre todo si los comparamos con computadores de sobremesa.

6.3.1.1. Notación

A continuación se introduce la notación necesaria para entender la propuesta, centrándonos en las entidades involucradas.

Nodo .- Cualquier dispositivo móvil actuando, ya bien sea como cliente o como servidor dentro de la red, se denominará a partir de ahora como *nodo*.

Nodo verificador .- Un nodo verificador es aquel nodo que requiere de la información de revocación para validar certificados.

Nodo proveedor .- Un nodo proveedor es aquel nodo de la red que posee información de revocación, y por lo tanto, puede realizar las funciones de servidor de información para los nodos verificadores. Un nodo verificador puede llegar a ser un nodo proveedor, tanto para sí mismo, como para otros nodos verificadores de la red. Dentro de esta definición se pueden encontrar dos roles:

- nodo repositorio.- Un nodo repositorio es un nodo proveedor que permite a otros nodos de la red la obtención de la lista completa del estado de revocación de los certificados.
- nodo *responder*.- Un nodo *responder* es aquel nodo que soporta un protocolo ligero para dar respuesta a las peticiones de verificación del estado de revocación realizadas por los nodos verificadores para un determinado certificado.

Nodo intermedio .- Un nodo intermedio es todo aquel nodo que no es ni origen ni destino en el servicio de revocación. El nodo se encarga de reenviar la información de revocación (peticiones, respuestas, etc).

RL .- Lista de certificados revocados (RL - *Revocation List*). Se utilizará la nomenclatura RL para referirnos tanto a una CRL como a una ACRL.

6.3.1.2. Requisitos

A continuación se enumeran una serie de requisitos que ha de cumplir nuestra propuesta:

1. Red y Conectividad.- Los nodos pueden estar desconectados de la infraestructura fija de gestión de los certificados durante periodos largos.

2. Infraestructura de certificación.- Con respecto a la infraestructura de gestión de certificados:

   - Cada autoridad raíz gestiona la información de revocación bajo su dominio. Eso no quiere decir que no pueda existir información de revocación de diferentes autoridades raíz dentro de la red.

- La distribución de la información de revocación se realizará a través de una lista de certificados revocados (RL), la cual contendrá información adicional para permitir la utilización de un protocolo de obtención de información de revocación y de validación adecuado a las limitaciones del escenario.

3. Eficiencia.- El objetivo es alcanzar la máxima eficiencia de la solución desde dos puntos de vista:
   - La solución, siguiendo la tendencia generalizada en MANET, será completamente reactiva. Es decir, si el uso del servicio de revocación es nulo, el tráfico generado también ha de ser nulo.
   - Los paquetes que transporten la información de revocación entre el proveedor y el verificador, han de ser de pequeño tamaño.
4. Disponibilidad del servicio de revocación.- Se pretende conseguir un elevado grado de disponibilidad del servicio sin reducir la eficiencia del sistema:
   - Cualquier nodo de la red con recursos suficientes debería poder convertirse en un nodo proveedor.
   - Si en algún momento algunos de los nodos de la MANET posee una RL, el servicio estará disponible y cualquier nodo podrá hacer consultas de estado. La RL debe poder replicarse para generar redundancia y aumentar la disponibilidad del servicio, pero una replicación sin control puede llevar a una solución ineficiente, ya que se ha de tener en cuenta que tanto la red como los dispositivos pueden tener recursos limitados (restricciones de batería, capacidad de cómputo y almacenamiento). Por lo tanto, esta operación debería poder realizarse de forma controlada.

Por lo tanto, la autoridad raíz u otra entidad de confianza autorizada (TTP) emite RLs, las cuales incluyen una extensión donde figura la estructura criptográfica necesaria para dar el servicio. Un nodo que durante un periodo de conectividad haya conseguido una RL, puede actuar como nodo proveedor de datos de revocación. A través de diferentes protocolos, un nodo que necesita verificar el estado de un certificado puede descubrir dónde hay un nodo proveedor (apartado 6.3.3), obtener los datos de revocación (apartado 6.3.4) y convertirse en nodo proveedor (apartado 6.3.5).

6.3.2. RL extendida: Definición de la estructura y generación de los datos de revocación

El esquema que se propone está basado en listas de certificados revocados (RL). Utilizando la funcionalidad del campo extensiones de la RL, se define una nueva ex-

tensión que contendrá información necesaria para llevar a cabo un protocolo de obtención de información de revocación ligero basado en árboles de *Merkle*. A la estructura que contiene esta nueva extensión la llamaremos a partir de ahora, eRL (*Extended Revocation List*). Esta estructura, que es completamente compatible con la definición X.509 de la ITU-T (28) y del IETF (39; 52), puede ser almacenada en los nodos proveedores. La eRL contiene toda la información de revocación necesaria para que el nodo que la contiene pueda funcionar como nodo *responder* y como nodo repositorio.

A continuación se explica con más detalle, la estructura de la eRL, centrándonos en su definición y generación por parte de la TTP.

6.3.2.1. Estructura eRL

Para poder usar un árbol MHT (*Merkle Hash Tree*), el nodo proveedor que va a dar el servicio necesita, además de la lista de certificados revocados, el valor del nodo raíz del árbol de Merkle firmado por la TTP. Este valor se ha de incluir en la respuesta para que el nodo verificador pueda comprobar la autenticidad e integridad de la información.

La TTP ha de proporcionar la información suficiente para que tanto un nodo proveedor como un nodo verificador puedan reconstruir el árbol de Merkle. Para ello, la autoridad ha de incluir en la RL una extensión no crítica, que permita a las entidades reconstruir el árbol de Merkle y poder verificar su autenticidad. La definición en ASN.1 de la estructura de la extensión es la siguiente:

```
Extension ::= SEQUENCE {
        extnID          OBJECT IDENTIFIER,
        critical        BOOLEAN DEFAULT FALSE,
        extnValue       OCTET STRING
   }
-- TreeDigest extension OID and syntax
id-ce-TreeDigest OBJECT IDENTIFIER ::= { id-ce TD }
```

Una extensión de la RL está formada por: 1) un identificador, el cual permite determinar el tipo de contenido, 2) un campo que indica si la extensión es crítica (ha de ser no crítica) y 3) el valor de la extensión. En nuestro caso el valor de la extensión es la estructura `TreeDigest` la cual está formada por los siguientes campos:

- `issuerHash` y `issuerKeyHash`: el valor de *hash* del DN y la clave pública de la autoridad emisora de los datos de revocación. Estos campos son opcionales, ya que tanto el nodo proveedor como el nodo verificador pueden obtenerlos del certificado de la autoridad.
- `thisUpdate` y `nextUpdate`: indican el periodo de validez de los datos del árbol MHT. Normalmente coinciden con el periodo de validez de la RL y por este

motivo son opcionales. En el caso de que no se hayan producido nuevas revocaciones (aunque sí expiraciones), en uno o más periodos de tiempo, la autoridad emisora puede optar por extender el periodo de validez de estos datos. Para ello envía el periodo de validez posterior a la última revocación y así mantiene los datos del MHT intactos (y por lo tanto su raíz). Para extender este periodo de validez, la autoridad utiliza el campo `currentUpdateValue` (ver apartado 6.2.4).

- `digestAlgorithm`: algoritmo de *hash* utilizado para obtener el valor de los nodos del árbol.
- `rootHash`: *hash* de la raíz del árbol MHT.
- `baseUpdateValue`: valor distribuido en la generación inicial de los datos de revocación.
- `signature`: cadena de bits que representa el valor de la firma de la estructura `TBSTreeDigest`. Esta estructura incluye todos los campos mencionados hasta ahora, junto al `signatureAlgorithm`, que es el mismo identificador de algoritmo de firma que especifica la autoridad en su certificado.

Si un nodo de la MANET sabe interpretar la extensión, entonces puede crear de nuevo la estructura MHT (ver apartado 6.3.2.2). Si el nodo no sabe interpretar la extensión, puede seguir utilizando la RL de forma estandarizada.

```
   TreeDigest ::= SEQUENCE {
           tbsTreeDigest        TBSTreeDigest,
           signatureAlgorithm   AlgorithmIdentifier,
           signatureValue       BIT STRING
   }

-- Digest signature uses the same algorithm as RL signature

   TBSTreeDigest ::= SEQUENCE {
        issuerHash              BIT STRING OPTIONAL,
        issuerKeyHash           BIT STRING OPTIONAL,
        thisUpdate              Time OPTIONAL,
        nextUpdate              Time OPTIONAL,
        digestAlgorithm         AlgorithmIdentifier,
        rootHash                BIT STRING,
        baseUpdateValue         BIT STRING,
        signatureAlgorithm      BIT STRING,
   }
```

6.3.2.2. Generación de la información de revocación

Una TTP (autoridad emisora de la eRL) es la encargada de generar la información de revocación. Esta información la podemos dividir en dos. Por una parte está la lista de certificados revocados, la cual se genera siguiendo los pasos definidos por el IETF. Por otra parte, se ha de incluir la información necesaria del árbol MHT en la nueva extensión de la RL. Básicamente la extensión `TreeDigest` contiene el valor de la raíz del árbol de Merkle, firmado por la autoridad.

A modo de ejemplo, la autoridad crea la eRL realizando los siguientes pasos (ver figura 6.2):

(1) La autoridad crea la estructura ASN.1 `ToBeSigned` (ver apartado 6.3.2.1) de la RL en la que incluye los identificadores de los certificados revocados que todavía no han expirado.

(2) La autoridad crea la extensión. Para ello, crea el árbol MHT utilizando las entradas de su base de datos de certificados revocados. A continuación se explica de forma sencilla el procedimiento básico para la creación de un árbol de *hash* (ver figura 6.2):

   a. Cada hoja del árbol representa un certificado revocado. Los nodos A, B, C y D representan, en orden ascendente, a cuatro certificados revocados, $C1$, $C2$, $C3$, y $C4$ respesctivamente.

   b. Las hojas adyacentes se concatenan y se aplica una función de *hash* sobre el valor concatenado. El resultado de la función de *hash* forma el nodo del árbol del nivel inmediatamente superior, por lo tanto $h(A \mid B) = E$ y $h(C \mid D) = F$.

   c. El proceso anterior se repite de forma recursiva hasta que se tiene un único nodo del árbol en el nivel superior (nodo raíz). Por lo tanto, el nodo raíz contiene un resumen de todos los datos almacenados en el árbol. En el caso del ejemplo, $h(C \mid D) =$raíz.

(3) La extensión contiene, en líneas generales, el valor de la raíz y una firma. Esta extensión tiene el mismo periodo de validez que la RL que la contiene (vp_{RL}- estructura ASN.1). Hay que resaltar que a pesar de que la RL va firmada, la extensión también ha de ser una estructura firmada. De esta manera, se puede proporcionar autenticación e integridad al valor de la raíz del árbol de MHT, y así poder ser utilizada por cualquier nodo de la MANET, con independencia de si se posee o no la RL. La firma de la extensión es necesaria, ya que un nodo verificador sólo dispondrá de la extensión firmada para la validación de los datos de revocación. Por lo tanto, el valor de la raíz es la información necesaria para poder reconstruir la versión autenticada del árbol de hash (ver apartado 6.3.4).

(4) La extensión se incluye en la estructura `TBSCertList` de la RL. Como ya se ha comentado, esta extensión ha de ser no crítica.

(5) La autoridad genera la eRL firmando la estructura `TBSCertList` de la RL.

(6) Por último, la autoridad distribuye copias de la eRL a los repositorios designados, de la misma manera que se haría con la RL estándar. En este momento, cualquier entidad que necesite verificar el estado de un certificado, puede acceder al repositorio y obtener la información.

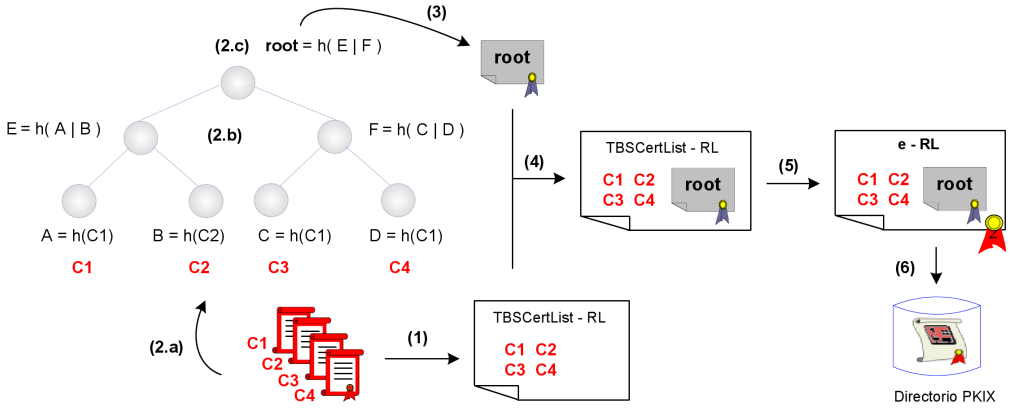

Figura 6.2: Generación y publicación de la eRL por la autoridad

Después de este proceso, los repositorios almacenan una eRL la cual tiene exactamente la misma información que la RL estándar. La eRL contiene, por una parte, la información necesaria para reconstruir el árbol de hash, y por otra, la extensión `TreeDigest` con el valor de la raíz firmada por la autoridad, la cual es válida para todo el periodo de validez de la eRL. La inclusión de la extensión no comporta un aumento significativo en el tamaño total de la RL, como se verá en el apartado 6.4.

6.3.3. Descubrimiento y selección del servicio de revocación

Una vez existe un nodo proveedor en la red, un nodo verificador ha de poder localizar al nodo proveedor. En redes con acceso a Internet la localización de los servicios suele ser sencilla. Por ejemplo, los certificados X.509 pueden utilizar la extensión `authorityInfoAccess` para proporcionar la URL en la que se puede encontrar información de la TTP que emitió el certificado (28). Sin embargo, en entornos donde no se sabe a priori quién puede proporcionar los datos de revocación en la red, es necesario incluir algún mecanismo que permita localizarlo de forma dinámica. Por lo tanto, una vez un nodo ha obtenido los datos de revocación a partir de la infraestructura de

la PKIX y entra a formar parte de una MANET, el nodo proveedor puede decidir compartir los datos de revocación con otros nodos de la red que lo necesiten. En general, los nodos de la red han de ser capaces de descubrir y compartir servicios de forma dinámica sin depender de una infraestructura fija. Para llevar a cabo este proceso, los nodos utilizan un protocolo de descubrimiento de servicio (SD - *Service Discovery*). SD puede ser simplemente explicado como un protocolo que permite localizar y/o anunciar servicios en una red. En otras palabras, SD permite a un dispositivo localizar servicios ofrecidos por otros dispositivos, sin tener un conocimiento previo de su localización.

Para el funcionamiento de la eRL en MANET, se ha propuesto utilizar el protocolo de descubrimiento PDP (*Pervasive Discovery Protocol*) para localizar los nodos que funcionan como proveedores[1]. PDP es un protocolo completamente distribuido que ha sido especialmente diseñado para entornos como las MANET. Una de las principales ventajas de PDP es que permite reducir el consumo de batería de los dispositivos. Para ello, prioriza las réplicas de descubrimiento de servicio de los nodos con un mayor número de recursos. Para más información sobre PDP se puede consultar (68; 114).

La búsqueda del nodo proveedor ha de tener asociada unos ciertos atributos de búsqueda (apartado 6.3.3.1), es decir, si un nodo verificador intenta validar el estado de revocación de un certificado, siempre es aconsejable tener la información más actualizada posible. Cuando un nodo verificador requiere obtener información de revocación, es posible que obtenga información de más de un nodo que ofrezca el servicio que se solicita. Entre las respuestas, el nodo verificador podrá seleccionar el nodo proveedor que ofrezca la información más actualizada (apartado 6.3.3.2). Es evidente que la disponibilidad de datos de revocación actualizados dependerá de la conectividad de los nodos proveedores a la infraestructura de la PKIX. Por lo tanto, no habrá información de revocación disponible en la MANET si nunca se ha tenido conectividad con la PKIX, o si los nodos proveedores han abandonado la red.

6.3.3.1. Descripción del servicio

Para poder seleccionar el servicio adecuado se han de poder especificar, de alguna manera, los atributos que definen el servicio. Los atributos que especifican el servicio de revocación los podemos dividir en función de si están incluidos en la petición o en la respuesta de descubrimiento del servicio, de la siguiente manera:

1. Incluidos en la petición de descubrimiento de servicio.- Los atributos incluidos en la petición describen el servicio que se ha de localizar. En este caso, se han definido

[1] Este protocolo es el mismo protocolo de descubrimiento reactivo utilizado en la ePKI (apartado 3.6).

cuatro atributos:

- `hashAlgorithm`: identificador de objeto que define el algoritmo de *hash* utilizado para obtener el valor de los atributos tipo `issuerNameHash`.
- `issuerNameHash`: especifica el valor de *hash* del `DistinguishedName` de la entidad emisora reconocida para la expedición de los datos de revocación que se solicitan.
- `issuerKeyHash`: especifica el valor de *hash* de la clave pública de la entidad emisora reconocida para la expedición de los datos de revocación que se solicitan.
- `riskThreshold`: especifica el valor de riesgo máximo que se acepta para la información de revocación que se necesita en la validación. El riesgo hace referencia a la caducidad de la información de revocación. Una especificación del cálculo del valor de riesgo se puede encontrar en (115).

2. Incluidos en la respuesta de descubrimiento del servicio.- Los atributos incluidos en la respuesta describen el servicio ofrecido por el nodo proveedor. El nodo puede especificar ciertos atributos que fueron definidos para el mensaje de petición, como son: `hashAlgorithm`, `issuerNameHash`, `riskThreshold`, `issuerKeyHash`. Además de los anteriores, se han definido los siguientes atributos:

   - `serviceUDPPort`: especifica el valor del puerto UDP del nodo proveedor en el que se reciben las peticiones de información de revocación.
   - `statusDataVP`: especifica el periodo de validez de los datos de revocación. Este valor permite que el nodo verificador pueda seleccionar el servidor que ofrezca los datos de revocación más actuales.
   - `serviceTimeout`: especifica el tiempo estimado durante el cual el nodo proveedor puede ofrecer el servicio.
   - `serviceType`: especifica el tipo de servicio que puede ofrecer el nodo proveedor (0 - repositorio, 1 - *responder*, 2 - ambos).

6.3.3.2. Selección del servicio de revocación

El protocolo de descubrimiento de servicio puede llevar a que un nodo verificador reciba más de una respuesta, es decir, se obtenga información de más de un nodo proveedor, o que dentro de la misma respuesta haya información sobre más de un nodo proveedor. Cuando el nodo verificador recibe las respuestas, ha de verificar que la información ofrecida por los proveedores cumple con los atributos especificados en la petición. Es decir, si la información de estado solicitada es de una autoridad reconocida para la emisión de datos de revocación, o si el valor de riesgo asociado a la

información de revocación es igual o inferior al `riskThreshold` especificado. El riesgo asociado a la información de revocación es una función que depende del tiempo. Es decir, el riesgo de los datos de revocación aumenta cuanto más tiempo haya transcurrido desde su emisión.

A partir de la información recibida, se selecciona el nodo proveedor de la siguiente manera:

- Nodos proveedores que ofrezcan la información de revocación más actualizada, es decir, los que ofrecen un valor de riesgo menor, o un valor menor o igual al especificado en la petición.
- Dentro de los nodos que ofrezcan la información más actualizada, es más adecuado seleccionar los que se encuentren disponibles a un menor número de saltos. A mayor distancia en número de saltos, mayor consumo de recursos en los nodos intermedios, y mayor probabilidad de no poder acceder al servicio (tanto los nodos proveedores como intermedios pueden cambiar de posición o abandonar la red).
- Los que ofrezcan un tiempo de servicio suficiente para obtener la información, ya que existe una mayor probabilidad de que el nodo continúe ofreciendo el servicio en el momento de realizar la petición.

6.3.4. Obtención y verificación del estado del certificado

Una vez se ha seleccionado un nodo proveedor (apartado 6.3.3.2), se pueden seguir dos caminos: 1) obtener la eRL completa, y 2) iniciar el protocolo ligero de revocación. El primer caso es idéntico al procedimiento para replicar la eRL en la red, por lo que se explicará con más detalle en el apartado 6.3.5. En el segundo caso, el nodo inicia el protocolo de verificación de estado RETRIP (*RETrieval Revocation Information Protocol*). Este protocolo está basado en MHT y utiliza sólo una parte de los valores del árbol para ser enviados al nodo verificador. A continuación se detallan los pasos a seguir en el proceso de validación (ver figura 6.3), para ello se toma como ejemplo el caso en el cual un nodo quiere verificar si el certificado identificado por $C1$ en la figura, está o no revocado:

(1) El nodo utiliza PDP para localizar los repositorios de revocación existentes en la MANET.

(2) Una vez localizados, selecciona el más adecuado en ese momento utilizando los criterios especificados en el apartado 6.3.3. Si el nodo obtiene los datos de un nodo proveedor de la red, ha de seguir lo pasos definidos en el apartado 6.3.5.1, teniendo en cuenta que en este caso es el nodo verificador quien inicia la comunicación.

(3) El nodo verificador envía al nodo *responder* el identificador del certificado a verificar, en este caso $C1$.

(4) El nodo *responder* busca el identificador del certificado ($C1$) dentro del árbol de revocación que previamente ha generado. El identificador $C1$ es una hoja del árbol, por lo que el nodo *responder* envía al nodo verificador un mensaje de respuesta que contiene:

   a. los valores de *hash* necesarios para que pueda generar el valor del nodo raíz; en este caso necesitará el valor de *hash* de $C2$ (B) y el *hash* resultante de la concatenación de los identificadores $C3$ y $C4$ (F),

   b. el valor de la raíz firmada contenida en la extensión de la eRL.

(5) Con los datos anteriores, el nodo *responder* genera y envía un mensaje de respuesta (RRESP) al nodo verificador (la estructura del mensaje fue definida por otros autores en (116)). Para dar mayor robustez al sistema, se pueden aplicar técnicas de retransmisión de la petición de información de revocación (ver apartado 6.3.5.1).

(6) El nodo verificador valida los datos recibidos, realizando los siguientes pasos:

   a. Genera el *hash* del identificador del certificado a validar: $A' = h(C1)$.

   b. Concatena el resultado del paso anterior con el valor B obtenido en la respuesta (RRESP), y genera un nuevo valor de hash: $E' = h(A'|B)$.

   c. El resultado anterior (E') se concatena con el valor F obtenido también en la respuesta (RRESP), y genera un nuevo valor de *hash* que se corresponde con el valor raíz del árbol de revocación: $root' = h(E'|F)$.

   d. Los datos se consideran correctos si el valor obtenido $root'$ coincide con el valor del $root$ proporcionado en la respuesta.

A pesar de que la eRL va firmada, y por lo tanto la extensión está autenticada, es necesario firmar de forma independiente el contenido de la extensión. Esto es así porque el nodo verificador ha de poder verificar la autenticidad de las respuestas proporcionadas por el nodo *responder*, sin que se haya de transmitir la eRL completa.

6.3.5. Gestión de los datos en el nodo proveedor

Una vez un nodo obtiene la eRL desde un repositorio centralizado, éste podría convertirse en un nodo proveedor dentro de la red. Imaginemos que un usuario (Alicia) ha accedido al repositorio y ha obtenido la última eRL emitida por la TTP. Posteriormente, entra a formar parte de una MANET en la que se ha perdido temporalmente la conexión. En este punto, el dispositivo de Alicia podría convertirse en un nodo

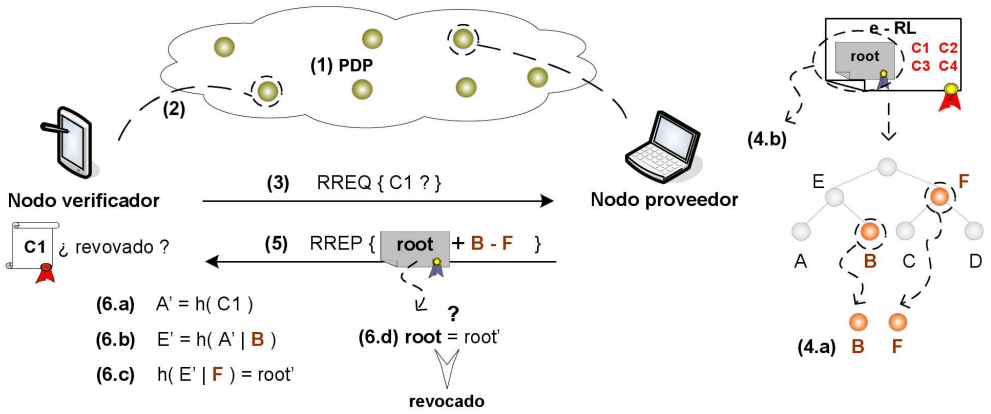

Figura 6.3: Proceso de descubrimiento, obtención y validación de la información de revocación

repositorio para la red MANET. El sistema permite que cualquier nodo de la MANET pueda llegar a ser un repositorio. Aunque cualquier nodo podría ser un repositorio, los nodos más adecuados son aquellos que poseen mayores capacidades, como puedan ser, portátiles o potentes PDAs. Esto es lo aconsejable, ya que los nodos que ofrezcan el servicio de repositorio deberían tener suficiente memoria para poder crear la estructura en árbol, y suficiente batería para responder a las peticiones del resto de nodos en la red.

Para poder ser un nodo repositorio, el nodo ha de seguir los siguientes pasos:

1. El nodo ha obtenido previamente la eRL por uno de los siguientes caminos: 1) un repositorio PKIX a través de una conexión con la red fija, o 2) un repositorio de revocación existente en la MANET.

2. Con la eRL obtenida, genera el árbol de *hash* siguiendo los mismos pasos que la TTP (apartado 6.3.2).

3. Una vez generado el árbol, comprueba que el valor de la raíz obtenida coincide con el valor de la raíz contenido en la extensión de la eRL.

4. En este punto, el nodo actuando como un repositorio de revocación puede empezar a escuchar las peticiones de estado de revocación por parte de otros nodos.

El árbol no tiene que volver a ser generado hasta que no exista una nueva eRL generada por la TTP, por lo que no ha de haber una sincronización entre la TTP y los repositorios.

6.3.5.1. Replicación

El proceso de replicado sirve para proporcionar mayor robustez al esquema en caso de que un nodo caiga o abandone la red. Aumentando el número de copias, se aumenta la disponibilidad del servicio. Sin embargo, la lista completa de datos de estado puede tener un coste considerable en recursos de red. Por lo tanto, la transmisión de la eRL debe realizarse de forma eficiente, teniendo en cuenta las características de la red y la utilidad de los datos. Es aconsejable que el proceso de replicado se haga a nodos que están a sólo un salto de distancia, para evitar el consumo excesivo de recursos en nodos intermedios.

Un nodo proveedor envía la eRL a un nodo verificador sólo si el nodo verificador que realizó la petición solicita de forma explícita obtener la eRL. Un nodo proveedor puede ejecutar PDP para localizar a los posibles nodos que ofrecen la posibilidad de convertirse en nodos proveedores. Una vez seleccionado, puede ejecutarse el protocolo de replicado (REPLIP - *REPLIcation Protocol*) realizando los siguientes pasos:

1. El nodo verificador manda una petición de replicación `PREQ` al nodo proveedor.

2. Si el nodo proveedor acepta enviar la eRL al nodo verificador, este último envía el paquete `PDATA`.

```
PDATA ::= SEQUENCE{
  eRL          CertificateRevocationList, Optional
  code         ResponseCode
}

ResponseCode ::= ENUMERATED{
  successful           (0), -- Accepted request
  malformedRequest     (1), -- Illegal request
  internalError        (2), -- Internal error in verifier
                       (3),(4) and (5) are not used
  notAskedFor          (6)  -- Replication not asked for
}
```

3. El nodo verificador iniciará un contador con valor `REPLIP_DATA_RTO` al comienzo de la transmisión. Una vez agotado el contador, si no se ha recibido respuesta alguna del nodo proveedor, se reintentará la comunicación enviando un paquete `PREQ`. Este proceso se repetirá un número `REPLIP_MAX_RT` de veces. Agotadas estas repeticiones, si la transmisión no se ha realizado con éxito, se aborta el proceso de replicación.

4. Una vez finalizada la transmisión con éxito, el nodo verificador ya puede realizar las funciones de nodo proveedor si así lo desea.

6.3.5.2. Actualización

Los nodos de la MANET pueden almacenar la validez de los certificados con los que trabajan, almacenando respuestas `RREP` de peticiones anteriores. El tiempo que pueden almacenarse estos datos es proporcional al riesgo que implica seguir usando un certificado que puede que ya haya sido revocado. Por lo tanto, es conveniente poder actualizar los datos de revocación. La actualización se hace de dos maneras:

1. Utilizando una petición directa a un nodo proveedor, tal como se ha explicado en el apartado 6.3.4. En ésta se incluye el valor de `rootHash`, y si el árbol no ha variado, el nodo proveedor proporciona el valor `currentUpdateValue` (apartado 6.2.4).

2. Cuando el nodo forma parte de una difusión de una petición de servicio puede examinar los posibles mensajes `RREP`. Estos mensajes viajan del nodo proveedor al nodo verificador pasando por nodos intermedios. Los nodos intermedios pueden utilizar esa información para verificar los valores `rootHash` y `currentUpdateValue` que transporta la `RREP`, y poder actualizar su propia información. De esta manera no se sobrecarga demasiado la capacidad de cómputo del nodo y se puede comprobar si puede actualizar su información cacheada.

De estas dos maneras se actualiza la información sin generar la misma cantidad de tráfico que si se desconociera por completo el estado de la validez de los certificados.

6.3.6. Análisis de seguridad

En este apartado se clarifican algunos de los aspectos de seguridad relacionados con eRL, como son: la modificación de la información de revocación y el envío de información obsoleta.

- Alteración de la información.- Un repositorio malicioso no puede distribuir información falsa. En el supuesto de la distribución de información falsa, pueden presentarse dos situaciones:

    a. En el caso en el que un certificado esté revocado, el nodo proveedor envía la información con los valores de *hash* y de los identificadores necesarios para que el nodo verificador pueda calcular el valor de *hash* de la raíz. Como el valor del $root$ está firmado por la TTP, el nodo verificador puede saber si los valores han sido modificados. Por lo tanto, el fallo de seguridad recaería en todo caso en la viabilidad de romper la función de *hash* utilizada. Es decir, conseguir modificar el resto de valores para obtener un valor del $root$ igual al generado por la TTP.

b. En el caso en el que un certificado no esté revocado y el nodo verificador ha enviado la información que demuestra que el certificado no está revocado, a pesar de estarlo. En este caso, como las hojas del árbol están en orden ascendente, y el nodo verificador tiene el valor de la raíz firmada, no es viable para el nodo proveedor modificar los valores de forma adecuada para que se pueda obtener el mismo valor que el *hash* de la raíz firmada.

- Envío de información caducada.- El periodo de validez es un campo generado por una TTP, y se incluye tanto en la RL, como en la extensión que contiene el valor del *root* del árbol MHT. Ambas estructuras están firmadas, con lo cual, el nodo verificador puede detectar si la información contenida ha sido modificada.

6.4. Análisis de Prestaciones

En este apartado se presentan los resultados preliminares de la evaluación del protocolo RETRIP. Para llevar a cabo las simulaciones, se ha utilizado el simulador de red ns-2 (*Network Simulator 2*)(117). ns-2 es un simulador de eventos discretos que permite la simulación de la gran mayoría de topologías de red. Es un simulador ampliamente utilizado a nivel de investigación tanto a nivel general de protocolos de redes como en particular para soluciones de PKI en MANET, por lo que se ha escogido para simular nuestra solución.

6.4.1. Desarrollo en ns-2

Para estudiar las prestaciones de la solución propuesta (eRL), se ha implementado el comportamiento de los diferentes nodos (nodo intermedio, nodo proveedor y nodo verificador) tanto en el caso de utilizar la distribución de la RL como en el caso de utilizar RETRIP. Para simular el comportamiento del sistema se han desarrollado dos componentes nuevos que se han integrado en el simulador:

- Aplicación para la gestión de los datos de estado de revocación.
- Un agente para el transporte de la información sobre UDP.

En ns-2 los agentes se encargan de gestionar la capa de transporte. A pesar que ns-2 tiene implementado un agente para la gestión de UDP, no tiene implementada la función para pasar los datos recibidos hacia la capa de aplicación, es decir, está pensado únicamente para tomar medidas de funcionamiento de UDP. Por lo tanto, ha sido necesaria la implementación de un nuevo agente para gestionar de forma adecuada la información recibida y/o enviada.

En el caso de la aplicación, se ha desarrollado el comportamiento del nodo proveedor y nodo verificador en una misma aplicación. Su comportamiento vendrá determinado por la configuración inicial de la aplicación, utilizando los siguientes parámetros:

- `RD_size` (*Revocation Data size*).- Especifica el tamaño de los datos de revocación: la RL o el mensaje de respuesta RETRIP.
- `eRL_exptime`.- Tiempo de validez de la eRL.
- `lambda`.- Define la tasa de peticiones del servicio de revocación cuando está habilitado RETRIP. Es decir, peticiones de estado de revocación por periodo de validez de la eRL, cuando se utiliza el árbol de revocación.
- `Provider`.- Este parámetro indica si el nodo va a realizar la función de nodo proveedor de la información de revocación, o va a asumir el rol de nodo verificador.
  - Si `Provider=1`, el nodo tiene las funcionalidades de un nodo proveedor por lo que sólo necesita utilizar el valor del parámetro `RD_size`.
  - Si `Provider=0`, el nodo tiene las funcionalidades de un nodo verificador por lo que necesita utilizar el valor de los parámetros `eRL_exptime` y `lambda`.

6.4.2. Simulaciones

La tabla 6.2 muestra los parámetros generales que se van a utilizar en la simulación. En términos globales se va a considerar un escenario en el que existe un nodo que realiza la funcionalidades de nodo proveedor, tanto para el caso RL general, como para el caso RETRIP. En el escenario considerado, los nodos se mueven dentro de un área limitada utilizando un canal *wireless* 802.11b, y en el que se utiliza un protocolo de encaminamiento reactivo como AODV (*Ad-hoc On Demand Distance Vector*) (118).

En este escenario, se va a considerar el caso más simple en el que sólo existe un nodo que realiza las funciones de proveedor y el número de nodos en la red varía entre 10 y 50, de los cuales, excepto el nodo que distribuye la información de revocación, el resto realiza las funciones del nodo verificador. Con este escenario y los parámetros de configuración de la tabla 6.2, se van a tomar medidas del *throughput* consumido por:

- el nodo proveedor, para estimar la carga que supone la existencia de un único servidor de información de revocación en toda la red. Este nodo siempre tendrá información de revocación actualizada de la eRL, por lo que podrá proporcionar datos de revocación actualizados durante todo el tiempo de simulación.

Tabla 6.2: Parámetros generales de la simulación

área (m,m)	(150 x 150)
número de nodos MANET (NumN)	10,20,30,40,50
velocidad máxima del nodo (m/s)	2
rango de transmisión (m)	60
modelo de movilidad	Random Waypoint
MAC	IEEE 802.11b
velocidad de transmisión	11 Mbps
protocolo de transporte	UDP
protocolo de enrutamiento	AODV
población de certificados	100000
certificados revocados (%)	10
periodo de validez de la eRL (s)	120
peticiones de estado por periodo de validez	2

- el resto de nodos, para estimar la carga que supone el servicio distribuido de revocación. Hay que tener en cuenta que durante una petición, la mayoría de nodos realizan funciones de encaminamiento.

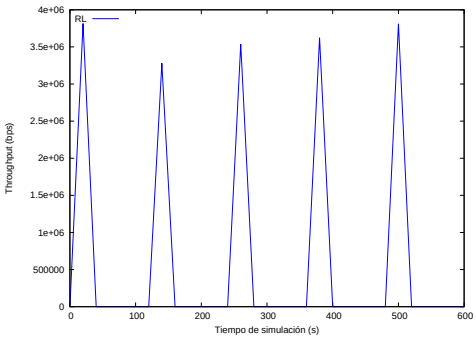

Figura 6.4: *Throughput* en el nodo proveedor con RL

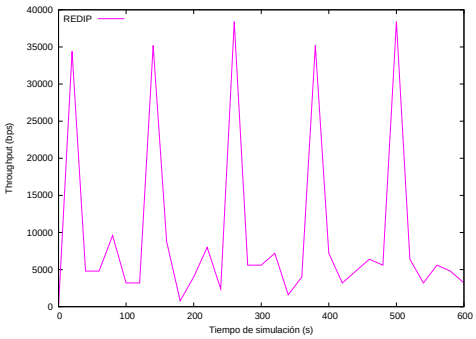

Figura 6.5: *Throughput* en el nodo proveedor con RETRIP

Las figuras 6.4 y 6.5 muestran el *throughput* instantáneo en el nodo proveedor en el caso de transmisión de la eRL y en el caso de utilizar RETRIP, respectivamente. Como se puede observar, el *throughput* cuando se utiliza RETRIP es aproximadamente de unas decenas de *kbps*, mientras que en el caso de enviar la RL completa, se obtiene un consumo de pico de algunos *Mbps*. El efecto de los picos es debido a la obtención de la RL. Hay que recordar que los nodos verificadores almacenan la RL hasta que finaliza su periodo de validez. Por lo tanto, los nodos no realizan nuevas peticiones hasta que la RL almacenada haya expirado. Este es el motivo por el que después del periodo de validez de la eRL ($120s$), el nodo repositorio recibe el mayor número de peticiones de obtención de la RL.

Como el canal que se está considerando es *wireless*, los nodos se mueven durante toda la simulación, lo que influye en el rendimiento del sistema, por lo que también

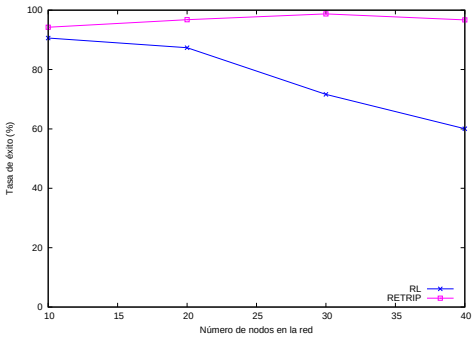

Figura 6.6: Disponibilidad servicio de información de revocación

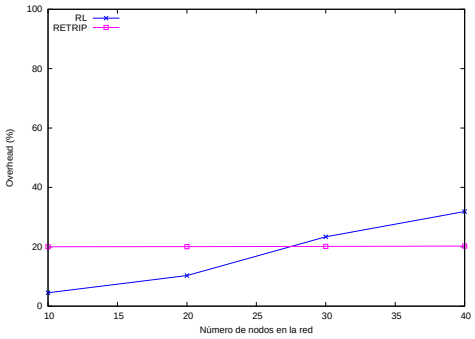

Figura 6.7: *Overhead* en la red debido a la tasa de pérdidas

se ha evaluado la disponibilidad del servicio de revocación y el *overhead* en los nodos. La disponibilidad del servicio viene determinada por la probabilidad de errores y de disponibilidad del canal, y por la movilidad de los nodos. Como puede apreciarse en la figura 6.6, RETRIP tiene una disponibilidad cercana al 100 %, mientras que en RL, la disponibilidad se reduce con el aumento del número de nodos verificadores. Si se aplican técnicas de retransmisión (ver apartado 6.3.4), ambos sistemas mejoran, aunque la mejora se hace más evidente en RL (ver figura 6.8).

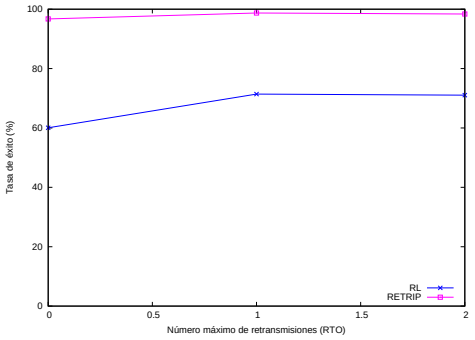

Figura 6.8: Disponibilidad servicio de información de revocación con RTO

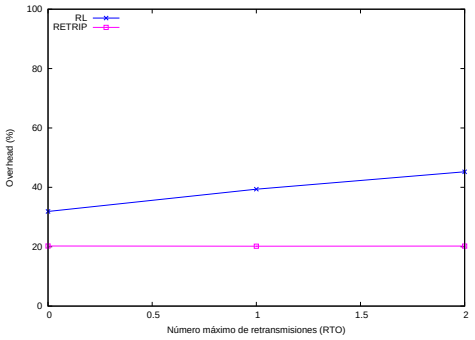

Figura 6.9: *Overhead* en la red debido a RTO

La red es MANET, lo que implica que los propios nodos verificadores realizan funciones de encaminamiento para otros nodos. Para evaluar la carga adicional sobre los nodos de la red debido al hecho de encaminar información de revocación para otros nodos, se ha evaluado el *overhead* medio en la red. En RETRIP, el *overhead* permanece cercano al 20 % con independencia tanto del número de nodos en la red (figura 6.7), como de la utilización de retransmisiones (RTO - *ReTransmissiOn*) (figura 6.9). Sin embargo, con RTO, la disponibilidad del servicio de RL aumenta (figura 6.8), a costa de aumentar el *overhead* en la red (figura 6.9).

Para finalizar, se presenta una tabla resumen comparando RETRIP con otros dos protocolos de validación de certificados utilizados en redes MANET: RL y ADOPT. Co-

mo muestra la tabla 6.3, el esquema eRL permite obtener las ventajas de la RL (no requiere TTPs *on-line* y disponibilidad elevada), y de ADOPT (tamaño de respuesta y capacidad de almacenamiento bajo). Por lo tanto, el esquema es apropiado para entornos de desconexión, en los que los recursos tanto del canal como de los dispositivos pueden ser limitados.

Tabla 6.3: Comparativa de eRL con otros protocolos de validación de certificados: RL y ADOPT

	RL	ADOPT	eRL
Requiere autoridades intermedias de confianza	No (utiliza repositorios)	Sí (requiere *responders*)	No (utiliza repositorios)
Capacidad de almacenamiento requerida por los nodos clientes	Elevado	Bajo	Bajo
Disponibilidad datos revocación	Elevado (contenida en la RL)	Bajo (respuestas previamente cacheadas)	Elevado (contenida en la eRL)
Tamaño de las respuestas	$\sim 100 kbytes$	$\sim 0{,}5 kbytes$	$\sim 0{,}8 kbytes$

6.5. Conclusiones

La validación de estado de certificados en redes móviles *ad-hoc* no es una tarea fácil, especialmente cuando los certificados fueron emitidos por una autoridad externa. En este caso, los mecanismos estándares de comprobación de estado de certificados pueden no ser viables, ya que la información y los servidores de confianza pueden no estar disponibles.

Las arquitecturas PKIX especialmente diseñadas para MANETs, tales como las PKI auto-organizadas y las PKI basadas en criptografía umbral generalmente proveen mecanismos de validación de certificados dentro de la MANET, pero la validez local de los certificados y la falta de soporte para la movilidad limitan su utilización.

La eRL permite llevar a cabo un mecanismo de validación de credenciales que aumenta la disponibilidad del mecanismo de validación de certificados, sin que sea necesaria la existencia de TTPs *on-line* y proporcionando respuestas de tamaño pequeño. Mediante la simulación de la propuesta eRL se ha mostrado que es viable validar certificados en entornos con capacidades limitadas tanto de comunicación como de almacenamiento.

En conclusión, la eRL completa el mecanismo para la gestión de la información de revocación para el modo desconectado, desde su generación (CARE, PREON) hasta su distribución (eRL) y validación (ePKI).

CAPÍTULO 7

CONCLUSIONES Y LÍNEAS FUTURAS

En esta tesis se ha presentado un análisis de las características de las redes ubicuas que influyen de forma directa en la validación de credenciales para ofrecer los servicios de autenticación y autorización. Como se ha expuesto, los nuevos tipos de redes y dispositivos permiten una gran flexibilidad para los usuarios, sin embargo también tienen ciertas peculiaridades que conllevan ciertas limitaciones tanto de los dispositivos de los usuarios como del tipo de red. La movilidad tanto de los clientes como de las entidades que ofrecen servicios, requiere de soluciones de seguridad de carácter global, basados en estructuras estandarizadas, que permitan una gestión distribuida y contemplen las limitaciones tanto de los dispositivos como de las nuevas redes.

Los certificados X.509 son una buena solución para la distribución de los atributos de un usuario, ya bien sea la clave pública para el servicio de autenticación, como sus privilegios para el servicio de autorización. Sin embargo, la gestión del ciclo de vida de estas credenciales requiere de la existencia de cierta infraestructura. Uno de los procesos más críticos es el de la gestión del estado de revocación de los certificados. Es decir, que cualquier entidad que quiera verificar un certificado, puede comprobar que ese certificado no ha dejado de ser válido antes de finalizar su periodo de validez. Si este proceso ya es considerado de por sí costoso, lo es aún más cuando aparecen cadenas de certificados donde se ha de verificar cada uno de los certificados que la componen.

La construcción y validación de caminos de certificados son procesos que consumen recursos, los cuales son especialmente críticos para dispositivos limitados, tales como PDAs, teléfonos móviles, etc. Por lo tanto, se requieren nuevas soluciones que permitan aumentar la disponibilidad de los servicios de autenticación/autorización

en entornos de computación ubicua y que permitan disminuir la carga de verificación en los procesos de validación de credenciales, teniendo en cuenta, tanto la disponibilidad de los servicios de seguridad, como la reducción en el consumo de recursos.

Para poder ofrecer las soluciones apropiadas, el primer paso ha sido analizar los procesos de validación, y detectar cuáles son los procesos que se ven afectados por los nuevos entornos, donde la conectividad global se puede perder y algunos servicios pueden estar temporalmente no disponibles. Como se ha presentado, los procesos que se ven directamente afectados son: la disponibilidad de la información tanto de revocación como de los certificados que se han de validar, las operaciones criptográficas (verificación de firma), y el volumen de información a obtener, sobre todo el de información de revocación. Evidentemente, el aumento de la disponibilidad de los servicios o la reducción de la carga en los procesos de validación de credenciales no ha de provocar una reducción del nivel de seguridad de los procesos involucrados.

La primera propuesta realizada se ha centrado en el diseño de una arquitectura de validación de credenciales para usuarios móviles, llamada ePKI. Esta arquitectura se ha diseñado de forma totalmente modular para simplificar la modificación o ampliación de funcionalidades, e intentando independizar la autorización de la solución para la validación de la identidad del usuario. La autenticación y autorización se proporcionan a los usuarios y aplicaciones a través de la combinación de los nuevos servicios de PMI y de un sistema de autenticación basado en confianza (PTM). Con esta solución un dispositivo móvil puede adaptarse al entorno (sistema *network-centric*). Es decir, cuando la conexión a sistemas centralizados de PKIX es posible, la validación puede llevarse a cabo de forma tradicional tanto para certificados de identidad como para certificados de atributo. Por otro lado, cuando esa conexión no es posible, la validación de autenticación se realiza en base a un sistema basado en confianza. De esta manera, la validación de autorización puede dar lugar a un mecanismo mixto que permite aumentar la disponibilidad del servicio tanto de autenticación como de autorización.

Sin embargo, uno de los procesos críticos sigue siendo el de la revocación, sobre todo cuando aparecen cadenas de certificados, como es el caso de la delegación. La delegación es un mecanismo que permite una gran flexibilidad, pero implica una gran carga en su validación. Esto se ha constatado en el análisis realizado sobre la revocación en cadenas de delegación. Durante el análisis se han identificado y definido los parámetros principales para evaluar los mecanismos de validación del estado de revocación en cadenas de certificados. Estos parámetros son la probabilidad de revocación de un certificado dentro de una cadena de certificados, el número de certificados a obtener y validar dada una cadena de certificados, y el volumen de información de revocación a distribuir.

El análisis realizado previamente indicaba que la aplicación de un sistema de re-

vocación en cascada podría reducir la carga en el sistema de validación. Por ese motivo, se ha realizado el diseño de la arquitectura CARE, la cual permite llevar a cabo la revocación en cascada. CARE permite reducir la carga en la validación del estado de revocación de cadenas de certificados. Sin embargo, se requiere centralizar ciertos procesos de la emisión de certificados. Además, se aumenta el volumen de la información de revocación que se ha de distribuir. Por lo tanto, sería conveniente buscar mecanismos que nos permitan mantener la reducción lograda con CARE y que además consigan reducir el volumen de datos de revocación a distribuir.

Como resultado del análisis y de la propuesta anterior se ha diseñado un nuevo sistema que permite mantener la reducción de la carga en el proceso de validación, eliminando la necesidad de centralizar la emisión de certificados y haciendo viable la distribución de la información de revocación mediante procedimientos *off-line*. Esta propuesta la hemos llamado PREON, y está basada en la utilización de códigos prefijo para identificar a los certificados bajo el dominio de una SOA. Este sistema se ha validado tanto de manera estadística como empírica. En el primer caso se han utilizado las técnicas y parámetros definidos en la evaluación de CARE. Como los resultados apuntaban mejoras sustanciales, se decidió utilizar una plataforma de *test* para su evaluación en escenarios más reales. Para el análisis de la solución anterior se han extendido las funcionalidades de CERVANTES, una plataforma de test para el estudio de los mecanismos de revocación. Entre las funcionalidades desarrolladas se encuentran: 1) permitir la generación de cadenas de delegación en forma de árbol con distribución probabilística; 2) análisis de las estadísticas de revocación, como por ejemplo obtener el porcentaje de certificados revocados de forma implícita o explícita; 3) permitir la modificación de la distribución de los eventos sobre el árbol generado, como por ejemplo, configurar qué certificados tienen mayor probabilidad de ser revocados o validados en función de su posición dentro del árbol, etc. Los resultados obtenidos sobre CERVANTES avalan los resultados analíticos.

En este punto ya se han solucionado muchos de los problemas que aparecían a priori. Se ha logrado aumentar la disponibilidad del servicio de validación de credenciales centrándonos en la disponibilidad de la información de autenticación, descentralizando el sistema de delegación, permitiendo la viabilidad de la distribución de la información de revocación al reducir su volumen. Pero todavía queda un punto importante. A pesar de que la información puede distribuirse de forma *off-line*, el acceso a dicha información se debía realizar desde un repositorio accesible. Con eRL se proporciona una solución para aumentar la disponibilidad de la información de revocación, reduciendo los recursos necesarios tanto de red como de los dispositivos que obtienen y ofrecen el servicio de revocación. Se ha utilizado una estructura estandarizada para ofrecer compatibilidad con las soluciones existentes. Además, la solución contempla la posibilidad de seleccionar el servicio de revocación en función de dife-

rentes parámetros, como por ejemplo, servidor más cercano o con información más actualizada (menor riesgo en la toma de decisiones). Para poder mejorar la disponibilidad de los datos de revocación, también se ha definido un protocolo que permite a los nodos de la red colaborar para ofrecer el servicio de revocación. Este mecanismo permite distribuir la carga entre los dispositivos y aumentar la disponibilidad del servicio de revocación. Para verificar las prestaciones de la solución de distribución de información de revocación en redes MANET, se ha desarrollado tanto un nuevo agente UDP como una nueva aplicación en el simulador de red ns-2. La modificación de UDP permite la obtención por parte de la capa de aplicación de los datos recibidos, algo que no contempla de forma directa el simulador. De la misma manera, la aplicación puede servir como base para desarrollos que dependan de la distribución de información de forma periódica, ya que la aplicación desarrollada es independiente del contenido de los datos distribuidos. Los resultados obtenidos confirman tanto la reducción del *throughput* requerido, como el aumento de la tasa de éxito en la obtención de información con respecto a otras soluciones.

Las líneas futuras de esta tesis son en su mayoría una continuación del trabajo ya realizado. En PREON sería interesante estudiar las características de la delegación que produce grafos acíclicos en general. Es decir, contemplar la posibilidad de que la delegación a una misma entidad por parte de dos entidades diferentes pueda llevarse a cabo a través de la emisión de un único certificado. También cabe la posibilidad de que desde una misma entidad con dos certificados diferentes puedan delegarse todos o parte de esos privilegios utilizando un único certificado. Estas situaciones conllevan el aumento de la carga en la validación ya que implican el descubrimiento y la validación de más de un camino de delegación. Además de conceptos teóricos sería interesante obtener ciertos resultados de las prestaciones tanto de PREON como de *eRL*. En PREON, el análisis realizado utilizando CERVANTES permite observar la carga del sistema tanto en cliente como en servidor, así como el *throughput* consumido. Sin embargo, se podría utilizar el sistema de emulación, bien sea en ns-2 o en su nueva versión ns-3, para conseguir entornos de red más realistas. De esta manera, se podrían obtener medidas de retardos, tasa de pérdidas, etc. También sería interesante implementar las soluciones en dispositivos móviles actuales para evaluar la carga real.

La ePKI podría ampliarse para proporcionar todos los servicios desarrollados (CARE, PREON y eRL), consiguiendo de esta manera una solución completa de validación de credenciales tanto para entornos conectados como desconectados. También se podrían introducir nuevos mecanismos para la toma de decisiones. Aún existe cierta probabilidad de no poder llevar a cabo la autenticación y autorización de una entidad (no encontrar un repositorio, información obsoleta, falta de recursos de red y del dispositivo, etc). Por lo tanto, se podrían introducir nuevos parámetros para seleccionar el mecanismo de validación de credenciales más adecuado en cada instante de tiempo.

BIBLIOGRAFÍA

[1] M.F. Hinarejos and J.L. Muñoz, J. Forné and O. Esparza. PREON: An Efficient Cascade Revocation Mechanism for Delegation Paths. *Computers & Security*, Accepted Manuscript, 2010.

[2] J. Forné, M.F. Hinarejos, A. Marín, F. Almenárez, J. López, J.A. Montenegro, M. Lacoste and D. Díaz. Pervasive Authentication and Authorization Infrastructures for Mobile Users. *Computers & Security*, 29(4):501–514, 2010.

[3] J. Forné and J.L. Muñoz and M.F. Hinarejos and O. Esparza. Certificate Status Validation in Mobile Ad-Hoc Networks. *IEEE Wireless Communications*, 16(11):55–62, 2009.

[4] M.F. Hinarejos, J. Forné. Revocation Scheme for PMI Based Upon the Tracing of Certificates Chains. In *Computational Science and Its Applications (ICCSA)*, Lecture Notes in Computer Science, pages 1098–1106. Springer, 2006.

[5] M.F. Hinarejos and J. Forné. A Lightweight Delegated Privileges Revocation Scheme Based on Coding. In *the conference on Applied Public Key Infrastructure (IWAP)*, pages 207–221. IOS Press, 2005.

[6] F. Almenarez, M. Carbonell, J. Forne, M.F. Hinarejos, M. Lacoste, A. Marin, and J.A. Montenegro. Design of an Enhanced PKI for Ubiquitous Networks. In *Sixteenth International Workshop on Database and Expert Systems Applications*, pages 262–266, Aug. 2005.

[7] M.F. Hinarejos J. Forné. Web-based authorization based on X.509 privilege management infrastructure. In *IEEE Pacific Rim Conference on Communications, Computers and signal Processing (PACRIM)*, 2003.

[8] J. Forné M.F. Hinarejos. L-DPR: un esquema ligero de revocación de privilegios delegados. In *V Jornadas de Ingeniería Telemática (JITEL)*, 2005.

[9] M.F. Hinarejos, J. Forné, F. Almenárez, A. Marín, M. Carbonell and J.A. Montenegro. Hacia una PKI mejorada para Comercio Electrónico Móvil. In *Simposio Español de Comercio Electrónico (SCE)*, 2005.

[10] M.F. Hinarejos, J. Forné. Seguimiento de cadenas de certificados para un sistema de revocación. In *VIII Reunión Española Sobre Criptología y Seguridad de la Información (RECSI)*, 2004.

[11] J. Forné M.F. Hinarejos. Actualización de la Información de Revocación de Privilegios mediante el Seguimiento de Cadenas de Certificados. In *XII Jornadas Telecom I+D*, 2004.

[12] J. Forné M.F. Hinarejos. Autorización en web basada en Infraestructuras de Gestión de Privilegios X.509. In *II Simposio Español de Comercio Electrónico (SCE)*, 2003.

[13] IEEE 80211. Wireless LAN medium access control and physical layer specification. Technical report, The Working Group for WLAN standards, 1997.

[14] R. Shirey. Internet Security Glossary, Version 2. RFC 4949 (Informational), August 2007.

[15] ITU-T Recommendation X.800. Security architecture - Open Systems Interconnection for CCITT applications, 1991.

[16] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992.

[17] NIST. Secure Hash Algorithm (SHA-1). FIPS PUB 180-1.

[18] NIST. Data Encryption Standard (DES). FIPS PUB 46-1.

[19] NIST. Triple DES. FIPS PUB 46-3.

[20] NIST. Advanced Encryption Standard (AES). FIPS PUB 197.

[21] Bluetooth. Web site: http://www.bluetooth.com/.

[22] L.M. Kohnfelder. Towards a practical public-key cryptosystem. Master's thesis, 1978. MIT Laboratory for Computer Science.

[23] SDSI/SPKI - Simple Public Key Infrastructure/Simple Distributed Security Infrastructure. Web site: http://www.syntelos.com/spki/.

[24] SESAME Project. Web site: https://www.cosic.esat.kuleuven.be/sesame/.

[25] M. Thompson, A. Essiari, S. Mudumbai. Certificate-based authorization policy in a PKI environment. *ACM Transactions on Information and System Security (TISSEC)*, 6(4):566–588, 2003.

[26] ITU/ISO Recommendation X.509. Information technology Open Systems Interconnection - The Directory: Public Key and Attribute Certificate Frameworks, 1997.

[27] ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection - The Directory: Autentication Frameworks, 2000. Technical Corrigendum.

[28] ITU-T Recommendation X.509. Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, 2005. Technical Corrigendum.

[29] International Telecommunication Union. ITU-T Recommendation X.680.

[30] C. Ellison. SPKI Requirements. RFC 2692 (Experimental), September 1999.

[31] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693 (Experimental), September 1999.

[32] SPKI Working Group. Web site: http://datatracker.ietf.org/wg/spki/charter/.

[33] B. Lampson R. L. Rivest. SDSI - A Simple Distributed Security Infrastructure. Technical report, 1996.

[34] P. Ashley and M. Vandenwauver. Practical Intranet Security: Overview of the State of the Art and Available Technologies, 1999.

[35] Computer Manufacturers' Association. Authentication and Privilege Attribute Security Application with Related key distribution functions, 1994.

[36] T. Parker. Sesame v4 overview, 1995.

[37] SPKI Working Group. http://acs.lbl.gov/akenti/. (último acceso 29-12-2009).

[38] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280 (Proposed Standard), April 2002. Obsoleted by RFC 5280, updated by RFCs 4325, 4630.

[39] S. Farrell, R. Housley, S. Turner. An Internet Attribute Certificate Profile for Authorization. RFC 5755 (Proposed Standard), January 2010.

[40] P. Zimmerman. The official pgp users guide, 1995.

[41] J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang. Providing robust and ubiquitous security support for mobile ad hoc networks. In *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols*, pages 251–260, Washington, DC, USA, 2001. IEEE Computer Society.

[42] Haiyun Luo and Songwu Lu. Ubiquitous and robust authentication services for ad hoc wireless networks. Technical report, Department of Computer Science, UCLA, 2000.

[43] L. Zhou and Z. J. Haas. Securing Ad-Hoc Networks. *IEEE Network Magazine*, 13:24–30, 1999.

[44] Lidong Zhou, Fred B. Schneider, and Robbert van Renesse. COCA: A Secure Distributed On-line Certification Authority. Technical report, Ithaca, NY, USA, 2000.

[45] Y.G. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 307–315, New York, NY, USA, 1989. Springer-Verlag New York, Inc. http://www.uml.org/.

[46] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 146–155, New York, NY, USA, 2001. ACM.

[47] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, 2003.

[48] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[49] ETSI TS 102 158. Electronic signatures and Infrastructures (ESI); Policy Requirements for Certification Service Providers issuing attribute certificates usable with Qualified certificates, 2003.

[50] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. RFC 4510 (Proposed Standard), June 2006.

[51] S. Farrell, R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC 3281 (Proposed Standard), April 2002.

[52] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

[53] L. Zhu, K. Jaganathan, and N. Williams. Online Certificate Status Protocol (OCSP) Support for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). RFC 4557 (Proposed Standard), June 2006.

[54] Ralph C. Merkle. A Certified Digital Signature. In *Advances in Cryptology*, Lecture Notes in Computer Science, pages 218–238. Springer, 1990.

[55] S. Lloyd. PKI Forum: Understanding Certification Path Construction. Technical report, White Paper, 2002.

[56] UBISEC Project. Web site: http://websrv2.c-lab.de/ubisec/.

[57] Mozilla. Web site: http://www.mozilla.com/.

[58] Google Chrome. Web site: http://www.google.es/chrome/.

[59] Interet Explorer. Web site: http://www.microsoft.com/spain/windows/internet-explorer/.

[60] UBISEC Project. Deliverable D2.2: Specification of Security Modules. 2006.

[61] OpenCA PKI Project. Web site: http://www.openca.org/projects/openca/.

[62] The Object Management Group (OMG). Unified Modeling Language (UML). Web site: http://www.uml.org/.

[63] UBISEC Project. Deliverable D2.3: Design and Proof-of Concept Implementation of the Enhanced PKI. 2006.

[64] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFC 3546.

[65] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681.

[66] F. Almenarez, A. Marin, D. Diaz, and J. Sanchez. Developing a model for trust management in pervasive devices. In *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 5–271, March 2006.

[67] UBISEC Project. Deliverable D4.1.2: Specification of UBISEC Applications and Scenarios. 2006.

[68] C. Campo, C. García-Rubio, A. Marín and F. Almenárez. PDP: a lightweight discovery protocol for local-scope interactions in wireless ad hoc networks. *Comput. Networks*, 50(17):3264–3283, December 2006.

[69] P. Quan, J. Reid, A. McCullagh, E. Dawson. Commitment issues in delegation process. In *Proceedings of the sixth Australasian conference on Information security (AISC)*, volume 81, pages 27–38. Australian Computer Society, Inc., 2008.

[70] R. Sandhu. Role activation hierarchies. In *Proceedings of the third ACM workshop on Role-based access control (RBAC)*, pages 33–40. ACM, 1998.

[71] J. Crampton, H. Khambhammettu. Delegation in role-based access control. In *Computer Security – ESORICS*, Lecture Notes in Computer Science, pages 174–191, 2006.

[72] A. Schaad, J.D. Moffett. A framework for organisational control principles. In *Computer Security Applications Conference*, pages 229–238, 2002.

[73] E. Barka, R. Sandhu. Framework for role-based delegation models. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC)*, page 168. IEEE Computer Society, 2000.

[74] A. Hagstrom, S. Jajodia, F. Parisi-Presicce, D. Wijesekera. Revocations-a classification. In *Proceedings of the 14th IEEE workshop on Computer Security Foundations (CSFW)*, pages 44–58. IEEE Computer Society, 2001.

[75] D. Clarke, J.E. Elien, C. Ellison, M. Fredette, A. Morcos, R.L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[76] B. Ramsdell. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. RFC 3851 (Proposed Standard), July 2004.

[77] S. Kent, K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005.

[78] T. Dierks, E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008.

[79] F. Almenarez, D. Díaz, A. Marín. http://www.it.uc3m.es/pervasive/wce-lite-compat/. This work has received the support of the European Commission through the IST programme, as part of the IST UBISEC project.

[80] International Telecommunication Union. Information Technology — ASN.1 Encoding Rules — Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER). ITU-T Recommendation X.690, July 2002.

[81] D. Mundy, D. Chadwick. Comparing the Performance of Abstract Syntax Notation One (ASN.1) vs eXtensible Markup Language (XML). In *Terena Networking Conference, Zagreb*, 2003.

[82] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820 (Proposed Standard), 2004.

[83] A. Arnes. Public Key Certificate Revocation Schemes. PhD thesis, Norwegian University of Science and Technology, 2000.

[84] T.P. Hormann, K. Wrona and S. Holtmanns. Evaluation of certificate validation mechanisms. *Computer Communications*, 29(3):291–305, 2006.

[85] K. Papapanagiotou, G.F. Marias and P. Georgiadis. Revising centralized certificate validation standards for mobile and wireless communications. *Computer Standards and Interfaces*, 2009. DOI: 10.1016/j.csi.2009.07.001.

[86] José L. Muñoz Tapia. Desing and Evaluation of Certificate Revocation Systems . In *Tesis Doctoral*. Universitat Politècnica de Catalunya, 2003.

[87] D. Chadwick, S. Otenko and T. Nguyen. Adding support to XACML for multi-domain user to user dynamic delegation of authority. *International Journal of Information Security (IJIS)*, 8(2):137–152, 2009.

[88] H. Khurana and V.D. Gligor. Review and revocation of access privileges distributed with pki certificates. In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 100–112, London, UK, 2001. Springer-Verlag.

[89] M. Blinov and C. Adams. Alternative Certificate Formats for the Public-Key Infrastructure Using X.509 (PKIX) Certificate Management Protocols. RFC 4212 (Informational), October 2005.

[90] T. Freeman, R. Housley, A. Malpani, D. Cooper, and W. Polk. Server-Based Certificate Validation Protocol (SCVP). RFC 5055 (Proposed Standard), dec 2007.

[91] Steve Lloyd. PKI Forum: Understanding Certification Path Construction, sep 2002.

[92] D. Pinkas and R. Housley. Delegated Path Validation and Delegated Path Discovery Protocol Requirements. RFC 3379 (Informational), September 2002.

[93] S. Eichler and B. Muller-Rathgeber. Performance Analysis of Scalable Certificate Revocation Schemes for Ad Hoc Networks. In *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*, pages 382–391, Washington, DC, USA, 2005. IEEE Computer Society.

[94] D. Chadwick. Delegation Issuing Service. In *NIST 4th Annual PKI Workshop*, pages 62–73, 2005.

[95] A. Cecchini, R. Alfieri, R. Cecchini, V. Ciaschini, Á. Frohner, A. Gianoli, K. Lőrentey, F. Spataro. Voms, an authorization system for virtual organizations. In *Proceedings of the 1st European Across Grids Conference*, Lecture Notes in Computer Science, pages 33–40. Springer Berlin / Heidelberg, 2004.

[96] J.L. Muñoz, J. Forné, O. Esparza and M. Soriano. Cervantes. a certificate validation test-bed. In *Public Key Infrastructure*, volume 3093 of *LNCS*, pages 28–42. Springer-Verlag, 2004.

[97] CERVANTES. Web site: http://sourceforge.net/projects/cervantes/.

[98] J. Berstel, D. Perrin. *Theory of Codes*. Academic Press, Inc., Orlando, FL, USA, 1985.

[99] H. Huang, S.F. Wu. An approach to certificate path discovery in mobile ad hoc networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 41–52. ACM, 2003.

[100] F. Templin, T. Gleeson, M. Talwar, and D. Thaler. Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). RFC 4214 (Experimental), October 2005. Obsoleted by RFC 5214.

[101] M. Myers, X. Liu, J. Schaad, and J. Weinstein. Certificate Management Messages over CMS. RFC 2797 (Proposed Standard), April 2000. Obsoleted by RFC 5272.

[102] C. Adams, S. Farrell, T. Kause, T. Mononen. Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard), September 2005.

[103] J. Schaad. Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF). RFC 4211 (Proposed Standard), September 2005.

[104] M. Brown and R. Housley. Transport Layer Security (TLS) Authorization Extensions. RFC 5878 (Experimental), May 2010.

[105] C. Adams, S. Farrell. Internet X.509 Public Key Infrastructure Certificate Management Protocols. RFC 2510 (Proposed Standard), March 1999. Obsoleted by RFC 4210.

[106] J. Schaad and M. Myers. Certificate Management over CMS (CMC). RFC 5272 (Proposed Standard), June 2008.

[107] R.S. Sandhu, J.S. Park. Binding identities and attributes using digitally signed certificates. In *16th Annual Computer Security Applications Conference (ACSAC)*, pages 120–127, 2000.

[108] J. Lopez I. Agudo, C. Fernandez-Gago. Delegating Privileges over Finite Resources: A Quota Based Delegation Approach. pages 302–315, 2009.

[109] M. Nelson. *The Data compression book*. M & T Books, New York, NY, 1995.

[110] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501 (Informational), January 1999.

[111] H. W. Go, P. Y. Chan, Y. Dong, A. F. Sui, S. M. Yiu, Lucas C. K. Hui, and Victor O. K. Li. Performance evaluation on crl distribution using flooding in mobile ad hoc networks (manets). In *ACM-SE 43: Proceedings of the 43rd annual Southeast regional conference*, pages 75–80, New York, NY, USA, 2005. ACM.

[112] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), February 2007.

[113] G.F. Marias, K. Papapanagiotou, V. Tsetsos, O. Sekkas, and P. Georgiadis. Integrating a trust framework with a distributed certificate validation scheme for manets. *EURASIP Journal Wireless Communication Network*, 2006(2):77–77, 2006.

[114] Pervasive Discovery Protocol. Web site: http://www.it.uc3m.es/celeste/pdp/.

[115] J.L. Muñoz, O. Esparza, C. Gañan and J. Parra-Arnau. Pkix certificate status in hybrid manets. In *WISTP '09: Proceedings of the 3rd IFIP WG 11.2 International Workshop on Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks*, pages 153–166, Berlin, Heidelberg, 2009. Springer-Verlag.

[116] J.L. Muñoz, J. Forné, O. Esparza and M. Soriano. Certificate revocation system implementation based on the Merkle hash tree. *International Journal of Information Security*, 2(2):110–124, 2004.

[117] Network Simulator 2. Web site: http://nsnam.isi.edu/nsnam.

[118] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.